Universidade Católica de Brasília

PROGRAMA DE PÓS-GRADUAÇÃO
LATO-SENSU EM ENGENHARIA DE SOFTWARE

Especialização

Processo de Gestão de Configuração e Mudanças Apoiado por Software Livre - PGCM

Autores: Alexandre Antônio Antunes de Almeida – UC04083839

José Junio Evangelista da Silva – UC04084009

Orientador: Prof. MSc. Candido Guerrero Salgado

Co-orientador: Prof. MSc. Fábio Bianchi Campos

BRASÍLIA 2006

ALEXANDRE ANTÔNIO ANTUNES DE ALMEIDA

JOSÉ JUNIO EVANGELISTA DA SILVA

PROCESSO DE GESTÃO DE CONFIGURAÇÃO E MUDANÇAS APOIADO POR SOFTWARE LIVRE - PGCM

Dissertação apresentada ao Programa de Pós Graduação "Lato Sensu" em Engenharia de Software da Universidade Católica de Brasília, como requisito para obtenção do Título de Especialista em Engenharia de Software.

Orientador: Prof. MSc. Candido Guerrero Salgado

Co-Orientador: Fábio Bianchi Campos

Brasília/2006

TERMO DE APROVAÇÃO

Dissertação defendida e aprovada como requisito parcial para obtenção de título de Especialista em Engenharia de Software, defendida e aprovada em 08 de abril de 2006, pela banca examinadora constituída por:

_____

Prof. MSc. Cândido Guerrero Salgado

Orientador

_____

Prof. MSc. Vilson Carlos Hartmann

Examinador

_____

Prof. MSc. Vilmondes Gomes da Silva

Examinador

DEDICATÓRIA

Á Deus, sem o qual nada seria possível.

Á minha querida esposa, que foi paciente e compreensiva nas horas e dias que precisei me dedicar quase que exclusivamente a este trabalho.

Aos mestres, que na verdadeira essência da palavra, souberam cumprir seus objetivos como orientadores e amigos.

Á todos aqueles que de alguma maneira contribuíram para que este projeto fosse realizado.

Alexandre Antonio Antunes de Almeida

Á Deus, por me iluminar com o Espírito Santo. Espírito este que me conduziu e me deu coragem para enfrentar todas as dificuldades e assim alcançar a mais esta graça.

Á minha sábia mãe e ao meu pai (*in_memorian*), fontes inspiradoras de luta, de coragem e de perseverança.

À minha irmã querida, exemplo de simplicidade e sabedoria.

À minha namorada, que me motivou nos momentos de dificuldade.

Aos meus colegas de trabalho, nos quais encontrei apoio e motivação na construção deste trabalho.

José Junio Evangelista da Silva

AGRADECIMENTOS

Aos nossos professores, que nos apoiaram e nos conduziram ao conhecimento tão necessário neste mundo globalizado. E em especial, aos professores Cândido Guerrero e Fábio Bianchi que, com dedicação e profissionalismo, nos auxiliaram na construção deste trabalho.

LISTA DE QUADROS

Quadro 1 - Grupo de Atividades do RUP para GCS 25
Quadro 2 - Perspectivas do CMMI [CMMI02] 29
Quadro 3 - Níveis e Processos do MR-MPS 36
Quadro 4 - Abordagens das subáreas da Área de Conhecimento de GCS do SWEBOK ... 40
Quadro 5 - Comparativo dos padrões e modelos 48
Quadro 6 - Estágios e Produtos do Trabalho [TOMER05] 69
Quadro 7 - Exemplo da aplicação do USDP 72
Quadro 8 - Artefatos das Disciplinas Fundamentais (USDP) 74
Quadro 9 - Artefatos das Disciplinas de Apoio (USDP) 75
Quadro 10 - Papéis do PGCM 77
Quadro 11 - Sugestão de Estrutura de Diretórios 155

LISTA DE TABELAS

Tabela 1 - Avaliação das atividades do PGCM 102
Tabela 2 - Avaliação dos papéis do PGCM. 104
Tabela 3 - Avaliação dos papéis do PCM. 104

LISTA DE FIGURAS

Figura 1 - Gerência de Configuração de Software e Controle de Mudanças [RUP01] 8
Figura 2 - Controle de Versões [ASKLUND02] 15
Figura 3 - Métricas por Área Funcional [FARAH04]..... 19
Figura 4 - Execução das atividades de GCS na fase de iniciação [RUP01] 27
Figura 5 - Execução das atividades de GCS nas fases, exceto iniciação [RUP01]..... 27
Figura 6 - Representação Contínua do CMMI [CMMI05] 30
Figura 7 - Representação Estagiada do CMMI [CMMI05] 30
Figura 8 - Metas de GC do CMMI (Nível 2 de maturidade) [FIORINI98] 31
Figura 9 - Estrutura da norma NBR ISO/IEC 12207 [ROCHA04] 33
Figura 10 - Subáreas da Área de Conhecimento de GCS do SWEBOK [SWEBOK04]..... 40
Figura 11 - Arquitetura CVS..... 51
Figura 12 - Fluxo de Trabalho do CVS 54
Figura 13 - Ciclo de vida do bug no Bugzilla [BARNSON05] 60
Figura 14 - Fases e Disciplinas do USDP [RUP01] 67
Figura 15 - Ambiente de Gerência de Configuração para grandes Organizações 78
Figura 16 - Bibliotecas de Gestão de Configuração de Software 79
Figura 17 - Comparativo dos valores obtidos na avaliação das atividades do PGCM 103
Figura 18 - Exemplos de Modelos da UML 120
Figura 19 - Exemplo de Ramos 122
Figura 20 - Processo básico de solicitação de mudanças..... 125
Figura 21 - Grafo de Evolução..... 127
Figura 22 - Estrutura de Diretórios 157
Figura 23 - Exemplo de Ciclo de Vida de uma CR 163
Figura 24 - Diagrama de Atividades de Controle de Mudanças. 164

LISTA DE FIGURAS DO TUTORIAL

Figura Tutorial 1 - Inicialização do Repositório 130
Figura Tutorial 2 - Seleção de Perspectiva 131
Figura Tutorial 3 - Perspectiva “CVS Repositories” 131
Figura Tutorial 4 - Adicionando um Repositório 132
Figura Tutorial 5 - Repositório do Projeto 132
Figura Tutorial 6 - Estrutura do Repositório 133
Figura Tutorial 7 - Saída do Comando de Importação 134
Figura Tutorial 8 - Projeto Importado para o Repositório 134
Figura Tutorial 9 - Dados para alteração 135
Figura Tutorial 10 - Módulo de Arquivos Fontes 135
Figura Tutorial 11 - Assistente de “check-out” 136
Figura Tutorial 12 - Alterando Perspectiva 136
Figura Tutorial 13 - Módulo “01.Fontes” 137
Figura Tutorial 14 - Abrindo arquivo para edição 138
Figura Tutorial 15 - Efetuando Alteração 138
Figura Tutorial 16 - Executando “commit” 139
Figura Tutorial 17 - Comentando uma alteração 140
Figura Tutorial 18 - Progresso do “commit” 140
Figura Tutorial 19 - Nova Versão: 1.2 141
Figura Tutorial 20 - Tela inicial do Bugzilla 142
Figura Tutorial 21 - Produtos (projetos) cadastrados 142
Figura Tutorial 22 - Informações sobre o projeto 143
Figura Tutorial 23 - Confirmação de cadastro do novo produto 143
Figura Tutorial 24 - Cadastro de Componente 144
Figura Tutorial 25 - Confirmação de Cadastro de Componente 144
Figura Tutorial 26 - Pesquisando Usuários 144
Figura Tutorial 27 - Cadastro de Usuários 145
Figura Tutorial 28 - Permissões para o Usuário/Grupo 145
Figura Tutorial 29 - Confirmação de atualização 146
Figura Tutorial 30 - Selecionando o Produto 146
Figura Tutorial 31 - Cadastro de uma nova CR (*Bug*) 147

Figura Tutorial 32 - Estado da CR 148
Figura Tutorial 33 - Tipos de Resolução de CR 149
Figura Tutorial 34 - Desenvolvedor efetuando *Login* 149
Figura Tutorial 35 - Página principal do Bugzilla 150
Figura Tutorial 36 - Página de Pesquisa de CR 150
Figura Tutorial 37 - CR cadastradas 151
Figura Tutorial 38 - CR Atribuída 151
Figura Tutorial 39 - Alterando resolução para Corrigida (*Fixed)* 151
Figura Tutorial 40 - Fechamento da CR 152
Figura Tutorial 41 - Fechamento da CR 152

RESUMO

A Engenharia de Software é uma disciplina da área de Tecnologia da Informação que vem evoluindo a passos largos nos últimos anos, e como a engenharia tradicional (civil e elétrica, por exemplo), vem procurando estabelecer processos e padrões que possam ser amplamente utilizados. Esta busca por padrões nos seus processos objetiva a obtenção da qualidade dos produtos gerados. Estes processos, por sua vez, possuem elementos fundamentais e elementos de apoio, sem os quais seria muito difícil o seu estabelecimento de maneira madura. Neste contexto, surge a disciplina de Gerência de Configuração e Controle de Mudanças, que vem apoiar o processo de desenvolvimento do software, controlando versões dos artefatos produzidos, bem como registrando suas mudanças ao longo do tempo. Entretanto, por se tratar de um elemento "meramente" administrativo, estes processos tendem a serem deixados de lado. Este, dentre outros motivos, conduziu à realização deste trabalho, que define um processo de gerência de configuração e controle de mudanças apoiado por ferramentas de software livre. Os processos aqui definidos encontram fundamento nas normas vigentes, bem como atende ao CMMI – Capability Maturity Model Integration, do Software Engineering Institute, tendo como modelo de desenvolvimento o Processo Unificado de Desenvolvimento de Software (USDP – Unified Software Development Process). O processo proposto foi definido e sumariamente avaliado por dois órgãos do Governo Federal. Uma vez obtida esta avaliação, o trabalho foi concluído com um processo seguramente mais maduro e próximo da completude ideal, não deixando de ser didático, por ser de caráter acadêmico.

PALAVRAS-CHAVE: Engenharia de Software, Gerência de Configuração de Software, Controle de Mudanças, Controle de Versões, Software Livre.

SUMÁRIO

Lista de Quadros vi
Lista de Tabelas vii
Lista de Figuras viii
Lista de Figuras do Tutorial ix
Resumo xi
Introdução 1
Objetivo 5
Capítulo 1 Referencial Teórico 6
1.1 Gerência de Configuração de Software e Controle de Mudanças 6
1.1.1 O cerne do problema: a modificação 6
1.1.2 O que é Gerenciamento de Configuração e de Solicitações de Mudança 8
1.1.3 Finalidade dos Sistemas de Gerência de Configuração de Software 11
1.1.4 Relação com Outras Disciplinas 12
1.1.5 Gerência de Configuração sob o ponto de vista gerencial 13
1.1.6 Gerência de Configuração sob o ponto de vista do desenvolvimento 14
1.1.7 Plano de Gerenciamento de Configuração (PGC) 16
1.1.8 Atividades Críticas de Gerência de Configuração 17
1.1.9 Métricas para Gerência de Configuração de Software 18
1.2 Normas, Padrões e Modelos de Melhorias de Processos 23
1.2.1 USDP/RUP 23
1.2.2 CMMI-SW 27
1.2.3 NBR ISO/IEC 12207 32
1.2.4 NBR ISO 10007 34
1.2.5 MPS.BR 35
1.2.6 SWEBOK (Software Engineering Body of Knowledge) 38
1.2.7 CMBOK 41
1.2.8 ITIL 43
1.2.9 Comparativo das atividades das normas, padrões e modelos de melhoria de Processos 48
1.3 Ferramentas de Apoio à Gerência de Configuração e Controle de Mudanças 49
1.3.1 CVS – Concurrent Version System 50
1.3.2 Bugzilla 56
1.3.3 Eclipse 61
Capítulo 2 Processo de Gestão de Configuração e Mudanças – PGCM 65
2.1 O PGCM e o USDP 66
2.1.1 Estágios do Nível do Sistema 69
2.1.2 Estágios do Nível de Item de Configuração 70
2.1.3 Um exemplo de desenvolvimento iterativo 72
2.1.4 Seleção dos Itens de Configuração de Software 73
2.1.5 Método de Promoção das Linhas Bases 75

2.1.6 Definição dos Papéis do Sistema de GCS e CM 76
2.2 O PGCM 77
2.2.1 Papel: Diretor de Gerência de Configuração e de Controle de Mudanças ... 78
2.2.2 Papel: Gerente de Configuração 82
2.2.3 Papel: Solicitante 86
2.2.4 Papel: Gerente de Controle de Mudanças 87
2.2.5 Papel: Integrador 89
2.2.6 Papel: Desenvolvedor 92
2.2.7 Papel: Qualquer Papel 95
Capítulo 3 Estudo de Caso 97
3.1 Introdução 97
3.2 Metodologia 97
3.3 Questionários de Avaliação 100
3.4 Estratégias de Avaliação 101
3.5 Características das Empresas e dos Avaliadores 101
3.6 Análise dos resultados da avaliação do PGCM 102
3.6.1 Quanto às atividades 102
3.6.2 Quanto aos papéis 103
3.7 Ajustes no PGCM 105
3.8 Lições Aprendidas 105
Capítulo 4 Conclusão 107
Referências Bibliográficas 110
Glossário 115
Apêndice A – Tutorial das Ferramentas de GC 129
Apêndice B – Identificação dos Produtos do Projeto 153
Apêndice C – Processo de Controle de Mudanças: Detalhamento 158
Apêndice D – Questionário de Avaliação da Empresa e do Perfil do Avaliador 165
Apêndice E – Questionário de Avaliação do PGCM 166
Apêndice F – Questionário de Avaliação do PCM 170

INTRODUÇÃO

Contexto

O processo de desenvolvimento de software vem ao longo dos anos sofrendo constantes modificações. Estas modificações são provenientes da necessidade da melhoria da qualidade de seus processos, pois cada vez mais a concorrência vem exigindo das empresas produtoras de software a certificação dos seus processos produtivos. Para que estas empresas tenham sucesso, uma disciplina de suporte ao processo de desenvolvimento é necessária. Neste contexto, surge a disciplina de Gerência de Configuração de Software. Originalmente, esta disciplina possuía seu foco no controle de código-fonte. Entretanto, esta disciplina abrange uma grande quantidade de artefatos relacionados ao processo de desenvolvimento do software, indo além do armazenamento de versões de código-fonte. O Gerenciamento de Configuração de Software abrange o desenvolvimento e a aplicação de padrões e procedimentos para gerenciar um produto de um sistema em desenvolvimento. Fica evidente a necessidade de gerenciar os sistemas em desenvolvimento, pois à medida que eles são desenvolvidos, são criadas muitas versões do software. Essas versões incorporam propostas de mudanças, bem como soluções de defeitos, além de adaptações para outros tipos de plataformas de hardware e sistemas operacionais. É necessário manter o controle das mudanças que foram implantadas e de como essas mudanças foram incluídas no software. As modificações de um projeto de software podem ocorrer a qualquer momento. Por este motivo, a disciplina de Gerência de Configuração de Software é considerada uma disciplina de suporte. Neste momento, torna-se importante fazer uma distinção entre suporte de software e o suporte fornecido pela disciplina de Gerência de Configuração de Software. Por suporte de software entende-se como o conjunto de atividades que ocorrem depois que o software foi entregue ao cliente e colocado em operação. Gerência de Configuração de Software é o conjunto de atividades de acompanhamento e controle. Começa no início do projeto do software, e só termina quando o software é retirado de operação.

O processo de gerenciamento da configuração e da documentação associada deve ser baseado em padrões estabelecidos. Dentro de uma organização empresarial, estes padrões devem ser publicados em um manual de gerenciamento de configuração ou como parte de um manual de qualidade. Na verdade, não importa qual padrão seja considerado ponto de partida, uma vez que todos os padrões definidos por órgãos internacionais apresentam processos

comparáveis. Entretanto, tanto nos requisitos de qualidade da ISO 9000, por exemplo, quanto nos modelos de maturidade de processo de software, as organizações devem definir e seguir os requisitos de maneira formal, para que possam obter a certificação da qualidade dos seus processos de desenvolvimento.

Outro elemento importante a ser contextualizado é a questão do software livre. Na última década, houve uma crescente demanda na utilização e produção de software com licença pública. Assim, foram surgindo ferramentas para usos diversos, linguagens de programação, bem como aplicativos de uso específico. Várias ferramentas de apoio ao desenvolvimento do projeto de software foram criadas, além de outras que surgiram em décadas anteriores e que evoluíram. Enfim, isto se tornou uma tendência mundial. Diversos governos institucionalizaram a adoção do software livre, pois esta medida propiciaria uma diminuição drástica do orçamento empregado na aquisição de licenças de software. Aliado a este argumento, tem-se também o crescente aumento da qualidade deste tipo de software, bem como a possibilidade de modificação dos códigos-fontes, tornando-se assim fáceis de serem adaptados às necessidades governamentais. Como não poderia ser diferente, diversas ferramentas de apoio ao processo de gerência de configuração de software foram desenvolvidas. Muitas destas ferramentas são comerciais. Por outro lado, existem aquelas ferramentas desenvolvidas sob licença pública, e que atendem perfeitamente às necessidades de apoio ao referido processo. Assim, a partir do momento que se deseja definir um processo de gerência de configuração de software aderente a normas e padrões, nada mais justo que apoiar este processo com o uso de ferramentas de software livre, uma vez que este tipo de software encontra-se em ampla utilização nos órgãos do governo brasileiro.

Motivação

A disciplina de Gerência de Configuração encontra-se presente como requisito em vários modelos de maturidade de desenvolvimento de software, além de ser definida a partir de uma série de padrões internacionais. Muitas organizações governamentais, assim como empresas privadas, têm buscado melhorar a produtividade e qualidade a partir da melhoria dos seus processos de desenvolvimento de software. Uma quantidade razoável de padrões é publicada freqüentemente, porém, em muitos casos, estes padrões são difíceis de serem interpretados e seguidos. Além do mais, a bibliografia produzida nacionalmente não dá a

devida importância para o processo de Gerência de Configuração, ocasionando a falta de subsídio para que as empresas possam implantar estes processos de maneira formal. Por este motivo, julgou-se viável a elaboração de um Processo de Gerência de Configuração e Controle de Mudanças que pudesse servir de parâmetro para as organizações que desejarem implantar um processo formal.

Para que a implantação do processo em questão se torne economicamente viável, definiu-se que este seria acompanhado de sugestões referentes a ferramentas livres que pudessem apoiá-lo. Desta maneira, o binômio "processo-ferramenta" mostrou-se ideal, pois seria um valoroso guia, facilitando a complexa tarefa de estabelecimento e implantação de um processo padrão de gerência de configuração e controle de mudanças.

Organização

Este documento encontra-se organizado em quatro capítulos, além de um conjunto de apêndices. O primeiro capítulo, denominado "Referencial Teórico", apresenta a terminologia adotada na disciplina de Gerência de Configuração. Além desta terminologia, apresenta um conjunto de normas e padrões que definem seus conceitos e o emprego da Gerência de Configuração e Controle de Mudanças, comparando-os entre si através de um quadro resumo. Alguns modelos de melhoria de processos são apresentados, de forma que o leitor visualize o contexto onde se enquadra a real necessidade da disciplina de Gerência de Configuração, uma vez que esta se encontra presente nestes modelos, sendo de cunho fundamental para a sua formalização.

O segundo capítulo deste trabalho, denominado "Processo de Gestão de Configuração e Mudanças – PGCM", é que define o processo propriamente dito. Quanto à apresentação dos processos de gerência de configuração e controle de mudanças, optou-se por apresentá-los separadamente, pois o primeiro surge em função do segundo, ou seja, a partir da necessidade do controle das modificações do software, o controle de versões torna-se necessário. Esta separação objetiva o melhor entendimento dos processos, pois facilita ao leitor o seu entendimento, atingindo assim um bom nível didático.

O terceiro capítulo denomina-se "Estudo de Caso". Neste capítulo é apresentada a metodologia de avaliação do processo, bem como os resultados obtidos. O quarto capítulo apresenta a conclusão do trabalho, indicando a possibilidade de trabalhos futuros na área de gerência de configuração e controle de mudanças. Quanto aos apêndices, estes incluem o

tutorial de utilização das ferramentas de apoio do processo, um guia para identificação dos produtos do projeto, o detalhamento do Processo de Controle de Mudanças, bem como os modelos de questionários de avaliação empregados no estudo de caso.

OBJETIVO

Este trabalho tem como objetivo a definição de um processo de gerência de configuração e controle de mudanças, apoiado por ferramentas de software livre, baseado no USDP (Unified Software Development Process - Processo de Unificado de Desenvolvimento de Software) e aderente a normas e padrões atuais da disciplina de Gerência de Configuração. Será um processo padrão, que poderá ser adaptado conforme a organização e o projeto onde será utilizado.

Para que o objetivo acima descrito seja alcançado, alguns objetivos intermediários foram definidos, no sentido de abordar os principais conceitos da disciplina, bem como as normas, padrões e modelos de maturidade vigentes no mercado. Além disso, os processos propostos por este trabalho foram submetidos à avaliação por parte de dois órgãos do governo, sendo um da área de prestação de serviços (autarquia) e outro da área de segurança nacional, de maneira que já pudessem ter algum nível de experimentação prática.

CAPÍTULO 1 REFERENCIAL TEÓRICO

1.1 Gerência de Configuração de Software e Controle de Mudanças

Gerenciamento de Configuração de Software (GCS) é uma disciplina de Engenharia de Software que compreende as ferramentas e técnicas (processos ou metodologia) que uma organização usa para gerenciar modificações do projeto do software. A norma IEEE 828-1998 (Padrão para Planos de Gerência de Configuração de Software) define a GCS da seguinte maneira:

A disciplina de Gerência de Configuração de Software é constituída de boas práticas de engenharia de software para qualquer tipo de projeto de software. É adequada para qualquer modelo de ciclo de vida de desenvolvimento. Ela aumenta a confiabilidade e qualidade do software através das seguintes práticas [BELLAGIO05]:

- Provendo estrutura para identificação e controlando a documentação e o código, interfaces, e bancos de dados para suportar todas as fases do ciclo de vida do projeto do software;
- Através do suporte de uma metodologia de desenvolvimento e de manutenção que se adaptam aos requisitos, padrões, políticas, organização e disciplina de gerenciamento;
- Produzindo informações referentes ao estado das linhas bases, controle de modificações, testes, liberações, auditorias e assim sucessivamente.

1.1.1 O cerne do problema: a modificação

Inequivocamente, o software é muito propenso a modificações. Diferentemente dos sistemas de hardware, o software por não ser tangível fisicamente, fica restrito apenas pelo limite da criatividade humana. Desta forma, a desorganização poderá surgir no ambiente de desenvolvimento.

A necessidade do Gerenciamento da Configuração de Software é reconhecida atualmente pela maioria dos modelos de maturidade de software, tais como CMMI

(Capability Maturity Model Integration) [CMMI02], SPICE (Software Process Improvement and Capability Determination) [SPICE03], MPS.BR [MPS-BR05], dentre outros. Entretanto, os projetos ainda continuam a mercê de problemas que poderiam ser evitados através do uso das ferramentas de Gerência de Configuração de Software e de seus processos apropriados. Estas falhas acabam gerando software com baixa qualidade, com problemas de atrasos na entrega do produto, estouro de orçamento, além da incapacidade de atender de forma eficiente às demandas dos clientes.

Para entender corretamente a gerência de configuração de software, faz-se uma analogia com um sistema de hardware, onde seus componentes possuem características físicas que fazem com que a ausência do gerenciamento de configuração seja facilmente percebida. Um exemplo simples a ser considerado é o computador pessoal. Um computador pessoal possui um processador, uma placa-mãe, memória, um disco rígido, um monitor, além do teclado e o mouse. Cada um destes itens possui interfaces que permitem a perfeita conexão com os outros itens de hardware. Todos esses itens devem ser compatíveis. Se o conector do mouse não for compatível com a referida porta na placa-mãe, tem-se um problema de compatibilidade, e o sistema poderá não funcionar corretamente, pelo menos como esperado. Se o processador não for compatível com o respectivo encaixe (slot) da placa-mãe, o problema é ainda mais grave.

Fica fácil perceber que quando o hardware é fabricado, as interfaces que são essenciais para a operação do sistema final devem ser compatíveis. Assim, quaisquer modificações no hardware devem ser cuidadosamente examinadas e avaliadas através de um projeto detalhado. Para sistemas de hardware, cada subsistema é numerado ou identificado, possuindo também um número de versão. Cada número de versão identifica diferentes projetos do mesmo componente de hardware [BELLAGIO05].

Os problemas encontrados num ambiente de desenvolvimento de software são análogos e mais observáveis ainda que os de um ambiente de produção de hardware. Cada parte do software possui seu próprio nome (Documento de Visão, Requisitos, Plano de Testes, por exemplo). Estas partes podem ser agrupadas em subsistemas, módulos ou componentes. Devem ser identificados e atribuídos números de versões. Estas partes de software também devem possuir interfaces compatíveis, sendo que diferentes versões podem ter diferentes interfaces. Finalmente, deve-se ter uma relação contendo os componentes compatíveis para a construção do sistema de software.

Entretanto, o gerenciamento de configuração de software é difícil de ser desempenhado na plenitude, pois o software tende a ser mais volátil no que se refere à estabilidade do que o hardware. Pequenas alterações no código são motivos suficientes para a criação de uma nova versão do software. Diferente do hardware, a construção do software pode sofrer centenas de modificações diárias pela equipe de desenvolvimento.

Neste ambiente dinâmico, a disciplina de Gerência de Configuração de Software e Controle de Mudanças surge para garantir que, quando uma versão final de um produto for liberada para utilização, todos os componentes do sistema possam ser integrados ao mesmo tempo, numa mesma localização física, de forma que o produto final faça aquilo que se propõe corretamente. Ainda que as equipes de desenvolvimento saibam que a Gerência de Configuração é necessária, a implantação desta disciplina é falha. Não pela sua complexidade, mas devido ao fato de que não existe um entendimento claro sobre as suas vantagens.

1.1.2 O que é Gerenciamento de Configuração e de Solicitações de Mudança

O Modelo Integrado de Maturidade de Capacidade (CMMI) do Software Engineering Institute (SEI) define que o Gerenciamento de Configuração e de Solicitações de Mudança controla mudanças feitas nos artefatos de um projeto e mantém a sua integridade. Contudo, a definição mais aceita e utilizada caracteriza a Gerência de Configuração de Software como "uma disciplina que visa identificar e documentar as características de itens de configuração, controlar as suas alterações, armazenar e relatar as modificações aos interessados e garantir que foi feito o que deveria ter sido feito" [IEEE90].

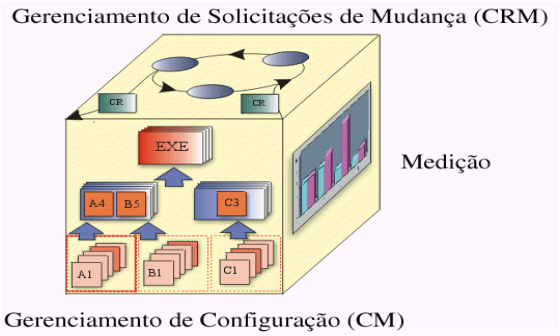


Figura 1 - Gerência de Configuração de Software e Controle de Mudanças [RUP01]

Analisando o cubo da figura 1, verifica-se nas 03 faces que estão em primeiro plano, os seguintes elementos integrantes de um sistema de Gerência de Configuração e Controle de Mudanças:

a) Gerenciamento de Solicitações e Mudanças

Durante todo o ciclo de vida do desenvolvimento do software, mudanças necessariamente ocorrerão. Estas mudanças refletirão em diversos artefatos, desde os requisitos, análise e design até o código propriamente dito. Para que estes artefatos estejam consistentes, controles formais devem ser colocados em prática, de forma que se tenha total controle sobre a efetivação das mudanças, bem como sua situação (estado).

b) Gerenciamento de Configuração

São atribuídos números de versões aos diversos artefatos que compõem o software. Estes artefatos são selecionados e agrupados para formar uma liberação (release) do produto, que corresponde ao executável da aplicação (build). Esta liberação, por sua vez, também possuirá um número de versão atribuída pelo sistema de Gerência de Configuração. Deve ser possibilitada a rastreabilidade das diversas versões dos artefatos, bem como de toda a sua documentação.

c) Medição.

Como todo processo, um modelo de melhoria deve ser previsto. Para que a melhoria seja efetivamente obtida, medições devem ser executadas. Para execução dessas medições, modelos de métricas são utilizadas de forma a se obter indicadores que servirão de parâmetros para futuras comparações. Oportunamente, serão apresentadas algumas sugestões de métricas a serem aplicadas no sistema de Gerência de Configuração.

Por exemplo, a informação sobre o tempo de processamento de uma solicitação de mudança (da abertura da solicitação de mudança até seu fechamento) requerido para vários tipos de solicitações deverá estar disponível na avaliação do critério para determinar quais níveis de autoridade são ideais para certo tipo de solicitação. Ou seja, para solicitações de mudanças simples, um nível menor de aprovação é requerido. Para as solicitações de mudanças mais complexas, um nível de autoridade maior poderá ser requerido para sua aprovação [SCOTT03].

Os métodos, os processos e as ferramentas utilizados para o gerenciamento de configuração e mudança de uma organização podem ser considerados como o Sistema de Gerência de Configuração da organização. O Sistema de Gerenciamento de Configuração e Solicitações de Mudança de uma organização contém informações-chave sobre os processos de desenvolvimento, promoção, implantação e manutenção de produtos da organização e armazena a base de ativos de artefatos potencialmente reutilizáveis, resultantes da execução desses processos.

Os principais aspectos de um Sistema de Gerência de Configuração são:

a) Gerenciamento de Solicitações de Mudança

Abrange a infra-estrutura organizacional necessária para avaliar o custo e o impacto de uma mudança solicitada sobre o produto existente. O Gerenciamento de Solicitações de Mudança envolve o trabalho da Equipe de Revisão de Mudança ou do Comitê de Controle de Mudança (CCB) [HUMPHREY90].

b) Estimativa do Estado de Configuração (Métrica)

É utilizada para descrever o “estado” do produto de acordo com tipo, número, nível e gravidade dos defeitos encontrados e corrigidos durante o desenvolvimento do produto. As métricas desses aspectos são obtidas por meio de auditorias ou de dados brutos e são úteis para determinar o estado geral de conclusão do projeto.

c) Gerenciamento de Configuração

Descreve a estrutura do produto e identifica os itens de sua configuração que são tratados como entidades de uma única versão no processo de gerenciamento de configuração. A Gerência de Configuração trata das definições de configurações, da criação, identificação e obtenção de artefatos de versão utilizados nos conjuntos de componentes, como também, da rastreabilidade entre as versões.

d) Rastreamento de Mudança

Descreve o que é feito nos artefatos do projeto, o motivo e a hora da mudança. Funciona como um histórico e um relatório das mudanças.

e) Seleção de Versão

A finalidade da “seleção de versão” é garantir que as versões corretas dos itens de configuração sejam selecionadas para mudanças ou implementações. A seleção de versão baseia-se na “identificação da configuração”.

f) Manufatura de Software

Aborda a necessidade de automatização dos passos para compilar, testar e compactar o software para distribuição.

1.1.3 Finalidade dos Sistemas de Gerência de Configuração de Software

Um Sistema de GCS é fundamental para controlar os inúmeros artefatos produzidos pelas muitas pessoas que trabalham em um mesmo projeto. O controle ajuda a evitar confusões dispendiosas e garante que os artefatos resultantes não entrem em conflito devido a algum dos seguintes problemas [HUMPHREY90]:

a) Atualização Simultânea

Quando dois ou mais membros da equipe trabalham separadamente no mesmo artefato, o último membro a fazer mudanças desfaz o trabalho realizado pelo anterior. O problema básico é que, se um sistema não permite a atualização simultânea, ocorrerá naturalmente a diminuição do ritmo do processo de desenvolvimento. Entretanto, com a atualização simultânea, o desafio é detectar se ocorreram atualizações simultaneamente e resolver quaisquer problemas de integração quando essas mudanças forem incorporadas.

b) Notificação Limitada

Quando um problema é corrigido nos artefatos compartilhados por vários desenvolvedores e alguns deles não são notificados da mudança.

c) Várias Versões

A maioria dos programas de grande porte é desenvolvida em liberações evolutivas. Uma liberação pode estar sendo usado pelo cliente, enquanto outra está em teste e uma terceira ainda está em desenvolvimento. Se forem encontrados problemas em qualquer uma

das versões, as correções deverão ser propagadas entre elas. Isso pode levar a confusão, que acarretam correções dispendiosas e retrabalho, a menos que as mudanças sejam cuidadosamente controladas e monitoradas.

Um Sistema de GCS é útil para gerenciar diversas variantes de sistemas de software em desenvolvimento, controlando as versões que são usadas em determinados construções do software, compilando construções de programas individuais ou de liberações inteiras, de acordo com especificações de versão definidas pelo usuário e impondo políticas de desenvolvimento específicas da organização.

Alguns dos benefícios diretos obtidos pelo uso de um Sistema de Gerência de Configuração incluem suporte a métodos de desenvolvimento, preservação da integridade do produto, garantia de abrangência e precisão do produto configurado. Também propicia um ambiente estável no qual o produto deve ser desenvolvido, restrição das mudanças feitas nos artefatos com base nas políticas do projeto e rastreamento de auditoria.

Além disso, um Sistema de GCS armazena dados detalhados sobre a "contabilidade" do processo de desenvolvimento: quem criou uma versão específica (e também quando e por que), quais versões dos códigos-fonte foram usadas em uma determinada construção, além de outras informações relevantes para o processo.

1.1.4 Relação com Outras Disciplinas

O Sistema de Gerência de Configuração de uma organização serve como suporte ao desenvolvimento de software, abrangendo desde a concepção do produto, a análise e projeto, implementação, instalação e manutenção. Contempla ainda o processo de aquisição, estendendo-se até a descontinuação do produto de software [NBR10007]. O sistema de Gerência de Configuração contém as versões atuais e históricas dos arquivos-fonte dos artefatos de requisitos, projeto e implementação que definem uma determinada versão de um sistema ou de um componente do sistema.

Os conhecimentos necessários de Engenharia de Software para a implementação de um sistema de gerência de configuração e controle de mudanças (versões, liberações, dentre outros) são exclusivos desta disciplina. Entretanto, sua abrangência vai além, envolvendo outras disciplinas da Engenharia de Software. Como exemplo de relacionamento com outras

disciplinas, temos a Garantia da Qualidade (QA – Quality Assurance), que tem como premissa básica a monitoração das atividades de gerência de configuração, garantindo assim a qualidade do processo como um todo. A documentação dos itens de configuração (IC) deve estar consistente com os itens respectivos. A Garantia da Qualidade atesta que os planos previstos estão sendo seguidos. Além da Garantia da Qualidade, a disciplina de Gerência de Projetos também possui um relacionamento profundo com a Gerência de Configuração [PMBOK00]. O Project Management Institute – PMI prevê o controle dos itens de configuração do projeto, pois esses itens devem ser controlados, sendo que a Gerência de Configuração é que estabelece os parâmetros para este controle.

1.1.5 Gerência de Configuração sob o ponto de vista gerencial

A disciplina de Gerência de Configuração pode ser tratada sob diferentes perspectivas em função do papel exercido pelo participante do processo de desenvolvimento de software [ASKLUND02]. Na perspectiva gerencial, a GCS é dividida em quatro funções, que são [NBR10007]:

a) Identificação da Configuração

A função de identificação da configuração tem por objetivo possibilitar:

- A seleção dos Itens de Configuração (IC) que são os elementos passíveis de GCS;
- A definição do esquema de nomes e números, que possibilite a identificação inequívoca dos IC;
- A descrição dos IC, tanto física quanto funcionalmente.

b) Controle da Configuração

A função de Controle da Configuração é designada para o acompanhamento da evolução dos IC selecionados e descritos pela função de identificação. Para que os IC possam evoluir de forma controlada. Esta função estabelece as seguintes atividades [PRESSMAN03]:

- **Requisição de modificação**: iniciando um ciclo da função de controle dado um pedido de manutenção, que pode ser corretiva, evolutiva, adaptativa ou preventiva;

- **Classificação da modificação**: que estabelece a prioridade do pedido em relação aos demais pedidos efetuados anteriormente;
- **Análise da modificação**: que visa relatar os impactos em esforço, cronograma e custo, e definir uma proposta de implementação da manutenção;
- **Avaliação da modificação pelo Comitê de Controle da Configuração**: que estabelece se o pedido será implementado, rejeitado ou postergado;
- **Implementação da modificação**: caso o pedido tenha sido aprovado;
- **Verificação da modificação**: através da aplicação de testes e validação do sistema;
- **Geração da linha base.**

c) Acompanhamento da Configuração (Contabilização)

A função de acompanhamento da configuração visa armazenar as informações geradas pelas demais funções permitindo que essas informações possam ser acessadas em função de necessidades específicas. Essas necessidades específicas abrangem o uso de métricas para a melhoria do processo, a estimativa de custos futuros e a geração de relatórios gerenciais.

d) Auditoria da Configuração (AC)

A função de Auditoria da Configuração ocorre quando a configuração de referência, gerada na função de controle da configuração, é selecionada para ser liberada para o cliente. Suas atividades compreendem a Auditoria de Configuração Física e Auditoria de Configuração Funcional [NBR10007].

1.1.6 Gerência de Configuração sob o ponto de vista do desenvolvimento

Sob a perspectiva de desenvolvimento, a disciplina de GCS é dividida em três sistemas principais, que são [ASKLUND02]:

a) Controle de Modificações

O sistema de controle de modificações é encarregado de executar a função de controle da configuração de forma sistemática, armazenando todas as informações geradas durante o andamento das requisições de modificação, e relatando essas informações aos participantes interessados e autorizados, assim como estabelecido pela função de acompanhamento da configuração.

b) Controle de versões

O sistema de controle de versões permite que os itens de configuração sejam identificados, de acordo com a função de identificação da configuração, para que evoluam de forma distribuída e concorrente, porém disciplinada. Essa característica é necessária para que diversas requisições de modificação possam ser tratadas em paralelo, sem, no entanto, causar transtornos característicos de um ambiente sem o referido controle. O controle de versões é o ponto central na disciplina de GCS, e sua funcionalidade reside na aplicação de ferramentas que automatizam o processo. Entretanto, o controle de versões não é tudo no processo. O controle de alterações também compartilha de igual importância para a maturidade e completude de qualquer processo de gerência de configuração. No que diz respeito ao controle de versão em si, surge uma propriedade que caracteriza sobremaneira o conceito de versão: a imutabilidade.

Cada IC estável encontra-se rotulado com um número de versão. A versão, por sua vez, por ser imutável, e por fazer parte de uma linha base, não poderá mais ter seu conteúdo modificado. Se a modificação for necessária, uma nova versão deverá ser criada.

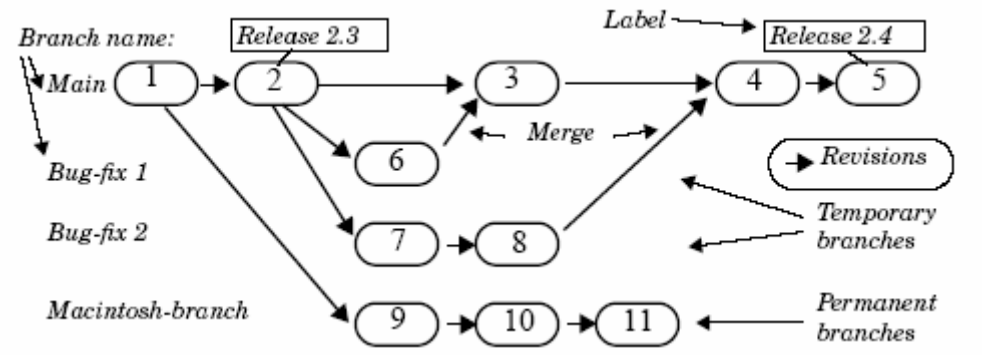


Figura 2 - Controle de Versões [ASKLUND02]

Na figura 2, por exemplo, as versões 1,2,3,4,5 são revisões do ramo (Branch) principal ‘Main’. A versão 2 foi nomeada como sendo a "Release 2.3", assim como a versão 5 foi rotulada como ‘Release 2.4’. ‘Bug-fix 1 e 2’ são ramos temporários (contendo uma e duas revisões, respectivamente), os quais foram unidos ao ramo principal ‘Main’. O exemplo também ilustra um ramo permanente que possui três revisões (9,10,11). Este ramo indica uma variante que será adotada em outra plataforma, sendo que não poderá ser unido ao ramo principal.

c) Controle de Construções e Liberações

O sistema de controle de construções e liberações automatiza o complexo processo de transformação dos diversos artefatos de software que compõem um projeto no sistema executável propriamente dito, de forma aderente aos processos, normas, procedimentos, políticas e padrões definidos para o projeto.

As quatro funções descritas na perspectiva gerencial são implementadas através dos três sistemas descritos na perspectiva de desenvolvimento, acrescidos de processos, normas, procedimentos, políticas e padrões.

1.1.7 Plano de Gerenciamento de Configuração (PGC)

O Plano de Gerenciamento de Configuração é um artefato essencial para a disciplina de Gerência de Configuração e Controle de Mudanças. Ele faz referência às etapas e atividades que descrevem como o Gerenciamento de Controle de Configuração e Mudança é executado no desenvolvimento do software [RUP01]. O conteúdo mínimo requerido para um PGC é definido pela norma IEEE Std 828-1998.

Este plano descreve todas as atividades de gerenciamento que serão executadas durante todo o ciclo de vida do projeto, como elas serão realizadas, quais os papéis envolvidos nestas atividades, quando elas acontecerão, bem como especifica os recursos que serão requeridos para consecução das atividades [IEEE98].

1.1.8 Atividades Críticas de Gerência de Configuração

O CMBOK [CMBOK] define as atividades consideradas críticas na disciplina de Gerência de Configuração. São elas:

- **Planejamento das Atividades:** geração de um PGC.

- **Aquisição e Suporte:** todos os requisitos para o suporte aos dados, processamento, armazenamento, integridade, segurança e manutenção deverão estar descritos na documentação de configuração. Esta documentação deve ser mantida por todo o ciclo de vida do projeto. Deverão ser previstos serviços automatizados para processamento e submissão de formulários, bem como um serviço de informações integradas para a extração de dados dos sistemas automatizados de Gerência de Configuração e Controle de Mudanças.

- **Identificação da Configuração:** estabelecer e manter, de maneira incremental, uma base para controle e contabilidade dos IC através do ciclo de vida.

- **Controle da Configuração:** é a sistemática adotada para a avaliação da necessidade da mudança, sua aprovação ou não, de um item de configuração, bem como o estabelecimento de uma linha base do item sob gerência de configuração.

