VANESSA RODRIGUES BORGES

IMPLANTAÇÃO DE PRÁTICAS DE GERÊNCIA DE CONFIGURAÇÃO EM UMA FÁBRICA DE SOFTWARE: UM ESTUDO DE CASO

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS – BRASIL
2006

VANESSA RODRIGUES BORGES

IMPLANTAÇÃO DE PRÁTICAS DE GERÊNCIA DE CONFIGURAÇÃO EM UMA FÁBRICA DE SOFTWARE: UM ESTUDO DE CASO

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de concentração:

Engenharia e Qualidade de Software

Orientador:

Prof. André Luiz Zambalde

LAVRAS
MINAS GERAIS – BRASIL
2006

Ficha Catalográfica preparada pela Divisão de Processo Técnico da Biblioteca Central da UFLA

Borges, Vanessa Rodrigues

Implantação de Práticas de Gerência de Configuração em uma Fábrica de Software: Um Estudo de Caso / Vanessa Rodrigues Borges – Minas Gerais, 2006.

Monografia de Graduação – Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Engenharia e Qualidade de Software. 2. Melhoria do Processo de Software. 3. Gerência de Configuração de Software. 1. BORGES, V. R.. II. Universidade Federal de Lavras. III. Título.

VANESSA RODRIGUES BORGES

IMPLANTAÇÃO DE PRÁTICAS DE GERÊNCIA DE CONFIGURAÇÃO EM UMA FÁBRICA DE SOFTWARE: UM ESTUDO DE CASO

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 18 de setembro de 2006.

_____
Prof. Guilherme Bastos Alvarenga

_____
Prof. Reginaldo Ferreira de Souza

_____
Prof. André Luiz Zambalde
(Orientador)

LAVRAS
MINAS GERAIS - BRASIL

Dedico esse trabalho aos meus pais, Humberto e Ida, e às minhas irmãs, Ludmila e Jacqueline – que fizeram desse sonho uma realidade! Agradeço todo apoio, educação, empenho, amor, carinho e dedicação que sempre recebi de vocês!

Agradecimentos

Agradeço a Deus por toda força, luz, saúde!

Aos meus professores, que foram peças imprescindíveis para a minha formação!

Ao Alessandro pelas figuras e co-orientação nesse trabalho. Pelo apoio, dedicação e carinho que serão para a vida inteira!

Ao Leandro e Breno pelos n trabalhos feitos juntos e noites sem dormir!

Às farras feitas em Lavras!

À Ângela e o Deivson pelos "quebra-galho" essenciais à minha graduação!

À República Kara Karamba Kara Kara Ô, por tornar Lavras meu Lar Doce Lar!

Á Casa das 7 Mulheres pelo tempo de hospedagem!

À Geração Saúde pelas farras Bom Sucessenses!

Á Gabi por ser a minha eterna princesinha e meu remédio contra o estresse e tristeza!

À Joana, Keka, Nina e Pati por toda amizade!

Às minhas irmãs por tudo!

À família SW que despertou em mim, a paixão por minha profissão!

Ao professor André Luiz Zambalde pela orientação e conselhos!

E a todos que de alguma forma fizeram parte da minha vida e contribuíram positiva e negativamente para o meu crescimento pessoal e profissional!

A todos: Muito obrigada!