- **Contabilidade do Estado da Configuração**: garante a identificação precisa de cada IC. Cada documentação aprovada deverá ter uma numeração associada com o respectivo IC. Os registros dos estados das requisições de mudanças devem ser mantidos por todo o ciclo de vida do projeto, bem como os registros de auditorias.

- **Auditoria da Configuração (AC):** a auditoria é executada antes do estabelecimento de uma linha base para um determinado IC. Ela é complementar à Revisão Técnica Formal (RTF). A AC divide-se em duas: Auditoria de Configuração Funcional e Auditoria de Configuração Física, descritas no Glossário.

1.1.9 Métricas para Gerência de Configuração de Software

A utilização de métricas nas organizações que produzem software constitui um fator essencial para a obtenção de vantagem estratégica, pois é fator crítico para a melhoria do processo adotado pela organização.

Algumas razões para medição de software são [HAZAN00]:

- Formar uma base para estimativas;
- Determinar se as metas de produtividade e qualidade do processo estão sendo atingidas;
- Determinar se as metas de qualidade do produto estão sendo atingidas;
- Avaliar os benefícios de novos métodos, treinamentos e ferramentas de software;
- Melhorar a gerência de contratos de software e relacionamentos com terceiros;
- Melhorar a gerência de projetos de desenvolvimento de software;
- Entender e aperfeiçoar o processo de software.

Neste contexto, a Gerência de Configuração de Software (GCS), como processo de apoio aos demais processos, também lança mão da utilização das métricas. Para que qualquer processo de desenvolvimento de software tenha sucesso, a melhoria contínua deve ser almejada, necessitando do estabelecimento de um processo automatizado de medições para coleta de dados, análise e execução de procedimentos de forma sistematizada.

Um dos maiores objetivos dos sistemas de GCS automatizados é a possibilidade da extração automatizada de dados, de forma a possibilitar a utilização de métricas. As métricas são utilizadas no gerenciamento para verificação de possíveis problemas no processo, de forma a corrigi-los, possibilitando assim a melhoria contínua do processo [CMBOK].

O processo de GCS possui algumas métricas próprias a serem avaliadas, divididas dentro de grupos funcionais, como indica a figura 3:

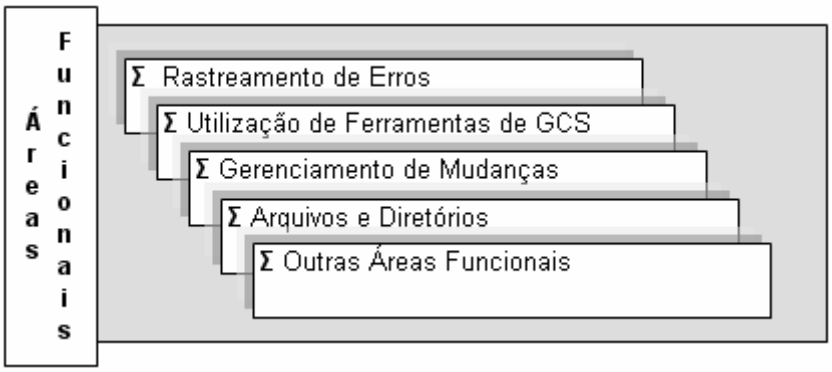


Figura 3 - Métricas por Área Funcional [FARAH04]

Rastreamento de erros

Normalmente, esta métrica é adotada separadamente para cada linha de desenvolvimento, pois os problemas a serem atacados por esta métrica podem ter causas semelhantes, facilitando o processo de comparação entre linhas de desenvolvimento.

a) Problemas solucionados e funcionalidades implementadas por liberação

Esta métrica indica o domínio de cada liberação, sendo um indicador para a quantidade de treinamento, documentação e outros recursos requeridos.

b) Duração dos problemas por prioridade

Esta é uma métrica de monitoramento do processo. Indica se os problemas estão sendo resolvidos dentro dos períodos indicados conforme a prioridade atribuída.

c) Identificação de problemas por fase do ciclo de desenvolvimento

Esta métrica permite o conhecimento de quantos erros surgem ao longo da fase propriamente dita. Permite uma avaliação da efetividade de processos tais como verificação e testes, por exemplo.

d) Freqüência de problemas duplicados

É uma métrica difícil de ser implementada, pois apenas se percebe o problema da duplicação dos erros quando eles surgem pela segunda vez, além do que, se a freqüência for alta, torna-se complexa a tarefa de identificação das duplicatas.

Utilização de Ferramentas de GCS

a) Requisitos de largura de banda

Esta métrica é calculada a partir do número de transações registradas na rede oriundas do sistema de GCS. Dois tipos de métricas são indicadas: Média (MBytes/dia) e o Pico Máximo (MBytes/hora). Através da análise deste comportamento, fica fácil prever o que acontecerá quando alguém necessitar fazer uma grande operação de carga de dados no ambiente de GCS. Esta métrica é utilizada para análise de impacto no sistema de rede da organização.

b) Taxa de crescimento do repositório

A taxa de crescimento do repositório tem um reflexo direto sobre a performance do sistema de GCS e o espaço em disco. Permite o acompanhamento do sistema frente às características e limites das ferramentas utilizadas, bem como propicia um planejamento adequado do crescimento do repositório.

c) Tempo de preparação para uma construção (*build*) simples

Esta métrica é utilizada para ajudar a identificar a eficiência das ferramentas utilizadas no ambiente de GCS. Quanto de esforço é necessário para definir o conteúdo da construção, as notas de construção, recuperação do código fonte, iniciação do processo de construção do sistema. Se o esforço for significante, novas métricas podem ser criadas para cada elemento do processo de construção, de forma a tornar precisa a métrica, e em conseqüência, o indicador.

Gerenciamento de Mudanças

a) Mudanças por projetista (Designer)

Se a um determinado projetista está sendo atribuído uma grande quantidade de requisições de mudanças (CR), esta métrica poderá indicar que um possível retrabalho está sendo executado por este projetista. Gera um indicador da eficiência da equipe de projeto.

b) Arquivos por mudanças (correção de erros, funcionalidades)

Esta métrica indica quantos arquivos são modificados em média para cada correção de erros ou funcionalidades.

c) Linhas de código por mudança

Esta métrica indica quantas linhas de código são adicionadas, modificadas ou excluídas por solicitação de mudança. Deve ser verificada com freqüência, pois se estiver com grande tendência de aumento, a política de modularidade não está sendo realizada a contento, ou está mal planejada.

d) Arquivos modificados com maior freqüência

Esta métrica indica quais arquivos são candidatos, em termos arquiteturais, à reengenharia.

e) Interfaces modificadas com maior freqüência

Quando uma interface é modificada com alta freqüência, esta métrica indica que há um problema no projeto inicial, ou até mesmo, no levantamento de requisitos.

Arquivos e Diretórios

a) Número de Arquivos por Produto ou Subsistema

Esta métrica indica o tamanho e a complexidade do produto ou subsistema. Também permite a verificação das alterações ao longo do tempo. Esta métrica pode ser relacionada a outras métricas (linhas de código, por exemplo) para obter algumas tendências significativas.

b) Linhas de Código (por arquivo, por subsistema, por tipo de arquivo)

Tradicionalmente, esta métrica é utilizada para comparação do tamanho dos projetos. Além deste indicador, pode-se comparar a taxa de erros contra o tamanho de cada artefato, podendo-se ainda estender esta métrica para avaliação da efetividade e consistência do projeto do sofware.

c) Ramos por Arquivo

Esta métrica, quando adotada em ambientes que utilizam apenas um ramo por linha de desenvolvimento, indica a porcentagem de arquivos que são modificados por cada liberação do produto.

Outras Áreas Funcionais

Outras métricas também podem ser utilizadas, desde a cobertura dos casos de teste (por problemas, por funcionalidades ou requisitos), além de taxas de falhas na execução dos testes. Na área de gerenciamento, métricas associadas à quantidade de documentos produzidos/modificados por período de tempo, tempo médio de revisão, tempo médio de aprovação, dentre outras possíveis.

Há um consenso de que o verdadeiro poder das métricas vem da habilidade de comparar os indicadores com bases históricas, isto é, execução da análise de tendência. Além disso, quanto maior for o período de apuração de dados, a avaliação se torna mais valorosa e precisa [FRANK05].

1.2 Normas, Padrões e Modelos de Melhorias de Processos

Nesta seção serão descritos algumas normas, padrões e modelos de melhoria de processos existentes no mercado e que têm contribuído na construção e melhoria dos processos de software existentes nas organizações. O objetivo comum destas normas, padrões e modelos é prover um conjunto de boas práticas necessárias para melhorar a qualidade tanto dos produtos quanto dos processos de software.

Serão apresentados o propósito, breve histórico, a estrutura e o processo de Gerência de configuração e mudança presentes em cada uma destas normas, padrões e modelos. O objetivo é mostrar a importância dada, por cada um desses instrumentos de qualidade, ao processo de Gerência de Configuração de Software.

E por fim será apresentado um quadro comparativo das atividades de Gerência de Configuração propostas pelos padrões, normas e modelos descritos nesta seção, identificando as suas atividades de GCS.

1.2.1 USDP/RUP

O Rational Unified Process é um exemplo específico e detalhado de um processo mais genérico descrito por Ivar Jacobson, Grady Booch e James Rumbaugh no livro “The Unified Software Development Process – UDSP”, mantido atualmente pela IBM-Rational.

O RUP consiste de uma instância do USDP, e descreve “melhores práticas” que são comumente realizadas nas organizações produtoras de software. Por possuir caráter comercial, vem apoiado com inúmeras ferramentas de propriedade da IBM-Rational. Estas “melhores práticas” são descritas através de diretrizes, modelos de documentos e características de ferramentas necessárias para a execução dos diversos processos presentes no RUP. Dentre as principais práticas do RUP, pode-se citar [RUP01]:

- Desenvolvimento iterativo e incremental;
- Gerência de Requisitos;
- Arquiteturas baseadas em objetos;

- Representação dos diversos componentes do software através de modelos, utilizando a notação UML (Unified Model Language, ou seja, Linguagem de Modelagem Unificada);

- Verificação da Qualidade do Software;

- Controle de mudanças de software;

O RUP oferece uma estrutura de processo adaptável para a Engenharia de Software. Uma das suas principais características é o uso da iteração (baseado no ciclo de vida em espiral) que, através de refinamentos sucessivos, melhora o entendimento do problema. A cada ciclo, o produto torna-se mais completo. O produto IBM RUP contém várias configurações e visões de processos que podem conduzir os diversos envolvidos a desenvolver um produto de software [GALLAGHER01].

O RUP possui cinco elementos principais [LUIZ04]:

- **Papel (ou perfil):** define o comportamento e as responsabilidades de um determinado indivíduo ou grupo de indivíduos trabalhando como uma equipe, como por exemplo: Gerente de Configuração, Gerente, Integrador.

- **Atividade:** é uma unidade de trabalho que um indivíduo executa quando está exercendo um determinado papel e produz um resultado importante para o contexto do projeto.

- **Artefato:** é um pedaço de informação que é produzido, modificado ou utilizado em um processo. São os documentos e códigos produzidos durante o desenvolvimento do projeto. Artefatos são utilizados como entradas de atividades e são produzidos como saída.

- **Fluxo de Trabalho:** é uma seqüência de atividades que são executadas para a produção de um resultado valioso para o projeto.

- **Disciplina:** é uma coleção de atividades relacionadas que fazem parte de um contexto comum em um projeto. As disciplinas de processo são: Modelagem de Negócios, Requisitos, Análise e Projeto, Implementação, Teste e Distribuição. As

de suporte são: Configuração e Gerenciamento de Mudanças, Gerenciamento de Projeto, e Ambiente.

As fases indicam a ênfase que é dada no projeto em um dado instante. Para capturar a dimensão do tempo de um projeto, o RUP divide o projeto em quatro fases diferentes [RUP01]: **Iniciação**, com ênfase no escopo do sistema, **Elaboração**, com ênfase na arquitetura, **Construção**, com ênfase no desenvolvimento e **Transição**, com ênfase na implantação. O USDP e consequentemente o RUP, são processos genéricos que se adaptam perfeitamente ao paradigma de Análise, Projeto e Implementação orientada à objetos. Além disso, tem maior efetividade se as atividades de engenharia estiverem apoiadas pela UML (Unified Modeling Language), que também foi desenvolvida pela IBM-Rational e tem sido adotada como padrão comercial [OMG01].

Disciplina Gerência de Configuração de Software

O fluxo de trabalho da disciplina de Gerência de Configuração e Mudança, proposto pelo RUP, é constituído dos seguintes grupos de atividades descritos no Quadro 1 [RUP05]:

Grupo de Atividade	Finalidades
Planejar Configuração do Projeto e Controle de Mudanças (Ativado no início do projeto)	• Estabelecer políticas para o gerenciamento de configuração do projeto; • Estabelecer políticas e processos para controlar as mudanças feitas no produto; • Documentar essas informações no Plano de Gerenciamento de Configuração (incluso no Plano de Desenvolvimento de Software);
Criar Ambiente para Controle de Mudanças do Projeto; (Ativado no início do projeto)	• Estabelecer um ambiente mediante a criação e manutenção de repositórios de dados, nos quais o produto poderá ser desenvolvido, compilado e disponibilizado para manutenção e reutilização.

Quadro 1 - Grupo de Atividades do RUP para GCS

Grupo de Atividade	Finalidades
Alterar e Liberar Itens de Configuração; (Execução contínua durante o ciclo de vida)	• Possibilitar que a partir do espaço de trabalho de integração, deve poder compilar o produto, criar linhas base (baselines) e torná-las disponíveis para os demais membros da equipe de desenvolvimento.
Gerenciar Linhas Base (Baselines) e Liberações (Releases); (Execução contínua durante o ciclo de vida)	• Assegurar que, ao atingir um nível específico de maturidade, os sistemas tenham sua linha base definida e que depois sejam disponibilizadas para liberações ou reutilização em futuras iterações do projeto e/ou em outros projetos.
Monitorar e Relatar Estado de Configuração; (Execução contínua durante o ciclo de vida)	• Determinar se o produto atende aos requisitos funcionais e físicos; • Determinar se os artefatos necessários estão armazenados em uma biblioteca controlada e se a respectiva linha base (baseline) foi definida; • Assegurar que os artefatos e as linhas base (baselines) estão disponíveis; • Auxiliar as atividades de Estimativa do Estado de Configuração do projeto que são baseadas em um registro formalizado e relatar o estado das mudanças propostas assim como o estado de implementação das mudanças propostas; • Facilitar a revisão do produto por meio de atividades de relatórios e controle de defeitos; • Assegurar que os dados sejam “agregados” e reportados para fins de controle de andamento e tendências;
Gerenciar Solicitações de Mudança. (Execução contínua durante o ciclo de vida)	• É assegurar que as mudanças feitas em um projeto sejam consistentes e que os envolvidos adequados sejam informados do estado do produto, das mudanças feitas nele e do impacto de custo e programação gerado por essas mudanças.

Quadro 1 – Continuação: Grupo de Atividades do RUP para GCS

As figuras 4 e 5 demonstram a freqüência da execução das atividades de Gerência de Configuração durante as fases de um projeto [RUP01].

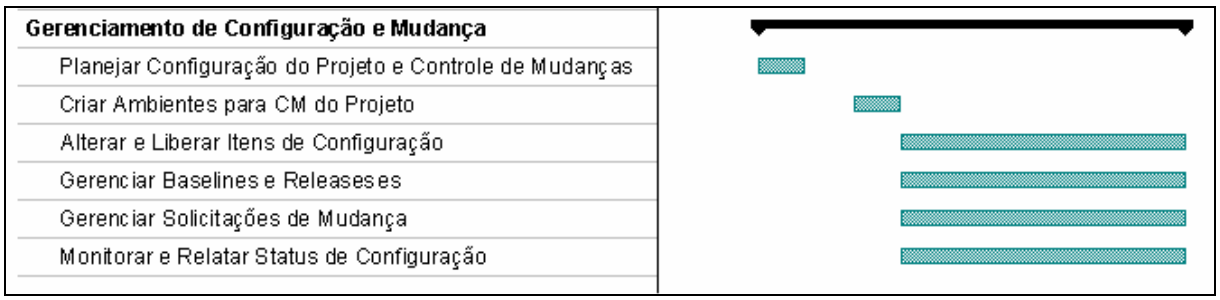


Figura 4 - Execução das atividades de GCS na fase de iniciação [RUP01]

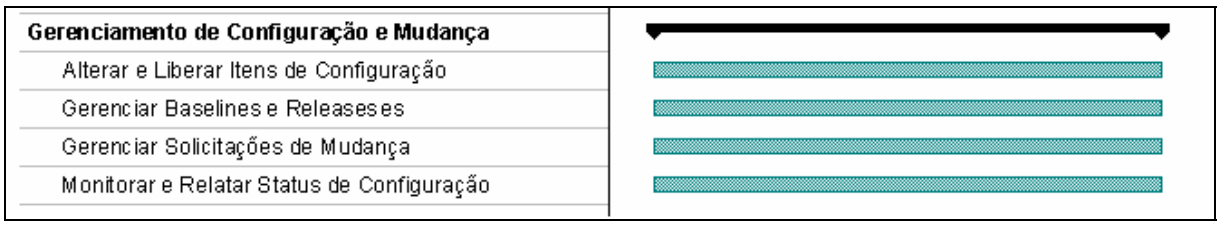


Figura 5 - Execução das atividades de GCS nas fases, exceto iniciação [RUP01]

1.2.2 CMMI-SW

O CMMI-SW (Capability Maturity Model Integration for Software Engineering) é um guia criado pelo SEI (Software Engineering Institute) e destinado a melhorar, de forma gradativa e disciplinada, a qualidade do processo de software de uma organização e a habilidade deste processo em gerenciar o desenvolvimento, aquisição e a manutenção dos software [CMMI02].

O SEI foi criado com a finalidade de ajudar o Departamento de Defesa dos Estados Unidos (US DOD), um dos maiores compradores de software do mundo, a selecionar empresas cada vez mais qualificadas e capazes de fornecer software e serviços de qualidade. A partir do CMM (Capability Maturity Model), os modelos de maturidade e métodos de avaliação criados pela entidade tornaram-se referência mundial para a realização de melhores práticas de Engenharia de Software e de melhoria de processos de software.

O CMMI integra disciplinas como Engenharia de Software (CMMI-SW), Engenharia de Sistemas (CMMI-SE) e Capacidade de Pessoas (P-CMMI). Esta integração partiu do

mercado, patrocinadores e mantenedores do SEI que exigiam modelos de maturidade para as disciplinas diferentes das de software e que também contribuem na maturidade da organização.

Existem dois tipos de representação no CMMI: em estágios e contínua. Cada representação permite que determinada organização siga por caminhos diferentes de melhoria. A representação em estágios é a representação usada no CMM-SW. Esta representação define um conjunto de Áreas de Processo que definem um caminho de melhoria para a unidade organizacional, descrito em termos de níveis de maturidade. A representação contínua é o enfoque utilizado no SECM (System Engineering Capability Model), no IPD-CMM (Integrated Product Development Capability Maturity Model) e também na ISO/IEC 15504. Este enfoque permite que uma organização selecione uma área de processo específica e melhore com relação a esta área. A representação contínua usa níveis de capacidade para caracterizar melhoria relacionada a uma área de processo, conforme indica a figura 6 [MPS-BR05].

As Áreas de Processos são constituídas de metas genéricas e especificas e que são alcançadas com a execução de práticas também genéricas e específicas. Como forma de garantir a correta execução das tarefas e o alcance dessas metas, o CMMI apresenta ainda um conjunto de características comuns.

Quanto às diferenças entre os modelos CMM e CMMI, pode-se citar o conjunto de melhores práticas definidas pelo modelo de melhoria, que no caso do CMMI estão mais apuradas e modernas. Além disso, novas Áreas de Processo e novos objetivos genéricos foram adicionados. Por exemplo, a Área de Processo de Medição e Análise, agora se encontra formalizada no nível 2 de maturidade. No CMM, estas práticas estavam dispersas nas demais Áreas de Processo.

A Área Chave de Processo denominada Engenharia do Produto de Software foi substituída no CMMI por outras Áreas de Processo mais detalhadas, conforme indicadas abaixo [SEI05]:

- Desenvolvimento de Requisitos;
- Soluções Técnicas;
- Integração do Produto;
- Verificação (incorporou as Revisões por Pares);

- Validação;
- Gerência de Riscos (Nova Área de Processo);
- Análise de Decisões e Resoluções (Nova Área de Processo).

Representação Contínua e Estagiada

A representação contínua usa níveis de capacidade para medir a melhoria do processo, enquanto a representação estagiada utiliza níveis de maturidade. A principal diferença entre níveis de maturidade e de capacidade é a apresentação dos mesmos ao longo do tempo, e também a maneira que serão aplicados. Níveis de capacidade (representação contínua) aplicam-se para cada Área de Processo. Existem seis níveis de capacitação, numerados de zero a cinco. Cada nível corresponde a uma meta genérica e a um conjunto de práticas genéricas e específicas.

Comparativo: Representação Contínua e Estagiada			
Nível de Capacitação	**Representação Contínua**	**Nível de Maturidade**	**Representação Estagiada**
0	Incompleto	---	-----
1	Executado	1	Inicial
2	Gerenciado	2	Gerenciado
3	Definido	3	Definido
4	Gerenciado Quantitativamente	4	Gerenciado Quantitativamente
5	Otimizado	5	Otimizado

Quadro 2 - Perspectivas do CMMI [CMMI02]

Níveis de maturidade (representação estagiada) são aplicados para definir a maturidade da organização como um todo. Existem cinco níveis de maturidade, numerados de 1 a 5. Cada nível compreende um conjunto de Áreas de Processos.

A representação continua tem práticas mais específicas que a estagiada, pois se divide em básicas e avançadas. Já a representação estagiada possui somente um tipo de prática específica. Na representação contínua, práticas genéricas existem para os níveis de capacidade de 1 a 5, enquanto que na representação estagiada, aparecem somente nos níveis 2 e 3. Não há práticas genéricas para os níveis 1, 4 e 5 [CMMI02].

Figura 6 - Representação Contínua do CMMI [CMMI05]

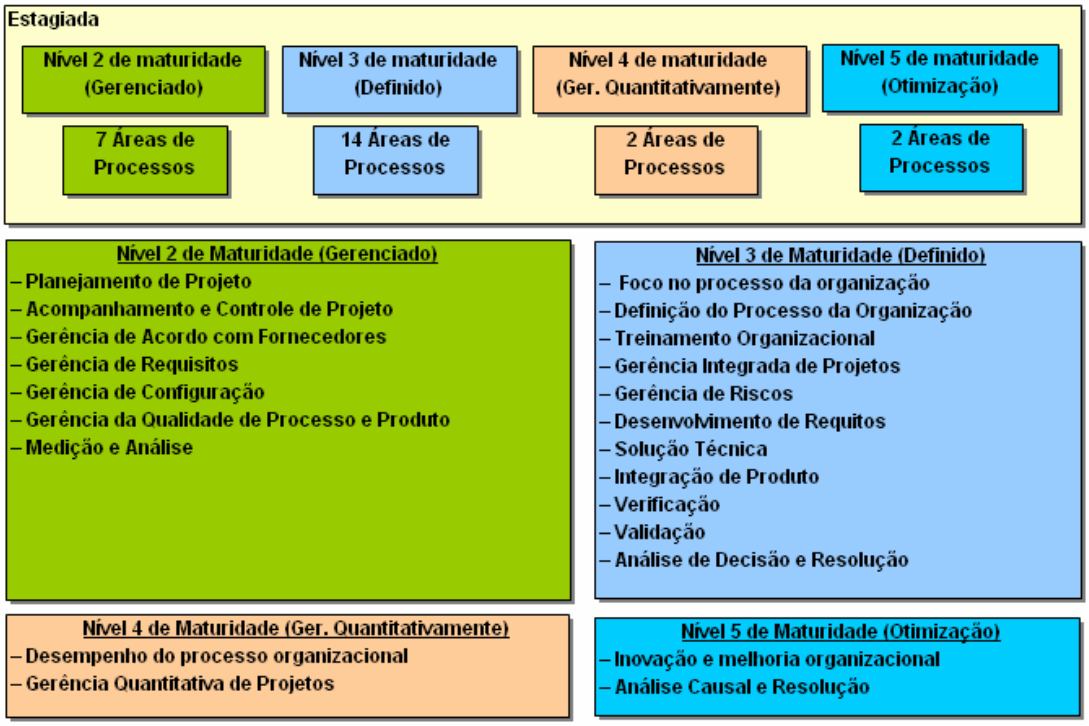


Figura 7 - Representação Estagiada do CMMI [CMMI05]

A Gerência de Configuração no CMMI

A Gerência de Configuração no CMMI aparece como uma prática genérica, destinada a estabelecer e manter a integridade dos produtos do trabalho, através de todo o ciclo de vida. Além disso, constitui uma Área de Processo específica. A figura 8 descreve as Metas da Área de Processo de Gerência de Configuração.

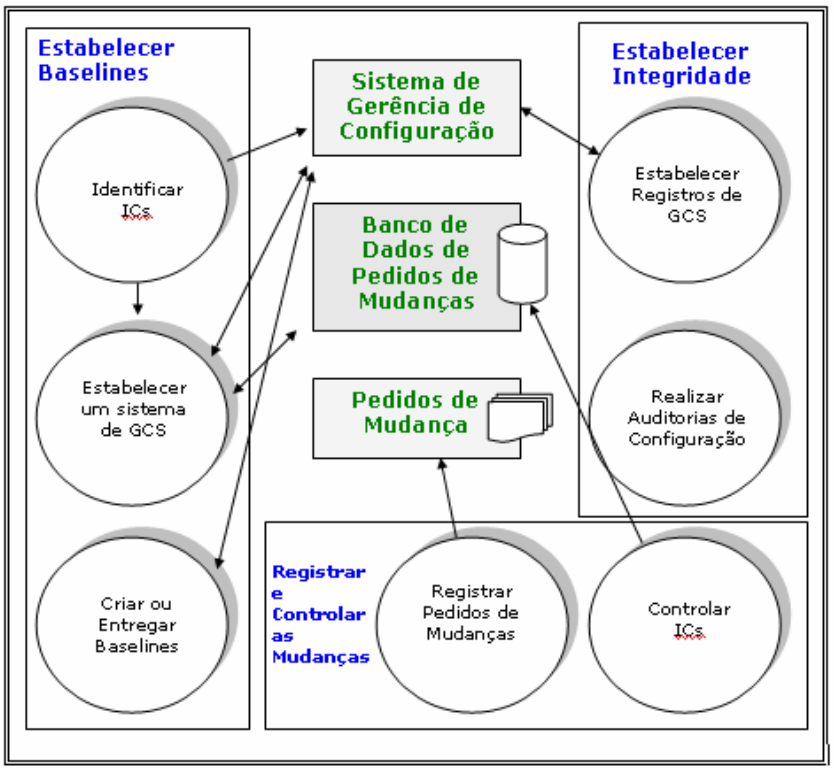


Figura 8 - Metas de GC do CMMI (Nível 2 de maturidade) [FIORINI98]

As Metas e Práticas abaixo relacionadas são pertencentes ao nível 2 de maturidade, sendo que a Meta Genérica 2 (Institucionalizar um Processo Definido) e suas práticas são requeridas somente para os níveis 3 e superiores.

Relacionamento entre Metas e Práticas

Meta 1: Estabelecer Linhas Bases
Prática 1 Identificar IC
Prática 2 Estabelecer um Sistema de GCS
Prática 3 Criar ou Liberar Linhas Bases

Meta 2: Registrar e Controlar Mudanças
Prática 1 Registrar e Acompanhar Pedidos de Mudanças
Prática 2 Controlar IC

Meta 3: Estabelecer Integridade
Prática 1 Estabelecer Registros de Gerenciamento de Configuração
Prática 2 Realizar Auditoria de Configuração

Meta Genérica 1: Institucionalizar um Processo Gerenciado
Prática 1 Estabelecer uma Política Organizacional
Prática 2 Planejar o Processo
Prática 3 Prover Recursos
Prática 4 Atribuir Responsabilidades
Prática 5 Treinar Recursos Humanos
Prática 6 Gerenciar Configurações
Prática 7 Identificar e Envolver os Interessados no Processo
Prática 8 Monitorar e Controlar o Processo
Prática 9 Avaliar Objetivamente a Aderência do Processo
Prática 10 Rever o Estado do Processo através de um Gerenciamento de Alto Nível

Meta Genérica 2: Institucionalizar um Processo Definido
Prática 1 Estabelecer um Processo Definido
Prática 2 Coletar informações de melhoria

1.2.3 NBR ISO/IEC 12207

A norma NBR ISO/IEC 12207 estabelece uma estrutura comum para os processos do ciclo de vida de software. A estrutura desta norma contém processos, atividades e tarefas que servem de referência quando uma empresa necessitar realizar aquisição, fornecimento, desenvolvimento, operação e manutenção de produtos de software [NBR12207].

São apresentadas várias referências normativas e definições necessárias para o bom entendimento dos processos por ela definida, incluindo conceitos próprios da Gerência de

Configuração. Sua organização compreende cinco processos fundamentais, oito processos de apoio e quatro processos organizacionais de ciclo de vida, como ilustrados na figura 9:

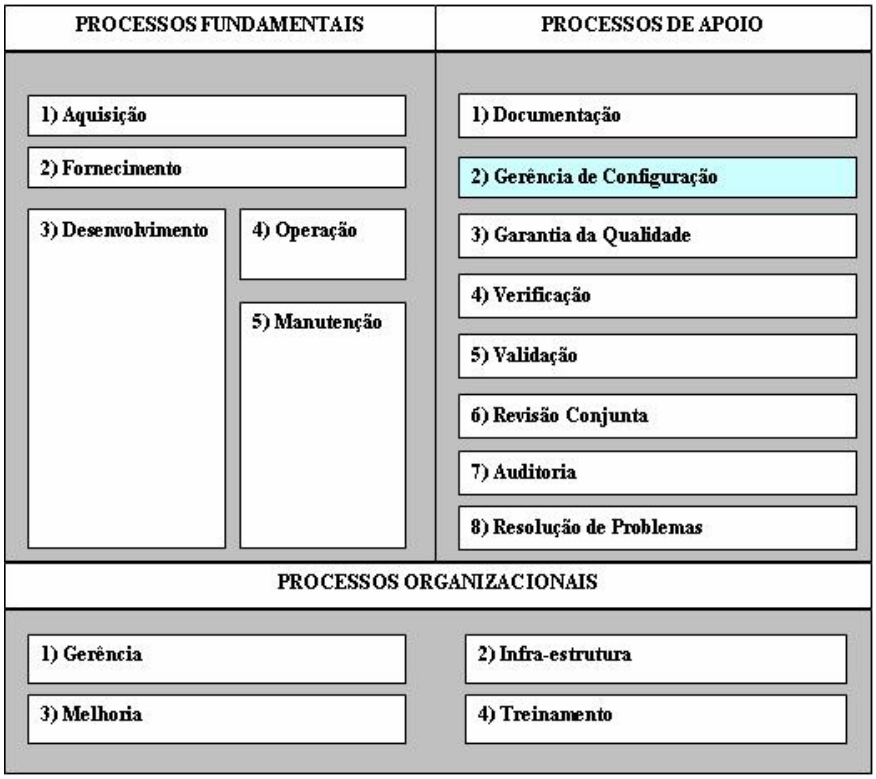


Figura 9 - Estrutura da norma NBR ISO/IEC 12207 [ROCHA04]

Processo de Gerência de Configuração da NBR ISO/IEC 12207

O processo de GCS pertence ao processo de apoio do ciclo de vida e está definido como sendo um processo de aplicação de procedimentos administrativos e técnicos, por todo o ciclo de vida de software. Este processo é executado para garantir a completeza, a consistência e a correção dos itens de software produzidos nos diversos processos do ciclo de vida do software [ROCHA04].

As atividades do processo de GCS são apresentadas de forma sucinta e sem muitos detalhes. Não são descritas informações de infra-estrutura organizacional e nem dos papéis e responsabilidades dos envolvidos no processo. As atividades do processo de GCS propostas por essa norma são [NBR12207]:

- Implementação do Processo;
- Identificação da Configuração;
- Controle da Configuração;
- Relato da Situação da Configuração;
- Avaliação da Configuração;
- Gerência de Liberação e Distribuição.

1.2.4 NBR ISO 10007

A norma NBR ISO 10007 fornece algumas diretrizes quanto à gestão de configuração de software e a integração com outros sistemas de gestão. Os objetivos desta norma são basicamente [NBR10007]:

- Aumentar o entendimento quanto à Gerência de Configuração;
- Incentivar organizações a aplicarem Gerência de configuração e assim melhorar a qualidade dos seus produtos;
- Alinhar a abordagem de Gerência de Configuração em toda a indústria;
- Melhorar a cooperação nacional e internacional quanto ao assunto.

Processo de Gerência de Configuração da NBR ISO/IEC 10007

Ao contrário da norma NBR ISO/IEC 12207, a norma ISO 10007 trata de aspectos organizacionais importantes e necessários para que os objetivos da GCS possam ser alcançados com sucesso. Outro ponto importante apresentado é quanto ao plano de Gestão de Configuração, tratado como elemento chave e que define políticas, atividades e convenções que devem ser executadas nos projetos de software da empresa.

As atividades de GCS propostas por essa norma são as seguintes:

a) Construção de planos e procedimentos de Gestão de Configuração;

b) Identificação de Configuração;

c) Controle de Configuração;

d) Contabilização da situação de Configuração;

e) Auditoria de Configuração.

Na seção "Procedimentos de Gestão de Configuração" da referida norma, são descritas, com mais detalhes, as tarefas de cada uma das atividades apresentadas acima. Segundo a norma NBR ISO/IEC 12207, estes procedimentos devem estar documentados nos Processos de GCS da organização ou nos Planos de Gerência de Configuração dos Projetos.

1.2.5 MPS.BR

O projeto MPS-BR (Melhoria de Processo do Software Brasileiro) é uma iniciativa envolvendo universidades, grupos de pesquisa e empresas brasileiras que estão sob a coordenação da SOFTEX (Sociedade para Promoção da Excelência do Software Brasileiro).

O objetivo do MPS-BR é definir um modelo de processo de software capaz de atender às necessidades das empresas brasileiras de pequeno porte e produtoras de software, no sentido de aumentar a qualidade dos seus produtos [MPS-BR05]. O MPS.BR também define regras para sua implementação e avaliação, dando assim, sustentação e garantia de que está sendo empregado de forma coerente com as suas definições [WEBER06]. Por este motivo o MPS-BR, está sendo construído seguindo as orientações das seguintes normas e modelos internacionais [MPS-BR05]:

- **NBR ISO/IEC 12207** – Processo de Ciclo de Vida de Software e suas emendas 1 e 2;
- **ISO/IEC 15504 (SPICE)** – Avaliação de Processo e seu Modelo de Avaliação de Processo de Software ISO/IEC 15504-5;
- **CMMI-SEI:** Maturidade de Processos de Software.

O MPS.BR está descrito em documentos que tem formato de guias [MPS-BR05]:

a) Guia Geral: contém a descrição geral do MPS.BR e detalha o Modelo de Referência (MR-MPS).

b) Guia de Aquisição: contém recomendações para a condução de compras de software e serviços correlatos. Foi descrito como forma de apoiar as instituições que queiram adquirir produtos de software e serviços correlatos apoiando-se no MR-MPS.

c) Guia de Avaliação: contém a descrição do processo de avaliação, os requisitos para o avaliador, os requisitos para a avaliação, o método e os formulários para apoiar a avaliação.

Com base nos níveis de maturidade do CMMI, o MR-MPS apresenta os seus níveis de maturidade, descrevendo as suas áreas de processos, juntamente com as suas capacidades. No quadro 3 estão descritos os 7 níveis de maturidade propostos pelo MPS.BR [MPS-BR05]:

Nível de maturidade	Área de processo
A **(Em Otimização)**	• Inovação e Implantação na Organização • Análise e Resolução de Causas
B **(Gerenciado Quantitativamente)**	• Desempenho do Processo Organizacional • Gerência Quantitativa do Projeto
C **(Definido)**	• Análise de Decisão e Resolução • Gerência de Riscos
D **(Largamente Definido)**	• Desenvolvimento de Requisitos • Solução Técnica • Integração do Produto • Instalação do Produto • Liberação do Produto • Verificação e Validação
E **(Parcialmente Definido)**	• Treinamento • Avaliação e Melhoria do Processo Organizacional • Definição do Processo Organizacional • Adaptação do Processo para Gerência de Projeto
F **(Gerenciado)**	• Medição • **Gerência de Configuração** • Aquisição • Garantia da Qualidade
G **(Parcialmente Gerenciado)**	• Gerência de Requisitos e Gerência de Projeto • Gerência de Projeto

Quadro 3 - Níveis e Processos do MR-MPS

Esta divisão tem uma gradação diferente do CMMI em estágios, para possibilitar uma implementação mais gradual e adequada às empresas brasileiras de qualquer porte [MPS-BR05].

A capacidade dos processos é definida através de um conjunto de atributos. Uma vez atendidos estes atributos, a organização obtém um nível de maturidade maior.

Processo de Gerência de Configuração - GCO

O propósito do Processo de Gerência de Configuração é estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los a todos os envolvidos no projeto.

Segue abaixo a descrição do processo de gerência de Configuração do MR-MPS, apresentado no Guia Geral do MPS-BR [MPS-BR05]:

a) GCO 1 - Os itens de configuração são selecionados, segundo critérios documentados;

b) GCO 2 - É especificado o responsável por cada item de configuração e quando o item é colocado sob uma linha base;

c) GCO3 - Os itens de configuração que necessitam ser independentemente identificados, armazenados, testados, revisados, usados, alterados, entregues e/ou mantidos são identificados, definidos e colocados sob uma linha base;

d) GCO 4 - É estabelecido um Sistema de Gerência de Configuração que contenha: múltiplos níveis de controle, mecanismos para armazenamento, retirada e recuperação dos itens de configuração e mecanismos para registros de solicitações de mudanças no Sistema de Gerência de Configuração;

e) GCO 5 - O conteúdo do Sistema de Gerência de Configuração é preservado;

f) GCO 6 - Antes de se criar ou liberar as linhas base dos itens de configuração, é obtida autorização do Grupo de Gerência de Configuração;

g) GCO 7 - As modificações e liberações dos itens de configuração são acompanhadas e controladas;

h) GCO 8 - As modificações e liberações dos itens de configuração são disponibilizadas para todos os envolvidos;

i) GCO 9 - As situações dos itens de configuração e solicitações de mudanças são registradas, relatadas e o seu impacto analisado;

j) GCO 10 - Executar revisões para assegurar que mudanças não causem efeitos inesperados nas linhas base;

k) GCO 11 - Os itens de configuração são controlados. Por isso, os itens de configuração alterados só são inseridos no Sistema de Gerência de Configuração após aprovação;

l) GCO 12 - A completeza e a consistência dos itens de configuração são asseguradas;

m) GCO 13 - O armazenamento, o manuseio e a liberação dos itens de configuração são controlados;

n) GCO 14 - A integridade das linhas base é estabelecida e mantida, através de auditoria da configuração e dos registros da Gerência de Configuração.

1.2.6 SWEBOK (Software Engineering Body of Knowledge)

O SWEBOK é um guia criado pelo IEEE (THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS) Software, com a colaboração de vários profissionais ligados a empresas e a instituições educacionais. O propósito de se utilizar estes colaboradores era de aproveitar o conhecimento e experiências na definição e na descrição das várias Áreas de Conhecimento (Knowledge Area - KA) presentes nesse guia.

Os cinco principais objetivos deste guia são [SWEBOK04]:

a) Promover uma visão padronizada entre todos os profissionais da Engenharia de Software;

b) Caracterizar o conteúdo das disciplinas de Engenharia de Software, descrevendo normas e práticas recomendadas na execução de cada uma delas;

c) Estruturar o Corpo de Conhecimento de Engenharia de Software;

d) Ser um guia para o desenvolvimento de currículos e certificações dos profissionais de tecnologia;

e) Promover uma integração da disciplina de Engenharia de Software com outras disciplinas, como por exemplo: Ciência da Computação, Gerência de Projetos, Engenharia da Computação e Matemática.

O SWEBOK está dividido em 12 capítulos, contendo: 01 capítulo com a introdução sobre o SWEBOK; 10 capítulos que descrevem, em detalhes, cada Área de Conhecimento proposta; e 01 capítulo descrevendo a avaliação das KAs.

Cada KA descreve a sua estrutura, apresentando introdução, breve descrição da KA e visão geral do seu escopo, além da sua relação com as outras KA's. As KA's são decompostas em subáreas, tópicos, e sub-tópicos. As 10 áreas de conhecimento de engenharia de software propostas pelo SWEBOK são:

a) Requisitos de Software (Software Requirements);

b) Desenho de Software (Software Design);

c) Construção de Software (Software Construction);

d) Teste de Software (Software Testing);

e) Manutenção de Software (Software Maintenance);

f) Gerência de Configuração de Software (Software Configuration Management);

g) Gerência de Engenharia de Software (Software Engineering Management);

h) Processo de Engenharia de Software (Software Engineering Process);

i) Ferramentas e Métodos de Engenharia de Software (Software Engineering Tools and Methods);

j) Qualidade de Software (Software Quality).

Área de Conhecimento de Gerência de Configuração de Software

No SWEBOK, a Gerência de Configuração de Software possui várias atividades que permitem o controle de configuração e mudança. Para que estes controles possam ser

realizados de forma adequada e de acordo com os objetivos da GCS, o SWEBOK dividiu esta Área de Conhecimento em 6 subáreas, conforme apresentado na figura 10:

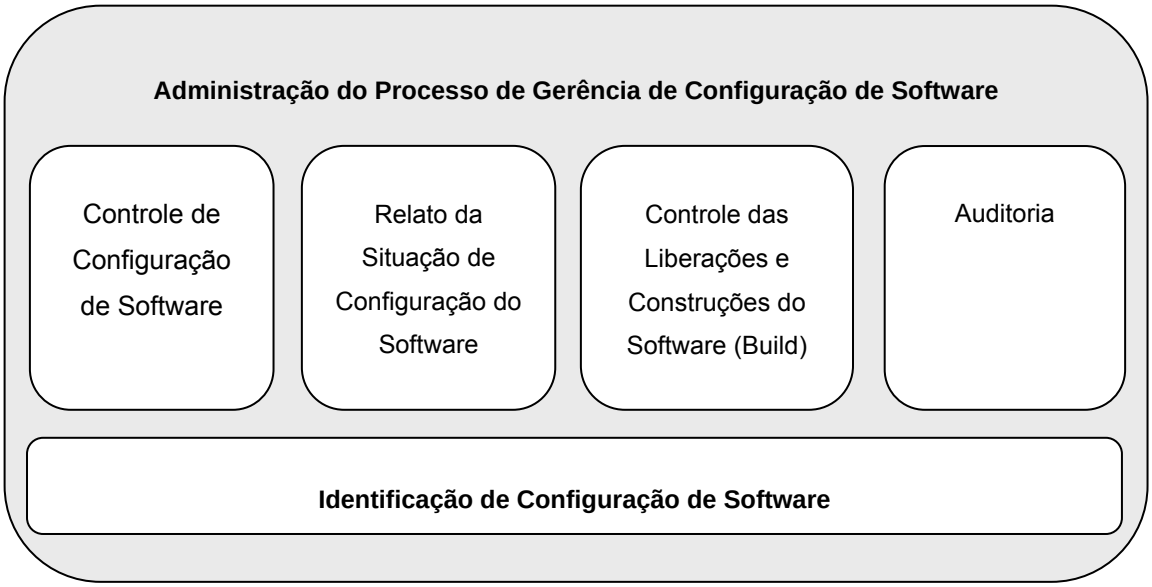


Figura 10 - Subáreas da Área de Conhecimento de GCS do SWEBOK [SWEBOK04]

Cada uma destas subáreas aborda aspectos que devem ser observados e executados durante todo o ciclo de vida do software, conforme descrito no quadro 4:

ÁREA DE CONHECIMENTO DE GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE	
SUBÁREA DE CONHECIMENTO	**ABORDAGEM**
Administração do processo Gerência de Configuração de Software.	• Contexto organizacional para GCS; • Restrições e orientação para GCS; • Planejamento para GCS; • Conteúdo do plano de GCS; • Acompanhamento da GCS.
Identificação de configuração de software.	• Identifica os itens a serem controlados e onde serão armazenados (Biblioteca de Software); • Estabelece métodos de identificação para os itens e suas versões; • Estabelece as ferramentas e técnicas a serem utilizadas na inserção e gerenciamento dos itens controlados.
Controle de configuração do software.	• Requisitando, avaliando e aprovando a modificação do software; • Implementando a modificação do software; • Divergências e desistências.

Quadro 4 - Abordagens das subáreas da Área de Conhecimento de GCS do SWEBOK

SUBÁREA DE CONHECIMENTO	ABORDAGEM
Relato da situação de configuração do software.	• Informações da situação de configuração de software contidas nos Relatórios de Situação de Configuração de Software.
Auditoria de configuração de software.	• Trata da auditoria funcional e auditoria física de Configuração de Software; • Processo de auditoria interna de uma linha base do software.
Controle das liberações e construções do software	• Controle de liberações do software; • Controle das diversas construções do software durante as iterações do desenvolvimento ou manutenção do software.

Quadro 4 – Continuação: Abordagens das subáreas da Área de Conhecimento de GCS do SWEBOK

1.2.7 CMBOK

O CMBOK (Configuration Management Body of Knowledge) é um guia que contem conhecimentos sobre diversos aspectos importantes ligados à Gerência de Configuração e Mudança de Software e Hardware. O CMBOK foi construído e tem sido aprimorado por vários profissionais e estudantes de tecnologia que utilizam os seus conhecimentos e experiências, juntamente com metodologias e artigos que tratam do assunto.

O objetivo do CMBOK é prover informações aos envolvidos nos processos de Gerência de Configuração das organizações, para que estes possam construir produtos de qualidade.

Estas práticas são bastante aceitas entres os profissionais de GCS, mas não obrigatórias. Isto porque depende de vários aspectos para serem executadas, tais como ambiente organizacional e tipo de projeto.

A primeira seção, o CMBOK descreve a estrutura da Gerência de Configuração: os ciclos de vida de aquisição e de desenvolvimento juntamente com as suas fases; os papéis envolvidos; e fatores organizacionais que podem influenciar as atividades da Gerência de Configuração.

Ainda na primeira seção são apresentados vários pontos importantes do processo de Gerência de Configuração: importância; conscientização dos executivos e demais pessoas da organização; treinamento; estratégias de implantação e de manutenção do processo de Gerência de Configuração.

Na segunda seção, são descritas as 7 áreas de conhecimento do CMBOK:

a) **Planejamento de Gerência de Configuração**: Esboça o processo, procedimentos e diretrizes que devem ser considerados na fase de planejamento da Gerência de Configuração. Também são disponibilizados exemplos de Planos de Gerência de Configuração presentes no CMMI e IEEE;

b) **Gerência de Ferramentas de Configuração**: É descrita a importância das ferramentas de GC no apoio à Gerência de configuração, bem como, os vários tipos ferramentas existentes no mercado;

c) **Gerência de Configuração (atividades e tarefas):** Nesta área são descritas as atividades e tarefas, a seqüência delas no ciclo de vida e a duração destas atividades;

d) **Contribuições da Gerência de Configuração aos Projetos:** Nesta área são apresentados os vários benefícios da GCS quando executada nos projetos organizacionais;

e) **Gerência de Configuração e Riscos:** Tendo como base o PMBOK, esta área de conhecimento descreve como os riscos de projeto, quanto à Gerência de Configuração, devem ser considerados e monitorados durante o Ciclo de Vida do Projeto. As atividades descritas são: Plano de Gerência de Riscos, Identificação de Riscos; Evitando os Riscos, Análise Quantitativa Versus Qualitativas de Riscos e Controlando e Monitorando Riscos;

f) **Aquisição para Gerência de Configuração:** Nesta área de conhecimento são descritas as orientações quanto à obtenção de recursos necessários para a execução do processo de Gerência de Configuração. As orientações apresentadas são embasadas no que é proposto no PMBOK. Dentre os recursos possíveis a serem adquiridos pode-se citar: serviços; consultores; ferramentas; e sistemas. As atividades propostas nesta área de conhecimento podem ser agrupadas sobre os seguintes aspectos: Seleção de Recursos necessários, Administração de Contratos e Encerramento de Contratos.

g) **Métrica para Gerência de Configuração:** Nesta área de conhecimento é descrita a importância das métricas dentro da Gerência de Configuração. Através de vários relatórios a equipe de Gerência de Configuração pode monitorar tudo o que acontece durante o ciclo de vida do projeto, sob o aspecto de mudanças ocorridas nos diversos itens de configuração.

A terceira seção do CMBOK traz assuntos complementares em forma de Apêndices. E na última seção, glossário e índices com definições e terminologias utilizadas na Gerência de Configuração.

1.2.8 ITIL

O ITIL (Information Tecnology Infrastructure Library) é uma biblioteca de estrutura de tecnologia de informação, onde contem um conjunto de melhores práticas destinadas à obtenção de qualidade nos serviços de Tecnologia da Informação (TI) das empresas.

O ITIL é publicado em uma série de livros que descrevem diretrizes para diferentes aspectos. Estes livros foram criados para atender à área de Tecnologia da Informação fornecendo informações quanto às necessidades das empresas e as suas metas [ITIL05]. Os assuntos de cada livro estão agrupados em 8 coleções e divididas em disciplinas que abordam assuntos específicos, conforme descrito abaixo:

a) **Serviço de Entrega:** Serviço que um centro de dados deve oferecer ao negócio para que o suporte possa ser adequado.
   - Gerência Financeira de TI (IT Financial Management);

- Gerência de Capacidade (Capacity Management);
- Gerência de Disponibilidade (Availability Management);
- Gerência de Continuidade de TI (IT Continuity Management);
- Gerência do Nível de Serviço (Service Level Management).

b) **Serviço de Suporte**: Como fazer para que o cliente tenha acesso ao serviço adequadamente.
   - Gerência de Mudança (Change Management);
   - Gerência de Liberação (Release Management);
   - Gerência de Problema (Problem Management);
   - Gerência de Incidente (Incident Management);
   - Gerência de Configuração (Configuration Management);
   - Serviço de Ajuda (Service Desk).

c) **Planejamento para implantar a Gerência de Serviço**: Como realizar a comutação para o ITIL. Explica os passos para identificar os benefícios produzidos pelo ITIL e como tirar proveito destes benefícios.

d) **Gerência de Segurança**.

e) **Gerência de Infra-estrutura:** Quais os processos, organizações e ferramentas são necessários para prover uma estável infra-estrutura de TI e de comunicação.
   - Gerência dos Serviços de Rede (Network service Management);
   - Gerência de Operações (Operations Management);
   - Gerência de processador local (Management of local processors);
   - Instalação e Aceitação de Computador;
   - Gerência de Sistemas (Systems Management).

f) **A Perspectiva Empresarial**: Explica os princípios fundamentais e exigências do negócio e da operação, e como estes se relacionam com o desenvolvimento, com a entrega e com o serviço de suporte de TI.

g) **Gerência de Aplicações:** Como gerenciar o ciclo de vida de desenvolvimento do software, ampliando os assuntos abordados no ciclo de vida de desenvolvimento de Software e teste de Serviços de TI.

h) **Gerência dos Recursos de Software**. Como gerenciar os recursos de software existentes na organização.

Gerência de Configuração do ITIL

O principal propósito da disciplina de Gerência de Configuração do ITIL é rastrear a estrutura e relacionamentos dos Itens de Configuração durante o ciclo de vida de um sistema.

Os objetivos principais da GCS no ITL são:

a) Prover informações de infra-estrutura de TI aos outros processos e gerentes de TI;

b) Habilitar o controle da infra-estrutura através de monitoramento e manutenção das informações (situação, histórico e relacionamentos dos Itens de Configuração) de todos os recursos necessários na prestação do serviço;

O ITIL descreve 4 tarefas primordiais na disciplina de Gerência de Configuração que seguem descritas abaixo:

a) Identificação dos Itens de Configuração e os relacionamentos entre eles;

b) Gerência dos Itens de Configuração e seus incidentes, problemas, modificações e liberações;

c) Registro de Situação e Relatórios;

d) Verificações através de Auditorias e Inspeções.

Gerência de Mudança do ITIL

O objetivo principal desta disciplina é aprovar modificações aprovadas eficientemente, com custo baixo e com mínimo risco para a nova e a já existente infra-estrutura.

O Processo de Gerência de Mudança do ITIL propõe as seguintes atividades:

- Solicitação de Mudança;
- Registro e Classificação;
- Monitorando e Planejando;
- Aprovação;
- Construção e Teste;
- Implementação aprovada;
- Implantação;
- Validação;

Gerência de Liberações do ITIL

Liberações são executadas sob o controle de Gerência de Mudança e têm os seguintes objetivos:

- Proteger todo o software e os seus itens de configuração;
- Assegurar que somente versões testadas/corretas do software e hardware estão sendo utilizadas;

As tarefas descritas no ITL para a Gerência de Liberações são:

- Definir políticas de liberações;
- Controle da Biblioteca de Definições de Software (DSL);
- Controle do Armazenamento das Definições de Hardware (DHS);
- Distribuição do software e seus itens associados;
- Gerenciar as liberações de software;

- Conduzir auditorias de software, onde serão avaliados, quanto à qualidade e conformidade, os itens gravados no banco de dados de configuração do projeto, chamado no ITIL de CMDB (Configuration Management Data Base).
- Supervisionar as construções das liberações do software.

1.2.9 Comparativo das atividades das normas, padrões e modelos de melhoria de Processos

PADRÕES / ATIVIDADES	RUP	CMMI	NBR / ISO 12207	NBR / ISO 10007	MPS-BR	SWEBOK	CMBOK	ITIL
Planejar Configuração do projeto e Controle de Mudanças	S	S	S	S	S	S	S	N*
Criar Ambiente para GCS do Projeto	S	S	S	S	S	S	S	S
Identificação de Configuração	S	S	S	S	S	S	S	S
Alterar e Liberar Itens de Configuração	S	S	S	S	S	S	S	S
Gerenciar Linhas de Base (Baselines)	S	S	S	S	S	S	S	S
Gerenciar Liberações (Releases)	S	S	S	N	S	S	S	S
Monitorar e Relatar Estado de Configuração	S	S	S	S	S	S	S	S
Gerenciar Solicitações de Mudança	S	S	S	S	S	S	S	S
Auditoria de Configuração	S	S	S	S	S	S	S	S

Quadro 5 - Comparativo dos padrões e modelos

Legenda: S – Sim N – Não

* O ITIL prevê o Controle de Mudanças, mas não abrange o Planejamento de Configuração do Projeto

1.3 Ferramentas de Apoio à Gerência de Configuração e Controle de Mudanças

Nesta seção serão apresentadas ferramentas de apoio para a disciplina de Gerência de Configuração e Controle de Mudanças. As ferramentas selecionadas são amplamente utilizadas no mercado. São elas:

- CVS – Concurrent Version System (Controle de Versões) – Open Source
- Bugzilla (Controle de Mudanças) – Open Source
- Eclipse (Ambiente Integrado de Desenvolvimento) - Eclipse Public License - EPL

As ferramentas para GCS objetivam automatizar e facilitar a aplicação das melhores práticas da disciplina de GCS. Assim como um compilador, um editor ou um ambiente integrado de desenvolvimento, as ferramentas de GCS são elementos essenciais do kit de ferramentas do engenheiro de software. Estes ferramentas ajudam a automatizar a tarefa manual que é susceptível a erros, podendo garantir que o projeto possa suportar todas as melhores práticas dessa disciplina [BELLAGIO05].

O subsistema de GCS que mais recebeu contribuições tanto de pesquisa quanto comercialmente é o de controle de versões. Várias soluções comerciais, como, por exemplo, *ClearCase*, da Rational Software e *Visual SourceSafe*, da Microsoft, fornecem características satisfatórias para o problema quando os IC são artefatos simples, definidos através de arquivos do sistema operacional. Além disso, outras tantas soluções livres estão disponíveis, como, por exemplo, CVS e RCS (Revision Control System).

Existe também um outro conjunto de ferramentas que dão apoio ao subsistema de Controle de Solicitação e Mudanças. Dentre elas pode-se citar o Bugzilla (open source, licença MPL - *Mozilla Public License*) e o Rational Clear Quest (proprietário, IBM/Rational).

As ferramentas de Controle de Solicitação e Mudanças proporcionam o registro das solicitações de mudanças dos IC que estão sob gerência de configuração, bem como o acompanhamento da implementação da mudança, através de relatórios que informam a situação (estado) da modificação. O objetivo é promover visibilidade e registro histórico ao processo de

GCS, além de melhorar a comunicação entre os envolvidos no desenvolvimento ou manutenção de software.

Como forma de aprimorar a comunicação entre os envolvidos e registrar o que acontece durante o ciclo de vida da modificação do IC, as ferramentas de Controle de Solicitação e Mudanças oferecem alguns recursos importantes:

- Formulários para registros e manutenção dos atributos das solicitações de mudanças.
- Arquivos e comentários anexados aos registros de solicitações de mudanças;
- Controle de acesso através de contas de usuários;
- Mecanismos de consultas e relatórios;

Neste trabalho será apresentado para **Controle de Versões** o software CVS – Concurrent Version System e para **Controle de Solicitação e Mudanças** o software Bugzilla.

1.3.1 CVS – Concurrent Version System

CVS é uma ferramenta de código livre, utilizada para a execução de duas tarefas essenciais ao controle de versões, seja de arquivos ASCII ou Binários. A primeira destas tarefas é o registro de cada modificação realizada pelos desenvolvedores, incluindo informações sobre o quê foi modificado, quando e quem efetuou a modificação. A segunda tarefa é assegurar que as mudanças feitas por cada desenvolvedor não sobrescrevam acidentalmente o trabalho realizado por outros desenvolvedores. Através do registro feito por cada desenvolvedor, outros desenvolvedores do mesmo projeto podem rapidamente contatar o responsável pela mudança. O histórico das modificações é mantido no CVS. Além disso, os desenvolvedores podem facilmente reverter o código fonte para uma situação anterior às modificações feitas por ele. CVS garante a consistência dos artefatos, pois possui a capacidade de mesclar as alterações feitas nestes códigos. Caso ocorra um conflito em que o CVS não possa fazer a mesclagem automática, o desenvolvedor é notificado, de forma que possa intervir e solucionar o conflito manualmente, garantindo assim a integridade do artefato.

Através do armazenamento dos artefatos em um ambiente centralizado, há a garantia de que existem pelo menos duas cópias dos artefatos sob gerência de configuração. Uma delas no ambiente de desenvolvimento do desenvolvedor, denominada “working copy”, e a outra no repositório do

CVS, normalmente disponível em um servidor centralizado. Esta simples característica permite a prevenção de perda de dados em caso de falhas de hardware, ou até mesmo de operações incorretas por parte dos desenvolvedores, fato que normalmente é catastrófico em grandes projetos. Além disso, a otimização do espaço em disco é garantida, pois o CVS armazena uma cópia do artefato, sendo que as demais mudanças são armazenadas como diferenciais, não havendo redundância de artefatos. A redundância ocorre apenas nos artefatos binários, onde não é possível o armazenamento de diferenciais, sendo necessária a duplicação do produto sob gerência de configuração.

A arquitetura do CVS é cliente servidor, sendo multiplataforma para cliente e servidor simultaneamente.

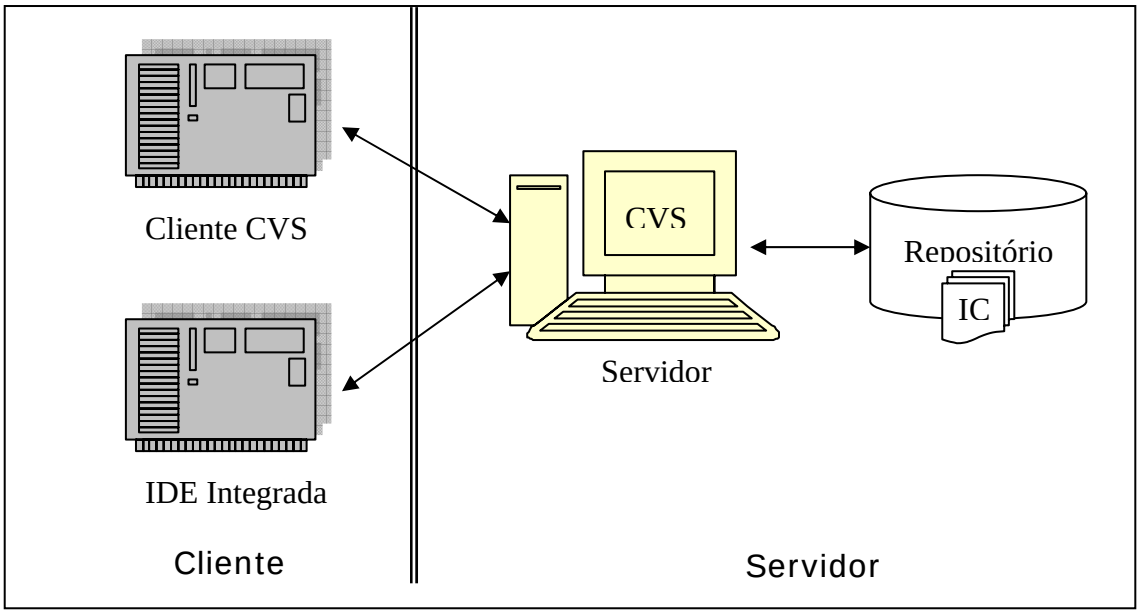


Figura 11 - Arquitetura CVS

Histórico

O CVS surgiu a partir de um conjunto de Shell scripts escritos por Dick Grune, que por sua vez foram postados no newsgroup *com.sources.unix*, em dezembro de 1986. Atualmente, muitos dos algoritmos de resolução de conflitos ainda são originários dos scripts de Dick Grune. Em abril de 1989, Brian Berliner projetou e codificou o CVS. Em dezembro de 1999, Tony Hoyle converteu a base CVS do Unix para o Windows NT, criando-se o CVSNT.

Atualmente, a organização responsável por manter o código do projeto disponibiliza a CVS e toda a sua documentação no site www.cvshome.org. O CVS é utilizado no controle de versões de

projetos como o Linux, Jboss, Tomcat, Apache, MYSQL, além de incubadoras de software livre como SourceForge, Tigris e Savannah [CEDERQUIVIST05].

Terminologia Básica do CVS

Cliente CVS: O cliente CVS é necessário para conectar-se ao servidor do CVS remoto, tendo em vista que CVS utiliza outros protocolos para comunicação. Dentre eles, o PSERVER é um dos mais utilizados (sem criptografia para senhas). Entretanto, o protocolo SSH também poderá ser utilizado (senhas criptografadas).

Repositório CVS: O CVS armazena uma cópia dos dados do projeto que são submetidos pelos desenvolvedores. Este servidor retém uma cópia da mais recente versão estável dos artefatos que estão sob gerência de configuração, bem como todo o histórico de modificações (versões). Normalmente, cada projeto de uma organização fica armazenado em um repositório diferente.

Módulo: CVS provê uma forma de armazenar diferentes tipos de dados dentro do mesmo repositório. Cada conjunto de dados distinto armazenado dentro de um repositório será colocado dentro do seu próprio módulo. Quando o desenvolvedor recupera os dados do repositório, deverá saber o nome do projeto usado para o módulo que contém os dados desejados. Módulos contém arquivos e diretórios, assim como o sistema de arquivos de um computador. Módulos são diretórios de maior nível dentro do repositório.

Cópia de Trabalho: Quando o desenvolvedor resgata os dados do repositório do CVS, utilizando o cliente CVS, uma cópia dos artefatos são salvos no sistema de arquivos local do desenvolvedor. Esta cópia é denominada “Working Copy” - Cópia de Trabalho ou “Workspace” – Espaço de Trabalho. Este processo de obtenção da cópia de trabalho é denominado “Check-out”. Para a correta realização do check-out, o desenvolvedor deverá informar o nome do módulo desejado. Somente os arquivos e diretórios localizados no módulo informado serão copiados para a cópia de trabalho.

Commit: É o processo de enviar os dados alterados pelo desenvolvedor, a partir de sua cópia de trabalho, para o repositório CVS. Quando o desenvolvedor executa o *commit*, somente os arquivos que foram alterados são enviados para o repositório. Considerando o processo de Controle de Mudanças, definido pela disciplina de Gerência de Configuração e Controle de Mudanças, a

execução do *commit* deverá somente ser realizada após a validação e aprovação das mudanças. Commit é uma nomenclatura do CVS como sinônimo de *Check-in*.

Update: Quando o check-out é realizado, a última versão de cada arquivo é baixada para o ambiente de trabalho do desenvolvedor. O desenvolvimento continua, entretanto, sua cópia poderá ficar desatualizada, pois outro desenvolvedor poderá ter realizado *commit* a partir de outro ambiente de trabalho. Para atualizar sua cópia, o desenvolvedor deverá realizar o comando "*update*". A partir deste instante, todas as mudanças são mescladas na sua cópia de trabalho.

Modo de trabalho otimista: No CVS, por padrão, não é necessário travar os arquivos para edição, sendo que vários desenvolvedores podem trabalhar no mesmo artefato ao mesmo tempo. CVS possui modo de bloqueio otimista, ou seja, parte do princípio que conflitos são raros, sendo que o desenvolvedor pode fazer alterações em qualquer recurso que tenha acesso.

Números de Revisão (Revision Numbers): representam as alterações efetivadas em cada um dos arquivos. Cada arquivo sob o controle do CVS possui o seu número de revisão. São representados por uma parte fixa, e uma parte que varia de acordo com as alterações, por exemplo: 1.1, 1.2, 1.3. A parte fixa pode ser configurada de acordo com as necessidades.

Rótulos (Tags): Representam uma posição dos arquivos de um determinado produto em um determinado tempo. Agrupam os arquivos e seus números de revisão sob um único nome.
Ex.: Considere os seguintes arquivos e seus números de revisão:

- Cliente.java – 1.1
- NotaFiscal.java – 1.2
- ItemNotaFisca.java – 1.1

Pode-se criar então um rótulo chamado Versao1, por exemplo, e dizer que os arquivos acima estão neste rótulo. Se alterarmos o arquivo Cliente.java para a revisão 1.2, por exemplo, e solicitarmos ao CVS os fontes do rótulo Versao1, ele vai retornar o arquivo Cliente.java na revisão 1.1, que é a revisão que faz parte deste rótulo. Existe um rótulo com o nome *HEAD*, que é padrão do CVS, e corresponde a linha base de desenvolvimento.

Ramos (Branches): Quando um arquivo passa a ser alterado em um ramo, seu número de revisão passa a ter um número a mais, sempre par, iniciando por dois, que indica o ramo, e mais um

número seqüencial de acordo com as alterações. O CVS também permite replicar as alterações dos diversos ramos para o ramo *HEAD*, que contém a linha base de desenvolvimento.

O modelo de trabalho CVS

O CVS possibilita que vários programadores desenvolvam a aplicação simultaneamente. O modelo de trabalho CVS é baseado no conceito (copiar – modificar - mesclar) e acontece na forma do fluxo definido na figura 12.

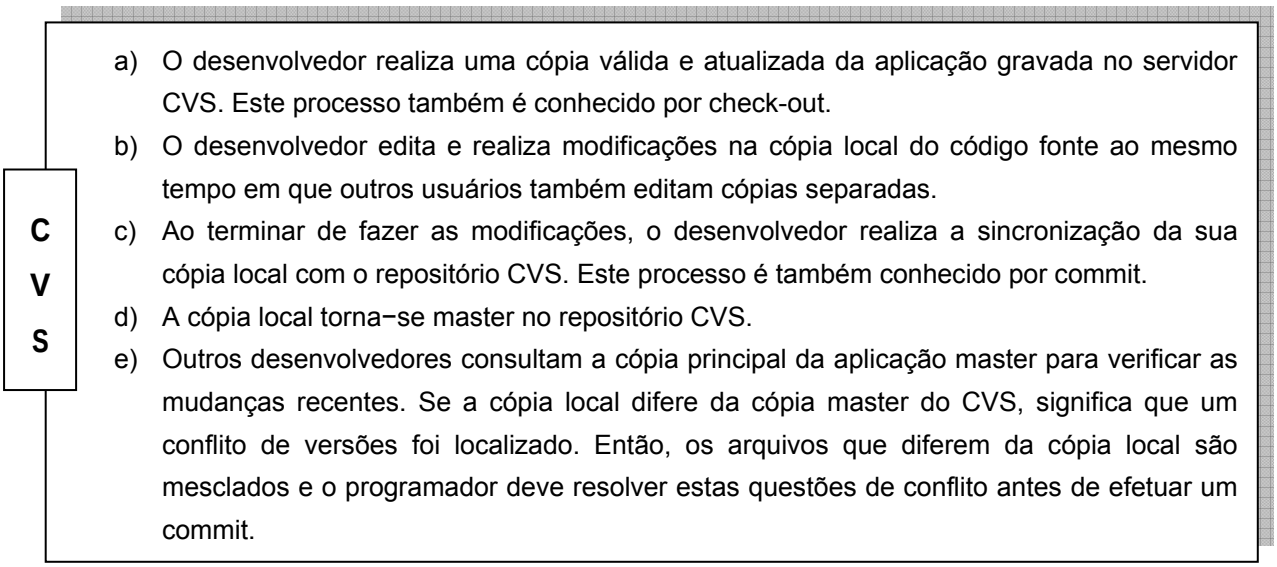


Figura 12 - Fluxo de Trabalho do CVS

Estratégia CVS para tratamento de arquivos binários

Arquivos Especiais

O CVS somente pode armazenar arquivos que possam ser convertidos para o formato RCS. Arquivos não podem ser adicionados se indicarem dispositivos (etc/dev, no caso do sistema operacional Linux, por exemplo) ou ligações simbólicas com outros arquivos ou diretórios, bem como outros arquivos que não sejam persistentes ou que não possam ser abertos ou modificados. Para armazenar estes arquivos como parte do projeto sob gerência de configuração, os comandos necessários para a criação destes arquivos devem ser armazenados em scripts, que por sua vez serão armazenados no repositório CVS.

Arquivos Binários

O método de trabalho padrão do CVS, relativo a questão de solução de conflitos, está baseado na manipulação de arquivos texto, pois o CVS pode determinar qual linha foi modificada ou adicionada, ou removida. CVS não possui esta capacidade quando o tipo de arquivo for binário, pois este tipo de arquivo não possui a característica de separar as linhas de texto com o caractere ASCII correspondente ao "*Carriage Return*".

Alguns arquivos com conteúdo textual não são exatamente arquivos texto. Os arquivos produzidos pelo Microsoft Word ou OpenOffice, por exemplo, são arquivos que possuem conteúdo textual, porém devem ser assinalados no CVS como binários, pois a mesclagem de dados linha a linha poderá corromper o respectivo arquivo.

Para arquivos binários, CVS utiliza um método diferente para resolução de conflitos. Os dois métodos disponíveis no CVS são *Merge* e *Copy*. O método *Merge* é o padrão. O método *Copy* instrui o CVS a prover o usuário com ambas as versões do arquivo se houver um conflito, então o usuário poderá solucionar as mudanças manualmente e efetuar novamente o *commit*.

Arquivos binários deverão ser adicionados ao CVS através da utilização do parâmetro - kb no comando *cvs add.* Este comando instrui o CVS a não efetuar modificações nas finalizações de linhas e a utilizar o método *Copy* para resolução de conflitos.

A listagem abaixo exemplifica o agendamento para o envio de dois arquivos binários do Microsoft Word para um repositório CVS:

```
>cvs add -kb Requisitos.doc
   cvs server: scheduling file "Requisitos.doc" for addition
   cvs server: use "cvs commit" to add this file permanently
>cvs add TesteAceitacao.doc
   cvs server: scheduling file "TesteAceitacao.doc" for addition
   cvs server: use "cvs commit" to add this file permanently
>cvs commit
```

O exemplo anterior é muito bem aplicado quando for adicionar arquivo por arquivo. Entretanto, para o trabalho em projetos que contenham dezenas, centenas ou talvez milhares de arquivos, esta tarefa se torna improdutiva. A solução proposta pelo CVS é a utilização de caracteres

coringa, denominados "*wildcards*". O nome do arquivo é o elemento chave para este trabalho (normalmente utilizado através da extensão do arquivo). Os mais importantes símbolos especiais a serem utilizados são: "*" (qualquer seqüência de caracteres, incluindo seqüência nula, e "?" - qualquer caractere individual) [VERPERMAN03].

1.3.2 Bugzilla

Bugzilla foi criado em 1998 pelo projeto Mozilla (*www.mozilla.org*) como uma ferramenta de Engenharia de Software para controle de erros e alterações realizados nos projetos, tornando-se parte do processo de desenvolvimento [REIS02].

O Bugzilla é utilizado pelos desenvolvedores e mantenedores de software com o objetivo de aumentar a produtividade e qualidade no desenvolvimento dos produtos, através do registro de bugs. Estes bugs caracterizaram tanto um defeito quanto uma evolução de um software.

Utilizando o Bugzilla como ferramenta de apoio ao Sistema de Controle de Solicitação e Mudanças, os envolvidos no projeto têm condições de organizar, gerenciar e rastrear adequadamente as alterações. Podem também utilizar recursos como anexos, integração com correio eletrônico, gráficos, buscas, grupos, contas de usuário, consulta, relatórios, e outros [REIS02].

O Bugzilla é um software de servidor composto de diversas páginas *www*, implementadas como scripts CGI. A linguagem principal de desenvolvimento é Perl e o código gerado após o processamento CGI inclui HTML, JavaScript e CSS (Cascade Style Sheet). Para que o Bugzilla seja customizado e internacionalizado por outros projetos ou empresas, um template denominado *Toolkit* é disponibilizado para tal finalidade. As plataformas suportadas pelo Bugzilla são UNIX, Linux e Windows. Utiliza banco de dados MySQL e um servidor web (Apache).

Uma característica importante do Bugzilla é a integração com várias ferramentas, como Bonsai e CVS. Um exemplo prático desta integração seria a partir de uma listagem de modificações no CVS, obtida pela ferramenta Bonsai (bonsai.mozilla.org) é possível identificar e acessar o bug associado àquela alteração. Esta propriedade auxilia a tarefa de identificar as causas de regressões funcionais ou de performance. [BARNSON05]

Histórico do Bugzilla

Segundo [BUGZILLA04], quando o Mozilla foi criado em 1998, o Bugzilla foi um dos primeiros produtos do projeto a ser disponibilizado aos desenvolvedores do projeto. Bugzilla foi desenvolvido originalmente em TCL (*Tool Command Language*) por Terry Weissman. Antes de mozilla.org tornar-se software livre, Terry decidiu migrar o Bugzilla para Perl, com esperança que mais pessoas poderiam contribuir no seu desenvolvimento e aperfeiçoamento. Surgiu então o Bugzilla 2.0. A partir daí, um número grande de projetos começaram a adotá-lo como ferramenta de apoio ao Gerenciamento de Controle de Solicitações e Mudanças.

Terminologia do Bugzilla

Bug: Serve para caracterizar tanto defeitos, quanto requisições de uma nova funcionalidade. Cada bug é identificado por um número seqüencial (e, opcionalmente, por um alias ou apelido), e é associado ao módulo (produto e componente) e à versão em que está presente [REIS02]. Além da identificação, cada bug possui atributos que o caracterizam e que proporcionam o acompanhamento da sua evolução no processo de modificação.

Situação e Resolução (Status and Resolution): Esta propriedade define exatamente a situação do bug, podendo ser Não Confirmado (unconfirmed), Novo (New), Assigned (Nomeado), Resolvido (Resolved), Reaberto (Reopen), Verificado (Verified), Fechado (Closed).

Nomeado (Assigned To): Esta propriedade indica a pessoa responsável por registrar o bug.

URL: A URL associada ao bug, se existir.

Resumo (Summary): Propriedade que apresenta um resumo do bug (problema ou evolução).

Notas (Estado Whiteboard): Campo destinado ao registro de notas curtas sobre o bug.

Palavras-chaves (Keywords): O Administrador pode definir palavras-chaves que servem como rótulos e para categorizar o bug.

Plataforma e Sistema Operacional (Platform and OS): Estes indicam o ambiente computacional onde o bug está localizado.

Versão (Version): Propriedade que serve para indicar a versão dos componentes nos quais o bug está vinculado.

Prioridade (Priority): Propriedade utilizada para indicar o grau de prioridade de solução do bug.

Severidade (Severity): Propriedade que indica o grau de seriedade do problema.

***Target*:** Indicação de qual será a próxima versão do bug após a sua conclusão.

Registrado por (Reporter): Pessoa que registrou o bug.

Lista CC: Lista dos e-mails das pessoas que deverão ser comunicadas quando o registro do bug sofrer qualquer modificação.

Anexos (Attachments): Arquivos que podem ser anexados ao bug.

Dependências (Dependencies): Lista dos bugs que não podem ser finalizados até que o bugs em questão seja finalizado. Ou ainda, são os bugs que precisam ser finalizados para que o bug em questão possa ser concluído.

Voto (Votes): Indica se o bug tem qualquer voto que indica a sua aceitação ou não.

Comentários adicionais (Additional Comments): Proporciona o registro de outras informações/comentários a respeito do bug.

Contas de Acesso (Accounts): Para que o usuário possa registrar e acompanhar bugs ele primeiro deverá criar uma conta de acesso, informando o seu e-mail. A senha é criada automaticamente e enviado ao e-mail informado. O Bugzilla dará a opção de alterar a senha gerada no primeiro acesso do usuário.

Lista de bugs (Bug list): Quando uma consulta é executada, o Bugzilla retorna uma página com uma lista de bug, localizados de acordo com os parâmetros informados na consulta. Esta lista também pode ser obtida através de outros programas, bastando acrescentar na URL de

chamada o parâmentro ctype=type. No caso da chamada ser feita pelo CVS, o parâmetro seria ctype=cvs.

Ciclo de vida do bug no Bugzilla

Como forma de organização do trabalho e das alterações, todo bug sofre alterações no seu estado (status) e pode percorrer caminhos intermediários entre os estados possíveis, até estabilizar-se em um estado final. As possibilidades de iterações pelos estados de ciclo de vida são infinitas, conforme mostrado na figura 13.

Inicialmente, um relato de bug é criado devido a erros encontrados, sugestão de melhorias ou adição de novas funcionalidades ao software. O relator deve preencher devidamente os campos das propriedades do bug de modo a mostrar as condições em que ele ocorreu e suas características. O relator deve ser sucinto ao preencher o campo resumo e detalhista para descrever o relato de modo que os envolvidos possam entendê-lo.

O responsável inicial (*assigned*) do bug é sempre o relator. Pode-se reatribuir (*reassign*) a responsabilidade do bug a qualquer outro usuário posteriormente. O relato de bug é encaminhado por e-mail para ao responsável pelo software e o bug assume o estado inicial de NÃO CONFIRMADO (*Unconfirmed*) ou Nova (*New*). Se necessário, pode ser enviado a outras pessoas, para isso deve ser preenchido o campo CC.

O responsável pelo bug ou outros usuários verificam se os dados do relato estão corretos e se ele já foi descrito anteriormente. Geralmente alguns campos são modificados (prioridade, severidade). Se já existe bug igual ou muito similar a um bug em questão, este último é marcado como Duplicado (*Duplicated*).

Tenta-se reproduzir o bug. O bug pode passar por votação. Se o bug for confirmado e aceito, assume o estado Novo (*New*). O próprio responsável ou alguém que se disponha a trabalhar no problema marca o bug como Atribuído (Assigned). Quando for encontrada uma solução, o bug é marcado como Resolvido (Resolved).

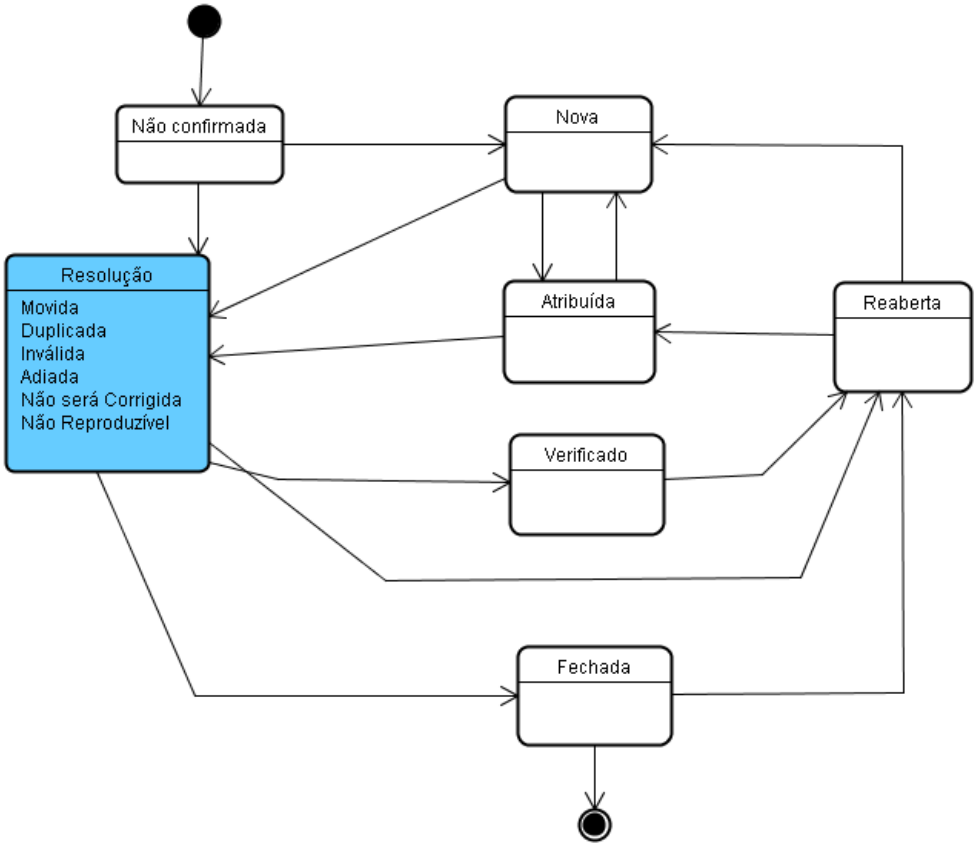


Figura 13 - Ciclo de vida do bug no Bugzilla [BARNSON05]

O estado RESOLVED (Resolvido) indica o tipo de solução dada ao problema, podendo ser **[QUEMELLO05]:**

- **Fixed (Corrigido)**: o bug foi solucionado (ou consertado).
- **Invalid (Inválido)**: o problema ou a notificação feita não é um bug (ou não é um bug válido).
- **Wontfix (Não será corrigido)**: o problema descrito é um bug que nunca será resolvido, solucionado ou implementado.
- **Later (Adiada):** o problema ou alteração descrita é um bug que não terá solução nesta versão do produto.
- **Remind (Pode ser reaberto)**: o problema ou alteração descrita é um bug que não terá solução nesta versão do produto, mas pode até ter uma solução.

- **Duplicate (Duplicada)**: o bug descrito é uma duplicação de um outro bug existente. É interessante indicar nos comentários do bug o número do outro bug replicado.
- **Worksforme (Não reproduzível):** todas as tentativas de reproduzir o bug foram inúteis, não foram encontraram pistas ou respostas pelas razões e motivos da ocorrência ou presença deste bug, e até possivelmente a leitura de código não mostrou nenhuma verdade sobre ele. Se alguma informação aparecer mais tarde, o bug pode ser reaceito, mas não por enquanto.

Posteriormente podem ser encontradas falhas na resolução e o bug tem que ser reaberto e alterado o seu estado para REOPENED (Reaberto). Se a solução definitiva for encontrada o bug é fechado (CLOSED) novamente [BARNSON05].

1.3.3 Eclipse

O Eclipse é uma plataforma IDE (Integration Development Environment) utilizada pelos desenvolvedores como um ambiente para integração de ferramentas de desenvolvimento e construção de novas ferramentas, utilizando várias linguagens de programação (HTML, Java, C++, C, JSP, EJB, XML e outros). São exemplos de IDE's para o Eclipse: a JDT (*Java Development Tools*); a CDT (*C Development Tools*) e a PDE (P*lug-in Development Environment*). A Plataforma do Eclipse é desenvolvida em Java e oferecida para vários ambientes operacionais, incluindo Linux, HP-UX, AIX, Solaris, QNX, Mac OS X e Windows [ECLIPSE05].

O Eclipse é um software livre mantido por colaboradores como IBM, Borland, Hitashi, TeamStudio. Possui uma arquitetura extensível baseada no uso e desenvolvimento de plug-ins que facilitam a integração das várias ferramentas instaladas no seu ambiente.

Um benefício desta integração é o apoio do Eclipse aos diversos processos e fases de um projeto de desenvolvimento do software (Modelagem UML, Implementação, Testes, Distribuição da aplicação, controle de versões e gerência de configuração). Isto porque esta ferramenta manipula várias extensões de arquivos.

Breve Histórico do Eclipse

Em novembro de 2001, líderes de indústria de informática - como IBM, Borland, MERANT, Rational - criaram o Consórcio *eclipe.org* com o objetivo de desenvolver uma ferramenta de plataforma aberta para integração de outras ferramentas e que deveria ser mantida pelos colaboradores do consórcio [ECLIPSE05].

Durante alguns anos, várias empresas, como *Oracle, Sybase, Fujitsu* e OMG, foram incluídas no Consórcio. A inclusão era por meio de aprovações realizadas em assembléias do conselho.

Em fevereiro de 2004, a IBM liberou o código fonte do Eclipse. Com esta mudança, acadêmicos, instituições de pesquisa, outras empresas e desenvolvedores individuais passaram a colaborar na manutenção e evolução dessa ferramenta. Os colaboradores também utilizam o Bugzilla (*bugzilla.eclipse.org*) como ferramenta de apoio ao controle das mudanças realizadas no Eclipse [ECLIPSE05].