IMPLANTAÇÃO DE PRÁTICAS DE GERÊNCIA DE CONFIGURAÇÃO EM UMA FÁBRICA DE SOFTWARE: UM ESTUDO DE CASO

RESUMO

O principal objetivo deste trabalho é descrever e discutir a implantação de práticas de Gerência de Configuração de Software de uma empresa de pequeno porte. A empresa participante da pesquisa, uma fábrica de software, buscou implantar atividades relacionadas à Gerência de Configuração em função de problemas relacionados à falta de controle sobre o desenvolvimento, comunicação precária entre os colaboradores, excesso de não conformidades nos softwares produzidos e grande quantidade de retrabalho decorrente de perdas de código-fonte e requisitos já documentados e implementados. A fim de facilitar o desenvolvimento deste trabalho levou-se em consideração o perfil da empresa, ou seja, empresa de pequeno porte e de recursos moderados (humano, tempo e capital). Foram definidas atividades dentro do processo de desenvolvimento que dessem suporte às práticas de identificação, controle, contabilização da situação e auditoria da configuração. Apesar da imaturidade no processo de gerência de configuração de software e da resistência causada pela cultura organizacional, a adoção das práticas definidas apontou para uma melhora na qualidade do processo e do produto da organização, diminuição de retrabalho, maior produtividade e uso racional de recursos.

Palavras-Chave: Engenharia de Software, Qualidade de Software, Melhoria do Processo de Software, Gerência de Configuração de Software.

IMPLANTATION OF CONFIGURATION MANAGEMENT PRACTICES IN A SOFTWARE FACTORY: A CASE STUDY

ABSTRACT

The main objective of this work is to describe and to argue the implantation of Configuration Management practices in a software development process of a small business company. The software factory of the participant company of the research searched to implant activities related to the Software Configuration Management in function of the control lack on the development, of the precarious communication between the collaborators, of the excess of non conformity in softwares produced and of the great amount of re-work decurrently of losses of code-source and requirements already documented and implemented. In order to facilitate the development of this work the profile of the company was taken in consideration, that is, small business company, young, moderate resources (human, time and capital). Activities had been defined within the development process that gave support to the identification practices, control, and situation accounting and configuration audit. Although the process immaturity of software configuration management and the resistance caused for the organizational culture, the adoption of the practices defined pointed to an improvement in the organization process and product quality, reduction of re-work, greater productivity and rational use of resources.

Key-Words: Software Engineering, Software Quality, Software Process Improvement, Software Configuration Management.

SUMÁRIO

LISTA DE FIGURAS ..... *ix*

LISTA DE TABELAS ..... *x*

LISTA DE ABREVIATURAS E SIGLAS ..... *xi*

1. INTRODUÇÃO *1*

1.1. Contextualização e Motivações **1**

1.2. Objetivos e Justificativas **2**

1.3. Organização do Trabalho **3**

2. REFERENCIAL TEÓRICO *4*

2.1. Engenharia e Qualidade de Software **4**

2.2. Melhoria do Processo de Software **6**

2.3. Fábrica de Software **12**

2.4. Gerência de Configuração de Software (GCS) **14**

2.4.1. Funções da Gerência de Configuração 18

2.4.2. Ferramentas de Apoio 22

2.5. Considerações Finais do Capítulo **24**

3. METODOLOGIA *25*

3.1. Tipo de Pesquisa **25**

3.2. Procedimentos Metodológicos **26**

4. RESULTADOS E DISCUSSÃO *28*

4.1. Caracterização da Empresa **28**

4.2. Diagnóstico das Práticas de Gerência de Configuração Atuais **28**

4.3. Planejamento da Implantação das Práticas de GCS na Empresa **32**

4.3.1. Auditoria da Implantação 32

4.3.2. Elaboração do Plano de Implantação 33

4.3.3. Execução do Plano de Implantação 33
4.3.4. Institucionalização das Práticas de GCS 37
4.3.5. Avaliação da Implantação 38
4.4. Implantação das Práticas de GCS 38
4.4.1. Plano de Gerência de Configuração 39
4.4.2. Plano de Ambiente 39
4.4.3. Identificação da Configuração 39
4.4.4. Controle da Configuração 40
4.4.5. Contabilização da Situação da Configuração 45
4.4.6. Auditoria da Configuração 45
4.5. Avaliação da Implantação das Práticas de GCS 46
5. CONCLUSÕES 47
6. REFERENCIAL BIBLIOGRÁFICO 49
Apêndice A – Glossário de Gerência de Configuração 52
Apêndice B – Plano de Gerência de Configuração 58
Apêndice C – Plano de Ambiente 63