Terminologia do Eclipse

Bancada (Workbench): É a área de trabalho do Eclipse contendo menus, barras de ferramentas, editores e visualizadores. Cada desenvolvedor possui o seu próprio Workbench de forma isolada.

Perspectiva: Um Workbench oferece um ou mais perspectivas. Uma perspectiva é um grupo de visualizadores e editores presentes no WorkBench e que representam uma funcionalidade do Eclipse. Um exemplo de perspectiva é o *CVS Repository Perpective* que proporciona a utilização dos recursos do CVS para o controle de versões dos componentes do projeto. Para cada ferramenta integrada ao Eclipse é atribuída uma perspectiva.

Projeto: Os diversos arquivos e componentes criados no Eclipse são incluídos em um projeto. Uma vez criado, o projeto pode ser mantido por um desenvolver ou uma equipe. Caso seja um desenvolvimento em equipe, o projeto dever ser compartilhado.

Diretório de Trabalho (Workspace): é o diretório local onde todos os arquivos do projeto são gravados.

Editor: Local onde o conteúdo dos componentes/arquivos é visualizado e editado. Podem existir múltiplas instâncias de editores dentro de um workbench, proporcionando a visualização de vários códigos simultaneamente. Dependendo do tipo de arquivo que estiver sendo editado (.txt, .gif), o Editor apresenta recursos diferenciados.

View (Visualizador): Apresenta alguma informação na área de trabalho. Isso pode ser uma árvore de diretórios e arquivos, uma descrição da estrutura de uma classe Java.

Plugin: Qualquer componente ou ferramenta completa que possa ser adicionado ao Eclipse seja um simples botão ou uma Perspective completa (com vários Editors e Views).

Controle de Versão no Eclipse

O Eclipse possui integração nativa com o CVS, o que permite aos desenvolvedores realizarem a gerência de configuração dos projetos. Como dito anteriormente, a utilização do CVS se dá através da perspectiva "*CVS Repository*", que é um ambiente gráfico onde são executados os diversos comandos do CVS sem precisar digitar linhas de código. Com esse cliente gráfico para o CVS, torna fácil o gerenciamento de projetos no repositório de versões.

O *Plug-in Team* se conecta ao CVS Server através dos protocolos *pserver* ou *ssh*, permitindo assim, que os membros da equipe de desenvolvimento trabalhem com um conjunto de arquivos fontes sem passar por cima de alterações efetuadas por outros membros da equipe [ECLIPSE05].

Através do Eclipse é possível gerenciar as diferentes versões da aplicação e as revisões dos arquivos fonte. Após a atualização de um arquivo ter sido realizada no repositório, todos os membros da equipe são notificados, para que assim, possam fazer a mesclagem das alterações concorrentes. Existe ainda a possibilidade de controlar diferentes ramos de um mesmo projeto (uma ramificação só para correções de uma versão mais antiga, por exemplo).

Fluxo de trabalho para utilizar o CVS no Eclipse

a) Criar um novo repositório através da perspectiva *CVS Repository Exploring*, informando os dados de conexão com o servidor (*Host*, conta de acesso, tipo de conexão - *pserver* ou ssh).

b) O projeto que será desenvolvido deve ser, então, compartilhado. Para isso deve-se utilizar o plugin *Team / Share Project*, onde será incluído no repositório criado anteriormente. Após a inclusão, o projeto é mostrado no visualizador *Synchronize*. Através deste visualizador é possível realizar o sincronismo (Check-in, check-out, update, edit) dos arquivos/componentes do projeto que estão no repositório.

c) Após compartilhar o projeto no repositório, os desenvolvedores poderão utilizar os diversos recursos do CVS. Por exemplo: criar novas versões de arquivos e projetos presentes no Workspace e no repositório, gerenciar as versões de um recurso, substituição de uma versão corrente por outra.

CAPÍTULO 2 PROCESSO DE GESTÃO DE CONFIGURAÇÃO E MUDANÇAS – PGCM

O processo proposto por este trabalho será definido a partir de um conjunto de atividades, que por sua vez poderão ser divididas em uma ou mais tarefas, devendo a execução das mesmas ser realizada de maneira integral, mesmo que com adaptações para cada projeto. As atividades propostas estão baseadas fundamentalmente no Processo Unificado de Desenvolvimento de Software (USDP), sendo que algumas atividades são complementadas, em termos de funcionalidades, a partir de práticas e recomendações existentes em normas internacionais anteriormente citadas nesta monografia, tais como ISO 12207, ISO 10007, dentre outras. Como todo processo, este que doravante será definido como "Processo de Gestão de Configuração e Mudanças Apoiado por Software Livre – PGCM", servirá como um guia para aqueles que desejam implementar um processo de Gerência de Configuração de Software e Controle de Mudanças apoiado por *Software Livre*.

Como já é de praxe, serão recomendadas as tarefas a serem executadas pelos inúmeros papéis envolvidos em um projeto padrão de desenvolvimento de software, não tendo o objetivo de esgotar a técnica ensinando "como fazer", mas sim dizendo "o que deve ser feito". Além destas recomendações, serão abordados os elementos essenciais para a utilização das ferramentas que servirão para automatizar os processos. Estas ferramentas, como já descrito anteriormente, são ferramentas do tipo *Open Source* (Código Aberto), com licenças compatíveis com o contrato GPL ou compatíveis [GPL05]. Serão considerados, para efeito deste processo e respectivo apoio por parte destas ferramentas, que a infra-estrutura de software adequada encontra-se previamente instalada, configurada e operante. Este trabalho não tem como objetivo servir como um tutorial de instalação e configuração destas ferramentas. O tutorial apresentado posteriormente será apenas um guia prático onde serão mostradas as práticas automatizadas do processo de GCS e CM, e como estas ferramentas interagem com o processo proposto.

O Capítulo 1 serviu como um referencial dos diversos modelos de maturidade, bem como um arcabouço de normas atinentes à Engenharia de Software. Pode-se constatar, a partir do quadro "Comparativo dos padrões e modelos" apresentado no Capítulo 1, que o processo de apoio denominado "Gerência de Configuração e Controle de Mudanças" possui, na maioria das vezes,

tópicos em comum. Estes tópicos variam na maneira como as atividades são abordadas, e em alguns casos, são idênticos. Aproveitando esta similaridade, o PGCM abordará as metodologias comuns de maneira integral, complementando o USDP com o que há de melhor nas demais normas ou modelos de maturidade. Para um perfeito entendimento, como exemplo, considere o processo proposto pela Norma ISO/IEC 12207, nas suas atividades: Implementação do Processo, Identificação da Configuração, Controle da Configuração, Relato da Situação da Configuração, Avaliação da Configuração e Gerência de Liberação e Distribuição. Considere também o proposto pelo USDP: Planejar Configuração do Projeto e Controle de Mudanças, Criar Ambientes para a CM do Projeto, Alterar e Liberar IC, Gerenciar Linhas Bases e Liberações, Monitorar e Relatar o Estado da Configuração e Gerenciar Solicitação de Mudanças. Após uma análise mesmo que superficial destas atividades, percebe-se que em sua essência, as atividades são as mesmas, mudando apenas a abordagem. O “Controle da Configuração” da ISO/IEC 12207 está para “Gerenciar Solicitação de Mudanças” assim como “Monitorar e Relatar o Estado da Configuração” do USDP está para “Relato da Situação da Configuração”. Tal similaridade pode ser encontrada em outras normas, haja vista que em muitos casos as mesmas são aderentes umas às outras.

A análise acima pode ser estendida aos modelos de maturidade. Um exemplo típico é o CMMI e o MPS.BR. As nomenclaturas dos níveis de maturidade de cada modelo são diferentes. Entretanto, nos dois modelos, já no segundo nível, o processo de apoio de GCS e CM é previsto como essencial. Além disso, o MPS.BR é aderente ao CMMI.

O próximo tópico será uma explanação sobre a metodologia de trabalho iterativa e incremental do USDP, no que se refere às suas fases e iterações das disciplinas previstas pelo modelo.

2.1 O PGCM e o USDP

Neste capítulo será definido o PGCM, sempre à luz do USDP. Para tal, será definida de maneira didática a estrutura do USDP, de maneira que fique clara a forte ligação entre as fases do processo, suas disciplinas, seus marcos e os produtos gerados em cada fase, e que, na maioria das vezes, são sérios candidatos a tornarem-se itens de configuração de software. Uma vez tendo sido apresentado o USDP, serão enumerados os itens de configuração mínimos de um projeto de desenvolvimento de software, e finalmente, será apresentado o processo propriamente dito.

O UDSP prevê um ciclo de vida do projeto dividido em 4 fases distribuídas ao longo do tempo: Iniciação, Elaboração, Construção e Transição. Na fase de *Iniciação*, o escopo do produto

está sendo definido. Uma visão geral do sistema é descrita. A fase de *Elaboração* é aquela em que os recursos, atividades e os produtos provenientes do trabalho são planejados, juntamente com a arquitetura do ciclo de vida do projeto. A fase de *Construção* é aquela em que o produto é verdadeiramente implementado, de maneira incremental. Como última fase do ciclo de vida proposto pelo UDSP, a fase de *Transição* é aquela em que versões operacionais do produto são entregues ao cliente. Cada uma destas fases é dividida em iterações, nas quais todo o fluxo de trabalho (disciplinas) pode ser aplicado, iterativamente e na intensidade relativa ao nível de desenvolvimento do projeto. Como pode ser observado na figura 14, a disciplina de Gerência de Configuração aparece durante todo o ciclo de vida do projeto. Surge já na fase de *Iniciação*, a partir do momento em que são elaborados os primeiros documentos do projeto (seleção dos itens de configuração), e se estende até a fase de *Transição*, quando ocorre a entrega do sistema para o cliente.

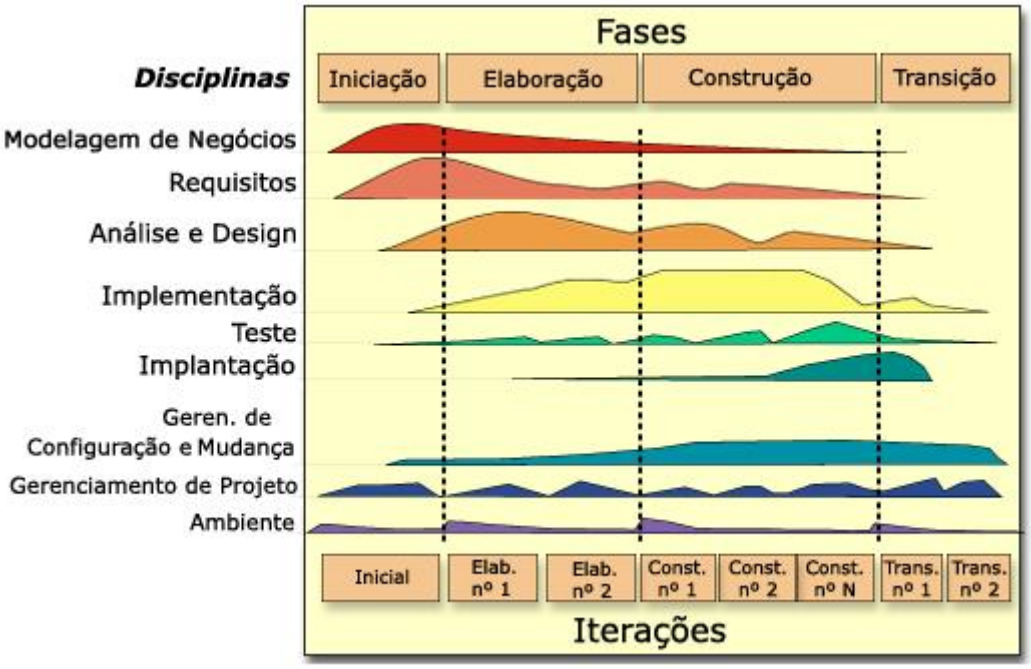


Figura 14 - Fases e Disciplinas do USDP [RUP01]

Como todo processo de desenvolvimento de software, o USDP compreende um conjunto de atividades divididas em estágios. A ordem e a freqüência em que estas atividades ocorrem constituem a diferença fundamental entre os processos de desenvolvimento de software. Para desmistificar o funcionamento iterativo do UDSP, os estágios serão divididos em dois níveis: nível de sistema e nível de item de configuração de software. Os estágios referentes ao nível de sistema

são aplicados primeiro, de forma a definir os requisitos gerais do produto a ser desenvolvido, o projeto e a arquitetura. A organização do sistema em função dos itens de configuração de software também é definida nestes estágios. Na seqüência, os estágios são aplicados ao nível dos itens de configuração (IC) para cada IC. Uma vez completo o desenvolvimento do IC, estágios de nível de sistema adicionais são aplicados de forma a conduzir o processo à entrega final do produto de software. Esta perspectiva torna-se ideal para a definição do PGCM, pois coloca em evidência um dos conceitos essenciais do processo de GCS, ou seja, o Item de Configuração. No quadro seguinte serão indicados os estágios do desenvolvimento do sistema e os produtos gerados pelo trabalho realizado no referido estágio.

Nível	Nº	Estágio	Produto do Trabalho	Fase
Sistema	1	Definição das fronteiras do Sistema	Documento de Visão	Iniciação
	2	Requisitos do Sistema	Especificação do Sistema Base de Requisitos do Sistema Modelo de Casos de Uso	
	3	Projeto (Design) do Sistema	Modelo de Projeto do Sistema	
	4	Planejamento e Análise de Risco	Plano de Desenvolvimento do Projeto Plano de Testes	
IC	5	Requisitos de Software	Especificação de Requisitos Base de Requisitos do Software Modelo de Casos de Uso	Elaboração / Construção
	6	Análise e Projeto do Software	Descrição da Arquitetura do Software Casos de Testes Modelo de Projeto baseado em ferramenta CASE.	
	7	Codificação e Teste de Unidade	Código testado com sucesso	
	8	Aprovação dos IC	IC aprovado	
Sistema	9	Integração do Sistema e Testes	Versão do sistema aprovada e pronta para entrega	

Quadro 6 - Estágios e Produtos do Trabalho [TOMER05]

2.1.1 Estágios do Nível do Sistema

Os seguintes estágios são aplicados ao nível do sistema, como indicado no quadro "*Estágios e Produtos do Trabalho*". Os estágios de 1 a 4 compreendem a fase de *Iniciação*, e são executadas apenas uma vez. Isto não quer dizer, que seus artefatos não serão submetidos à Gerência de Configuração. Pelo contrário, o PMI (Project Management Institute) preconiza, em todas as áreas, o Controle de Mudanças dos seus artefatos, a fim de manter sua integridade. O Estágio 9 é executado a cada iteração, através das fases de *Elaboração e Construção*, em seqüência, dos estágios 5 até o 8 no nível de Item de Configuração (IC).

Estágio Nº 1: Definição das Fronteiras do Sistema

O objetivo deste estágio é prover o entendimento comum sobre o sistema. Estão envolvidos neste estágio o cliente, o desenvolvedor e qualquer outro elemento interessado. O produto deste estágio é um documento completo denominado "Documento de Visão". Contém o propósito do

sistema, objetivos, a descrição do relacionamento com sistemas externos, bem como seus requisitos primários.

Estágio Nº 2: Requisitos do Sistema

O objetivo deste estágio é definir em detalhes os requisitos do sistema. Neste estágio surgem os modelos de caso de uso. Nestes modelos são definidos os requisitos funcionais do sistema, juntamente com as ligações externas do sistema em estudo.

Estágio Nº 3: Projeto (Design) do Sistema

Neste estágio é desenvolvido o projeto da solução do sistema, incluindo a arquitetura e a organização do sistema em termos de itens de configuração de software e hardware. Os Modelos de Análise e de Projeto são os documentos essenciais produzidos neste estágio. Neste momento é definida a alocação dos requisitos do sistema referentes aos IC que atendem a estes mesmos requisitos.

Estágio Nº 4: Planejamento e Análise de Risco

O Plano de Desenvolvimento do Software (PDS) é elaborado neste estágio. Este documento contém o detalhamento das fases do desenvolvimento do software para todos os IC, tendo a forma de um plano de iterações. O plano deve prever, para cada iteração, a forma de mitigação dos riscos prioritários (de maior impacto ou probabilidade), definidos na Análise de Riscos do projeto. Como parte do PDS, temos ainda o Plano de Testes.

Estágio Nº 9: Integração do Sistema e Testes

Este estágio é executado a cada iteração, após todos os quatro estágios do nível de IC terem sido executados para cada IC. O propósito deste estágio é possibilitar a entrega de uma versão operacional do sistema, uma vez tendo sido realizada a validação, a aprovação e integração das versões estáveis de todos os IC do sistema. Este estágio é aquele que de fato caracteriza o **desenvolvimento incremental**, pois possibilita a entrega de versões do sistema cada vez mais funcionais.

2.1.2 Estágios do Nível de Item de Configuração

Os estágios de 5 a 8, conforme indicado no quadro *"Estágios e Produtos do Trabalho"*, são executados separadamente para cada IC. Os produtos deste estágio são produzidos de forma

iterativa (a cada iteração, detalhes adicionais levantados na iteração em questão são adicionados naqueles documentos produzidos nas iterações anteriores, de maneira a refiná-los, mantendo-os atualizados). O 5º estágio de cada iteração deverá ser precedido por um planejamento detalhado da próxima iteração, que definirá de maneira precisa quais serão os IC envolvidos, bem como a seleção dos requisitos do sistema que serão implementados na referida iteração.

Estágio Nº 5: Requisitos do Software

Neste estágio são definidos detalhadamente os requisitos do sistema a serem atendidos na iteração em questão. O produto final deste estágio é a Especificação de Requisitos, normalmente orientada a Casos de Uso. A característica de rastreabilidade de requisitos é desejável, e plenamente obtida a partir da utilização de uma ferramenta CASE (Computer Aided Software Engineering).

Estágio Nº 6: Análise e Projeto do Software

Neste estágio, os requisitos em termos de IC são analisados para gerar ou atualizar o projeto do IC. As múltiplas visões arquiteturais são levadas em consideração para o projeto, pois o projeto de cada IC pode impactar em uma ou mais dessas visões [KRUTCHEN98]. Os IC são definidos a partir dos diagramas da UML (Unified Modeling Language), tais como os Diagramas de Classe, Diagramas de Seqüência, Diagramas de Atividades e de Estados. Estes modelos servem de subsídio para a descrição da arquitetura do software, um dos produtos gerados nesta fase, e também é colocado sob gerência de configuração, assim como os demais.

Estágio Nº 7: Codificação e Teste de Unidade

Esta fase corresponde a implementação do sistema. Os módulos do sistema são codificados e testados no ambiente de desenvolvimento, de acordo com o projeto. O código testado deve ser previamente aprovado, podendo tornar-se uma linha base para novas liberações, ou servindo de subsídio para iterações subseqüentes. O produto deste estágio é o próprio código testado e aprovado.

Estágio Nº 8: Aprovação dos Itens de Configuração

Os IC devem ser validados e aprovados. Os testes devem ser executados com sucesso, conforme o previsto nos planos de teste. Os resultados dos testes devem ser registrados e analisados.

2.1.3 Um exemplo de desenvolvimento iterativo

Para ilustrar a aplicação do USDP, de acordo com as suas fases e disciplinas, o quadro 7 mostra o processo iterativo de desenvolvimento de um produto hipotético, composto por uma amostra de dois IC, sendo desenvolvido através de quatro iterações, a partir da fase de *Elaboração*, sendo a primeira fase a de *Iniciação* (ocorre somente uma vez).

Iniciação		1. Fronteiras do Sistema	
		2. Requisitos do Sistema	
		Revisão dos Requisitos do Sistema	
		3. Projeto (Design) do Sistema	
		4. Planejamento e Análise de Risco	
		Revisão do Projeto do Sistema	
		IC Nº 1	**IC Nº 2**
Elaboração	"n" Iterações	Planejamento detalhado da iteração	
		Revisão do plano de iteração	
		5. Requisitos do Software	5. Requisitos do Software
		Revisão da Especificação do Software	
		6. Projeto do Software	6. Projeto do Software
		Revisão do Projeto do Software	
		7. Codificação e Teste de Unidade	7. Codificação e Teste de Unidade
		8. Aprovação dos IC	8. Aprovação dos IC
		Revisão da Completude da Iteração	
Construção	"n" Iterações	Planejamento detalhado da iteração	
		Revisão do plano de iteração	
		5. Requisitos do Software	5. Requisitos do Software
		Revisão da Especificação do Software	
		6. Projeto do Software	6. Projeto do Software
		Revisão do Projeto do Software	
		7. Codificação e Teste de Unidade	7. Codificação e Teste de Unidade
		8. Aprovação dos IC	8. Aprovação dos IC
		Revisão da Completude da Iteração	

Quadro 7 - Exemplo da aplicação do USDP

2.1.4 Seleção dos Itens de Configuração de Software

Quando um projeto é iniciado, a GCS também deve ser iniciada, de forma a garantir que os IC sejam corretamente identificados, produzidos, armazenados e liberados para utilização durante cada fase do ciclo de desenvolvimento do software. Como já definido anteriormente, o IC é um elemento essencial do processo de desenvolvimento. É constituído de todos os artefatos do projeto que podem ser nomeados e cujas alterações devem ser controladas. A linha base é composta de um IC ou conjunto destes, cuja característica essencial é terem sido revisados e aprovados pela autoridade competente do processo. A seleção dos IC é um fator crítico para o sistema, pois o excesso de IC pode levar o sistema de gerência de configuração e controle de mudanças a uma burocracia desnecessária, e o relaxamento na definição pode levar ao fracasso o sistema. A linha base é gerada para cada IC. É comum o tratamento de uma versão completa do sistema como uma linha base, entretanto, para efeito do PGCM, será adotado o conceito de linha base por IC.

A seleção dos IC, como demonstrado posteriormente a partir da apresentação do processo, é tratada como uma *Atividade* do processo. A exemplo do que prescreve o CMMI, “Identificar itens de Configuração”, ou no modelo proposto pelo IEEE, “Identificação da Configuração do Software”, dentre outros modelos citados no Capítulo 1, a Identificação ou Seleção dos IC é a base para as outras atividades de GCS [NBR12207]. O propósito neste momento e apresentar, de acordo com o USDP, uma sugestão de possíveis artefatos a serem mantidos sob gerência de configuração. O conjunto de artefatos sugerido não tem caráter obrigatório, porém desejável, de forma que a seleção possa estar de acordo com o porte do projeto. Outro aspecto a ser observado, é a aderência ao processo unificado, e consequentemente, ao paradigma de orientação a objetos. A sugestão a seguir apresentada está baseada conforme os artefatos produzidos num ambiente de desenvolvimento sob o paradigma da orientação a objetos. Muitos deles são artefatos propostos pela UML. Outros têm origem na área de garantia da qualidade, tais como Planos de Teste e Casos de Teste, por exemplo.

No quadro 8 são indicados os principais artefatos que podem ser classificados como IC, de acordo com a utilização por disciplina do processo unificado. Aqueles artefatos que aparecem em mais de uma disciplina serão apresentados apenas uma vez.

ARTEFATOS DAS DISCIPLINAS FUNDAMENTAIS		
Disciplina	**Objetivo**	**Artefatos**
Modelagem de Negócios	Apresentar o contexto de negócios do sistema	▪ Visão do Negócio ▪ Modelos de Casos de Uso de Negócios ▪ Modelo de Objetos de Negócios ▪ Casos de Uso de Negócios ▪ Realização de Caso de Uso de Negócios ▪ Documento de Arquitetura de Negócios
Requisitos	Define os recursos e funcionalidades do sistema	▪ Visão ▪ Plano de Gerenciamento de Requisitos ▪ Modelos de Caso de Uso ▪ Especificações Suplementares ▪ Caso de Uso ▪ Especificação de Requisitos ▪ Protótipo de Interface
Análise e Projeto	Apresentar a solução dos problemas propostos nos Requisitos	▪ Prova de Conceito Arquitetural ▪ Documento de Arquitetura de Software ▪ Modelo de Análise ▪ Modelo de Design (Projeto) ▪ Modelo de Implantação ▪ Realização de Casos de Uso ▪ Classes de Análise ▪ Classes de Design (Projeto) ▪ Pacotes de Design ▪ Modelo de Dados e Dicionário de Dados
Implementação	Realiza a solução proposta na Análise e Projeto	▪ Componentes ▪ Subsistema de Implementação ▪ Plano de Integração da Construção ▪ Construção (Build) ▪ Modelo de Implementação ▪ Códigos Fonte ▪ Ferramentas (compiladores, bibliotecas, editores)
Teste	Artefatos desenvolvidos como produto dos testes do sistema (unitário e integração)	▪ Plano de Teste ▪ Casos de Teste ▪ Scripts e Dados de Teste ▪ Resultados de Teste ▪ Guia de Teste ▪ Classes e Componentes de Teste
Implantação	Apresentar informações relacionadas à transição do sistema para a produção	▪ Plano de Implantação ▪ Unidade de Implantação (Contém o sistema testado, pronto para implantação) ▪ Artefatos de Instalação

Quadro 8 - Artefatos das Disciplinas Fundamentais (USDP)

ARTEFATOS DAS DISCIPLINAS DE APOIO		
Disciplina	**Objetivo**	**Artefatos**
Gerenciamento do Projeto	Artefatos associados ao planejamento e a execução do projeto e do processo	▪ Plano de Desenvolvimento de Software ▪ Plano de Iteração ▪ Plano de Garantia da Qualidade ▪ Plano de Métricas ▪ Plano de Gerenciamento de Riscos
Gerenciamento de Configuração e Mudança	Informações relacionadas à disciplina de GCS e CM	▪ Plano de Gerência de Configuração ▪ Registro de Auditoria
Ambiente	Garantir a consistência de todos os artefatos produzidos	▪ Caso de Desenvolvimento ▪ Infra-estrutura de Desenvolvimento ▪ Guia de Modelagem de Negócio e Casos de Uso

Quadro 9 - Artefatos das Disciplinas de Apoio (USDP)

O UDSP prevê a existência de muitos outros artefatos, entretanto, nos quadros anteriores são indicados os considerados essenciais, seja por motivo de manutenção de histórico, ou de controle de versões. Como dito anteriormente, conforme o porte do projeto, alguns artefatos não são necessários, e em conseqüência, não serão colocados sob gerência de configuração.

2.1.5 Método de Promoção das Linhas Bases

Com a evolução do andamento do projeto, a estabilidade das linhas base aumenta. O processo de promoção destas linhas base torna-se necessário, pois o nível de promoção é um atributo que representa a qualidade ou estabilidade da linha base. Os níveis de uma linha base podem ser definidos pelo projeto, de maneira personalizada. Entretanto, o USDP sugere os seguintes níveis de promoção:

- Rejeitada
- Integração Testada
- Sistema Testado
- Aceitação Testada
- Produção

Outros níveis de promoção também podem ser definidos, tais como:

- Inicial
- Compilada
- Testada
- Liberada

Os níveis devem ser ordenados de maneira que representem a progressão da qualidade. A tarefa de alterar o nível de uma linha base chama-se “promover” ou “rebaixar”. O ato de se rebaixar uma linha base normalmente ocorre quando um erro importante é encontrado em uma linha base recém criada. Para evitar que esta linha base seja copiada para a área de trabalho dos desenvolvedores, ela poderá ser rotulada como “rejeitada”, por exemplo.

Para cada projeto, a política de desenvolvimento definirá que uma linha base somente poderá ser utilizada após atingir certo nível de promoção. Esta política garantirá que a equipe de desenvolvimento tenha sempre em mãos uma linha base aceitável.

2.1.6 Definição dos Papéis do Sistema de GCS e CM

Para que qualquer processo seja eficaz, o entendimento do relacionamento entre os diversos papéis envolvidos no referido processo deve ser obtido. É importante salientar que a quantidade de papéis previstos nos processos quase sempre não corresponde à realidade encontrada nas empresas ou corporações. Entretanto, a definição dos papéis objetiva demonstrar o funcionamento cooperativo durante a execução do processo dentro da organização, bem como a efetiva distribuição de suas atividades. Na prática, um papel é desempenhado por uma pessoa ou grupo de pessoas trabalhando em conjunto. Um papel define um comportamento de uma pessoa ou equipe em um dado momento do projeto, sendo que o mesmo possui um conjunto de atividades relacionadas e combinadas em termos de funcionalidades. As atividades, por sua vez, são ligadas aos artefatos que servem de insumo e são gerados pelas respectivas atividades.

O PGCM, define os seguintes papéis que atuam no sistema de GC e CM:

Papéis	Especificação dos papéis definidos no PGCM
Diretor de Gerência de Configuração e Controle de Mudanças	Define e implanta as políticas do processo de Gerência de Configuração e Controle de Mudanças à nível organizacional.
Gerente de Configuração	Disponibiliza e mantém o ambiente e a infra-estrutura geral de GC para a equipe de desenvolvimento do produto.
Solicitante	Qualquer elemento envolvido no projeto que possa propor modificações que afetem seus artefatos.
Gerente de controle de mudanças	Supervisiona o processo de controle de mudanças. o Processo de Gerenciamento de Solicitações de Mudança, que é documentado no Plano CM.
Integrador	Responsável pela integração dos vários artefatos que fazem parte de uma construção do sistema (*Build*). Mantém as Linhas Base (*Baselines*) em um ambiente de integração.
Desenvolvedor	O elemento que recebe a responsabilidade para implementar as alterações nos artefatos, em resposta a uma CR aprovada pelo CCB. Deve manter o estado da CR sempre atualizado
Qualquer Papel	Qualquer elemento que possa efetuar alterações no estado de uma CR
Presidente do CCB	Líder do CCB. Tem poder de decisão em caso de indefinições por parte do CCB.
CCB	Formado por representantes de todas as partes interessadas, incluindo clientes, desenvolvedores e usuários.
Avaliador	Analisa o impacto da aprovação da CR para o projeto.
Gerente de Projeto	É responsável por todo o planejamento e acompanhamento das atividades do desenvolvimento do projeto.
Verificador	É o papel que assume o elemento responsável por determinar quando uma mudança foi realizada corretamente. Corresponde a atividade de Garantia da Qualidade.

Quadro 10 - Papéis do PGCM

2.2 O PGCM

Abaixo segue descrito o Processo de Gestão de Configuração e Mudança – PGCM. As atividades estão agrupadas pelos papéis envolvidos no processo. Para cada atividade estão descritos as tarefas, os insumos (artefatos necessários para a realização das atividades), os artefatos resultantes e a freqüência, ou seja, o momento em que cada atividade deve ser executada dentro do ciclo de vida de desenvolvimento do software. A Figura 15 representa o ambiente de Gerência de Configuração com o Papel do Diretor de Gerência de Configuração, tendo como subordinados os Gerentes de Configuração de cada projeto. Para organizações menores, o Papel de Diretor de

Gerência de Configuração pode ser absorvido pelo papel de Gerente de Configuração, uma vez que ele poderá estabelecer os procedimentos organizacionais, atendendo simultaneamente a todos os projetos.

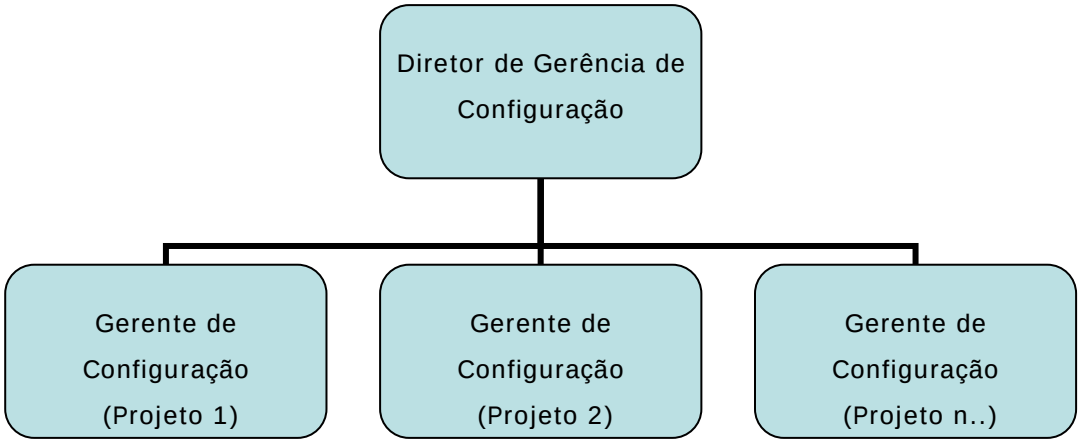


Figura 15 - Ambiente de Gerência de Configuração para grandes Organizações

2.2.1 Papel: Diretor de Gerência de Configuração e de Controle de Mudanças

A1. Atividade: Estabelecer Políticas de Gerenciamento de Configuração para a Organização.

As políticas de gerência de configuração são usadas para padronizar os procedimentos de adoção do processo de gerência de configuração e controle de mudanças, bem como de outros itens relevantes, tais como relatórios e utilização das linhas base. Estas atividades estão relacionadas com a padronização do processo como um todo, definindo diretrizes para a organização, que poderão ser instanciadas para cada projeto.

Tarefas

- Prever orçamento para implantação do Sistema de Gerência de Configuração;
- Definir diretrizes para os Planos de Gerência de Configuração da Organização;
- Definir práticas de Identificação de Configuração e Linhas Base;
- Prever a existência de bibliotecas de Gestão de Configuração distintas, uma para desenvolvimento e outra para manutenção de software [FILHO01], conforme indica a figura 16.

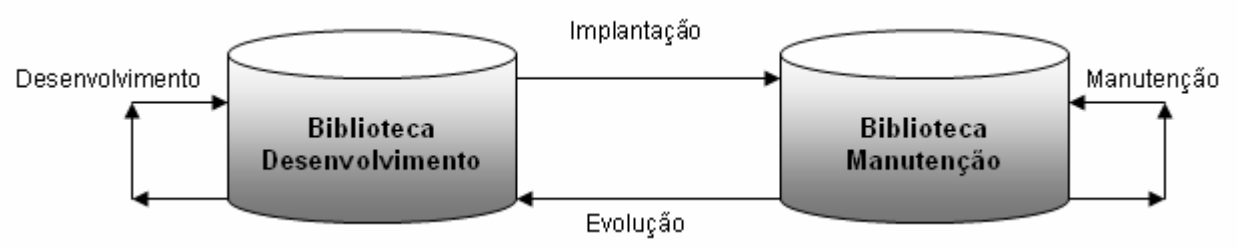


Figura 16 - Bibliotecas de Gestão de Configuração de Software

- Definir diretrizes de armazenamento de mídias e manuais;
- Definir requisitos para os relatórios oriundos do Sistema de Gerência de Configuração;
- Definir ferramentas automatizadas de apoio à Gerência de Configuração;
- Definir o ambiente de hardware do Sistema de Gerência de Configuração.

Insumos

- Processo de Desenvolvimento de Software Padrão da Organização;
- Inventário de Hardware e Software da Organização.

Artefatos Resultantes

- Diretrizes para Planos de Gerenciamento de Configuração da Organização;
- Modelo de Relatório de Estado de Configuração.

Freqüência

As práticas desta atividade precisam ser estabelecidas no início da implantação do Processo de Gerência de Configuração e Controle de Mudanças da Organização. Podem ocorrer ajustes no processo, devendo o mesmo ser revisado, atualizado e difundido na Organização.

A2. Atividade: Preparar Recursos Humanos para o Sistema de Gerência de Configuração.

A existência de membros com conhecimentos apurados das técnicas de Gerência de Configuração constitui um fator essencial para a organização. Com uma equipe bem treinada, o sistema de Gerência de Configuração torna-se mais abrangente e eficiente [CMBOK].

Tarefas

- Definir as equipes do Sistema de Gerência de Configuração;
- Prever orçamento para treinamento;
- Treinar as equipes do Sistema de Gerência de Configuração.

Insumos

- Aprovação da Gerência Executiva da Organização;
- Relação de Recursos Humanos e Qualificações da Organização.

Artefatos Resultantes

- Plano de cursos e estágios para as equipes de Gerência de Configuração;
- Relatório com a definição das equipes de Gerência de Configuração.

A3. Atividade: Estabelecer Processo de Controle de Mudanças dos Projetos da Organização

Esta atividade garante a consistência das mudanças feitas nos projetos, permitindo que os elementos envolvidos sejam comunicados sobre o estado de cada mudança. Além disso, estabelece políticas para avaliação das mudanças, definindo fluxos e estados possíveis de uma solicitação de mudança.

Tarefas

- Estabelecer o Processo de Solicitação de Mudança;
- Estabelecer Diretrizes para formação do Comitê de Controle de Mudança;

- Definir Protocolos e Fluxos de Trabalho para Notificação de Solicitação de Mudança;
- Definir o Modelo de Relatório de Avaliação de Solicitação de Mudança.

Insumos

- Processo de Desenvolvimento de Software Padrão da Organização;
- Diretrizes para Planos de Gerenciamento de Configuração.

Artefatos Resultantes

- Processo de Controle de Mudanças;
- Modelo de Relatório de Avaliação de Solicitação de Mudança.

A4. Atividade: Realizar Auditoria de Configuração dos Projetos da Organização.

Esta atividade é essencial para garantia da qualidade do PGCM. A auditoria física e funcional é realizada, de maneira a garantir a integridade física e lógica dos artefatos (códigos e requisitos, por exemplo).

Tarefas

- Reunir Relatórios de Auditoria de Configuração Física dos Projetos da Organização;
- Reunir relatórios de configuração funcional dos projetos da organização;
- Avaliar os processos;
- Tomar ações preventivas e corretivas, se for o caso.

Insumos

- Processo de Desenvolvimento de Software Padrão da Organização;
- Relatórios de Estado de Configuração dos Projetos;
- Planos de Gerência de Configuração dos Projetos.

Artefato Resultante

- Registro de Auditoria de Configuração;
- Diretrizes de Gerência de Configuração atualizadas.

Freqüência

De acordo com as iterações definidas para cada projeto, ao final de cada iteração.

2.2.2 Papel: Gerente de Configuração

A1. Atividade: Escrever o Plano de Gerenciamento de Configuração (PGC) do Projeto

O Gerente de Configuração deverá documentar todas as atividades relacionadas ao processo de gerência de configuração a serem executadas durante todo o ciclo de vida do projeto, descrevendo como e por quem serão executadas.

Tarefas

- Escrever o PGC;
- Manter atualizado o PGC.

Insumos

- Processo de Desenvolvimento de Software Padrão da Organização;
- Diretrizes para Planos de Gerenciamento de Configuração da Organização;
- Plano de Desenvolvimento de Software do Projeto.

Artefatos Resultantes

- PGC.

Freqüência

O PGC é redigido no início da Fase de Elaboração, sendo posteriormente, revisado no decorrer do ciclo de vida do projeto, no início das Fases de Construção e de Transição ou sempre que necessário.

A2. Atividade: Configurar Ambiente de Gerência de Configuração para o Projeto

Nesta atividade será estabelecido um ambiente onde o projeto possa ser desenvolvido. Repositórios e estruturas de diretórios são definidos, assim como a importação inicial dos itens de configuração deve ser realizada.

Tarefas

- Mapear a Estrutura de Diretórios do Projeto para o Repositório. Ver Apêndice “B”;
- Criar Conjunto Inicial de Itens de Configuração;
- Definir Níveis de Promoção da Linha Base para o projeto. Ver Métodos de Promoção das Linhas Base, no Capítulo 2.

Insumos

- Modelo de Implementação;
- Plano de Gerenciamento de Configuração.

Artefatos Resultantes

- Repositório do Projeto.

A3. Atividade: Realizar Auditoria de Configuração do Projeto

O Gerente de Configuração deve realizar a auditoria física e funcional no âmbito do projeto, garantindo a integridade física e lógica dos artefatos (códigos e requisitos, por exemplo).

Tarefas

- Executar Auditoria de Configuração Física (integridade dos artefatos);
- Executar Auditoria de Configuração Funcional (requisitos funcionais);
- Relatar descobertas e tomar ações corretivas;
- Verificar estados das solicitações de mudanças abertas;
- Relatar ao Diretor de Gerência de Configuração os resultados das avaliações.

Insumos

- PGC;
- Repositório do Projeto.

Artefato Resultante

- Registro de Auditoria de Configuração do Projeto.

Freqüência

Após o teste do sistema e antes do teste de aceitação, ao final de cada iteração, ou quando solicitado pelo Diretor de Gerência de Configuração e Controle de Mudanças.

A4. Atividade: Relatar o Estado da Configuração do Projeto

Esta atividade possui relação direta com a melhoria do processo, pois garante que dados referentes aos itens de configuração bem como as solicitações de mudanças sejam tabuladas, gerando indicadores que possibilitem a melhoria e a tomada de ações preventivas e corretivas.

Tarefas

- Coletar dados e relatar o estado da configuração dos IC;
- Relatar o estado da implementação das CR;
- Fornecer relatórios e controle de defeitos.

Insumos

- PGC;
- Repositório do Projeto.

Artefatos Resultantes

- Relatórios de Métricas do Projeto.

Freqüência

Ao final de cada iteração, conforme o plano de iterações, ou quando solicitado pelo Diretor de Gerência de Configuração e Controle de Mudanças.

A5. Atividade: Preparar Unidade de Implantação do Sistema

Esta atividade consiste na preparação de uma construção completa, com seus executáveis e componentes, de forma que possa ser instalada e operada em algum nó operacional.

Tarefas

- Gerar cópia dos produtos liberados;
- Preparar material de suporte para o usuário;
- Preparar artefatos para instalação.

Insumos

- Plano de Gerenciamento de Configuração;
- Conjunto de Executáveis do Sistema;
- Plano de Implantação;
- Modelo de Implantação;
- Artefatos de Instalação;

- Notas de Liberação;
- Materiais de Treinamento;
- Material de Suporte para o Usuário;
- Repositório do Projeto.

Artefato Resultante

- Unidade de Implantação do Sistema.

Freqüência

A cada construção do sistema.

2.2.3 Papel: Solicitante

Este papel representa o elemento envolvido no projeto que tem a permissão de submeter novas solicitações de mudanças ao Sistema de Controle de Mudanças. Qualquer pessoa envolvida no projeto poderá realizar esta atividade, seja ela gerente, desenvolvedor ou cliente.

A1. Atividade: Enviar Solicitação de Mudança (CR: *Change Request*)

Esta atividade registra as solicitações de mudanças de um IC, mantendo assim um histórico das mudanças realizadas no projeto.

Tarefas

- Preencher e submeter a Solicitação de Mudança (CR);

Insumos

- Não Possui.

Artefato Resultante

- Solicitação de Mudança (CR).

Freqüência

Toda vez que um artefato, que está sob Gerência de Configuração, precisar ser modificado, o solicitante terá que fazer uma Solicitação de Mudança (CR).