LISTA DE FIGURAS

Figura 2.1 - O Processo de Software e seus Componentes 7
Figura 2.2 - O Processo de Software 11
Figura 4.1 - Política "trava-modifica-destrava" 30
Figura 4.2 - Cópia dos arquivos do repositório para o local de trabalho 30
Figura 4.3 - Etapas da Implantação das Práticas de GCS 32
Figura 4.4 - Política "copia-modifica-resolve" 36
Figura 4.5 - Práticas de Gerência de Configuração 38
Figura 4.6 - Fluxo Sintético de Gerência de Configuração 41
Figura 4.7 - Fluxo de Operação do *Subversion* 42
Figura 4.8 - Fluxo de Controle de Mudanças Formal 43
Figura 4.9 - Fluxo de *bugs* adotado para a Ferramenta *Mantis* 44
Figura 4.10 - Fluxo de requisitos adotado para a Ferramenta *Mantis* 44
Figura A.1 - Páginas 1 e 2 do Glossário de Gerência de Configuração 52
Figura A.2 - Páginas 3 e 4 do Glossário de Gerência de Configuração 52
Figura A.3 - Páginas 5 e 6 do Glossário de Gerência de Configuração 53
Figura A.4 - Páginas 7 e 8 do Glossário de Gerência de Configuração 53
Figura A.5 - Páginas 9 e 10 do Glossário de Gerência de Configuração 54
Figura A.6 - Páginas 11 e 12 do Glossário de Gerência de Configuração 54
Figura A.7 - Páginas 13 e 14 do Glossário de Gerência de Configuração 55
Figura A.8 - Páginas 15 e 16 do Glossário de Configuração 55
Figura A.9 - Páginas 17 e 18 do Glossário de Configuração 56
Figura A.10 - Páginas 19 e 20 do Glossário de Configuração 56
Figura A.11 - Página 21 do Glossário de Configuração 57
Figura B.1 - Páginas 1 e 2 do Plano de Gerência de Configuração 58
Figura B.2 - Páginas 3 e 4 do Plano de Gerência de Configuração 58
Figura B.3 - Páginas de 5 e 6 do Plano de Gerência de Configuração 59
Figura B.4 - Páginas 7 e 8 do Plano de Gerência de Configuração 59
Figura B.5 - Páginas 9 e 10 do Plano de Gerência de Configuração 60
Figura B.6 - Páginas 11 e 12 do Plano de Gerência de Configuração 60
Figura B.7 - Páginas 13 e 14 do Plano de Gerência de Configuração 61
Figura B.8 - Páginas 15 e 16 do Plano de Gerência de Configuração 61
Figura B.9 - Páginas 17 e 18 do Plano de Gerência de Configuração 62
Figura B.10 - Página 19 do Plano de Gerência de Configuração 62
Figura C.1 - Páginas 1 e 2 do Plano de Ambiente 63
Figura C.2 - Páginas 3 e 4 do Plano de Ambiente 63
Figura C.3 - Páginas 5 e 6 do Plano de Ambiente 64
Figura C.4 - Páginas 7 e 8 do Plano de Ambiente 64

LISTA DE TABELAS

Tabela 2.1 - Características do processo..... 9

LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

CASE - *Computer-Aided Software Engineering* – Engenharia de Software Apoiada por Computador

CMG – *Configuration Management Group* – Grupo de Gerência de Configuração

CMMI-SE/SW - *Capability Maturity Model® Integration* – Modelo de Capacidade da Maturidade Integrado

ES - Engenharia de Software

EUA – Estados Unidos da América

GC - Gerência de Configuração

GCS - Gerência de Configuração de Software

HP - Hewlett-Packard Development Company

IEEE - *Institute of Electrical and Eletronic Engineers* – Instituto de Engenheiros Elétricos e Eletrônicos

IBM - *International Business Machines Corporation*

IC – Item de configuração

ISO/IEC - *International Organization for Standardization / International Electrotechnical Commission* - Organização Internacional para Padronização / Comissão Internacional Eletrônica.

MG - Minas Gerais

MPS.BR - Melhoria de Processo do Software Brasileiro

NBR – Norma Brasileira

SDO - *Software Development Organization* – Organização Desenvolvedora de Software

SEI - *Software Engineering Institute* - Instituto de Engenharia de Software.

SEPG - *Software Engineering Process Group* – Grupo de Engenharia de Processo de Software

SW-CMM® - *Capability Maturity Model* – Modelo de Capacidade da Maturidade

TI - Tecnologia da Informação

UFLA - Universidade Federal de Lavras

1. INTRODUÇÃO

A preocupação com a melhoria do processo de desenvolvimento de software vem sendo impulsionada por exigências do mercado por mais qualidade e produtividade. Muitas empresas têm revisto seus processos, procurando se capacitar no mercado cada vez mais competitivo.

Neste primeiro capítulo é apresentada uma breve introdução sobre o trabalho proposto, juntamente com as motivações para a realização do mesmo, os objetivos e as justificativas.

1.1. Contextualização e Motivações

Com o aumento da complexidade no desenvolvimento de sistemas e a existência cada vez maior de uma competitividade no mercado na área de tecnologia da informação, tornou-se crescente a necessidade de criação de produtos mais confiáveis e de boa qualidade.

Nos últimos anos, identifica-se uma forte tendência à busca contínua pela qualidade, principalmente porque o resultado de métodos que objetivam a melhora dessa qualidade pôde ser traduzido como lucro para as empresas (diminuição de retrabalho, maior produtividade e uso racional de recursos).

A fim de atender a essa tendência e exigências do mercado, diversas metodologias de desenvolvimento de software relacionadas ao controle foram elaboradas, visto que, ao longo do ciclo de vida de um projeto de software uma grande quantidade de itens de informação é produzida, tais como: documentos, código-fonte, dados e manuais. Muitos desses itens provavelmente irão sofrer algumas modificações durante o projeto devido a diversas causas, como mudanças nos requisitos ou correção de defeitos.

Para evitar a perda do controle do projeto em conseqüência das mudanças é preciso que essas sejam devidamente controladas e gerenciadas.

A Gerência de Configuração de Software (GCS) é uma disciplina, dentro da Engenharia de Software, que tem como objetivo gerenciar e controlar a evolução de um software através, basicamente, de controle formal de versão e solicitação de mudanças. Em outras palavras, é a prática de lidar com modificações de forma sistemática, permitindo que os sistemas tenham sua integridade mantida com o passar do tempo.

A falta da Gerência de Configuração de Software pode ocasionar problemas como: perda de código-fonte, bibliotecas que inesperadamente param de funcionar,

impossibilidade de determinar o que aconteceu com um programa ou parte dele, programa em execução e o seu código-fonte em versões diferentes ou, até mesmo, quem, por que e quando foram efetuadas modificações.

Bersoff (1984) garante que a Gerência de Configuração nasceu como resposta aos erros cometidos durante os anos 70, quando computadores começaram a ser utilizados de maneira geral para a solução dos mais diversos e complexos problemas e para os quais o gerenciamento tradicional mostrou-se inadequado. Principalmente, quando se descobriu que a complexidade de gerenciamento de um sistema não variava de maneira linear com o número de linhas de código, mas de maneira exponencial.

Como observa Babich (1986), ela é uma disciplina útil para a gerência do projeto, pois lhe permite executar tarefas complexas de uma maneira mais organizada, o que proporciona aumento de produtividade e redução de erros. Do ponto de vista da atividade de teste, o uso da gerência de configuração como elemento de melhoria de qualidade ainda é uma questão em aberto. Mas ela é profundamente importante no que tange aos benefícios trazidos por ela, pois o tipo e quantidade de informação advindos da atividade de teste que podem ser manipuladas com o uso dessa disciplina, possibilitam maior eficiência nas atividades de evolução, depuração e manutenção do software.