2.2.4 Papel: Gerente de Controle de Mudanças

O Processo de Controle de Mudanças poderá ser observado de maneira mais detalhada no Apêndice "C"

A1. Atividade: Estabelecer Processo de Controle de Mudanças do Projeto

Esta atividade garante a consistência das mudanças feitas nos projetos, permitindo que os elementos envolvidos sejam comunicados sobre o estado de cada mudança. Além disso, garante a verificação do impacto e custo da modificação, que são fatores essenciais para a aprovação ou rejeição de um pedido de mudanças.

Tarefas

- Adequar o Processo de Controle de Mudança da Organização ao projeto;
- Estabelecer o Comitê de Controle de Mudanças do projeto;
- Preparar as ferramentas de Controle de Mudanças com informações do projeto.

Insumos

- Processo de Controle de Mudanças da Organização;
- Plano de Desenvolvimento do Projeto;
- PGC.

Artefato Resultante

- Processo de Controle de Mudanças do Projeto.

Freqüência

As práticas de Controle de Mudança do Projeto são estabelecidas no início da Fase de Elaboração, e verificadas novamente ao longo do ciclo de vida do projeto, no início das Fases de Construção e Transição.

A2. Atividade: Avaliar CR

Quando uma nova CR der entrada no sistema de controle de mudanças, a mesma deve ser aprovada ou recusada. Para que seja aprovada, uma avaliação preliminar deve ser realizada (avaliação de custos, riscos, prioridade), de forma a verificar se está dentro do escopo da liberação atual.

Tarefas

- Programar Reunião de Revisão do CCB;
- Recuperar CR para Avaliação;
- Revisar CR Enviadas e preencher o Relatório de Avaliação de Solicitação de Mudança.

Insumos

- Modelo do Relatório de Avaliação de Solicitação de Mudança;
- CR.

Artefatos Resultantes

- Relatório de Avaliação de CR;
- CR avaliada.

Freqüência

Após a criação de uma linha base de um IC e sempre que uma nova CR der entrada no Sistema de Gerência de Controle de Mudanças. Alterações anteriores à criação da linha base não necessariamente necessitam de controle de mudanças.

A3. Atividade: Confirmar CR Duplicada ou Rejeitar uma CR

Quando há suspeita de envio de uma CR cujo conteúdo diz respeito a uma solicitação já cadastrada, ou a mesma encontra-se inválida, o Gerente de Controle de Mudanças deve certificar-se (no caso da duplicidade) ou solicitar maiores informações a quem a submeteu ao sistema.

Tarefas

- Recuperar o Formulário de CR;
- Confirmar Duplicação ou Validade CR;
- Atualizar o estado da CR.

Insumos

- Uma nova CR.

Artefato Resultante

- CR aprovada, duplicada ou rejeitada.

2.2.5 Papel: Integrador

A1. Atividade: Criar Espaços de Trabalho de Integração

Após a criação do repositório do projeto, o Integrador cria o ambiente onde será integrado um conjunto de itens de configuração (IC) que farão parte de uma nova construção (build) ou de uma liberação (release) do sistema. O espaço de trabalho de integração é semelhante ao Espaço de Trabalho de Desenvolvimento, descrito na Atividade “Criar Espaço de Trabalho de Desenvolvimento” do Papel Desenvolvedor. O Espaço de Trabalho de Integração é o local onde os componentes que foram construídos e testados separadamente são integrados, de maneira que possam trabalhar juntos, como um produto.

Tarefas

- Criar o Espaço de Trabalho de Integração.

Insumos

- Plano de Iteração;
- Documento de Arquitetura do Software;
- Repositório do Projeto.

Artefato Resultante

- Espaço de Trabalho de Integração.

Freqüência

Esta atividade deve ser executada no início do projeto e no início de cada iteração do desenvolvimento do sistema. A integração contínua é parte da política de desenvolvimento iterativo.

A2. Atividade: Criar Linha Base (Baseline)

As Linhas Base são criadas nos Espaços de Trabalho de Integração, onde vão conter os itens de configuração mais atuais e que servirão de base para as fases posteriores do projeto. A criação da linha base deve ser de acordo com as políticas de gerência de configuração do projeto.

Tarefas

- Criar uma nova linha base.

Insumos

- Repositório do projeto;
- Espaço de trabalho de integração.

Artefato Resultante

- Espaço de trabalho de integração.

Freqüência

Esta atividade ocorre da seguinte maneira:

a) Esta atividade deve ser executada no início do projeto e durante as primeiras iterações, mesmo que estas iterações não produzam construções (builds) do sistema.

b) No final de cada iteração do desenvolvimento do sistema.

A3. Atividade: Promover Linha Base (Baseline)

Esta atividade consiste em atribuir um rótulo que indique o nível de maturidade, estabilidade e qualidade da Linha Base. Uma vez rotulada, a linha base pode ser então disponibilizada para uma liberação (release) ou servir de base para as fases posteriores do projeto.

Tarefa

- Escolher o rótulo apropriado para a Linha Base, com base na política de promoção de linhas base. Ver Método de Promoção de Linhas Base, no Capítulo 2.

Insumos

- Repositório do projeto;
- Plano de Iteração;
- Espaço de trabalho de integração.

Artefato Resultante

- Repositório do projeto;
- Espaço de trabalho de integração.

Freqüência

Esta atividade deve ser contínua, ou seja, realizada durante todo o ciclo de desenvolvimento do software, conforme o plano de iteração do projeto.

2.2.6 Papel: Desenvolvedor

A1. Atividade: Criar Espaço de Trabalho de Desenvolvimento

O espaço de trabalho de desenvolvimento é um local privado onde os desenvolvedores podem armazenar os artefatos atuais do projeto e assim realizarem as alterações necessárias. Para cada membro da equipe deve ser criado um Espaço de Trabalho de Desenvolvimento. Geralmente, um projeto terá uma área de trabalho compartilhada (Espaço de Trabalho de Integração) e vários espaços de trabalho de desenvolvimento. O espaço de trabalho de desenvolvimento permite a um desenvolvedor trabalhar nas atividades que lhe foram destinadas e de maneira isolada.

Tarefa

- Criar Espaço de Trabalho de Desenvolvimento.

Insumos

- Repositório do Projeto.

Artefato Resultante

- Espaço de Trabalho de Desenvolvimento.

Freqüência

Esta atividade deve ser realizada continuamente, ou seja, durante todo o ciclo de vida do projeto e sempre que novo integrante aderir ao projeto.

A2. Atividade: Atualizar Espaço de Trabalho de Desenvolvimento

A realização desta atividade assegura aos membros da equipe do projeto que os artefatos presentes nos seus Espaços de Trabalho de Desenvolvimento são os mais recentes. Após atualizar o Espaço de Trabalho de Desenvolvimento, o desenvolvedor poderá visualizar e alterar qualquer um destes artefatos gravados.

Tarefas

- Atualizar Espaço de Trabalho de desenvolvimento.

Insumos

- Espaço de Trabalho de Integração;
- Repositório do Projeto.

Artefato Resultante

- Espaço de Trabalho de desenvolvimento atualizado.

Freqüência

Dependerá da freqüência de alterações dos artefatos. Por isso, esta atividade deverá ser realizada diariamente quando for iniciada uma tarefa de desenvolvimento.

A3. Atividade: Fazer Mudanças

Após a criação do repositório do projeto e dos espaços de trabalho (de desenvolvimento e de integração), os desenvolvedores poderão fazer as mudanças solicitadas nas CR's.

Operações de *check-in* e *check-out* são executadas para que os IC possam ser alterados e disponibilizados aos outros membros da equipe. O *check-in* do IC deve ser realizado após a aprovação do Integrador.

Tarefas

- Fazer check-out dos artefatos relacionados a CR;

- Implementar as mudanças dos IC;
- Fazer testes unitários para verificação das mudanças;
- Encaminhar os IC modificados ao Grupo de Garantia da Qualidade, para aprovação. Ver Apêndice “C” - Processo de Controle de Mudanças”.
- Após aprovação, fazer check-in dos IC modificados.

Insumos

- Solicitação de Mudança (CR);
- Espaço de trabalho de desenvolvimento atualizado;
- IC a serem alterados.

Artefato Resultante

- IC modificados;
- Espaço de Trabalho atualizado.

Freqüência

Esta atividade deve ser realizada continuamente, ou seja, durante todo o ciclo de vida do projeto.

A4. Atividade: Liberar Mudanças

Após as mudanças terem sido testadas e aprovadas pelo grupo de Garantia da Qualidade, os IC alterados devem ser liberados do Espaço de Trabalho do Desenvolvedor para o Espaço de Trabalho de Integração. Assim todos os membros da equipe terão acesso à nova versão do artefato, através de uma nova linha base.

Outras mudanças no IC realizadas por outros desenvolvedores devem ser mescladas antes da liberação final do artefato. A meta principal desta atividade é fazer com que os desenvolvedores

criem e testem seu trabalho nas áreas de desenvolvimento, com base no trabalho incluído nas linhas base estáveis, antes de liberá-lo para o espaço de trabalho de integração.

A liberação é concluída quando todas as mudanças foram submetidas ao check-in e assim estando todos liberados para que outros desenvolvedores possam utilizá-los

Tarefas

- Preparar mudanças para liberação;
- Liberar mudanças;
- Atualizar estado da Solicitação de Mudança (CR).

Insumos

- Espaço de Trabalho do Desenvolvedor;
- Itens de Configuração a serem liberados.

Artefato Resultante

- Espaço de Trabalho de Integração atualizado;
- Solicitação de Mudança (CR) atualizada.

Freqüência

Esta atividade deve ser realizada continuamente, ou seja, durante todo o ciclo de vida do projeto.

2.2.7 Papel: Qualquer Papel

A1. Atividade: Atualizar Solicitação de Mudança (CR)

As alterações na Solicitação de Mudança (CR) podem ser de estado, datas, motivos da mudança, alteração dos responsáveis do projeto e/ou da mudança. Qualquer papel pertencente ao processo (Gerente de Controle de Mudanças, Solicitante, Desenvolvedor, por exemplo) pode alterar

a Solicitação de Mudança (CR), desde que faça parte do fluxo previsto no Processo de Controle de Mudanças do projeto e que a mesma (CR) esteja sob sua responsabilidade no momento do ciclo de vida da CR.

Tarefas

- Recuperar o formulário de Solicitação de Mudança;
- Atualizar os dados da Solicitação de Mudança.

Insumos

- Solicitação de Mudança (CR).

Artefato Resultante

- Solicitação de Mudança (CR) atualizada.

Freqüência

Esta atividade deve ser executada toda vez que os dados da Solicitação de Mudança (CR) necessitarem serem alterados.

CAPÍTULO 3 ESTUDO DE CASO

3.1 Introdução

Com o objetivo de avaliar a aplicabilidade das atividades e papéis definidos no PGCM em projetos de desenvolvimento de software, este foi submetido a uma Empresa Pública e a um Órgão de Segurança Nacional do Governo Federal brasileiro. Esta avaliação tem o objetivo primordial de avaliar o processo e verificar as sugestões quanto a sua real aplicação em um ambiente corporativo. Devido à pequena amostra da avaliação (5 avaliações), não se pretende levantar dados estatísticos quanto à aplicação do processo nas organizações, uma vez que nas organizações avaliadoras não há um processo de gerência de configuração definido e implantado.

A metodologia consistiu na apresentação do processo proposto a um grupo de pessoas, na sua grande parte gerentes das áreas de sistemas das organizações. Após a apresentação, cópia do processo e seus apêndices foram entregues aos grupos, para que estes pudessem realizar a avaliação.

Ao final da avaliação foram entregues aos autores do processo PGCM os Questionários de Avaliação respondidos, onde foram registradas as críticas e as sugestões para a melhoria do processo proposto. Por fim, os dados da avaliação foram tabulados, sendo executadas as mudanças necessárias no PGCM.

3.2 Metodologia

Os estudos de caso foram executados seguindo 04 (quatro) etapas, que são descritas abaixo. Para cada etapa estão descritos os seus objetivos, os recursos utilizados e os resultados obtidos.

1ª etapa – Apresentação do PGCM.

Objetivos:

a) Sensibilizar os gerentes das organizações da importância da Gerência de Configuração e Mudança;

b) Buscar patrocínio, ou seja, apoio dos gerentes das organizações no sentido de avaliarem o processo proposto;

c) Definir os grupos avaliadores do PGCM.

Recursos utilizados:

a) Apresentação, em forma de slides, abordando os seguintes assuntos: Histórico do Projeto, Objetivo do Estudo de Caso, Etapas do Estudo de Caso, Conceito de GC, Finalidade da GC, Benefícios da GC, USDP - Processo Unificado, Atividades Críticas de GC, Papéis envolvidos, Ferramentas de GC, PGCM, Exemplo e Avaliação do PGCM;

b) Questionário de Avaliação da Empresa e do Perfil do Avaliador;

c) Questionários de Avaliação do PGCM;

d) O PGCM e os seus apêndices.

Resultados obtidos:

a) Apoio dos gerentes das organizações;

b) Lista dos responsáveis pela avaliação (grupo de avaliadores);

c) Estratégia para realização da avaliação;

d) Entrega dos Questionários de Avaliação.

2ª Etapa - Avaliação do PGCM

Objetivos:

a) Executar a estratégia de avaliação do PGCM;

b) Avaliar o PGCM quanto à aplicabilidade e adequação à realidade da organização;

c) Levantamento de informações quanto ao Controle de Configuração e Mudanças executado nas organizações.

Recursos utilizados:

a) Questionário de Avaliação da Empresa e do Perfil do Avaliador;

b) Questionários de Avaliação do PGCM;

c) O PGCM e os seus apêndices.

Resultados obtidos:

a) Os PGCM e os seus apêndices avaliados;

b) Questionário de Avaliação do PGCM preenchido.

3ª Etapa – Análise dos resultados da Avaliação.

Objetivos:

a) Gerar relatório estatístico da avaliação.

b) Extrair sugestões de melhoria do PGCM;

c) Analisar se existe necessidade de mudanças do PGCM.

Recursos utilizados:

a) Processo avaliado;

b) Questionários de avaliação preenchidos;

c) Planilha Estatística da Avaliação.

Resultados obtidos:

a) Relatório estatístico da avaliação do PGCM;

b) Definição dos ajustes do PGCM.

4ª Etapa – Executar Mudanças no PGCM.

Objetivos:

Modificar o PGCM para atender as considerações feitas pelos grupos avaliadores.

Recursos utilizados:

a) Processo avaliado;

b) Questionários de avaliação preenchidos;

c) Relatório estatístico da Avaliação;

d) Definição dos ajustes do PGCM.

Resultados obtidos:

- PGCM ajustado.

3.3 Questionários de Avaliação

Com o objetivo de registrar a avaliação do PGCM, foram entregues 03 (três) questionários aos grupos avaliadores. Seguem descritas abaixo, as finalidades de cada questionário e as suas características:

a) Questionário de Avaliação da Empresa e do Perfil do Avaliador - Os objetivos deste questionário são: obter informações do perfil do avaliador, quanto à sua área de atuação e ao papel exercido na empresa; obter informações sobre a Empresa, quanto à existência de um processo de Gerência de Configuração e Mudança. No Apêndice “D” deste trabalho encontra-se o modelo deste questionário.

b) Questionário de Avaliação do PGCM: Tem como objetivo ser o meio de registro da avaliação de cada item do Processo de Gerência de Configuração e Mudança. À medida que os avaliadores analisaram cada item do PGCM, valores, quanto à aplicabilidade, foram atribuídos. Considerações sobre os itens avaliados e sobre os valores atribuídos também foram registradas neste questionário. Os valores atribuídos aos itens foram: (D) DESNECESSÁRIO, (P) POUCO ÚTIL, (U) ÚTIL e (M) MUITO ÚTIL. No Apêndice “E” deste trabalho encontra-se o modelo deste questionário.

c) Questionário de Avaliação do Processo de Controle de Mudanças: Como forma de avaliar também o Processo de Controle de Mudanças proposto por este trabalho, os grupos avaliadores preencheram este questionário. A estrutura é semelhante ao do Questionário de Avaliação do Processo de Controle de Mudanças, onde cada item avaliado recebeu um valor e considerações. No Apêndice "F" deste trabalho encontra-se o modelo deste questionário.

3.4 Estratégias de Avaliação

As avaliações foram realizadas por avaliadores de perfis e departamentos diferenciados. A estratégia adotada por ambas as organizações, foi de realizar a avaliação de forma departamental, ou seja, cada departamento avaliou o PGCM de acordo com a sua realidade. Na Empresa Pública, participaram 03 (três) departamentos e no Órgão de Defesa brasileiro teve a participação de 02 (dois) departamentos, sumarizando 05 (cinco) avaliações do PGCM.

3.5 Características das Empresas e dos Avaliadores

Com o preenchimento do Questionário de Avaliação da Empresa e do Perfil do Avaliador, puderam-se extrair as características das organizações avaliadoras, quanto aos seguintes aspectos:

a) Existência de um Processo de Gerência de Configuração: Todos os departamentos avaliados declaram que não executam este tipo de processo nos seus departamentos, exceto o departamento de produção da Empresa Pública.

b) Utilização de Ferramentas de Gerência de Configuração e Mudanças: Todos os departamentos avaliados afirmaram que utilizam este tipo de ferramentas, mesmo sem terem definido um processo de Gerência de Configuração e Mudanças. As ferramentas utilizadas são, na maioria, para controle de versões, como CVS, Subversion, Microsoft Visual Source Safe. O Bugzilla (ferramenta de controle de mudanças) somente é utilizado pelo Órgão de Defesa brasileiro. A ferramenta HP Service Desk é utilizada pelo departamento de produção da Empresa Pública para gerenciar mudanças, configurações e incidentes.

c) Plano de implantar um processo de Gerência de Configuração e Mudanças: Ambas as organizações afirmaram que estão realizando estudos para a implantação deste processo, mas de forma padronizada para todos os departamentos envolvidos.

d) Fatores que impedem a implantação de um Processo de Gerência de Configuração: Dentre os motivos indicados no questionário os fatores que mais impedem a implantação deste processo são os fatores Culturais e Qualificação dos Recursos Humanos. Foi também citado, só que uma única vez, os fatores Organizacionais e Políticos/Estratégicos.

e) Papel exercido pelo avaliador na organização: Dentre os papéis presentes no PGCM, a maioria dos avaliadores responderam que se enquadram como Gerentes de Configuração e Gerentes de Mudanças, havendo a possibilidade de serem também membros de CCB, Solicitantes e Implementadores.

3.6 Análise dos resultados da avaliação do PGCM

Após a entrega dos questionários pelos grupos avaliadores, os valores atribuídos a cada item do PGCM foram consolidados em uma planilha. Os resultados seguem apresentados abaixo. Esta análise se dará sobre dois aspectos: quanto às atividades e quanto aos papéis presentes no PGCM.

3.6.1 Quanto às atividades

Conforme apresentado na Tabela 1 - Avaliação das atividades do PGCM, 90 atividades foram avaliadas onde obtiveram um alto índice de aceitação. Foram 97,78% das atividades classificadas como útil ou muito útil. Somente 2,22% foram consideradas Pouco Úteis e nenhuma foi considerada Desnecessária.

Tabela 1 - Avaliação das atividades do PGCM

Avaliação do PGCM - Referente às Atividades

Avaliação	Quantidade de itens	Percentual por item	Percentual por Nível de Aceitação	Nível de Aceitação
Desnecessário (D)	0	0	2,22	Inaceitável
Pouco útil (P)	2	2,22		
Útil (U)	51	56,67	97,78	Aceitável
Muito Útil (M)	37	41,11		
TOTAL	90	100	100	

O gráfico da Figura 17 - Comparativo dos valores obtidos na avaliação das atividades do PGCM apresenta uma visão da diferença entre os valores do nível INACEITÁVEL (D e P) e os valores do nível ACEITÁVEL (U e M).

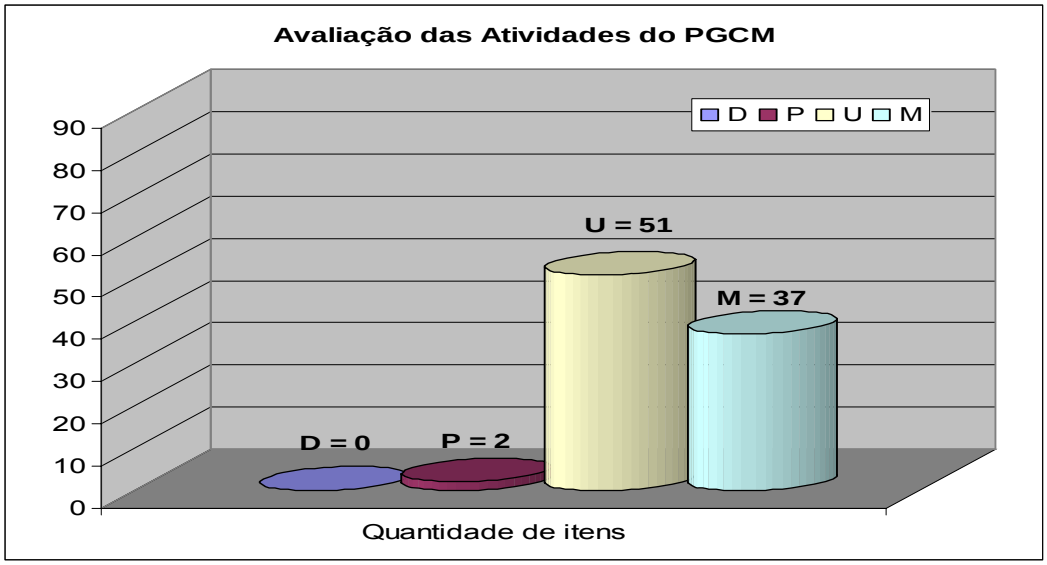


Figura 17 - Comparativo dos valores obtidos na avaliação das atividades do PGCM

3.6.2 Quanto aos papéis

A Tabela 2 - Avaliação dos papéis do PGCM apresenta o resultado da avaliação dos papéis do Processo de Gerência de Configuração e Mudanças. Os cinco avaliadores consideraram aceitáveis todos os papéis definidos no PGCM, com o nível Aceitável em 100%. Os papéis com os maiores índices de aceitação foram o de Gerente de Configuração e o de Gerente de Controle de Mudanças, com 80% e 100% respectivamente. O papel Qualquer Papel foi considerado pela maioria dos avaliadores como útil, pelo fato de aglomerar um conjunto de papéis que participam do Processo de Gerência de Configuração e Mudanças (Solicitante, Analista de Sistemas, Gerente de Projeto).

Tabela 2 - Avaliação dos papéis do PGCM.

Avaliação do PGCM - Referente aos Papéis

Papel	Nível Inaceitável				Nível Aceitável			
	D		P		U		M	
	Qtd	%	Qtd	%	Qtd	%	Qtd	%
Gerente de Configuração	0	0	0	0	1	20	4	80
Gerente de Controle de Mudanças	0	0	0	0	0	0	5	100
Integrador	0	0	0	0	3	60	2	40
Qualquer Papel	0	0	0	0	4	80	1	20
Total por valor	0	0	0	0	8	40	12	60
Total de itens por Nível de Aceitação	0				20			
Percentual total por Nível de Aceitação	0				100			

A Tabela 3 - Avaliação dos papéis do PCM apresenta o resultado da avaliação dos papéis do Processo de Controle de Mudanças. Os cinco avaliadores também consideraram aceitáveis todos os papéis definidos no PCM, com 96,43% de aceitação. O único papel que foi considerado Pouco útil foi o do Avaliador. O avaliador do PGCM considerou que as atividades atribuídas a este papel devem ser executadas pelo Gerente do Projeto. Os papéis que obtiveram maior índice de aceitação foram: o Comitê de Controle de Alterações (CCB), o Implementador, o Solicitante e o Gerente de Projeto. Todos os avaliadores considerados estes papéis muito úteis.

Tabela 3 - Avaliação dos papéis do PCM.

Avaliação do PCM - Referente aos Papéis

Papel	Nível Inaceitável				Nível Aceitável			
	D		P		U		M	
	Qtd	%	Qtd	%	Qtd	%	Qtd	%
Presidente do Comitê de Controle de Alterações (CCB)	0	0	0	0	3	75	1	25
Comitê de Controle de Alterações (CCB)	0	0	0	0	0	0	4	100
Avaliador	0	0	1	25	1	25	2	50
Implementador	0	0	0	0	0	0	4	100
Solicitante	0	0	0	0	0	0	4	100
Gerente de Projeto	0	0	0	0	0	0	4	100
Verificador	0	0	0	0	2	50	2	50
Total por valor	0	0	1	3,57	6	21,43	21	75,00
Total de itens por Nível de Aceitação	**1**				**27**			
Percentual por Nível de Aceitação	**3,57**				**96,43**			

3.7 Ajustes no PGCM

Ao avaliar o PGCM, inúmeras observações foram realizadas. A maioria destas observações diz respeito a comparações entre os papéis existentes na empresa e aqueles existentes no PGCM. De maneira simplificada, o Gerente de Projeto foi indicado inúmeras vezes como responsável por uma grande quantidade de atividades definidas no PGCM. Outras sugestões ficaram focadas na melhor descrição dos papéis.

Quanto a modificações dos papéis, após a avaliação do PGCM surgiu a necessidade da definição de um papel hierarquicamente superior ao do Gerente de Configuração e ao Gerente de Controle de Mudanças. Este papel foi denominado "Diretor de Gerência de Configuração e Controle de Mudanças" e absorveu algumas tarefas de nível organizacional que estavam sob responsabilidade de elementos de nível de projeto. Não houve indicação de novas atividades e novas tarefas por parte dos avaliadores, fator que indica a completude do processo apresentado.

3.8 Lições Aprendidas

Este estudo de caso proporcionou a execução da avaliação de um processo de software, submetendo-o a um conjunto de profissionais experientes em definições de processos e em desenvolvimento de sistemas, obtendo um conjunto de considerações a respeito de cada um dos itens presentes no PGCM.

A execução das atividades definidas na metodologia deste estudo de caso proporcionou aos autores deste trabalho uma visão prática de como um processo pode ser adequado à realidade de uma empresa que pretende implantar ou melhorar o seu Processo de Desenvolvimento de Software.

Além disso, pôde ser observado o grau de importância dado ao Processo de Gerência de Configuração nas organizações. Apesar de não possuírem um processo de gerência de configuração definido, ambas as organizações estão realizando estudos para tal finalidade. O objetivo é que esse processo atenda a toda Empresa e de forma padronizada.

Ambas as organizações haviam adotado estratégias semelhantes às de muitas empresas, ou seja, definem primeiro a ferramenta e depois o processo. Com a execução deste estudo de caso um dos avaliadores fez a seguinte consideração: "A apresentação elucidou muitos conceitos sobre o processo, uma vez que o domínio de ferramentas por si só não garante a maturidade necessária".

Outro aspecto observado foi a maturidade das organizações quanto ao assunto. O conhecimento a respeito deste tipo de processo limita-se a poucas pessoas, na maioria gerentes.

CAPÍTULO 4 CONCLUSÃO

A avaliação dos processos de Gerência de Configuração e Controle de Mudanças propostos por este trabalho proporcionou o levantamento de algumas questões importantes. A primeira delas, oriunda das observações realizadas pelos avaliadores do processo, é quanto ao acúmulo de papéis designado ao Gerente de Projetos. Este acúmulo torna claro que as organizações não possuem atividades e tarefas bem definidas, uma vez que as atividades concernentes à disciplina de Gerência de Configuração ficam sob responsabilidade do Gerente de Projetos. Não há papéis bem definidos, assim como não há uma tramitação vertical de informações referentes às atividades aqui propostas, ou seja, relatórios pertinentes ao estado da configuração dos projetos não fluem entre o Gerente de Projeto e a Gerência Executiva. Em um sentido mais amplo, confirmado pelas informações prestadas na avaliação, afirma-se a inexistência de um processo definido nas organizações avaliadoras do processo. O PGCM, a partir do momento que define políticas organizacionais para gerência de configuração e controle de mudanças que possam ser instanciadas para os projetos da organização, estabelece os princípios para a definição de papéis, alterando a cultura organizacional de que o Gerente do Projeto responde pela gerência de configuração e controle de mudanças.

Outra questão importante é o uso das ferramentas de apoio. Constata-se, a partir do perfil dos avaliadores, que ambas as organizações já possuem definidas ou estão avaliando ferramentas de apoio à Gerência de Configuração e Controle de Mudanças. Tal constatação mais uma vez afirma a idéia de que as ferramentas são definidas antes dos processos, o que de certa forma, caracteriza uma inversão lógica, uma vez que o ideal é a definição dos processos, e a conseqüente adoção de ferramentas que possam suportá-los. Neste contexto, a elaboração do PGCM apoiado por ferramentas de Software Livre vem preencher uma lacuna na área de Engenharia de Software, especificamente na disciplina de Gerência de Configuração e Controle de Mudanças, que é justamente a definição de um processo apoiado por ferramentas de suporte. Não se pode, no entanto, omitir a característica das ferramentas no que diz respeito ao tipo de licença. Em ambas as organizações são utilizadas ou há planos para adoção de ferramentas livres para gerência de configuração e controle de mudanças, confirmando assim o acerto na escolha das ferramentas definidas no PGCM.

O aspecto cultural das organizações avaliadoras também é um fator que deve ser salientado, pois é essencial para a implantação de processos de Engenharia de Software. Por este motivo, foram

realizadas apresentações do processo nos órgãos avaliadores, sendo observado o interesse pelo assunto, e em contrapartida, evidenciando a falta de conhecimento técnico para implantação do processo. Notou-se também a necessidade de treinamento na área de gerência de configuração e controle de mudanças, elemento que foi apontado como fator de impedimento para implantação da disciplina por parte de alguns avaliadores. Por este motivo, foram definidas tarefas que tratam do treinamento dos recursos humanos, sob responsabilidade do Diretor de Gerência de Configuração e Controle de Mudanças, e consequentemente, da direção geral da organização.

Quanto à aderência aos padrões e normas a que se propõe este processo, há de se deixar claro o seguinte: um processo define o quê deve ser feito, não como fazê-lo. Portanto, este trabalho não detalha os padrões na área de controle de versões, tais como Teste de Regressão [BERCZUK02], por exemplo. Neste contexto, o PGCM definiu com precisão uma grande quantidade de artefatos presentes na filosofia de desenvolvimento definida pelo UDSP, conforme suas disciplinas e etapas. Além disso, definiu atividades e tarefas que conjuntamente atendem o previsto como essencial nas normas apresentadas neste processo, cobrindo desde o planejamento até a auditoria do processo, fator que pôde ser confirmado nos resultados de avaliação pela ausência de sugestões de novas atividades ou tarefas.

Ainda no que diz respeito à questão do processo em si, verifica-se o quão importante é sua definição como padrão a ser adotado pela organização. Ambas as organizações avaliadoras encontram-se em busca da melhoria do processo de desenvolvimento de software, sendo de fundamental importância o estabelecimento do processo de apoio definido pelo PGCM. Deixou-se claro que o processo padrão poderá ser instanciado para diversos tipos de organização, uma vez que o mesmo foi avaliado por organizações distintas e de diferente porte, além de ter sido elaborado a partir de padrões e normas aceitos atualmente no mercado de tecnologia da informação. Este estudo poderá servir de referência para as empresas brasileiras que pretendem definir um processo de gerência de configuração e controle de mudanças, facilitando a tarefa de consolidação dos aspectos abordados nas diversas normas, padrões e modelos de maturidade.

Ao desenvolver este projeto, apresentando ferramentas de software livre para suporte aos processos propostos, foram surgindo questionamentos referentes às melhores práticas e padrões de gerência de configuração, bem como questões técnicas sobre quais ferramentas melhor suportariam estes padrões. Tais questões fundamentam-se especialmente no que diz respeito aos itens abaixo:

a) Completude das ferramentas livres, especialmente nos requisitos de emissão de relatórios de estado de configurações e de controle de mudanças;

b) Possibilidade de integração das ferramentas, de forma que possam, em conjunto, atender as perspectivas gerenciais e de desenvolvimento descritas nos itens 1.1.5 e 1.1.6 deste trabalho;

c) Melhores práticas no estabelecimento de liberações e linhas base.

d) Melhores práticas no controle de versões e integração de sistemas desenvolvidos por grandes equipes de desenvolvimento.

As respostas para estas indagações encontram-se fora do escopo deste trabalho, podendo seguramente servir de motivação para novos trabalhos na área.

REFERÊNCIAS BIBLIOGRÁFICAS

[ASKLUND02]
ASKLUND, U.; BENDIX, L. **A Study on Configuration Management in Open Source Software Projects**. IEE Proceedings – Software, v. 149, n. 1, p. 40–46, fevereiro 2002. Disponível em: http://www.lucas.lth.se/publications/CM4OSS.pdf. Acesso: 13 novembro 2005.

[BARNSON05]
BARNSON, Matthew P.; The Bugzilla Team. **The Bugzilla Guide - 2.16.10 Release**. Publicado em 18/05/2005. Disponível em: <www.bugzilla.org/docs216/pdf/Bugzilla-Guide.pdf>. Acesso: 20 de setembro de 2005.

[BELLAGIO05]
BELLAGIO, David E., MILLIGAN, Tom J. **Software Configuration Management Strategies and IBM Rational ClearCase.** A Practical Introduction. 2º ed. Addison-Wesley Professional, 2005.

[BERCZUK02]
BERCZUK, Steve. **Software Configuration Management Patterns: Effective Teamwork, Practical Integration**. 2002.

[BUGZILLA04]
The Bugzilla Team. **The Bugzilla Guide**. 2.19 Development Release, 2004. Disponível em <http://www.bugzilla.org/docs/>. Acesso em: 10 de novembro de 2005.

[CEDERQUIVIST05]
CEDERQUIVIST et al. **Version Management with CVS**. Disponível em <http://www.cvshome.org >. Acesso: 20 setembro 2005.

[CMBOK]
CMBOK. **Configuration Management Body of Knowledge**. Disponível em <http://www.cmbok.com>. Acesso: 20 agosto 2005.

[CMMI02]
SOFTWARE ENGINEERING INSTITUTE - SEI. Capability Maturity Model Integration for Software Engineering - Continuous Representation CMU/SEI-2002-TR-028 ESC-TR-2002-028. V 1.1.

[CMMI05]
SOFTWARE ENGINEERING INSTITUTE - SEI. **Capability Maturity Model Integration (CMMI) Overview.** 2005. Disponível em <http://www.sei.cmu.edu/cmmi/adoption/presentations.html>. Acesso: 02 janeiro 2006.

[DATASUS06]
Departamento de Informação e Informática do SUS – DATASUS. **Processo de Desenvolvimento de Software da DATASUS**. Disponível em <http://pds.datasus.gov.br>. Acesso: 07 fevereiro 2006.

[ECLIPSE05]
Eclipse Foundation Inc. **Eclipse documentation**. Disponível em <http://help.eclipse.org/help31/index.jsp>. Acesso: 15 de outubro de 2005.

[ESTUBLIER02]
ESTUBLIER, J. **Software Configuration Management: A Road Map**. In: Finkelstein, A. **The Future of Software Engineering.** 22nd Int. Conf. on Software Engineering, Limerick, Ireland, ACM Press (2000) 279-289.

[FARAH04]
FARAH, Joe. **Metrics and Process Maturity**. 2004. Disponível em <http://www.cmcrossroads.com>. Acesso: 1º agosto 2005.

[FILHO01]
FILHO, Wilson de Pádua Paula, 2001. **Engenharia de Software, Fundamentos, Métodos e Padrões**. 1º ed. LTC Rio de Janeiro, 2001.

[FIORINI98]
FIORINI, Soeli T. et al. **Engenharia de Software com CMM** – Brasport, 1998. Rio de Janeiro

[FRANK05]
FRANK, Helen – CM Metrics. Disponível em <http://www.cmcrossroads.com/>. Acesso: 05 agosto 2005.

[GALLAGHER01]
GALLAGHER, Brian; Brownsword, Lisa. **The Rational Unified Process® and the Capability Maturity Model® – Integrated Systems/Software Engineering.** Carnegie Mellon University. 2001

[GPL05]
MAGNUS LINUX. Tradução não oficial da Licença Publica Geral (GPL). Disponível em <http://www.magnux.org/doc/GPL-pt_BR.txt>. Acesso: 10 outubro 2005.

[HAZAN00]
HAZAN, C. Metodologia para o Uso de Indicadores na Gerência de Projetos de Desenvolvimento de Software. Tese de Mestrado, IME, Maio 1999.

[HUMPHREY90]
HUMPHREY, Watts. **Managing the Software Process**. Software Engineering Institute (SEI), Addison-Wesley Publishing Company, 1990.

[IEEE78]
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **Software Requirements, Baselining and Control.** (Proceedings, Twelfth Annual Configuration and Date Management Workshop) EIA, October 1978.

[IEEE90]
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, **IEEE Standard 610.12-1990**. **Standard Glossary of Software Engineering Terminology**. New York, Estados Unidos., 1990. Disponível em: <http://standards.ieee.org/reading/ieee/std/se/610.12-1990.pdf>. Acesso: 12 agosto 2005.

[IEEE98]
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. IEEE Standard 828-1998. **IEEE Standard for Software Configuration Management Plans**. New York, 1998.

[ITIL05]
ITIL IT Service Management Collaboration. Disponível em: <http://www.itilwiki.com>. Acesso: 02/11/2005.

[KRUTCHEN98]
KRUTCHEN, P. **The Rational Unified Process – An Introduction**. Addison-Wesley. Reading, MA, 1998.

[LUIZ04]
LUIZ, Ronaldo Rezende Vilela. **Obtendo Qualidade de Software com o RUP**. Sistemas de Informação – Universidade de Uberaba (UNIUBE) Uberaba, MG - Brasil. Publicado em: 6 de Dezembro de 2004.

[MPS-BR05]
Sociedade SOFTEX. MPS.BR - Melhoria de Processo do Software Brasileiro. **Guia Geral do MPS-BR V. 1.0**. Abril de 2005. Disponível em: <www.softex.com.br>. Acesso: 10 de outubro de 2005

[NBR10007]
NBR ISO 10007. Gestão da Qualidade – Diretrizes para a Gestão de Configuração. Nov. 1996.

[NBR12207]
NBR ISO/IEC 12207 - Tecnologia da Informação – Processos de Ciclo de Vida de Software.

[OMG01]
OMG Unified Modeling Language Specification, Version 1.4, September, 2001.

[PMBOK00]
PROJECT MANAGEMENT INSTITUTE - ANSI/PMI**. Project Manager Body of Knowledge** – PMBOK, 2000.

[PRESMAN03]
PRESSMAN, Roger S. **Engenharia de Software**. Mc Graw Hill, 5º ed. Rio de Janeiro, 2002.

[QUEMELLO05]
Quemello , Leonardo Jorge; Manduca, Alexandre Michetti, Fortes, Renata P. M. **Uma Introdução ao Bugzilla**. Universidade de São Paulo. 2005 Disponível em <http://safe.icmc.usp.br/safe/technical-reports/rt_253.pdf>. Acesso: 20 de janeiro de 2006.

[REIS02]
REIS, Christian Robottom. **Uma Visão Geral do Bugzilla, uma Ferramenta de Acompanhamento de Alterações**. Sessão de Ferramentas do SBES 2002. Gramado RS. 14-18 de outubro de 2002. Disponível em <http://www.async.com.br/~kiko/sbes2002/bugzilla.pdf >. Acesso: 20 de outubro de 2005.

[ROCHA04]
ROCHA, Ana Regina Cavalcanti da; Maldonado, José Carlos; Weber, Kival Chaves.Qualidade de Software: Teoria e Prática. Pearson Prentice Hall. 1ª ed. São Paulo. 2004.

[RUMBAUGH05]
RUMBAUGH, James, JACOBSON, Ivar, BOOCH, Grady. **The Unified Modeling Language – Reference Manual.** 2 Ed. Addison-Wesley Publishing Company, 2005.

[RUP01]
RATIONAL SOFTWARE CORPORATION. **Rational Unified Process** - RUP, 2001.

[RUP05]
RATIONAL SOFTWARE CORPORATION. **Rational Unified Process - Best Practices for Software Development Teams.** A Rational Software Corporation White Paper. Disponível em: <http://www.augustana.ab.ca/~mohrj/courses/2000.winter/csc220/papers/rup_best_practices/rup_bestpractices.pdf >. Acesso: Outubro de 2005.

[SCOTT03]
SCOTT, John A. **Software Configuration Management - Draft Version 0.5** - Lawrence Livermore National, 2003. Disponível em: <http://www.swebok.org/stoneman/version_0.5/> Acesso: 11 julho 2005.

[SPICE03]
INTERNATIONAL ORGANIZATION FOR STANDARDIZATION - ISO/IEC 15504 SPICE Software Process Improvement and Capability dEtermination 2003.

[SWEBOK04]
SOFTWARE ENGINEERING BODY OF KNOWLEDGE – **SWEBOK**. Iron Man Version, 2004. Disponível em <http://www.swebok.org>. Acesso: 02 junho 2005.

[SEI05]
SOFTWARE ENGINEERING INSTITUTE – SEI. **Upgrading from CMM-SW to CMMI**. Carnegie Mellon, 2005.

[SOMMERVILLE03]
SOMMERVILLE, Ian. **Engenharia de Software.** Pearson – Addison Wesley, 6º ed. São Paulo, 2003.

[TOMAYKO86]
TOMAYKO, J. E. **Support Materials for Software Configuration Management.** Software Engineering Institute (SEI) – 1986.

[TOMER05]
TOMER, Amir, SHANY, Boaz. Iterative Software Development – From Theory do Practice. RAFAEL, Israel, 2005.

[VERPERMAN03]
VERPERMAN, Jennifer. **Essential CVS**. O'Reilly & Associates, Inc., 2003.

[WHITE01]
WHITE, Brian. **The Power of Unified Change Management**. Rational Software, 2001. Disponível em <http://www.therationaledge.com>. Acesso: 25 junho 2005.

[WEBER06]
WEBER, Kival C.; Rocha, Ana Regina e outros. **Modelo de Referência para Melhoria de Processo de Software: Uma abordagem brasileira**. Disponível em: <www.softex.br/media/artigoCLEI_versao_final.pdf > . Acessad: 20 de agosto de 2006.

GLOSSÁRIO

Auditoria de Configuração Física

Identifica os componentes de um produto a serem implantados a partir do Repositório do Projeto. Todos os artefatos devem estar presentes na linha base. Além disso, a documentação do projeto deve estar consistente com os respectivos códigos.

Auditoria de Configuração Funcional

Confirma que uma linha base atende aos requisitos estabelecidos para ela. Deve-se ter a certeza de que cada requisito foi devidamente testado. Além disso, deve-se verificar junto ao sistema de controle de mudanças, se cada solicitação de mudança encontra-se concluída (fechada).

Caso de Desenvolvimento

O Caso de Desenvolvimento descreve o processo de desenvolvimento escolhido para ser seguido no projeto. Objetiva capturar o processo de desenvolvimento adaptado para o projeto individual [RUP01].