Hoje, independente do porte da empresa de desenvolvimento de software, tem-se que a GCS é imprescindível para maximizar a produtividade, minimizar os erros e diminuir o retrabalho.

1.2. Objetivos e Justificativas

O objetivo deste trabalho é propor o uso da gerência de configuração nas atividades de uma empresa desenvolvedora de software, neste trabalho tratada pelo nome fictício “Beta”, com o intuito de minimizar o esforço alocado, o retrabalho e aumentar a produtividade durante toda a evolução do software por ela desenvolvido.

Nessa perspectiva, a finalidade é apresentar um plano de gerência de configuração para a organização, criar um manual de instruções para todos os envolvidos no processo de desenvolvimento de software, elaborar um plano de ambiente, organizar a equipe e a infra-estrutura dentro da empresa e, por fim, implantar as práticas no processo da organização.

Com a implantação dessas práticas, espera-se que seja institucionalizada a metodologia de gerenciamento dos itens de configuração nos projetos da organização, que

se tenha um satisfatório controle de versões e de mudanças, bem como, que se obtenha um aumento na qualidade dos seus produtos desenvolvidos e redução do retrabalho.

A empresa participante como ambiente de execução e estudo de caso para a realização deste trabalho é de desenvolvimento de software, uma *Software Development Organization* (SDO), de pequeno porte, localizada na cidade de Lavras – MG.

Apesar de existir nesta empresa "Beta" um processo de desenvolvimento de software definido, esse possui carências no que se refere à Gerência de Configuração de Software. Para a empresa, a implantação das práticas de GCS é motivada pela busca da qualidade nos projetos sob sua orientação como ponto de diferenciação de seus produtos num mercado aberto e competitivo. Esse trabalho será o início da formalização e institucionalização do processo de gerência de configuração no desenvolvimento de software dentro da organização.

A empresa estudo de caso submeteu-se, recentemente, à implantação de melhorias em seu processo visando uma certificação formal nível G no modelo MPS.BR - Melhoria de Processo do Software Brasileiro. Nesse contexto, o trabalho se justifica, uma vez que, a organização já possui uma cultura institucionalizada de busca por qualidade. Na empresa se tem como bom investimento implantação de novas práticas em seu processo de desenvolvimento que auxiliarão o aumento e a garantia da qualidade de seus produtos e processo.

1.3. Organização do Trabalho

Os capítulos subseqüentes deste trabalho estão assim organizados: no Capítulo 2, são discutidos os conceitos de engenharia e qualidade de software; melhoria no processo de software; fábrica de software, além de gerência de configuração. A metodologia utilizada para a concepção deste trabalho é mostrada no Capítulo 3. No capítulo 4, são apresentados os resultados e discussões sobre o estudo realizado. Por fim, o Capítulo 5 aborda as conclusões deste trabalho. Nos Anexos de A até C, são exibidos modelos de documentos utilizados para apoiar a atividade e a implantação das práticas de Gerência de Configuração de Software.

2. REFERENCIAL TEÓRICO

O presente capítulo tem por objetivo facilitar a compreensão do trabalho proposto abordando os temas relacionados ao mesmo. São introduzidos conceitos gerais de engenharia e qualidade de software, melhoria no processo de software, fábrica de software e gerência de configuração de software; fornecendo, assim, embasamento para um melhor entendimento deste trabalho.

2.1. Engenharia e Qualidade de Software

Segundo Bauer *apud* Pressman (2002), "Engenharia de Software (ES) é a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe eficientemente em máquinas reais". O próprio significado de engenharia já traz os conceitos de criação, construção, análise, desenvolvimento e manutenção.

Sommerville (2003) diz que a Engenharia de Software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, depois que ele entrou em operação. Destacam-se dois fragmentos nesta definição: “disciplina da engenharia”, o que diz respeito à aplicação de teorias, métodos e ferramentas apropriadas em momentos apropriadas, de modo seletivo; e sempre procura descobrir soluções para os problemas, mesmo quando não existem teorias aplicáveis e métodos de apoio; e “todos os aspectos da produção de software”, o que elucida que a engenharia de software não se encarrega somente dos processos de desenvolvimento, mas também do gerenciamento de projetos de software e desenvolvimento de ferramentas, métodos e teorias que dêem apoio à produção de software.

Essa disciplina surgiu em meados dos anos 70 numa tentativa de contornar a crise do software e dar um tratamento de engenharia (mais sistemático e controlado) ao desenvolvimento de sistemas de software complexos. Um sistema de software complexo se caracteriza por um conjunto de componentes abstratos de software (estruturas de dados e algoritmos) encapsulados na forma de procedimentos, funções, módulos, objetos ou agentes e interconectados entre si, compondo a arquitetura do software, que deverão ser executados em sistemas computacionais.

A Engenharia de Software é uma área de conhecimento voltada para a especificação, desenvolvimento e manutenção de sistemas de software aplicando

tecnologias e práticas de ciência da computação, gerência de projetos, gerência de configuração e outras disciplinas, objetivando organização, produtividade e qualidade.

Empresas desenvolvedoras de software passaram a empregar os conceitos de Engenharia de Software, sobretudo para orientar suas áreas de desenvolvimento, muitas delas organizadas sob a forma de Fábrica de Software. O conceitos sobre fábrica de softwares serão apresentados na seção 2.3 deste trabalho.

A engenharia de sistemas é uma área mais ampla por tratar de todos os aspectos de sistemas baseados em computadores, incluindo *hardware* e engenharia de processos além do software.

Segundo o IEEE (2001) são definidos no SWEBOK as seguintes áreas de conhecimento da Engenharia de Software: requisitos de software; projeto (*design*) de software; construção de software; teste de software; manutenção de software; gerência de configuração de software; gerência de engenharia de software; ferramentas e métodos de engenharia de software e qualidade de software.

A Gerência de Configuração de Software (GCS) além de ser uma das áreas de conhecimento da ES é o foco deste trabalho e será apresentada de forma mais detalhada na seção 2.4 desse trabalho.

A ES está envolvida com todos os aspectos da produção de software de qualidade a um custo aceitável. Esses termos levam à busca de um processo de desenvolvimento que considere a componente "Qualidade".

Atingir um alto nível de qualidade de produto ou serviço é o objetivo da maioria das organizações. Atualmente, não é mais aceitável entregar produtos com baixa qualidade e reparar os problemas e as deficiências depois que os produtos foram entregues ao cliente, conforme diz Sommerville (2003).

A definição de qualidade é relativa. O que é qualidade para uma pessoa pode ser falta de qualidade para outra. A idéia de qualidade é aparentemente intuitiva; contudo, quando examinado mais longamente, o conceito se revela complexo. Definir um conceito de qualidade para estabelecer objetivos é, assim, uma tarefa menos trivial do que aparenta a princípio. A noção de qualidade de software pode ser descrita por um grupo de fatores, requisitos ou atributos, tais como: confiabilidade, eficiência, facilidade de uso, modularidade e legibilidade.

Em Engenharia de Software, a qualidade é normalmente tratada de forma diferenciada quanto a produtos e a processos de desenvolvimento de software.

Segundo a norma ISO/IEC 9126 (1991) - *International Organization for Standardization/International Electrotechnical Commission*, relacionada a produtos, qualidade é "Um conjunto de atributos que têm impacto na capacidade do software de manter o seu nível de desempenho dentro de condições estabelecidas por um dado período de tempo".

Pressman (2002) diz que qualidade é a conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software profissionalmente desenvolvido.

As duas vertentes – qualidade de produto e qualidade de processo – são complementares e interdependentes. Espera-se que a qualidade do processo de fabricação tenha um impacto positivo sobre o software obtido. Entretanto, tal objetivo será atingido se houver uma compreensão clara de que os processos devem fornecer todos os mecanismos necessários para especificar o produto e controlar a fabricação.

Atentas a essas exigências e buscando alcançar um alto padrão de qualidade dos seus produtos, as organizações de software vêm utilizando o conceito da engenharia de software na construção dos mesmos.

Em geral, a disciplina de Engenharia de Software adota uma abordagem sistemática e organizada para a realização do trabalho, uma vez que essa é, com freqüência, a maneira mais eficaz de produzir software de alta qualidade.

Sendo assim, para alcançar a qualidade, a Engenharia de Software utiliza-se de melhoria de processos, implementada através de modelos abstratos ou formais que permitem aos engenheiros especificar, projetar, implementar e manter sistemas de software, avaliando e garantindo suas qualidades. Além disto, a Engenharia de Software deve oferecer mecanismos para se planejar e gerenciar o processo de desenvolvimento.

A implantação de um sistema de qualidade no desenvolvimento de software permite um aumento de produtividade, uma melhoria da qualidade do produto final e um aumento da satisfação dos clientes e da própria empresa.

2.2. Melhoria do Processo de Software

Hoje em dia, a qualidade do processo é mais importante do que a qualidade final do produto e existem normas e padrões tanto para produtos quanto para processos.

O conceito de processo na indústria de manufaturas é quase intuitivo e pode ser visto como “as diversas operações pelas quais um produto passa até ficar pronto”.

Segundo o IEEE (*Institute of Electrical and Eletronic Engineers),* de forma mais geral, processo é uma seqüência de passos realizados para um determinado propósito.

Para software, o conceito de processo pode ser definido segundo Paulk *et al.* (1995) como um conjunto de atividades, métodos, práticas e tecnologias que as pessoas utilizam para desenvolver e manter o software e seus produtos relacionados. A Figura 2.1 ilustra esta definição.