Caso de Uso e Modelo de Caso de Uso

Uma descrição de comportamento do sistema em termos de seqüências de ações. Um caso de uso deve produzir um resultado de valor observável para um ator. Ele contém todos os fluxos alternativos de eventos referentes à produção do "resultado de valor observável". Mais formalmente, um caso de uso define um conjunto de instâncias de casos de uso ou cenários. Especificação de uma seqüência de ações (incluindo variantes) que um sistema (ou outra entidade) pode executar, interagindo com atores do sistema. Trata-se de uma descrição lógica de parte das funcionalidades do sistema modelado. Por sua vez, um Modelo de Caso de Uso descreve requisitos funcionais de um sistema em termos de Casos de Uso. O modelo de caso de uso modela o sistema assim como ele é percebido pelos agentes externos, denominados atores. A finalidade do modelo é listar os atores e casos de uso, bem como a interação entre estes elementos.

Check-out e Check-in

Check-out representa o processo de requisição, aprovação e cópia dos IC do repositório para o espaço de trabalho do desenvolvedor. Check-in, ao contrário, representa o processo de revisão, aprovação e cópia dos IC do espaço de trabalho do desenvolvedor para o repositório.

Ciclo (RUP – Rational Unified Process)

Uma passagem completa pelas quatro fases do RUP: iniciação, elaboração, construção e transição. O intervalo de tempo entre o começo da fase de iniciação e o final da fase de transição.

Ciclo de Vida

É o conjunto de processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, abrangendo toda a vida do sistema, desde a fase de levantamento de requisitos até a sua descontinuidade.

Comitê de Controle de Mudança (CCB – Change Control Board)

É o comitê que supervisiona o processo de mudanças. É integrado por representantes de todas as partes interessadas, inclusive clientes, desenvolvedores e usuários. Consiste de um mecanismo central que é necessário para assegurar que cada necessidade de mudança é corretamente controlada e coordenada. Cada mudança na linha base deve ser autorizada por este comitê. É responsável pela aprovação ou rejeição da mudança.

Configuração

Os requisitos, o design e a implementação que definem determinada versão de um sistema ou de um componente do sistema.

Construção (Build)

É o processo de compilação do sistema a partir dos itens fonte para uma configuração alvo. Utiliza um arquivo de comandos que descreve como deve ocorrer a construção. Ex.: Makefile.

CSS

Cascading Style Sheet, é um mecanismo para adição de estilos (cores, tamanho de fontes, espaçamento) às páginas escritas em HTML. É definida pelo W3C – World Wide Web Consortium.

Delta (Diferença)

Representa o que mudou entre duas versões consecutivas. A técnica de diferença para frente (*forward delta*) armazena a versão mais antiga do IC e as diferenças para as versões posteriores. Já a técnica de diferença para trás *(reverse delta)* armazena a versão mais recente do IC e as diferenças para as versões anteriores.

Descrições de Versão e Releases

As descrições de versão e/ou releases devem incluir no mínimo as seguintes informações:

- Mídia física e documentos da versão;
- Inventário do conteúdo do software (listagens de arquivos);
- Instruções de instalação;
- Possíveis problemas e erros conhecidos.

Espaço de Trabalho (Workspace)

A área de trabalho que contém todo o código que está sendo desenvolvido no momento presente, ou seja, as edições atuais.

Formulário de Solicitação de Mudanças

É um formulário, na sua forma impressa ou eletrônica, que deverá ser preenchido como forma de dar início ao processo de modificação do software. O solicitante relaciona as necessidades de modificação, expondo de maneira clara as necessidades da mudança.

Garantia de Qualidade

A função de Garantia de Qualidade (QA) é responsabilidade do Gerente de Projeto e visa garantir que os padrões do projeto estejam corretos e sejam seguidos por toda a equipe do projeto.

Gerência de Configuração de Software – (GCS)

Um processo de suporte utilizado para identificar, definir e criar uma linha base de itens; controlar modificações e liberações desses itens; relatar e registrar o estado dos itens e CR; assegurar abrangência, consistência e correção dos itens; e controlar armazenamento, gerenciamento e liberação dos itens.

Gerenciamento de Controle de Mudanças

A atividade de controle e rastreamento de mudanças em artefatos.

HTML

Hyper Text Markup Language, é a linguagem de programação não proprietária mais comumente utilizada na *internet*. É interpretada pelos navegadores e sua especificação é controlada pelo W3C – World Wide Web Consortium (www.w3c.org).

Implantação (deployment)

Representa a arrumação estratégica do sistema no ambiente de desenvolvimento ou produção.

Item de Configuração (IC)

Uma entidade em uma configuração que satisfaz uma função de uso final e que pode ser identificada de forma exclusiva em um determinado ponto de referência. O termo *item de configuração* representa a agregação de hardware, software, ou ambos, tratados pela GCS como um elemento único. O exemplo mais comum de IC é um arquivo de código fonte. Além disso, executáveis, documentos e modelos de toda espécie também são considerados IC.

Item de Configuração Fonte e Item de Configuração Derivado

Item de Configuração Fonte é o IC que serve como origem para a geração de IC derivados via um procedimento de construção. O termo Item de Configuração Derivado representa um IC que

pode ser obtido a partir de outros IC pela aplicação de um procedimento de construção, não sendo necessário o seu armazenamento no repositório.

Liberação (Release)

Um subconjunto do produto final que é o objeto de avaliação em um marco principal. Uma liberação é uma versão estável e executável do produto que vem acompanhada dos artefatos necessários para sua utilização (como notas de liberações ou instruções de instalação, por exemplo). Uma liberação pode ser interna ou externa. Uma liberação interna é usada apenas pela organização de desenvolvimento, como parte de um marco, ou para fazer uma demonstração para usuários ou clientes. Uma liberação externa é entregue aos usuários finais. Uma liberação não é necessariamente um produto completo, mas pode ser apenas uma etapa ao longo do caminho, com sua utilidade avaliada apenas do ponto de vista da engenharia. Uma liberação não é somente código executável, pois também inclui os arquivos de configuração que definem como a liberação deve ser configurada para determinadas instalações. Inclui também arquivos de dados, os programas de instalação, toda a documentação eletrônica e em papel que descreva o sistema, desde documentação técnica quando manuais diversos.

Linguagem de Modelagem Unificada - UML

A UML é uma linguagem gráfica para visualização, especificação, construção e documentação dos artefatos de um software. Está baseada exclusivamente no paradigma de programação orientada a objetos, sendo que vários de seus diagramas expressam seus comportamentos, estados e responsabilidades.

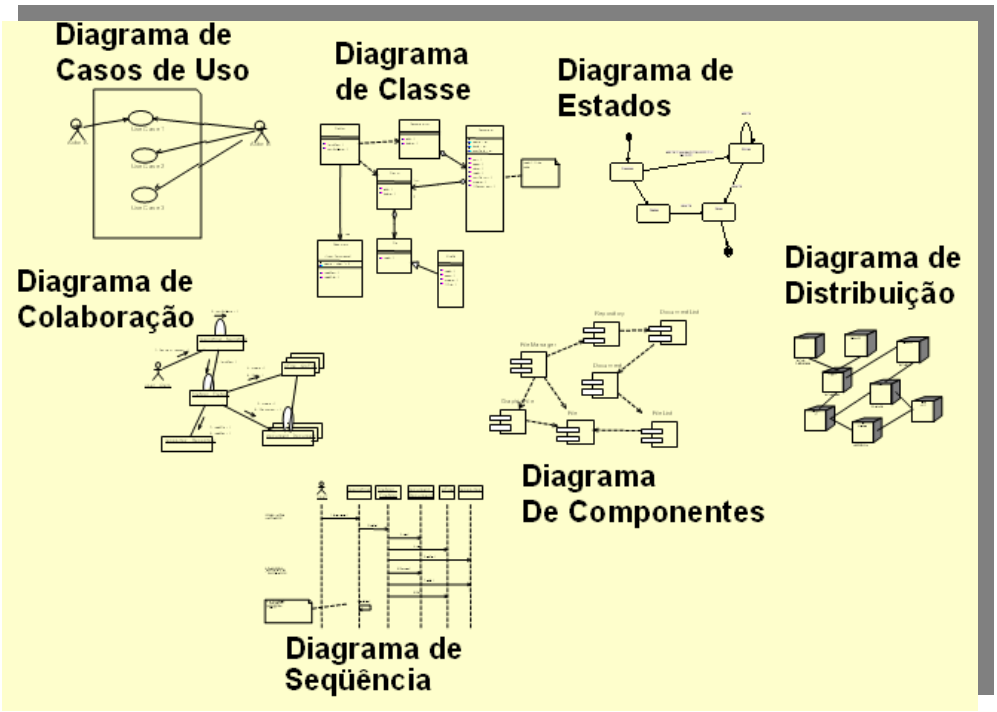


Figura 18 - Exemplos de Modelos da UML

Linha Base (Baseline)

Este conceito é o ponto central da disciplina de GCS. Consiste de uma liberação revisada e aprovada de IC que constitui uma base ajustada para desenvolvimento ou evolução posterior e que só pode ser alterado através de um procedimento formal, como gerenciamento de mudança e controle de configuração. Uma linha base deve conter um conjunto de IC, devendo ser rastreáveis entre si, ou seja, documentos de análise e projeto devem refletir com exatidão o código à que se refere.

Modelo de Design

Um modelo de objeto que descreve a realização de casos de uso. É utilizado como uma abstração do modelo de implementação e seu código-fonte. É utilizado para explorar a arquitetura do sistema e as formas de implementação.

Modelo de Implementação

Define a dependência entre os artefatos do projeto, sua estrutura de diretórios [DATASUS06]. Contém o conjunto de componentes e dos subsistemas que compõem o sistema principal. Trata-se de um modelo abrangente, que envolve todos os artefatos necessários para criar e gerenciar o sistema no ambiente em tempo de execução [RUP01].

Modelo de Domínio

Um modelo de domínio captura os tipos mais importantes de objetos no contexto de domínio. Os objetos de domínio representam as entidades que existem ou os eventos conhecidos no ambiente em que o sistema funciona. O modelo de domínio é um subconjunto do modelo de objetos de negócios.

Plano de Iteração

O gerente de projeto tem a responsabilidade de desenvolver o Plano de Iteração, que nada mais é que um conjunto de atividades e tarefas divididas por seqüências de tempo, com recursos atribuídos e dependências de tarefas [RUP01].

Ramos (Branch) e Versões

O termo ramo (*branch*) representa versões temporárias que não seguem a linha principal do desenvolvimento. Um ramo é utilizado para dividir um projeto. Normalmente existe o ramo mestre e os ramos secundários. Sua finalidade principal é o isolamento de várias linhas de desenvolvimento, que poderão ser mescladas em uma nova versão. Na figura 19, são indicados vários ramos independentes de desenvolvimento, que se destinam a ambientes de hardware diferentes.

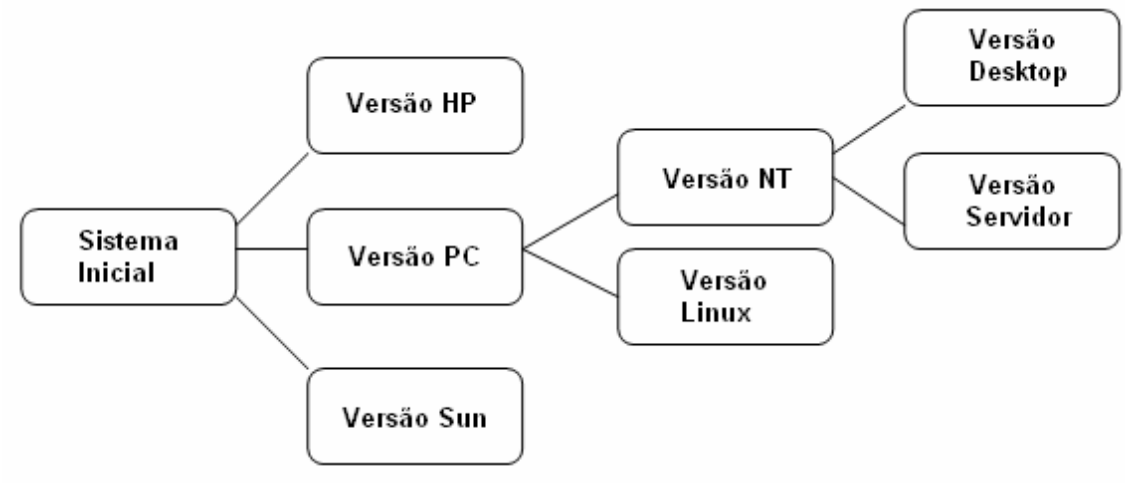


Figura 19 - Exemplo de Ramos

A possibilidade de armazenar, recriar e registrar o histórico do desenvolvimento dos IC constitui uma das características fundamentais de um sistema de GCS. Uma versão é uma instância de um IC, ou de um sistema, que difere de maneira substancial de outras versões. Novas versões podem ter acréscimos de funcionalidades, até mesmo solução de defeitos do IC ou sistema. Cada versão estável de um IC corresponde a uma versão deste item. Versões de um IC podem ser organizadas de diversas formas, através da atribuição de uma numeração específica. Esta numeração é seqüencial e frequentemente é chamada de revisão. Estes ramos (linhas de desenvolvimento) podem ser unidos em uma nova versão, que por sua vez possui dois ou mais ramos predecessores. Os ramos são criados por diversas razões. A primeira delas consiste da criação de ramos permanentes, os quais representam aplicações alvo diferentes, tais como sistemas operacionais ou plataformas de hardware, assim como aplicações destinadas a servidores e desktops. Num segundo momento, temos os ramos temporários, que simplesmente permitem o desenvolvimento paralelo. Assim que não houver mais a necessidade do desenvolvimento paralelo, os ramos são mesclados na linha de desenvolvimento principal. Normalmente, um ramo consiste de uma série de revisões, sendo que ramos adicionais podem ser criados a partir do original. Outra utilização dos ramos diz respeito à correção de erros (*bugs*). Neste caso a versão em produção de um sistema fica isolada da versão em desenvolvimento.

A criação de uma estratégia para manipulação de ramos é uma tarefa essencial do gerente de configuração. Em geral, as ferramentas para controle de versões identificam revisões, normalmente através uma técnica de numeração em diversos estágios. Além disso, o desenvolvedor pode atribuir

rótulos na forma de cadeias de caracteres, frequentemente chamados de Rótulos (*tags).* Estes rótulos identificam um conjunto de IC, que a qualquer momento, podem ser recuperados para a construção do sistema (build).

Relatório de Estado de Configuração

O sistema de controle de mudanças deve prover informações estatísticas sobre o andamento do processo. Estas informações, quando organizadas, servem de base para a melhoria do processo. São baseadas em modelos de métricas, gerando indicadores que possibilitam a elaboração de mudanças e adaptações no processo, de forma a torná-lo mais eficiente. As seguintes categorias de relatórios são desejáveis num sistema de controle de mudanças:

a) Vencimento (Relatórios baseados em períodos de tempo)

Há quanto tempo as Solicitações de Mudança de diversos tipos estão pendentes? Qual é o "tempo de retardo" de quando são encontrados defeitos no ciclo de vida em comparação com o tempo necessário para sua correção?

b) Distribuição (Relatórios baseados em contagens)

Existem quantas Solicitações de Mudança nas diversas categorias por proprietário, prioridade ou estado de correção?

c) Tendência (Relatórios relacionados a períodos e contagens)

Qual é o número acumulado de defeitos encontrados e corrigidos no decorrer do tempo? Qual é a classificação dos defeitos detectados e corrigidos? Relação de defeitos pendentes versus defeitos corrigidos? Qual é a média de tempo de correção de um defeito?

Repositório

Consiste de um local de acesso controlado onde as versões dos IC são armazenadas. Pode ser um banco de dados ou apenas uma estrutura hierárquica de diretórios, dependendo do tipo de sistema automatizado adotado.

Reunião de Revisão do CCB

A função dessa reunião é rever as CR Enviadas. Uma revisão inicial do conteúdo da CR é feita nesta reunião, de forma a determinar se a solicitação é válida. Se for, será decidido se a mudança está dentro ou fora do escopo dos releases atuais, conforme a prioridade, programação, recursos, nível de esforço, risco, gravidade e outros critérios relevantes definidos pelo grupo.

Revisões por Pares

Uma revisão por pares é a revisão do produto de trabalho executada por integrantes da equipe pertencentes a um mesmo papel durante o desenvolvimento. Objetiva a identificação dos defeitos para remoção. No CMMI, possui equivalência a "Inspeção do Produto do Trabalho" do CMM [CMMI02].

Revisões Técnicas Formais - RTF

A Revisão Técnica Formal (RTF) é uma atividade de garantia da qualidade de software realizada com o objetivo de se descobrir erros no software, sejam nas funções lógicas, implementação ou requisitos. A RTF é na realidade uma categoria de revisões que inclui *walkthroughs*, inspeções e revisões.

Rótulos (Tags)

Representam uma posição dos arquivos de um determinado produto em um determinado tempo. Agrupam os arquivos e seus números de revisão sob um único nome. Ex.: *Cliente.java – 1.1, NotaFiscal.java – 1.2, ItemNotaFisca.java – 1.1*

Um rótulo chamado *Versao1*, por exemplo, pode ser criado incluindo todos os arquivos acima exemplificados.

Solicitação de Mudança (Change Request - CR)

Um artefato formalmente submetido que é usado para rastrear todas as solicitações dos envolvidos (inclusive novas características, solicitações de melhoria, conserto de defeitos ou mudança de requisitos) junto com as informações de estado relacionadas durante todo o ciclo de vida do projeto. Todo o histórico de mudanças será mantido com a CR, o que inclui todas as

mudanças de estado, datas e motivos para as mudanças. Essas informações ficam disponíveis para revisões repetidas e para fechamento final.

Figura 20 - Processo básico de solicitação de mudanças

Abaixo são listadas as informações desejáveis para cada CR:

a) **Tamanho**: avaliação da quantidade de novas linhas de código ou classes que serão criadas ou alteradas;

b) **Alternativas**: quais são, se houver, as alternativas para execução da modificação. Se houver mais de uma forma de fazer, qual será adotada efetivamente;

c) **Complexidade**: abrangência da modificação;

d) **Cronograma**: especificação dos prazos;

e) **Custo**;

f) **Severidade**: quão crítica é a modificação;

g) Relacionamento com outras modificações;

h) **Testes**: quais são os requisitos especiais de testes;

i) Recursos necessários para a modificação;

j) **Impacto no sistema**: consumo de memória, performance ou outras conseqüências para o sistema;

k) Benefícios da modificação;

l) **Política**: considerações especiais sobre quem está solicitando a modificação, ou sobre a maneira que o solicitante será afetado pela modificação;

Se a origem da solicitação é para solucionar um defeito do sistema, outras informações são necessárias:

a) Definição do problema a ser corrigido;

b) As condições sobre os quais os problemas foram observados;

c) A descrição técnica do problema;

d) A relação dos demais programas afetados.

Informações adicionais são necessárias antes da liberação :

a) Código final gerado pela mudança;

b) Documentação final atualizada;

c) Resultado dos testes;

Teste de Software

O Teste de Software é um fator crítico da garantia da qualidade de software e representa a revisão final da especificação, projeto e geração de código. Na prática, é o processo de execução do programa na tentativa de se encontrar um erro. Pressman propõem a divisão do teste em duas categorias: *Caixa Preta* (interface do software) e *Caixa Branca (*análise do código). Para a execução dos testes, o *Caso de Teste* deve ser projetado. Trata-se dos passos necessários para a execução dos testes do programa, suas condições de execução e resultados esperados, de forma que o testador possa testar a maioria dos fluxos de controle providos pela interface. O caso de teste é executado e comparado com os resultados esperados. Um relatório de testes deve ser gerado após a conclusão da massa de testes.

Versão e Variante

Versão é o estado de um IC em um determinado momento do desenvolvimento de software. Para cada IC, um grafo de evolução poderá ser criado, como mostra a figura 21. Este grafo descreve o histórico de modificações de determinado IC. Para cada revisão, é incrementado o número de versão, de acordo com a política adotada pela organização.

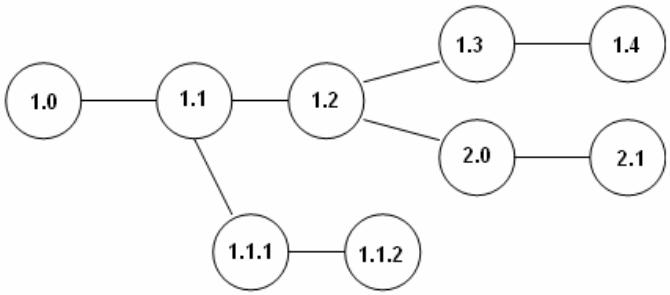


Figura 21 - Grafo de Evolução

Por sua vez, a variante consiste de uma versão funcionalmente equivalente à outra, mas projetada para ambiente de hardware ou software distinto.

Versão x Linha Base

Apesar de fisicamente representarem o estado dos IC em um dado momento, estes conceitos possuem alguns diferenciais:

- Uma versão pode ser gerada a qualquer instante, de acordo com a necessidade do desenvolvimento (testes, produção, correção de erros), sendo que o software poderá evoluir ou não a partir dela. Uma versão pode ser o resultado de uma linha base;

- Uma linha base necessariamente implica na integridade dos IC. É a base para a evolução do software, sendo que sua modificação somente ocorrerá a partir de um processo formal de controle de mudanças.

WWW

Acrônimo para *World Wide Web* (Rede de Alcance Mundial). Corresponde em termos gerais, a interface gráfica da internet.

APÊNDICE A – TUTORIAL DAS FERRAMENTAS DE GC

A.1 Descrição

Este tutorial demonstrará o relacionamento das ferramentas de apoio ao PGCM, dentro de um ambiente baseado em software livre. Para que este tutorial possa ser executado, algumas pré-condições devem ser obtidas. Estas pré-condições dizem respeito ao estabelecimento do ambiente operacional das ferramentas, compreendendo instalação e configuração, sendo que os mesmos encontram-se fora do escopo do trabalho.

As ferramentas utilizadas compreendem os softwares CVS – *Concurrent Version System* (controle de versões), o *Bugzilla* (controle de mudanças) e o ambiente de desenvolvimento *Eclipse*.

A.2 Objetivo

O objetivo deste tutorial é apresentar as ferramentas propostas pelo PGCM e o seu relacionamento no ambiente de Gerência de Configuração e Controle de Mudanças. Além disso, tem como objetivo secundário mostrar a simplicidade do uso das ferramentas, com um exemplo simples, porém esclarecedor.

A.3 Método de Apresentação

O tutorial está baseado no desenvolvimento de um software hipotético do tipo *Site da Internet* (*Website*), contendo as referidas páginas e documentação de requisitos. Por uma questão de concisão e simplicidade, não será levado em consideração aspectos tais como completude dos artefatos para um projeto orientado à objetos, por exemplo, nem iterações dentro das fases do USDP. Nem todas as funcionalidades das ferramentas serão abordadas pelo tutorial, resumindo-se somente àquelas necessárias para demonstração das ferramentas.

A.4 Ferramentas utilizadas e suas versões

Para execução deste tutorial, as seguintes ferramentas foram utilizadas.

CVS: CVSNT 2.5.03 (*Scorpio*) *Build* 2151 (disponível em www.cvsnt.org)

Eclipse: Eclipse SDK 3.1.1 (disponível em www.eclipse.org)

Bugzilla: versão estável 2.20 (disponível em www.bugzilla.org)

A.5 Configuração do CVS e do Eclipse

- **Inicialização do Repositório CVS**

Pré-requisito: Ter uma instalação válida do CVS.

Vamos assumir deste ponto em diante, que durante a instalação do CVS, um repositório será criado conforme indicado: *"d:\repositorio"*

Depois de instalado o CVS, o repositório deve ser inicializado, conforme indica a figura abaixo:

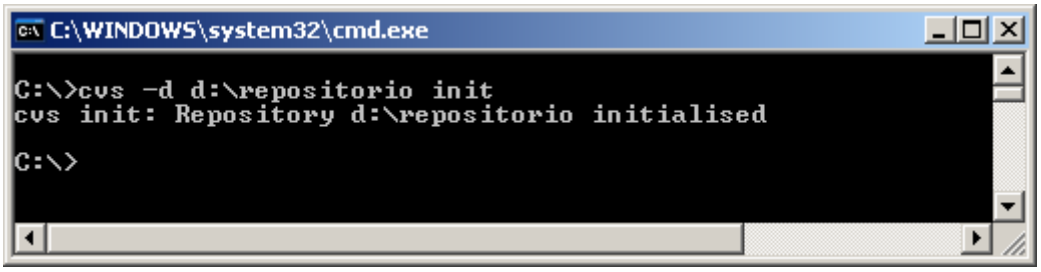


Figura Tutorial 1 - Inicialização do Repositório

- **Configurar o Eclipse para visualizar o repositório criado**

Abrir o Eclipse e selecionar a perspectiva "*CVS Repository Exploring*" através do menu *Window\Open Perspective\Other*, conforme indica a figura 2:

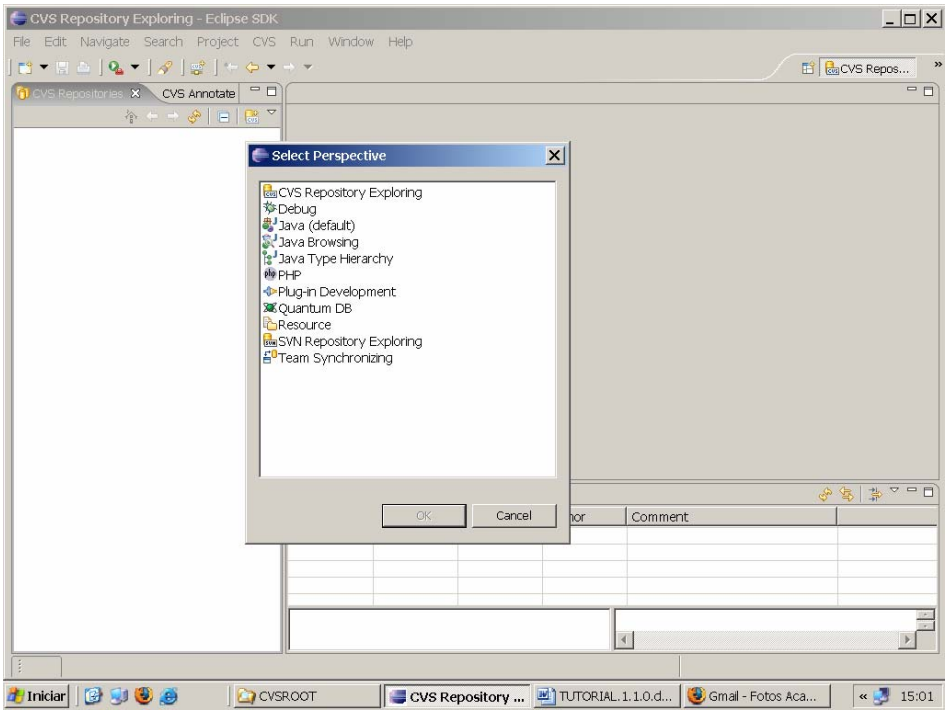


Figura Tutorial 2 - Seleção de Perspectiva

Em seguida, clique no ícone "*Add CVS Repository*", conforme indica a figura 3:

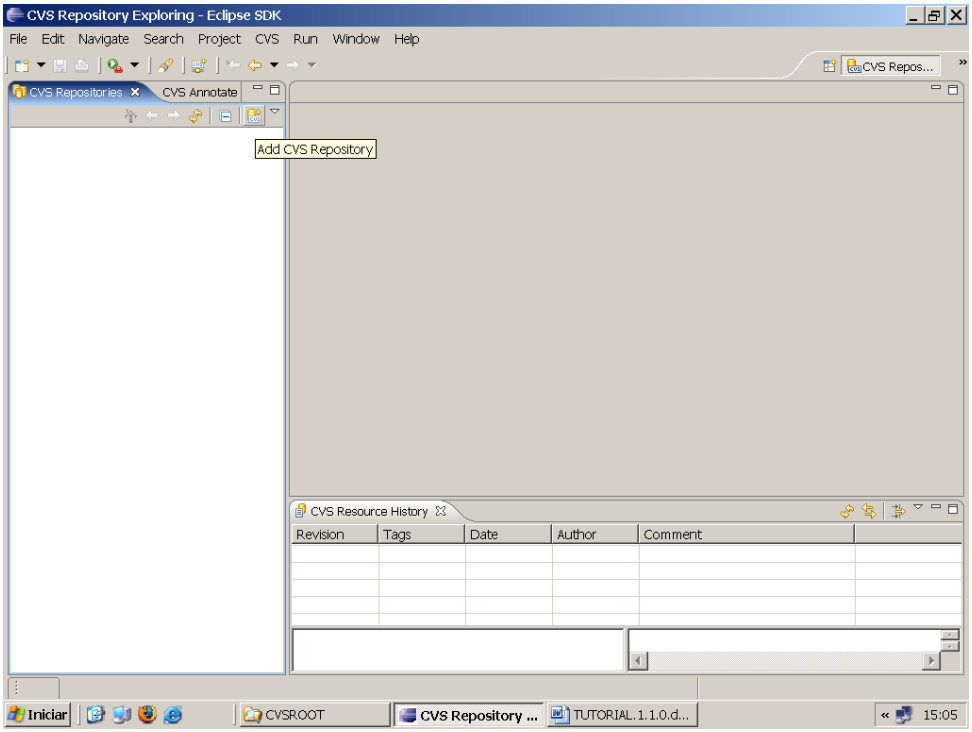


Figura Tutorial 3 - Perspectiva "CVS Repositories"

Na janela que se abre, informe os dados referentes ao repositório:

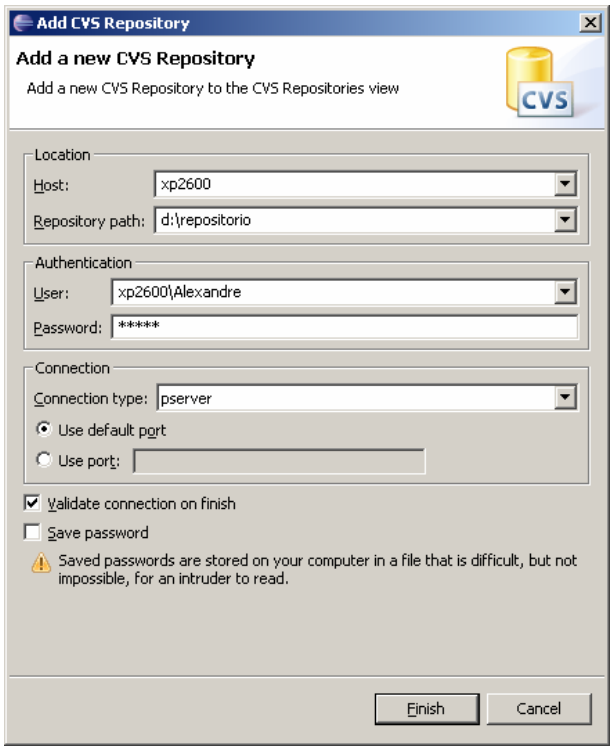


Figura Tutorial 4 - Adicionando um Repositório

O protocolo *PSERVER* será utilizado, por ser nativo do CVS.

Após clicar em "*Finish*", o painel esquerdo do Eclipse, que contém a visão do repositório, ficará como mostra a figura 5:

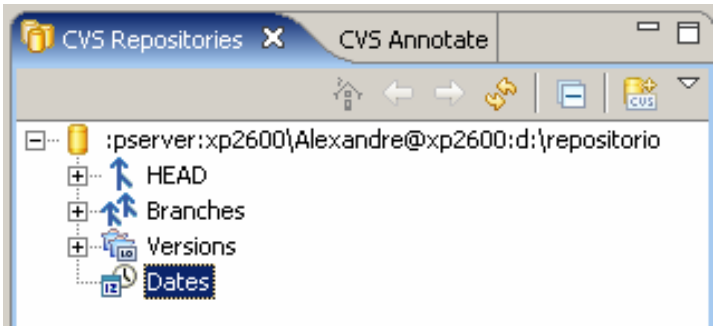


Figura Tutorial 5 - Repositório do Projeto

Neste momento, o Eclipse já estará conectado ao repositório CVS.

A.6 Preparação do Projeto para Importação para o Repositório CVS

Será criada a estrutura de diretórios do projeto que será importada para o repositório CVS. Como padrão, esta estrutura será definida com base no proposto no Apêndice "B" desta monografia, com algumas adaptações, como é previsto. A figura 6 apresenta uma sugestão de estrutura para o projeto:

Figura Tutorial 6 - Estrutura do Repositório

- **Importação**

Acessar a pasta raiz do site: *"01.webmestre.com.br"*

Executar o comando "**import**" definido abaixo:

```
$ cvs -d "d:\repositorio" -q import -m "Site do WebMestre"
webmestre.com.br ucb lib-01-0
```

Onde:

-d	Informa que será especificado o local do repositório
"d:\repositorio"	Local onde se encontra a pasta CVSROOT
-m	Especifica uma mensagem descrevendo o projeto importado. No caso "*Site do WebMestre*".
Webmestre.com.br	Designa o nome do projeto e, por conseguinte, o diretório onde os arquivos serão armazenados no repositório.
ucb	Atribui ao projeto o nome da empresa.
Lib-01-0	Um identificador de liberação.

Ao executar o comando, a saída (parcial) abaixo mostra os arquivos que foram importados para o repositório:

```
N webmestre.com.br/02.Projeto-e-Implementacao/01.Fontes/Alunos23A.php
N webmestre.com.br/02.Projeto-e-Implementacao/01.Fontes/Alunos31B.php
N webmestre.com.br/02.Projeto-e-Implementacao/01.Fontes/cadDisciplina.php
N webmestre.com.br/02.Projeto-e-Implementacao/01.Fontes/cadProf.php
N webmestre.com.br/02.Projeto-e-Implementacao/01.Fontes/cadSenhaProf.php
N webmestre.com.br/02.Projeto-e-Implementacao/01.Fontes/cadTurmas.php
N webmestre.com.br/02.Projeto-e-Implementacao/01.Fontes/cadTurmasAcomp.php
N webmestre.com.br/02.Projeto-e-Implementacao/01.Fontes/chat.php
```

Figura Tutorial 7 - Saída do Comando de Importação

A letra "N" ao lado precedendo o nome de cada arquivo, indica que se trata de um novo arquivo.

Através do Eclipse, após a atualização da Perspectiva "*CVS Repositories*", poderá ser observada toda a árvore de diretórios que foi importada, conforme indica a figura 8:

Figura Tutorial 8 - Projeto Importado para o Repositório

Após a importação ter sido concluída, o processo de controle de versões já está em funcionamento. Os desenvolvedores poderão agora executar o "*check-out*" do projeto para suas respectivas áreas de desenvolvimento.

- **Check-out**

Neste momento, surge a necessidade de alteração de uma funcionalidade do sistema. Uma nova CR foi criada pelo Gerente do Projeto, por exemplo, atribuindo-a a um determinado desenvolvedor. Esta CR especifica que deverá ser alterado o arquivo de "*Upload*" de Imagens, denominado "*uploadimagem.php*", conforme indicado abaixo:

Original	Alterar para

`// Largura máxima (pixels)`
`$config["largura"] = 350;`
`// Altura máxima (pixels)`
`$config["altura"]  = 180;`
	`// Largura máxima (pixels)`
	`$config["largura"] = 550;`
	`// Altura máxima (pixels)`
	`$config["altura"]  = 480;`

Figura Tutorial 9 - Dados para alteração

Para realização da alteração, o desenvolvedor deverá realizar o "*check-out*" do módulo "01.Fontes", conforme indica a figura 10:

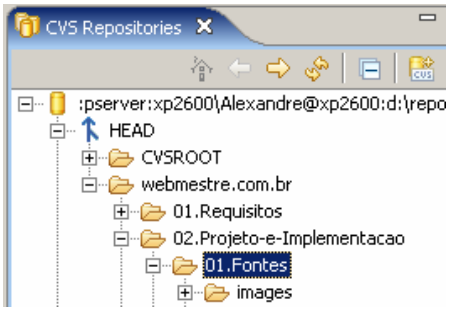


Figura Tutorial 10 - Módulo de Arquivos Fontes

Após ter selecionado a opção "*Check-out*", o Eclipse abrirá um assistente onde poderá ser escolhida a forma como será realizado o "*check-out*" do projeto, conforme indica a figura 11:

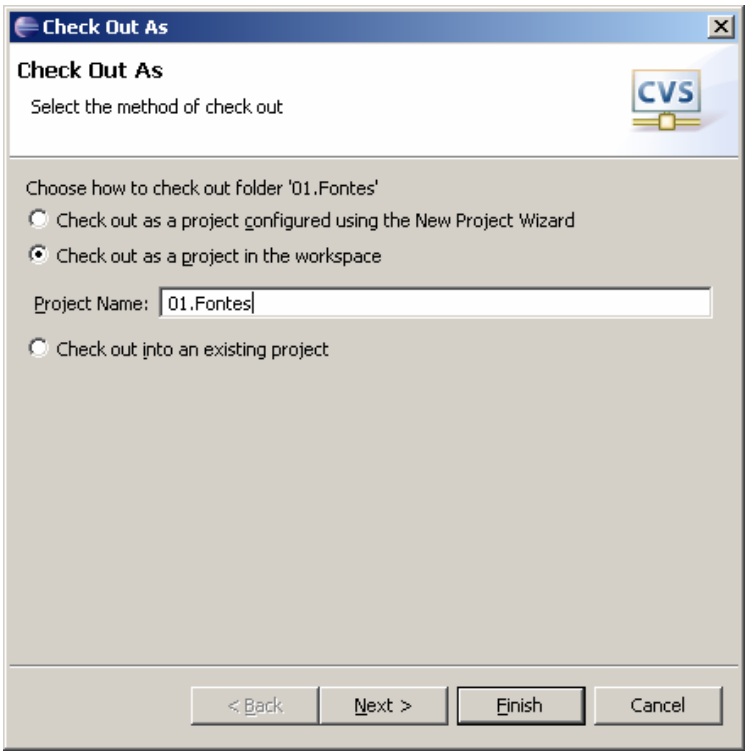


Figura Tutorial 11 - Assistente de "check-out"

Escolheremos então a opção "*Check-out As a Project in the Workspace*". A área de trabalho escolhida será a definida na abertura do Eclipse (c:\desenvolvimento). Selecionar o botão "*Finish*" para realizar o "*check-out*". Após este passo, o projeto será copiado para a área de desenvolvimento. Para visualização do projeto, a perspectiva do Eclipse deve ser alterada para a perspectiva Java, selecionando-se o ícone como indica a figura 12:

Figura Tutorial 12 - Alterando Perspectiva

Uma vez selecionada a perspectiva Java, a janela do Eclipse fica conforme indica a figura 13:

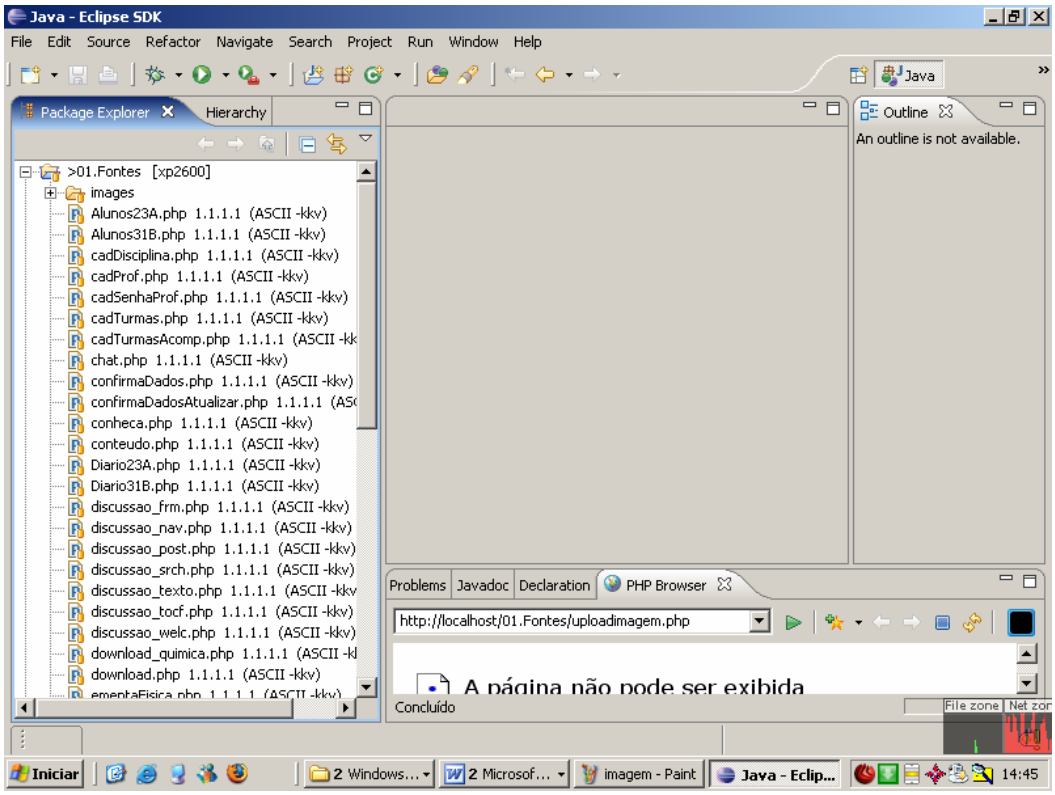


Figura Tutorial 13 - Módulo "01.Fontes"

Observe que na visão do "*Package Explorer*", os arquivos do módulo o qual foi realizado o "*check-out*", são identificados com o número de versão 1.1.1.1, além da indicação do tipo de arquivo (ASCII).

Na visão "*Package Explorer*", o arquivo "*uploadimagem.php*", deverá ser selecionado e aberto para alteração. Um duplo clique sobre o arquivo fará com que o mesmo seja aberto para edição, conforme indica a figura 14:

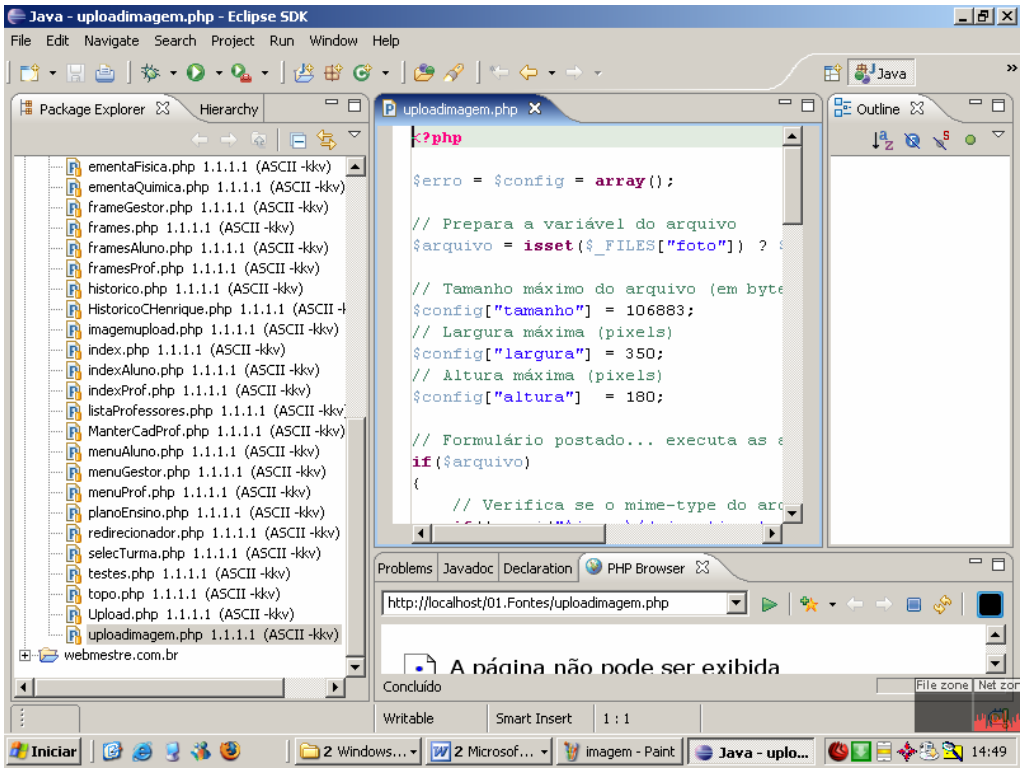


Figura Tutorial 14 - Abrindo arquivo para edição

O desenvolvedor então faz as alterações necessárias (conforme indicado anteriormente) e salva o arquivo (Pressione Ctrl + S). Feche o arquivo após a alteração.

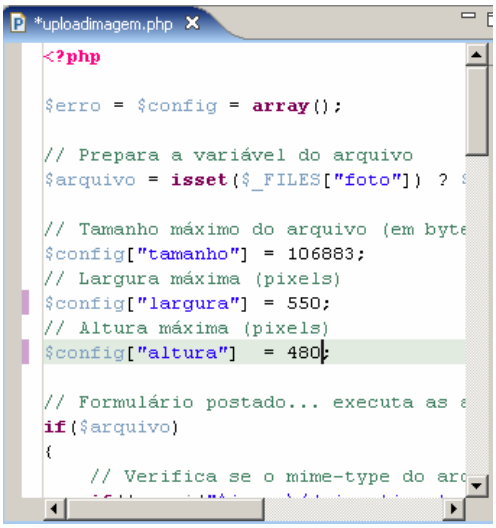


Figura Tutorial 15 - Efetuando Alteração

A partir do momento que as alterações foram realizadas, as atividades de verificação e validação devem ser iniciadas (pelo representante da Garantia da Qualidade), conforme o processo de desenvolvimento da organização. Estas atividades estão fora do escopo deste trabalho.

Uma vez aprovadas, o desenvolvedor deverá enviá-las para o repositório, de maneira que a mesma obtenha um novo número de versão. Este processo, no contexto do CVS chama-se "*commit*".

- **Commit (Check-in)**

Para o envio do módulo para o repositório, o processo de *commit* deve ser executado. Na visão "*Package Explorer*", selecione o projeto com botão direito do mouse, escolha a opção "*Team*" e em seguida "*commit*", como indicado na figura 16:

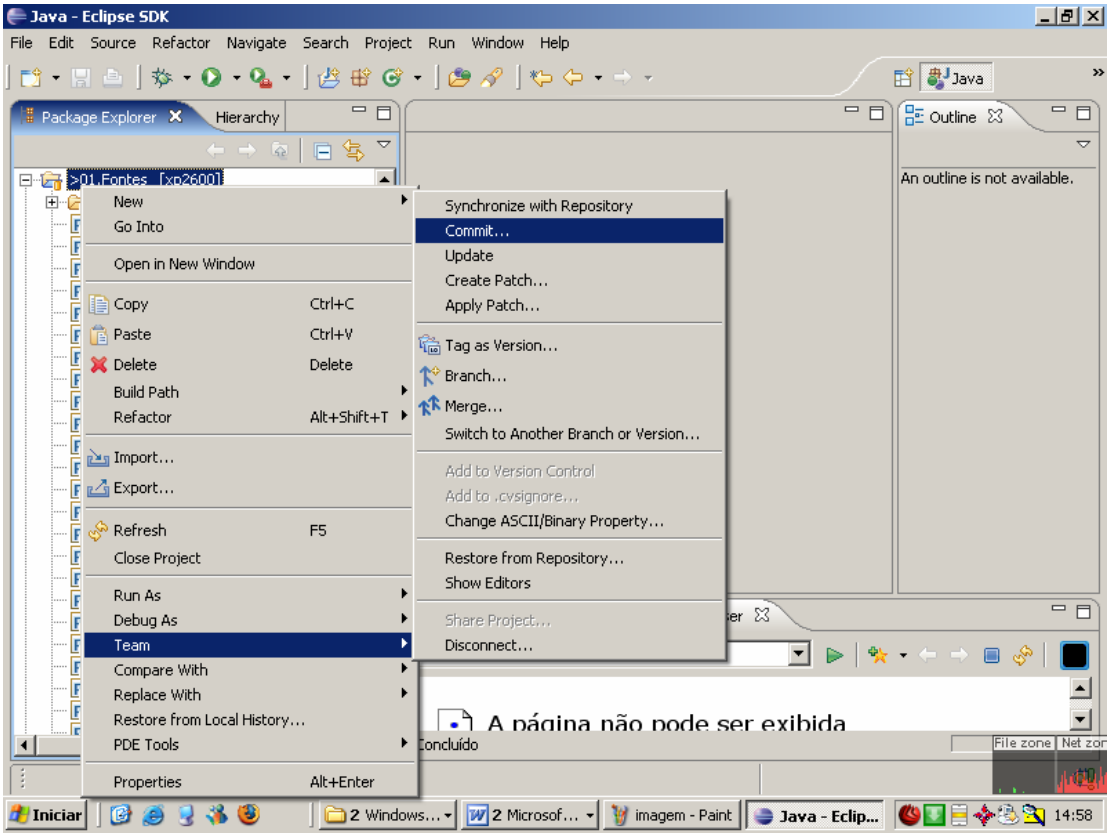


Figura Tutorial 16 - Executando "commit"

No momento da seleção, será aberta uma janela para que as observações referentes às alterações realizadas sejam cadastradas:

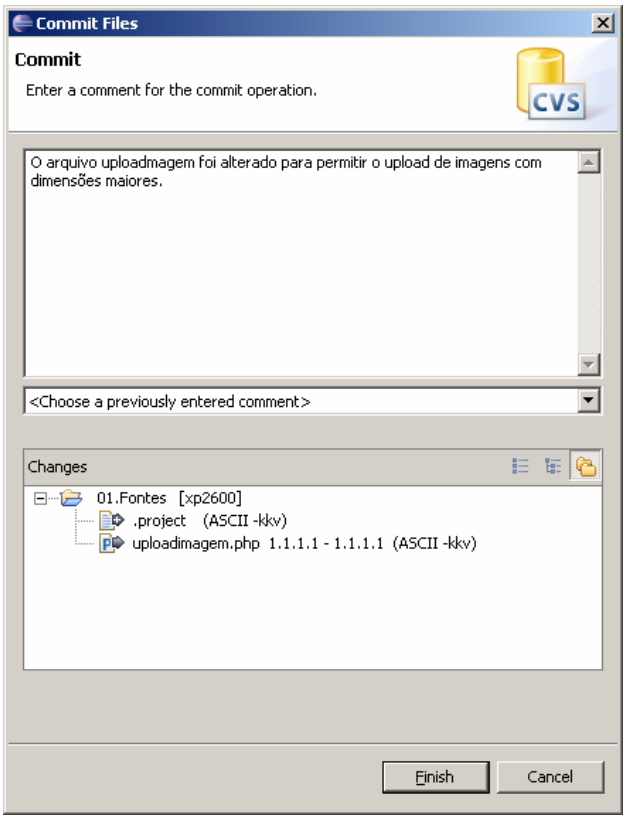


Figura Tutorial 17 - Comentando uma alteração

Durante a realização do "*commit*", a seguinte janela será observada:

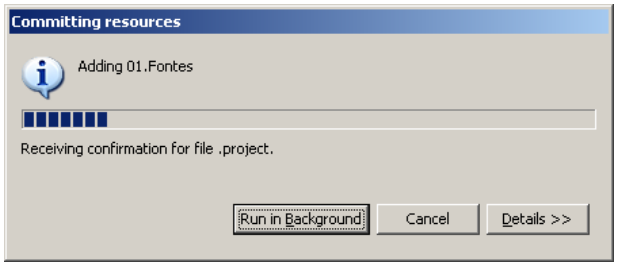


Figura Tutorial 18 - Progresso do "commit"

O projeto poderá em seguida ser fechado, bem como a visão "*Package Explorer*".

Retornando para a visão *"Cvs Repositories"*, após a atualização, poderá ser observado que neste o arquivo alterado anteriormente tem agora um novo número de versão: 1.2, conforme indicado na figura 19:

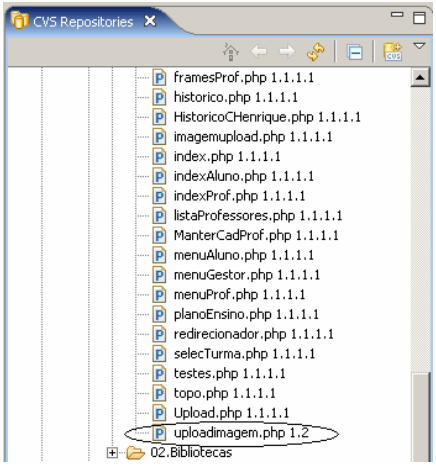


Figura Tutorial 19 - Nova Versão: 1.2

A.7 Controle de Mudanças: Bugzilla

Neste momento será dado início o controle de modificações do projeto, sendo apresentado algumas funcionalidades básicas do software Bugzilla, como definido anteriormente.

Para utilização do Bugzilla, o projeto deverá estar cadastrado, assim como os elementos que fazem parte do time de desenvolvimento. O gerente fará a atribuição da CR ao desenvolvedor que possui responsabilidade sobre determinado módulo do sistema. Uma vez tendo sido confirmada a necessidade da CR e atribuída, o controle de versões será iniciado, seguindo-se os passos previstos na seção A.6 deste tutorial.

- **Cadastro do Produto (Sistema)**

A partir da tela inicial do Bugzilla, conforme indica abaixo, o "produto" (projeto em questão) deverá ser cadastrado:

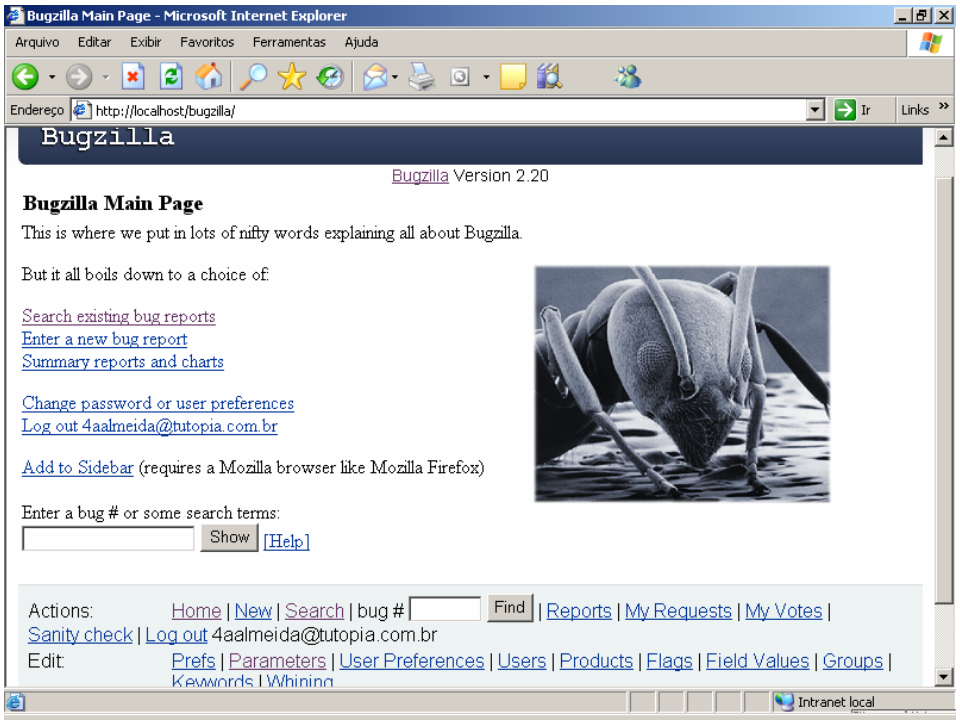


Figura Tutorial 20 - Tela inicial do Bugzilla

A partir da tela anterior, o administrador deverá cadastrar o sistema, através do menu "*Products*", na parte inferior da tela. Ao acessar este menu, o administrador poderá cadastrar o sistema, através do link "*Add a product*", conforme indicado abaixo:

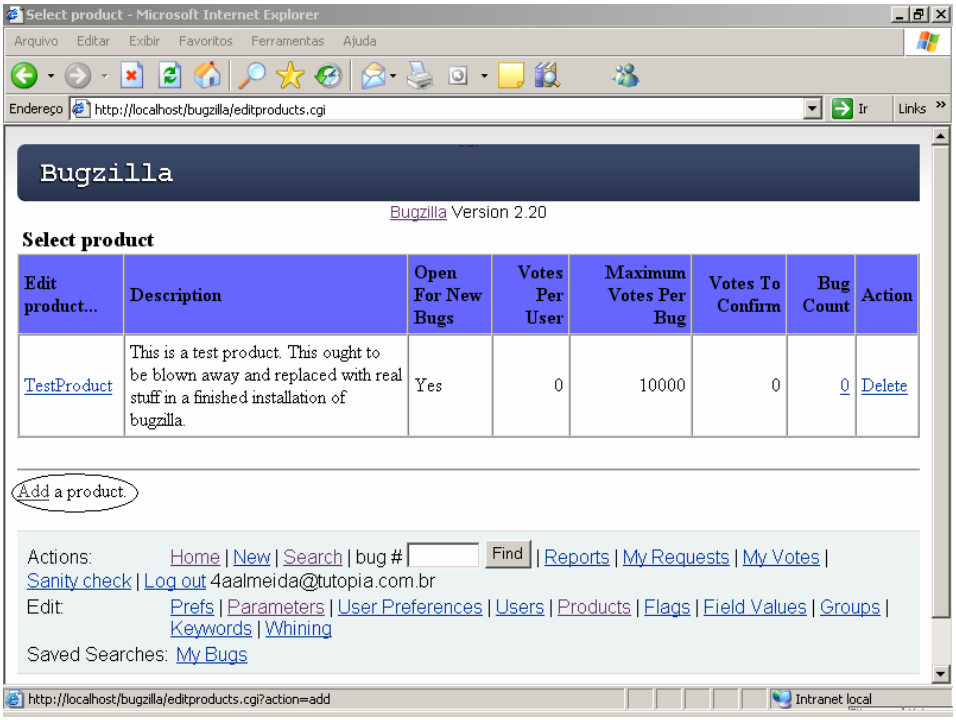


Figura Tutorial 21 - Produtos (projetos) cadastrados

Algumas informações básicas devem ser fornecidas a respeito do sistema, conforme indicado na figura 22:

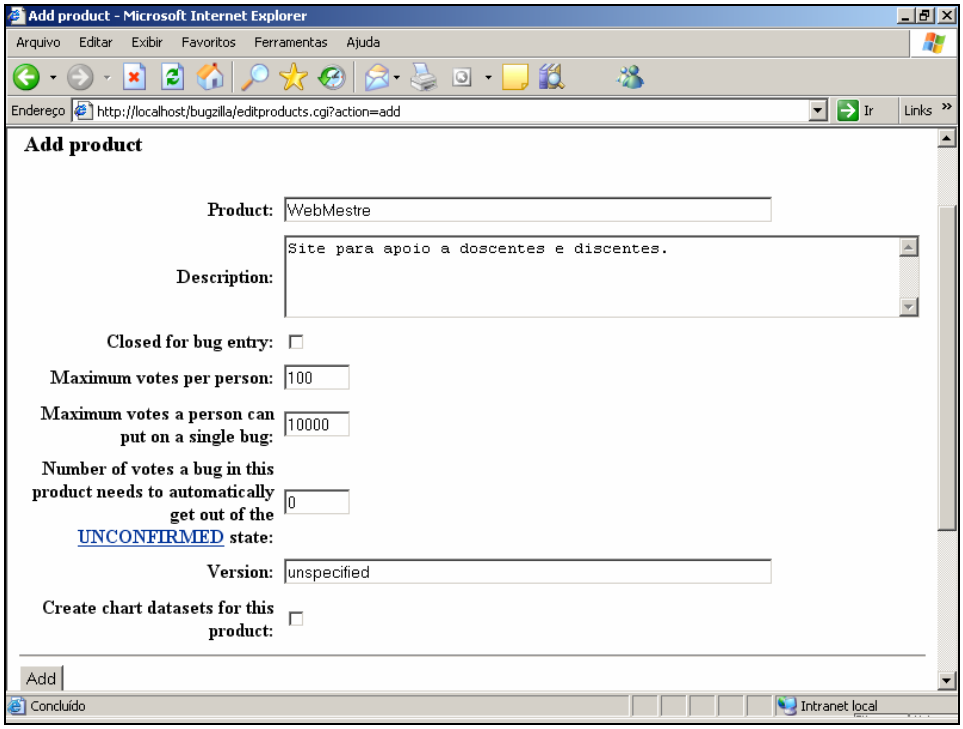


Figura Tutorial 22 - Informações sobre o projeto

Após terem sido informados os dados básicos, o administrador seleciona o botão "*Add*" na base da tela, sendo enviado para a tela de informação abaixo, onde é indicado que pelo menos um componente do produto em questão deve ser cadastrado para que se possa cadastrar uma nova CR:

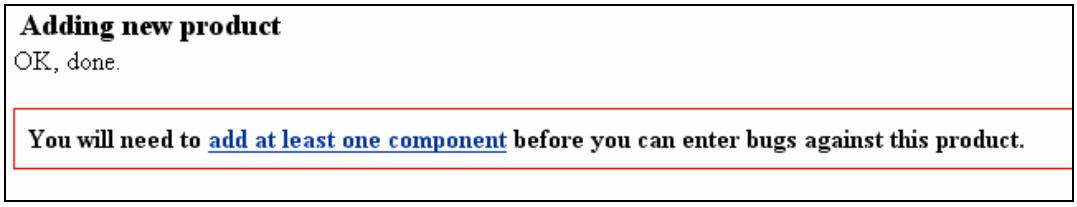


Figura Tutorial 23 - Confirmação de cadastro do novo produto

- ## Cadastro do componente do sistema

O administrador cadastrará o componente, selecionando o link "*add at least one component*", sendo redirecionado para a tela indicada na figura 24:

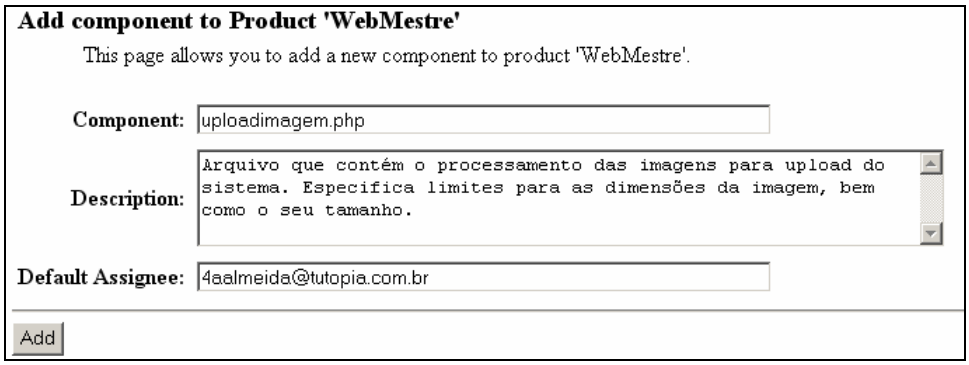


Figura Tutorial 24 - Cadastro de Componente

Serão preenchidos os dados do componente que está sendo alterado com exemplo deste tutorial, o arquivo "*uploadimagem.php*", sendo que ao final, o botão "*Add*" necessita ser acionado.

Ao ser acionado o botão "*Add*", o navegador será redirecionado para a tela de confirmação, conforme indicado na figura 25:

Figura Tutorial 25 - Confirmação de Cadastro de Componente

- ## Cadastro de Usuários

Para que uma CR possa ser atribuída, um conjunto de usuários, normalmente pertencentes ao time de desenvolvimento, deverá ser cadastrado. Para tal, o menu "*Users*" da tela principal do Bugzilla deverá ser acionado, sendo redirecionado para a tela indicada na figura 26:

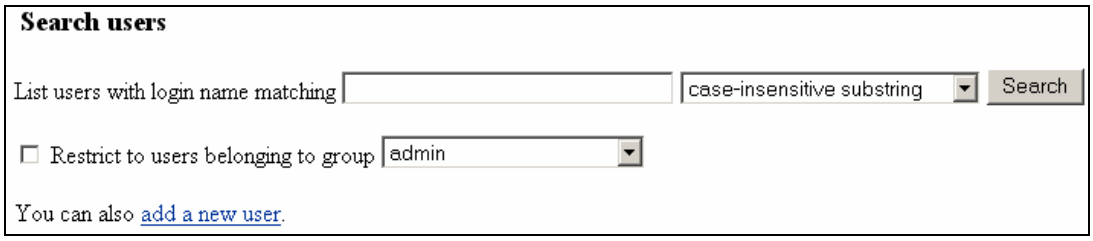


Figura Tutorial 26 - Pesquisando Usuários

A partir desta tela, o administrador poderá selecionar o link "*add new user*", sendo redirecionado para a tela a "*Add User*":

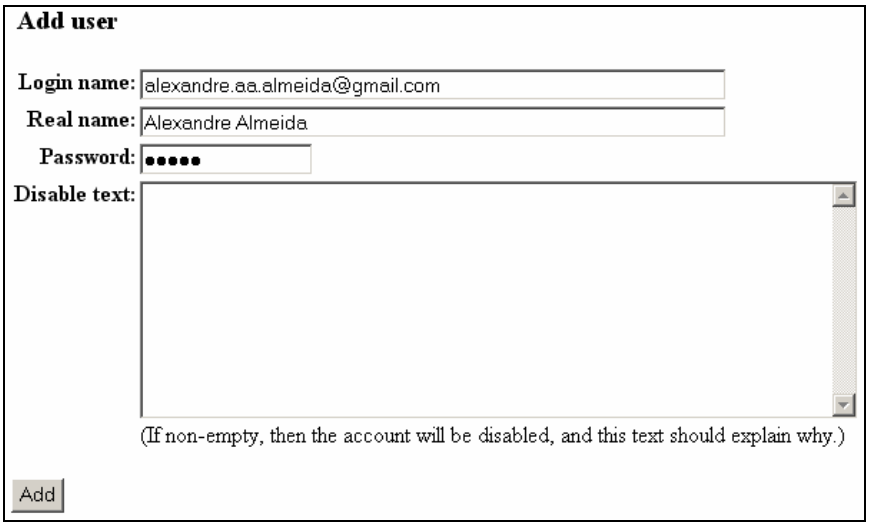


Figura Tutorial 27 - Cadastro de Usuários

Nesta tela, o administrador cadastra os dados do desenvolvedor, colocando como "*login*" o endereço de "*e-mail*" do usuário.

Uma vez cadastrado o usuário, a seguinte tela será apresentada, onde o administrador poderá cadastrar outras informações sobre o usuário e as suas operações permitidas. As opções abaixo compreendem uma sugestão de configuração para o usuário recém cadastrado. Maiores informações sobre estas opções podem ser encontradas no manual do Bugzilla, referenciado na bibliografia, estando fora do escopo deste tutorial.

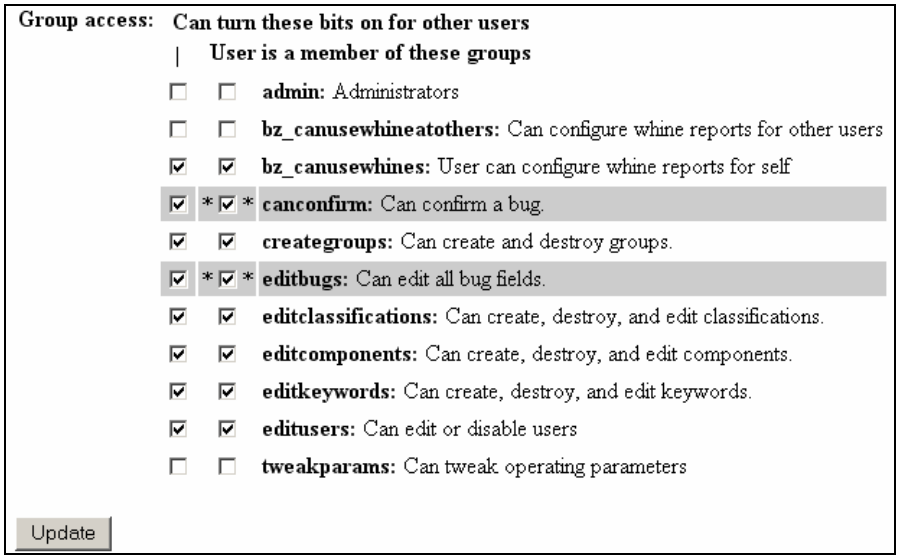


Figura Tutorial 28 - Permissões para o Usuário/Grupo

O botão "*Update*" deve ser selecionado para que as opções sejam salvas. Após o acionamento, a tela indicada na figura 29 é apresentada, indicando o cadastro com sucesso.

User alexandre.aa.almeida@gmail.com updated

The following changes have been made to the user account alexandre.aa.almeida@gmail.com:

- The account has been added to the bz_canusewhines, canconfirm, creategroups, editbugs, editclassifications, editcomponents, editkeywords, editusers groups.
- The account has been granted rights to bless the bz_canusewhines, canconfirm, creategroups, editbugs, editclassifications, editcomponents, editkeywords, editusers groups.

Figura Tutorial 29 - Confirmação de atualização

- ## Criando uma nova CR

Para que seja criada uma nova CR no Bugzilla, a partir da tela inicial, o interessado poderá selecionar o link *"New"*. Então, poderá selecionar o produto em questão, conforme indica a figura abaixo:

Enter Bug First, you must pick a product on which to enter a bug.

TestProduct: This is a test product. This ought to be blown away and replaced with real stuff in a finished installation bugzilla.

WebMestre: Site para apoio a doscentes e discentes.

Figura Tutorial 30 - Selecionando o Produto

Ao selecionar "*WebMestre*", o navegador será redirecionado para a página de cadastro de Novo Bug (Nova CR):

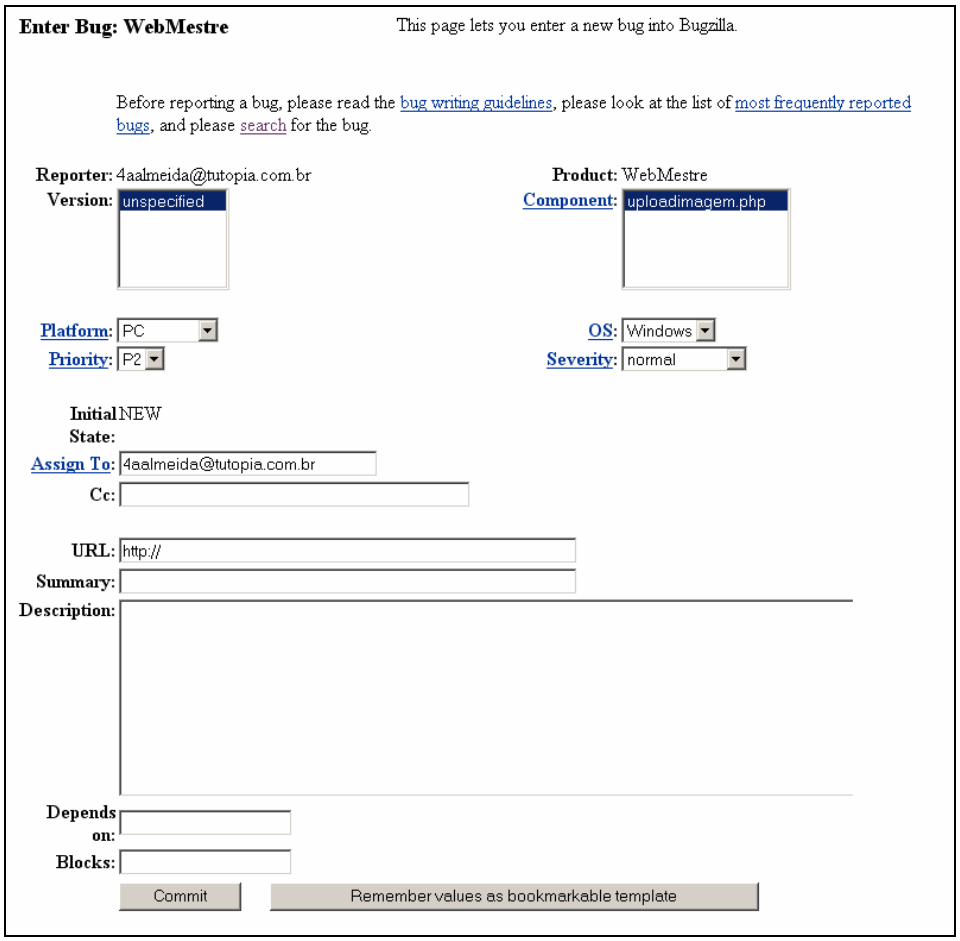
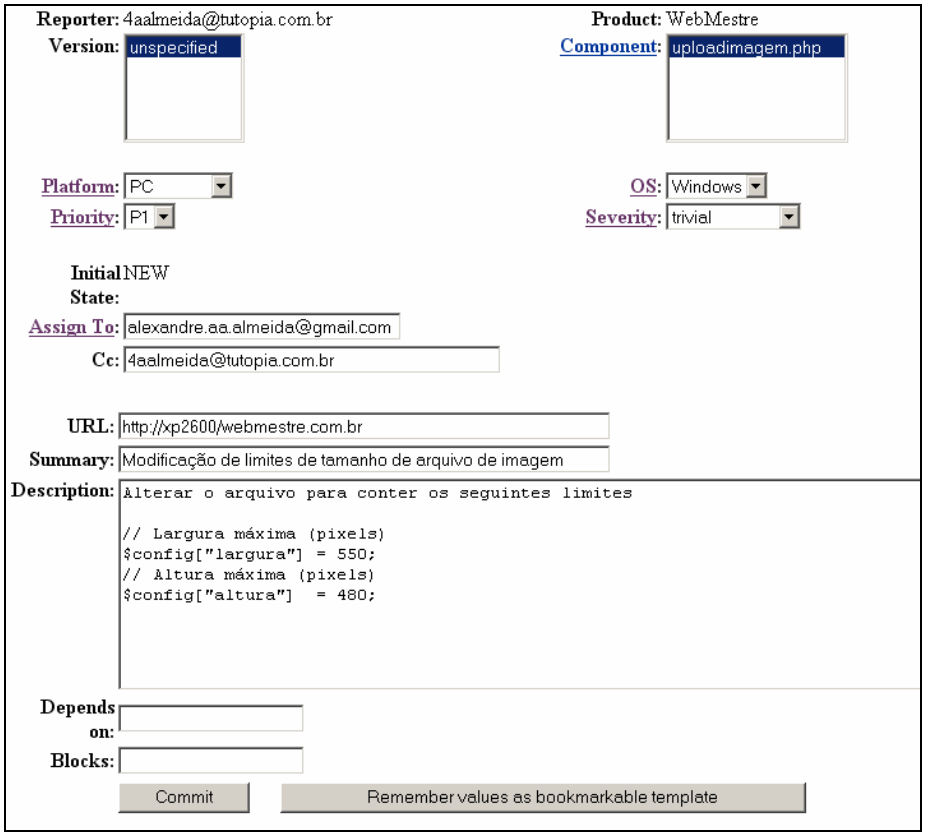


Figura Tutorial 31 - Cadastro de uma nova CR (*Bug*)

Os dados referentes às alterações propostas pelo tutorial são preenchidos, conforme indicado abaixo:



O campo *"Assigned To"* é de extrema importância, pois será este usuário cadastrado que receberá o "*e-mail*" para solução da CR em questão.

Após enviar a CR, a seguinte tela será apresentada, com o número do BUG (CR) atribuído, sendo definido o estado da CR como Nova (*New).*

Na mesma tela, o estado da CR poderá ser alterado, conforme indica a figura 32:

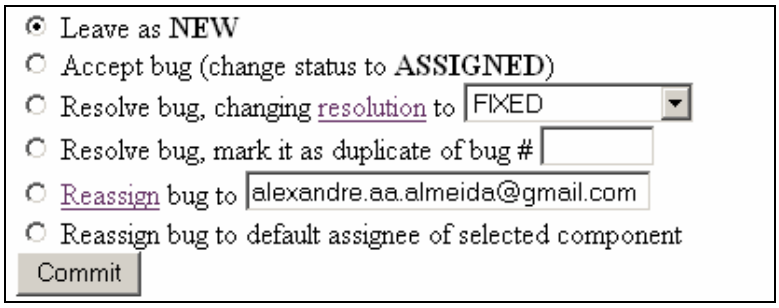


Figura Tutorial 32 - Estado da CR

Quando uma nova CR dá entrada no Bugzilla, seu estado inicial é definido como "*New*", até que alguém aceite formalmente a CR, sendo então definida para *"ASSIGNED"*. Nesta tela, a solução da CR poderá ser efetivada, podendo ser definido um dos seis estados para resolução, conforme indicado na figura 33:

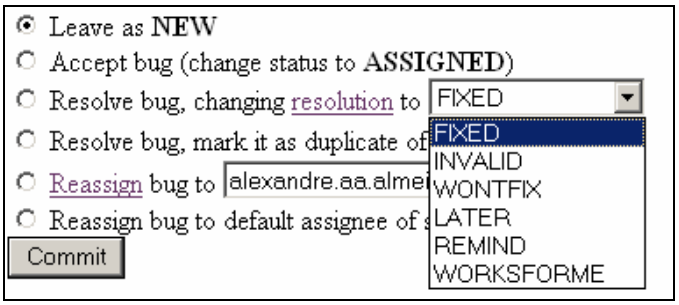


Figura Tutorial 33 - Tipos de Resolução de CR

O desenvolvedor receberá um *"e-mail"* informando sobre a nova CR, devendo o mesmo efetuar *"login"* no Bugzilla para verificar a CR.

- **Modificação do Estado da CR**

A partir deste momento, será assumido que o papel em questão será o do desenvolvedor ao qual foi atribuída a nova CR:

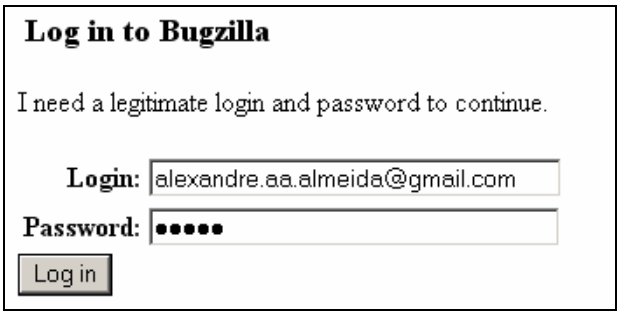


Figura Tutorial 34 - Desenvolvedor efetuando *Login*

Após o *"login"* ter sido realizado com sucesso, o desenvolvedor deverá localizar a CR a si atribuída, através do link *"Search existing bug reports"* indicado na figura 35:

Figura Tutorial 35 - Página principal do Bugzilla

Em seguida, selecione os dados do sistema em questão e acione o botão *"Search"*.

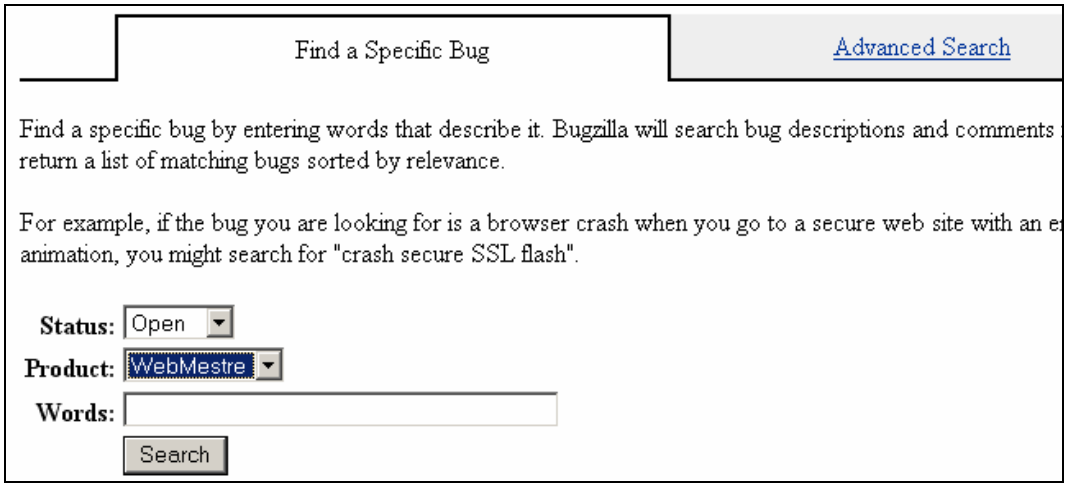


Figura Tutorial 36 - Página de Pesquisa de CR

Relação dos CR (*bugs*) cadastradas:

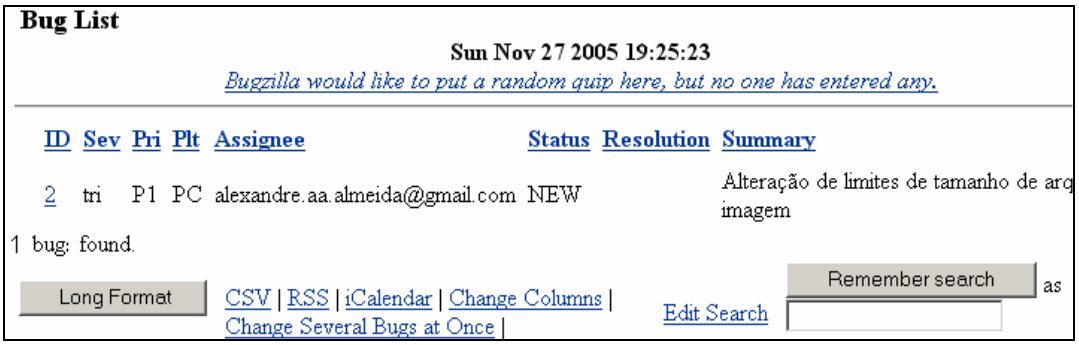


Figura Tutorial 37 - CR cadastradas

Neste momento, o desenvolvedor poderá aceitar a atribuição da CR (bug), alterando o estado da solicitação para "*Assigned*", ou seja, "Atribuído", conforme indica a figura 38:

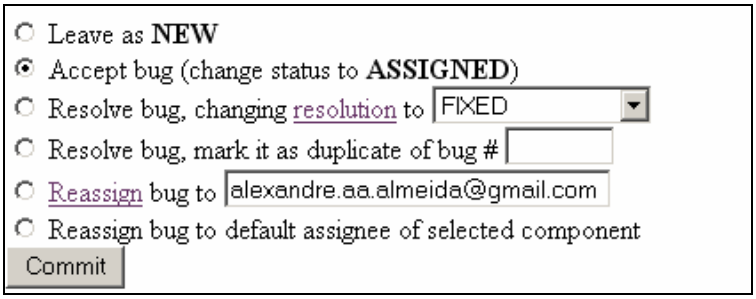


Figura Tutorial 38 - CR Atribuída

Após a atualização do estado da CR, o desenvolvedor deverá agora realizar as alterações no projeto, conforme previsto na CR, fazendo o "*check-out*" através do Eclipse. Após as devidas modificações e verificações por parte dos elementos de Garantia da Qualidade, uma vez estando aprovadas as modificações, o desenvolvedor deverá atualizar a CR, alterando seu estado para "Fixed" (corrigido), conforme a figura 39:

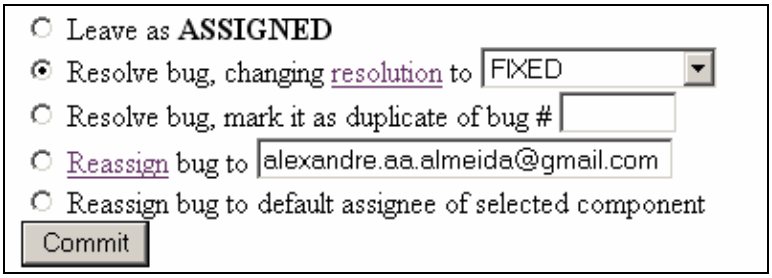


Figura Tutorial 39 - Alterando resolução para Corrigida (*Fixed*)

- **Fechamento da CR**

Após a alteração do estado da CR, ela ficará na situação "*Resolved*" (resolvido), com a resolução definida como "*Fixed*" (corrigido).

Figura Tutorial 40 - Fechamento da CR

Após esta CR ter sido solucionada, a mesma poderá ser fechada, ou então reaberta, como indica a figura 41:

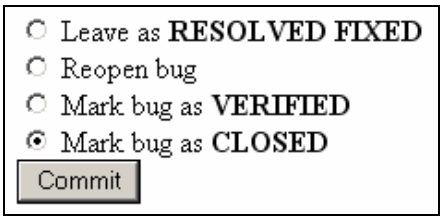


Figura Tutorial 41 - Fechamento da CR

A partir deste momento a CR torna-se Fechada, encerrando neste momento o fluxo do processo que corresponde ao Controle de Mudanças.

O desenvolvedor poderá agora realizar o "*check-in*" de suas alterações no repositório do CVS, utilizando para isso a ferramenta Eclipse, citada anteriormente.

APÊNDICE B – IDENTIFICAÇÃO DOS PRODUTOS DO PROJETO

B.1 Estrutura de Identificação de Projetos de Software

Cada projeto a ser desenvolvido deverá possuir uma módulo (diretório) correspondente no repositório. Cada diretório de projeto deverá ser dividido em subdiretórios que serão um espelho da estrutura de diretórios utilizada pelos desenvolvedores do projeto. Uma estrutura adicional poderá ser composta pelas linhas base de cada projeto. Um mesmo item de configuração pode pertencer a mais de uma linha base, desde que permaneça sem alteração ao longo dos marcos do projeto.

Os IC podem ser identificados com as seguintes informações:

a) Número de revisão (número interno de revisão);

b) Nome do IC (equivalente ao nome do arquivo);

c) Zero ou mais nomes de diretórios a que pertence;

d) Nome do projeto em que está inserido;

e) Nome da linha base a que pertence.