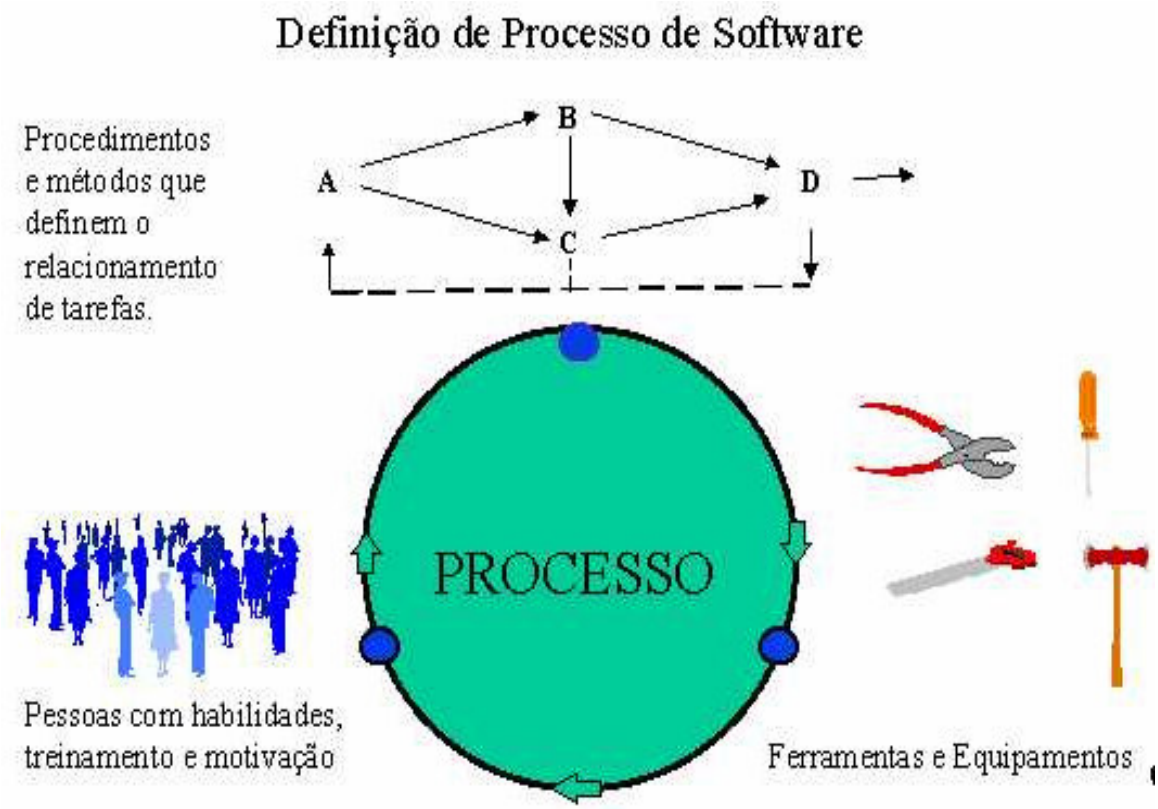


Figura 2.1 - O Processo de Software e seus Componentes
Fonte: Pessoa (2003)

Pressman (2002) trata a engenharia de software como uma “tecnologia em camadas”. Toda iniciativa de engenharia de software deve ser apoiada por um compromisso com a qualidade. Acima da camada da qualidade encontram-se os processos, acima destes os métodos e acima destes as ferramentas. O processo de software é o conjunto de atividades e resultados associados que levam à produção de software.

Ao longo da história da engenharia de software foram sendo criadas ferramentas computadorizadas para apoiar o desenvolvimento, vários modelos de processos de software foram concebidos.

Existem algumas atividades fundamentais, que se pode dizer “comuns em todos os processos de software”, são elas: especificação, projeto e implementação; validação e evolução do software.

A especificação de software, também conhecida como engenharia de requisitos, destina-se a estabelecer quais funções são requeridas pelo sistema e as restrições sobre a operação e o desenvolvimento do sistema.

A implementação converte a especificação produzida na atividade anterior em um sistema executável. Esta atividade, geralmente, envolve o projeto e a programação do software. O projeto é a descrição da estrutura do software, dos dados que fazem parte do sistema e das interfaces entre os componentes do sistema.

A atividade de validação, também chamada de verificação e validação, atesta que o sistema está de acordo com suas especificações e que atende às expectativas. Esta atividade inclui revisões e inspeções em cada estágio do processo de software.

A última atividade fundamental, a evolução de software, trata da demanda real por modificações no software, o que é cada vez mais comum visto que as necessidades dos usuários são mutáveis.

Segundo Sommerville (2003), durante os últimos anos, ampliou-se o interesse por parte das organizações que desenvolvem software pela melhoria de seus processos. A melhoria do processo significa compreender os processos existentes e modificá-los, a fim de melhorar a qualidade do produto e/ou reduzir os custos e o tempo de desenvolvimento. A maior parte da literatura relacionada a esse assunto tem se concentrado em aprimorar os processos para melhorar a qualidade do produto e, em particular, para reduzir o número de defeitos nos softwares fornecidos. Uma vez que esse objetivo é alcançado, a redução dos custos e do tempo pode se tornar a principal meta da melhoria.

A melhoria do processo de software envolve aspectos técnicos, gerenciais e culturais, tais como:

- Alinhamento das ações de melhoria ao contexto, estratégia e objetivos de negócio da organização;

- Escolha de um modelo (ou conjunto de modelos) de processo como referência para orientação do trabalho (por exemplo, a ISO/IEC 15504 e o CMMI-SE/SW - *Capability Maturity Model® Integration*);
- Estabelecimento de um programa de melhoria para gerenciar as ações a serem realizadas, conhecimento do estado atual das práticas da organização;
- Acompanhamento, medição e institucionalização da melhoria.

Tudo isto por meio da definição, utilização e melhoria contínua dos processos