B.2 Estrutura de identificação dos produtos de software

Os produtos gerados pelo projeto serão identificados com até 3 números inteiros, separados por ponto (X.Y.Z). Ex.: 2.1.10. Esta numeração identifica cada versão do artefato em questão. Parte desta numeração é controlada pelo sistema automatizado de controle de versões. A primeira parte da numeração é controlada pelo gerente de configuração. Não há partilha de IC entre projetos. Se for o caso, o IC deve ser duplicado, entrando na linha base do respectivo projeto.

A identificação das versões dos produtos tem como base classificações dos projetos, como exemplificado abaixo:

a) Novo projeto de desenvolvimento: criação de um novo produto;

b) Projeto de evolução: acréscimo de novas funcionalidades;

c) Projeto de Melhoria: acréscimo de pequenas melhorias;

d) Projeto de Transporte: transporte para outro ambiente operacional, sem variação de funcionalidade.

As versões serão então indicadas por números primário (X), secundário (Y) e terciário (Z). O número terciário poderá ser omitido quando for zero.

A numeração poderá variar, conforme a política adotada pela corporação, podendo ser a seguinte:

Projeto de evolução: número primário incrementado em uma unidade, e os demais zerados. Se o projeto for de melhoria, a alteração será no número secundário, mantendo-se o primário e zerando o terciário. Para projetos de transporte, o número terciário será incrementado, mantendo os outros números.

Rótulos poderão ser aplicados de forma a agrupar um conjunto de artefatos com diversas versões. Os rótulos identificam determinadas versões de IC. O conjunto de artefatos que constitui a versão de um subsistema é coletiva e individualmente identificável por uma determinada versão e rótulo.

B.3 Estrutura de Diretórios do Projeto

A estrutura de diretórios consiste de uma organização lógica para todos os artefatos de versão relacionados ao produto que se encontram sob gerência de configuração. Normalmente, um sistema desenvolvido de acordo com o processo unificado, possui o seu conjunto de artefatos organizados conforme o grupo de informações referentes às suas disciplinas (Modelagem de Negócios, Requisitos e assim sucessivamente). Se a organização for baseada em conjunto de informações correlatas, não será levado em consideração as fases de desenvolvimento do sistema.

O levantamento da estrutura de diretórios tem como base a arquitetura do software, que por sua vez é proveniente das atividades de Análise e Projeto. Após estas atividades, considera-se que a estrutura de diretórios esteja refinada, representando um bom começo para sua organização inicial.

A seguir, encontra-se representada a sugestão de uma organização inicial, de acordo com o conjunto de artefatos previstos no USDP.

Requisitos do Sistema	**Modelos**	Modelo de Casos de Uso	Pacote de Casos de Uso
		Protótipos de Interfaces de Usuário	
	Banco de Dados	Atributos de Requisitos	
	Documentos	Visão	
		Glossário	
		Especificações Suplementares	
		Especificação de Requisitos de Software	
Design e Implementação do Sistema	**Modelos**	Modelo de Análise	Realização de Casos de Uso
		Modelo de Design	Subsistema de Design
			Interface
			Pacote de Teste
		Modelo de Dados	
		Documento de Análise de Carga de Trabalho	
	Documentos	Documento de Arquitetura de Software	
		Relatório Sintético de Modelo de Design	
	Subsistema 1	Estrutura de Diretórios do Subsistema	
	Subsistema 2	Estrutura de Diretórios do Subsistema	
	Subsistema N	Estrutura de Diretórios do Subsistema	
Integração de Sistema	**Planos**	Plano de Integração	
	Bibliotecas		
Teste do Sistema	**Plano de Teste**	Conjunto de Testes	
	Casos de Teste	Scripts de Teste	
	Dados de Teste		
	Resultados do Teste		
Implantação de Sistema	**Plano de Implantação**		
	Documentos	Notas de Release	
	Manuais	Material de Suporte para o Usuário Final	
		Materiais de Treinamento	
	Artefatos de Instalação		

Quadro 11 - Sugestão de Estrutura de Diretórios

Gerenciamento de Sistema	**Planos**	Plano de Desenvolvimento de Software	
		Plano de Iteração	Plano de Gerenciamento de Requisitos
		Lista de Riscos	Plano de Gerenciamento de Riscos
		Caso de Desenvolvimento	Plano de Infra-estrutura
		Plano de Documentação	Plano de Avaliação de Qualidade
		Plano de Melhoria do Processo	Plano de Métricas
	Avaliações	Avaliação de Iteração	
		Avaliação da Organização de Desenvolvimento	
		Avaliação de Estado	
Padrões e Diretrizes	**Requisitos**	Atributos de Requisitos	
		Guia de Modelagem de Casos de Uso	
		Interface do Usuário	
	Design	Guia de Design	
	Implementação	Guia de Programação	
	Documentação	Manual de Guia de Estilo	

Quadro 11 – Continuação: Sugestão de Estrutura de Diretórios

Como pode ser observado na organização descrita no quadro 11, os diversos subsistemas que comporão o produto final estão representados recursivamente dentro da estrutura inicial. A estrutura principal se repete para cada subsistema, com estrutura de diretórios semelhante, porém com as informações necessárias ao seu próprio projeto e implementação. A figura seguinte mostra uma estrutura de diretórios possível para um dois sistemas (X e Y), sendo o sistema X formado por dois subsistemas (A e B).

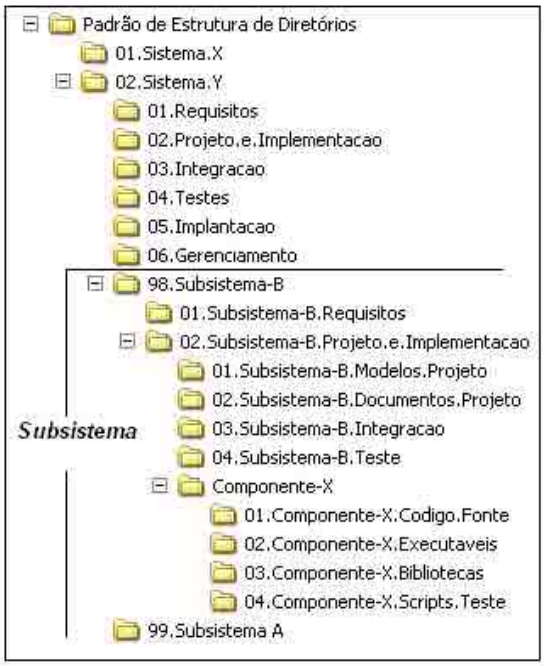


Figura 22 - Estrutura de Diretórios

Sugere-se que a numeração da pasta principal do subsistema comece com um valor bem maior que as demais pastas (Ex.: 99.Subsistema.A), sendo decrescente. Desta forma, os subsistemas ficam agrupados no final da hierarquia de pastas dos sistemas.

APÊNDICE C – PROCESSO DE CONTROLE DE MUDANÇAS: DETALHAMENTO

O presente apêndice tem por objetivo detalhar o processo de Controle de Mudanças. Para tal, será apresentada em detalhes a participação dos papéis envolvidos no processo. O tratamento da Solicitação de Mudança (CR) também será definido, tendo como base os possíveis estados de uma solicitação realizada a partir da ferramenta Bugzilla. O formato deste documento sugere a formatação de processo, possuindo seu propósito, escopo, definições, papéis e responsabilidades, o estado possível da CR (insumo para o processo, bem como resultante). Para a definição dos procedimentos, algumas pré-condições são exigidas, assim como pós-condições.

C.1 Introdução

Propósito	Este documento descreve o processo que será utilizado para solicitação e gerenciamento de mudanças dos produtos do trabalho de desenvolvimento do projeto em questão. Este processo irá facilitar a comunicação entre os envolvidos no processo, de maneira que seja provido um processo comum para solução de solicitações de mudanças e problemas relatados pelo sistema de CM.
Escopo	Qualquer elemento envolvido no projeto pode submeter os seguintes tipos de solicitações para o sistema de CM: • Mudanças de requisitos (adição, exclusão, modificação) no software atualmente em desenvolvimento; • Requisição de melhorias no produto; • Requisição para solução de erros; • Outros itens relevantes. Este processo de controle de mudanças aplica-se ao trabalho colocado sob gerência de configuração, na forma de linha base, criados ou gerenciados pelos membros do projeto, incluindo: • Programas que estão sendo liberados para produção ou em teste beta; • Especificação de Requisitos para o projeto; • Grupo de procedimentos e processos; • Documentação técnica e do usuário.

Definições	Termo	Definição
	CR	Requisição de Mudança (Change Request): Um documento enviado por qualquer elemento envolvido no projeto, normalmente em meio eletrônico, que informa os dados inerentes à solicitação de mudança no software.

C.2 Papéis e Responsabilidades

Papel	Descrição
Presidente do Comitê de Controle de Alterações (CCB)	Líder do Comitê de Controle de Alterações (Change Control Board), que possui a competência para dar a palavra final quando o CCB não entra em acordo a respeito de alguma CR; Designa os papéis de Avaliador e Modificador (das CR aprovadas) dentre os elementos da equipe.
Comitê de Controle de Alterações (CCB)	O grupo que decide sobre a aprovação ou rejeição das CR para um projeto específico.
Avaliador	O elemento que recebe a atribuição do Presidente do CCB para analisar o impacto da aprovação da CR.
Implementador	O elemento que recebe a responsabilidade para implementar as alterações nos artefatos, em resposta a uma CR aprovada pelo CCB. Deve manter o estado da CR sempre atualizado.
Solicitante	É o papel que representa o interessado na modificação. É quem propõem a solicitação de mudança (CR).
Gerente de Projeto	É responsável por todo o planejamento e acompanhamento das atividades do desenvolvimento do projeto.
Verificador	É o papel que assume o elemento responsável por determinar quando uma mudança foi realizada corretamente, conforme descrito na CR. Compreende a Garantia da Qualidade.

C.3 Estados da CR

Estados da CR	Uma CR irá passar por diversos estados durante seu ciclo de vida. Estes estados, bem como os critérios para transição entre os mesmos, são representados no diagrama de transição de estados da figura 23, e descritos no quadro de descrição dos "**Estados Possíveis**".
Notificações	Sempre que o estado de uma CR é modificado, a ferramenta de controle de alterações enviará uma notificação automática via e-mail ao Solicitante, ao Implementador, bem como ao Presidente do CCB.

Estados Possíveis

Estados	Significado
Não Confirmada (Unconfirmed) *	A CR fica aguardando mais informações para que seja processada.
Nova (New)*	Uma Nova CR dá entrada no sistema. Ainda não foi atribuída nenhuma tarefa a nenhum integrante do projeto.
Avaliada (Evaluated)*	O Avaliador executou a análise de impacto da CR.
Aprovada (Aproved)*	A CR será implementada e alocada na próxima liberação/construção do produto.
Atribuída (Assigned)*	A CR foi atribuída ao responsável que deverá implementar a mudança nela indicada.
Modificada (Modified)*	O Implementador executou a implementação do previsto na CR.
Rejeitada	O CCB decidiu não implementar as previstas na CR.
Verificada	O Verificador confirmou que as alterações nos artefatos a que se refere a CR foram realizadas corretamente.
Fechada	O produto gerado pela execução da CR foi verificado, foi integrado, sendo as que as modificações definidas na CR são consideradas fechadas.
Reaberta	Quando a solução da CR não foi correta, por qualquer motivo, a mesma poderá ser reaberta.
(*) Os termos em inglês correspondem à nomenclatura do Bugzilla	

C.4 Resoluções

Resoluções Possíveis

Resolução	Significado
Inválida (Invalid)	O problema descrito não corresponde a um erro ou não necessita correção, bem como nenhuma implementação é requerida.
Corrigida (Fixed)	O problema foi apresentado na CR foi solucionado.
Não será corrigida (Wontfix)	O problema apresentado na CR nunca será corrigido.
Adiada (Deferred)	A CR contém uma solicitação que somente será atendida numa versão futura do produto.
Duplicada (Duplicate)	A CR contém uma solicitação duplicada, já registrada em outra CR. O Número da CR duplicada deve ser informado na descrição da CR.
Não Reproduzível (Worksforme)	Todas as tentativas de reproduzir o problema apresentado na CR foram infrutíferas, não sendo possível sua correção. Se mais informações forem fornecidas, a CR poderá ser reaberta e reatribuída.
Movida (Moved)	O problema foi relatado para outro produto relacionado em outra base de dados de controle de mudanças. A CR foi movida para a referida base.

C.5 Procedimentos

Critérios de Entrada	☐ O CCB está estabelecido para o projeto. ☐ Os artefatos que farão parte da linha base existem e estão definidos. ☐ O solicitante enviou a CR com todas as informações requeridas pelo Presidente do CCB. ☐ A CR ao dar entrada na ferramenta de controle de mudanças, assume o estado de "Não Confirmada".
Tarefas	1. O presidente do CCB atribui um avaliador para avaliar o impacto da CR 2. O avaliador certifica a viabilidade, se realmente pertence ao projeto, se o problema relatado pode ser reproduzido (se for um erro, por exemplo), a estimativa em horas de trabalho para execução da modificação. Para mudança de requisitos, uma análise de impacto deve ser realizada para cada requisito modificado. O estado da CR torna-se "Avaliada". 3. O CCB decide se a modificação prevista na CR deverá ser realizada neste momento ou futuramente, ou não será realizada. Outras informações devem ser obtidas de outros interessados que porventura sejam afetados pela modificação em questão, antes de tomar a decisão. 4. Se a CR for aceita, seu estado será alterado para "Aprovada". O Presidente do CCB atribui a CR a determinado implementador, definindo seu estado para "Atribuída". O Gerente de Projeto negocia as mudanças necessárias no projeto juntamente como os elementos afetados pela mudança (se for o caso). 5. Se a solicitação for rejeitada, o Presidente do CCB altera o estado da CR para "Rejeitada" e informa os motivos no campo apropriado da ferramenta. Os envolvidos devem ser notificados. 6. O Presidente do CCB e o Solicitante determinam se a verificação formal da mudança será requerida, seguindo os procedimentos previstos na seção de Verificação. Se for requerida, o Presidente do CCB designa o Verificador para executá-la. 7. O Implementador executa as mudanças necessárias e notifica as demais partes afetadas se outras mudanças em artefatos relacionados também necessitarem modificações, tais como documentação, ajuda ou testes. O estado da CR é definido como "Modificada" 8. Se for verificado que a modificação é impraticável, o implementador notifica o Presidente do CCB. Este atribuirá a resolução como "Não será corrigida", ou outra resolução similar, tal como "Não Reproduzível", por exemplo. O Implementador recupera a versão disponível na última linha base. A ferramenta de controle de mudanças notifica o solicitante, o Presidente do CCB e o Gerente de Projeto. 9. Quando a modificação estiver completa, o Implementador define o estado da CR para "Modificada", atualizando a base de dados com a resposta adequada de acordo com os atributos da CR. Informa ainda o esforço (em horas de trabalho) que foi empregado na solução da CR. A ferramenta de controle de mudanças notifica o solicitante e o Presidente do CCB para que seja realizada a verificação.
Verificação	1. O Implementador notifica o Solicitante e o Verificador (se alguém possuir esta atribuição) que a mudança foi realizada e disponibiliza todos os artefatos aos responsáveis pelo processo de verificação. 2. O Verificador executa os passos necessários para garantir a aprovação da alteração implementada.

3. Se a verificação for realizada com sucesso, o Verificador define o estado da CR como “Verificada”. O Solicitante e o Implementador são notificados.
4. Se a verificação detectar alguma falha na implementação, não obtendo sucesso, o Verificador define o estado da CR para “Reaberta” descrevendo o problema no atributo respectivo da CR. O Solicitante e o Implementador são notificados pela ferramenta de controle de alterações. O processo continua na tarefa número 4, para uma reatribuição.
5. Para uma CR correspondente a solução de problemas, ou melhoria, o Implementador após a aprovação da verificação, submete os artefatos ao sistema de controle de versões, atualizando as linhas bases. Para mudanças de requisitos, o Implementador atualiza os números de versões de todos os artefatos relacionados à modificação, preparando para uma nova liberação do produto, compondo uma nova linha base.
6. O Presidente do CCB define o estado da CR para “Fechado”. A ferramenta de controle de mudanças notifica o Solicitante.

Relato do Estado da Configuração

O Gerente de Controle de Mudanças providencia a emissão de relatórios, de acordo com a periodicidade prevista no PGC, totalizando o estado de todo o conteúdo do sistema de controle de mudanças. Estes relatórios identificam todas as mudanças realizadas no período em questão (normalmente um mês), enumerando o estado de todas as CR que atualmente possuem estado diferente de “Fechada”. Este relatório torna-se um indicador do nível de atividades de modificação. O Gerente de Projeto revisa estes relatórios para determinar se alguma ação corretiva é necessária.

Critérios de Saída

☐ O estado da CR é definido como Fechada.
☐ Os artefatos sob gerência de configuração, que foram modificados em resposta à respectiva CR foram liberados.
☐ O Solicitante, o Presidente do CCB e o Gerente de Projeto receberam notificações sobre o estado atual da CR.
☐ As informações referentes à rastreabilidade de requisitos foram devidamente atualizadas, se for o caso.

A figura 23 representa um diagrama de estados de uma CR. Nem todas as transições são representadas neste diagrama, pois são inúmeras as combinações possíveis.

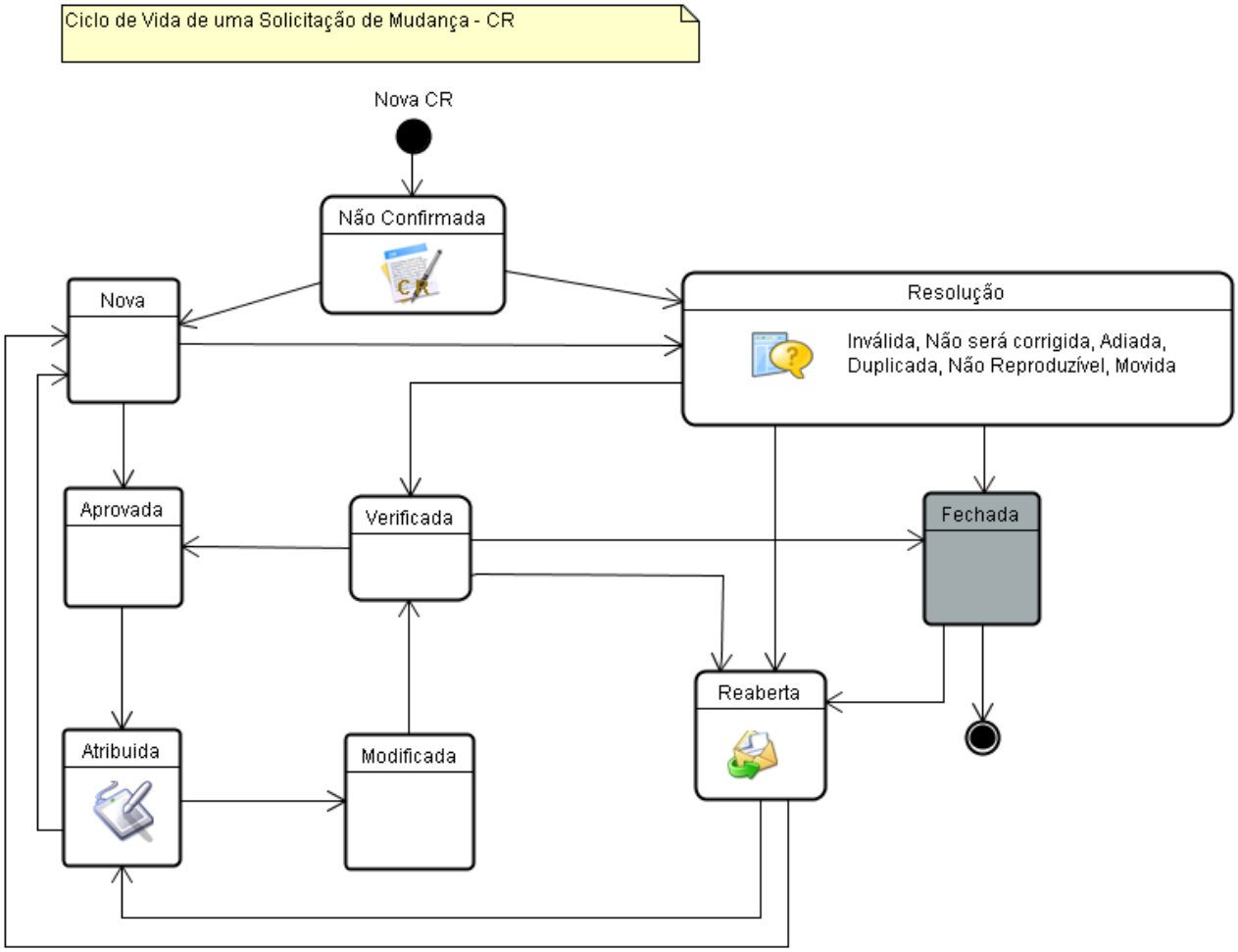


Figura 23 - Exemplo de Ciclo de Vida de uma CR

A figura 24, que representa um diagrama de atividades, contém o instanciamento de quatro papéis envolvidos no processo de controle de mudanças. O processo não se restringe a estes papéis, podendo ser adaptado conforme a necessidade da organização. Por exemplo, ao Gerente de Projeto poderia ser atribuída a tarefa de atribuir a CR ao Implementador, desonerando o CCB ou seu representante de tal tarefa.

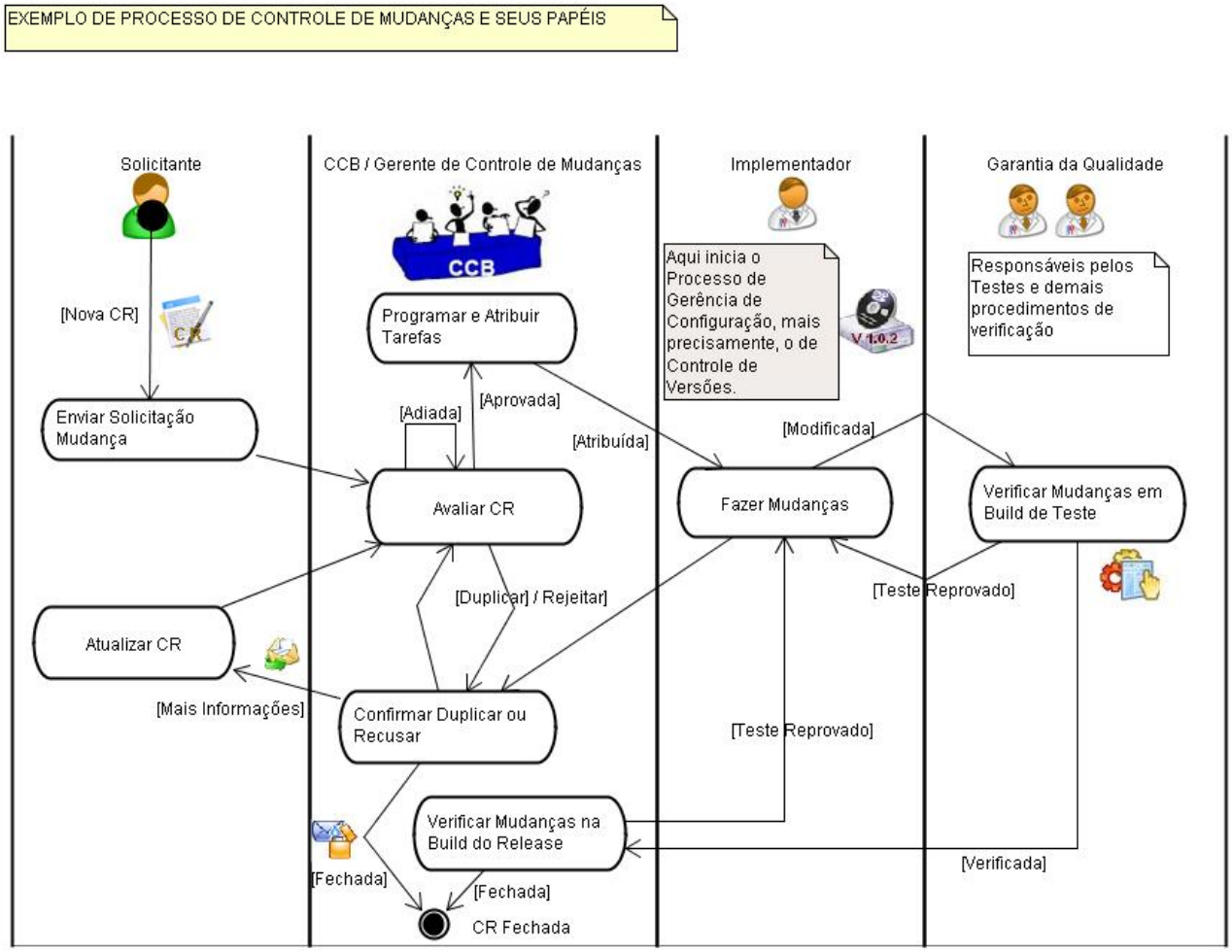


Figura 24 - Diagrama de Atividades de Controle de Mudanças.

APÊNDICE D – QUESTIONÁRIO DE AVALIAÇÃO DA EMPRESA E DO PERFIL DO AVALIADOR

OBJETIVO DO QUESTIONÁRIO: Levantar informações sobre a Empresa quanto a existência de um processo de Gerência de Configuração e Mudança.

Data: _____/_____/_____

Empresa:_____ Departamento: _____

1. A sua empresa possui Processo de Gerência de Configuração e Mudança?

   ☐ Não ☐ Sim.

2. A sua empresa utiliza alguma ferramenta de Controle de Configuração e Mudança?

   ☐ Não ☐ Sim. Quais? _____

   _____

3. A sua empresa possui algum plano de implantar um Processo de Gerência de Configuração e Mudança?

   ☐ Não ☐ Sim.

   3.1 - Caso afirmativo, em que estágio se encontra?

   ☐ Inicial ☐ Em estudo ☐ Projeto Piloto

4. Quais são os principais fatores que impedem a implantação de um Processo de Gerência de Configuração na sua empresa atualmente?

   ☐ Organizacionais ☐ Tecnológicos

   ☐ Financeiros ☐ Culturais

   ☐ Políticos / Estratégicos ☐ Recursos Humanos (Qualificação / Quantitativo)

   ☐ Outros: _____

5. Dentre os papéis citados na apresentação, em qual deles você estaria se enquadrando?

   ☐ CCB ☐ Gerente de Configuração ☐ Gerente Controle de Mudanças

   ☐ Solicitante ☐ Implementador ☐ Auditor ☐ Integrador ☐ Revisor

   ☐ Nenhum

6. Algo a considerar sobre o que foi apresentado?

APÊNDICE E – QUESTIONÁRIO DE AVALIAÇÃO DO PGCM

1. Dados do Avaliador:

Nome: _____

Departamento: _____

Função: _____

Cargo: _____

Data Avaliação: _____

2. Objetivo do Questionário:

- Avaliar todos os itens presentes no processo de Gerência de Configuração e Mudança – PGCM.

3. Orientações importantes:

- Para avaliar cada item do processo, o avaliador deverá marcar um 'X' no valor correspondente;
- A avaliação é quanto à utilidade do item do processo de Gerência de Configuração e Mudança proposto;
- O Avaliador poderá fazer considerações sobre o valor atribuído e/ou sobre o item avaliado;
- Caso exista algum item importante que não esteja descrito no PGCM, o Avaliador poderá descrevê-lo na seção "5 - Itens ausentes".
- A seqüência dos itens no questionário é a mesma apresentada no PGCM.
- Caso o espaço para considerações não seja suficiente, o Avaliador poderá utilizar a seção "7 – Considerações Complementares" para finalizar as suas considerações.

4. Valores de avaliação:

- (D) - Desnecessário
- (P) - Pouco útil
- (U) - Útil
- (M) - Muito útil

5. Questionário de Avaliação

Item	Classificação	Descrição	Avaliação	Considerações
1	**Papel**	**Gerente de Configuração**	**()** D **()** P **()** U **()** M	
2	Atividade	Configurar Ambiente de Gerência de Configuração	**()** D **()** P **()** U **()** M	
3	Tarefa	Definir o Ambiente de Hardware do CM	**()** D **()** P **()** U **()** M	
4	Tarefa	Mapear a Arquitetura para o Repositório	**()** D **()** P **()** U **()** M	
5	Tarefa	Criar Conjunto Inicial de Elementos Versionados.	**()** D **()** P **()** U **()** M	
6	Tarefa	Definir Níveis de Promoção da Baseline	**()** D **()** P **()** U **()** M	
7	Atividade	Estabelecer Políticas de Gerenciamento de Configuração (CM)	**()** D **()** P **()** U **()** M	
8	Tarefa	Definir Práticas de Identificação de Configuração	**()** D **()** P **()** U **()** M	
10	Tarefa	Definir Práticas de linhas base	**()** D **()** P **()** U **()** M	
11	Tarefa	Definir Práticas de Arquivamento	**()** D **()** P **()** U **()** M	
12	Tarefa	Definir Requisitos de Relatório de Estado da Configuração	**()** D **()** P **()** U **()** M	
13	Atividade	Escrever o Plano de Gerenciamento de Configuração	**()** D **()** P **()** U **()** M	
14	Tarefa	Escrever o PGC	**()** D **()** P **()** U **()** M	
15	Tarefa	Revisar e Aprovar o PGC	**()** D **()** P **()** U **()** M	
16	Tarefa	Manter o Plano PGC	**()** D **()** P **()** U **()** M	
17	Atividade	Preparar Unidade de Implantação	**()** D **()** P **()** U **()** M	
18	Tarefa	Gerar cópia dos produtos liberados	**()** D **()** P **()** U **()** M	
19	Tarefa	Preparar material de suporte para o usuário	**()** D **()** P **()** U **()** M	
20	Tarefa	Preparar artefatos para instalação	**()** D **()** P **()** U **()** M	
21	Atividade	Relatar o Estado da Configuração	**()** D **()** P **()** U **()** M	
22	Tarefa	Coletar dados e relatar o Estado da Configuração	**()** D **()** P **()** U **()** M	
23	Tarefa	Relatar o estado da implementação das CR	**()** D **()** P **()** U **()** M	
24	Tarefa	Fornecer relatórios e controle de defeitos, de forma a facilitar a revisão	**()** D **()** P **()** U **()** M	
25	Atividade	Realizar Auditoria de Configuração	**()** D **()** P **()** U **()** M	
26	Tarefa	Executar Auditoria de Configuração Física (integridade dos artefatos).	**()** D **()** P **()** U **()** M	
27	Tarefa	Executar Auditoria de Configuração Funcional (requisitos funcionais).	**()** D **()** P **()** U **()** M	
28	Tarefa	Relatar descobertas e tomar ações corretivas	**()** D **()** P **()** U **()** M	
29	**Papel**	**Gerente de Controle de Mudanças**	**()** D **()** P **()** U **()** M	
30	Atividade	Estabelecer Processo de Controle de Mudanças	**()** D **()** P **()** U **()** M	
31	Tarefa	Estabelecer o Processo de Solicitação	**()** D **()** P **()** U **()** M	

		de Mudança		
32	Tarefa	Estabelecer o Comitê de Controle de Mudança.	() D () P () U () M	
33	Tarefa	Definir Protocolos para Notificação de Revisão de Mudança (adaptar conforme a ferramenta utilizada).	() D () P () U () M	
34	Atividade	Revisar CR	() D () P () U () M	
35	Tarefa	Programar Reunião de Revisão do CCB.	() D () P () U () M	
36	Tarefa	Recuperar CR para Revisão.	() D () P () U () M	
37	Tarefa	Revisar CR Enviadas.	() D () P () U () M	
38	Atividade	Confirmar CR Duplicada ou Rejeitar uma CR	() D () P () U () M	
39	Tarefa	Recuperar o Formulário de Solicitação de Mudança	() D () P () U () M	
40	Tarefa	Confirmar Duplicação ou Validade	() D () P () U () M	
41	Tarefa	Atualizar o Estado da Solicitação de Mudança.	() D () P () U () M	
42	**Papel**	**Integrador**	() D () P () U () M	
43	Atividade	Criar Espaços de Trabalho de Integração	() D () P () U () M	
44	Tarefa	Criar o Espaço de Trabalho de Integração	() D () P () U () M	
45	Atividade	Criar Linha Base	() D () P () U () M	
46	Tarefa	Criar uma nova Linha Base	() D () P () U () M	
47	Atividade	Promover Linha Base	() D () P () U () M	
48	Tarefa	Escolher o rótulo apropriado para a Linha Base	() D () P () U () M	
49	**Papel**	**Qualquer Papel**	() D () P () U () M	
50	Atividade	Criar Espaços de Trabalho de Desenvolvimento	() D () P () U () M	
51	Tarefa	Criar Espaço de Trabalho de Desenvolvimento	() D () P () U () M	

52	Atividade	Fazer Mudanças	() D () P () U () M	
53	Tarefa	Fazer check-out dos arquivos que serão alterados	() D () P () U () M	
54	Tarefa	Fazer as mudanças	() D () P () U () M	
55	Tarefa	Fazer testes unitários para verificar as mudanças	() D () P () U () M	
56	Tarefa	Submeter as mudanças ao Integrador para aprovação	() D () P () U () M	
57	Tarefa	Após aprovação, fazer check-in das mudanças.	() D () P () U () M	
58	Atividade	Liberar mudanças	() D () P () U () M	
59	Tarefa	Preparar mudanças para liberação	() D () P () U () M	
60	Tarefa	Liberar mudanças (check-in).	() D () P () U () M	
61	Tarefa	Atualizar Estado da Solicitação de Mudança (CR).	() D () P () U () M	
62	Atividade	Atualizar Espaço de Trabalho	() D () P () U () M	
63	Tarefa	Atualizar Espaço de Trabalho.	() D () P () U () M	
64	Atividade	Enviar Solicitação de Mudança (CR)	() D () P () U () M	
65	Tarefa	Preencher Formulário de Solicitação de Mudança	() D () P () U () M	
66	Tarefa	Enviar Solicitação de Mudança.	() D () P () U () M	
67	Atividade	Atualizar Solicitação de Mudança (CR)	() D () P () U () M	
68	Tarefa	Recupera o formulário de Solicitação de Mudança	() D () P () U () M	
69	Tarefa	Atualizar e reenviar a Solicitação de Mudança.	() D () P () U () M	

6. Itens ausentes

Classificação	Descrição	Considerações

7. Considerações complementares

Neste local, o Sr(a) poderá relacionar quaisquer considerações relevantes para a melhoria do processo. Também poderá propor mudanças de atividades ou tarefas entre papéis, bem como propor a inclusão e/ou exclusão de novos papéis.

Item	Classificação	Descrição	Considerações

Observações:_____

_____

_____

APÊNDICE F – QUESTIONÁRIO DE AVALIAÇÃO DO PCM

1. Dados do Avaliador:

Nome: _____

Departamento: _____

Função: _____

Cargo: _____

Data Avaliação: _____

2. Objetivo do Questionário:

- Avaliar todos os itens presentes no processo de Gerência de Configuração e Mudança – PGCM.

3. Orientações importantes:

- Para avaliar cada item do processo, o avaliador deverá marcar um 'X' no valor correspondente;
- A avaliação é quanto à utilidade do item do Processo de Controle de Mudanças proposto;
- O Avaliador poderá fazer considerações sobre o valor atribuído e/ou sobre o item avaliado;
- Caso exista algum item importante que não esteja descrito no PCM, o Avaliador poderá descrevê-lo na seção "5 - Itens ausentes".
- A seqüência dos itens no questionário é a mesma apresentada no PCM.
- Caso o espaço para considerações não seja suficiente, o Avaliador poderá utilizar a seção "7 – Considerações Complementares" para finalizar as suas considerações.

4. Valores de avaliação:

- (D) - Desnecessário
- (P) - Pouco útil
- (U) - Útil
- (M) - Muito útil

5. Questionário de Avaliação do Processo de Controle de Mudanças

Item	Classificação	Descrição	Avaliação	Considerações
1	Papel	Presidente do Comitê de Controle de Alterações (CCB)	() D () P () U () M	
2	Papel	Comitê de Controle de Alterações (CCB).	() D () P () U () M	
3	Papel	Avaliador	() D () P () U () M	
4	Papel	Implementador	() D () P () U () M	
5	Papel	Solicitante	() D () P () U () M	
6	Papel	Gerente de Projeto	() D () P () U () M	
7	Papel	Verificador	() D () P () U () M	
8	Estado	Não Confirmada (Unconfirmed)	() D () P () U () M	
9	Estado	Nova (New)	() D () P () U () M	
10	Estado	Avaliada (Evaluated)	() D () P () U () M	
11	Estado	Aprovada (Aproved)	() D () P () U () M	
12	Estado	Modificada (Modified)	() D () P () U () M	
13	Estado	Rejeitada	() D () P () U () M	
14	Estado	Verificada	() D () P () U () M	
15	Estado	Fechada	() D () P () U () M	
16	Estado	Reaberta	() D () P () U () M	
17	Resolução	Inválida (Invalid)	() D () P () U () M	
18	Resolução	Não será corrigida (Wontfix)	() D () P () U () M	
19	Resolução	Adiada (Deferred)	() D () P () U () M	
20	Resolução	Duplicada (Duplicate)	() D () P () U () M	
21	Resolução	Não Reproduzível (Worksforme)	() D () P () U () M	
22	Resolução	Movida (Moved)	() D () P () U () M	
23	Relatório	Relato do Estado da Configuração	() D () P () U () M	
24	Critério de Entrada	O CCB está estabelecido para o projeto. Os artefatos que farão parte da linha base existem e estão definidos.	() D () P () U () M	

25	Critério de Entrada	O solicitante que enviou a CR com todas as informações requeridas pelo Presidente do CCB.	() D () P () U () M	
26	Critério de Entrada	A CR ao dar entrada na ferramenta de controle de mudanças, assume o estado de “Enviada”.	() D () P () U () M	
27	Tarefa	O presidente do CCB atribui um avaliador para avaliar o impacto da CR	() D () P () U () M	
28	Tarefa	O avaliador certifica a viabilidade, se realmente pertence ao projeto, se o problema relatado pode ser reproduzido (se for um erro, por exemplo), a estimativa em horas de trabalho para execução da modificação. Para mudança de requisitos, uma análise de impacto deve ser realizada para cada requisito modificado. O estado da CR torna-se “Avaliada”.	() D () P () U () M	
29	Tarefa	O CCB decide se a modificação prevista na CR deverá ser realizada neste momento ou futuramente, ou não será realizada. Outras informações devem ser obtidas de outros interessados que porventura sejam afetados pela modificação em questão, antes de tomar a decisão.	() D () P () U () M	
30	Tarefa	Se a modificação for aceita, o Presidente do CCB atribui ao implementador, e define o estado da CR como “Aprovada”. O Gerente de Projeto negocia as mudanças necessárias no projeto juntamente como os elementos afetados pela mudança (se for o caso). A ferramenta de controle de mudanças envia notificações para o responsável por implementar a modificação, bem como para o solicitante.	() D () P () U () M	
31	Tarefa	Se a mudança for rejeitada, o Presidente do CCB altera o estado da CR como “Rejeitada” e informa os motivos no campo apropriado da ferramenta. A ferramenta de controle de alterações notifica o solicitante.	() D () P () U () M	
32	Tarefa	O Presidente do CCB e o Solicitante determinam se a verificação formal da mudança será requerida, seguindo os procedimentos previstos na seção de Verificação. Se for requerida, o Presidente do CCB designa o Verificador para efetuar a verificação.	() D () P () U () M	
33	Tarefa	O Implementador executa as mudanças necessárias e notifica as demais partes afetadas se outras	() D () P () U () M	

		mudanças em artefatos relacionados também necessitarem modificações, tais como documentação, ajuda ou testes.		
34	Tarefa	O Gerente de Projeto atualiza a documentação do projeto.	() D () P () U () M	
35	Tarefa	Se for verificado que a modificação é impraticável, o implementador notifica o Presidente do CCB. Este definirá o estado da CR como “Cancelada”. O Implementador recupera a versão disponível na última linha base. A ferramenta de controle de mudanças notifica o solicitante, o Presidente do CCB e o Gerente de Projeto.	() D () P () U () M	
36	Tarefa	Quando a modificação estiver completa, o Implementador define o estado da CR para “Mudança Executada”, atualizando a base de dados com a resposta adequada de acordo com os atributos da CR. Informa ainda o esforço (em horas de trabalho) que foi empregado na solução da CR. A ferramenta de controle de mudanças notifica o solicitante e o Presidente do CCB.	() D () P () U () M	
37	Tarefa	Relato do Estado da Configuração	() D () P () U () M	
38	Verificação	O Implementador notifica o Solicitante e o Verificador (se alguém possuir esta atribuição) que a mudança foi realizada e disponibiliza todos os artefatos para os responsáveis pelo processo de verificação.	() D () P () U () M	
39	Verificação	O Verificador executa os passos necessários para garantir a aprovação da alteração implementada.	() D () P () U () M	
40	Verificação	Se a verificação for realizada com sucesso, o Verificador define o estado da CR como “Verificada”. O Solicitante e o Implementador são notificados pela ferramenta de controle de alterações.	() D () P () U () M	
41	Verificação	Se a verificação detectar alguma falha na implementação, não obtendo sucesso, o Verificador define o estado da CR para “Aprovada” descrevendo o problema no atributo respectivo da CR. O Solicitante e o Implementador são notificados pela ferramenta de controle de alterações. O processo continua na tarefa número 7.	() D () P () U () M	
42	Verificação	Para uma CR correspondente a solução de problemas, ou melhoria, o Implementador após a verificação,	() D () P () U () M	

		submete os artefatos ao sistema de controle de versões, atualizando as linhas bases. Para mudanças de requisitos, o Implementador atualiza os números de versões de todos os artefatos relacionados à modificação, preparando para uma nova liberação do produto, compondo uma nova linha base.		
43	Verificação	O Implementador define o estado da CR para Fechado. A ferramenta de controle de mudanças notifica o Solicitante e o Presidente do CCB.	**()** D **()** P **()** U **()** M	
44	Critério de Saída	O estado da CR é definido como Rejeitada ou Fechada.	**()** D **()** P **()** U **()** M	
45	Critério de Saída	Os artefatos sob gerência de configuração, que foram modificados em resposta à respectiva CR foram liberados.	**()** D **()** P **()** U **()** M	
46	Critério de Saída	O Solicitante, o Presidente do CCB e o Gerente de Projeto receberam notificações sobre o estado atual da CR.	**()** D **()** P **()** U **()** M	
47	Critério de Saída	As informações referentes à rastreabilidade de requisitos foram devidamente atualizadas.	**()** D **()** P **()** U **()** M	

6. Itens ausentes

Classificação	Descrição	Considerações

7. Considerações complementares

Neste local, o Sr(a) poderá relacionar quaisquer considerações relevantes para a melhoria do processo. Também poderá propor mudanças de atividades ou tarefas entre papéis, bem como propor a inclusão e/ou exclusão de novos papéis.

Item	Classificação	Descrição	Considerações

Observações: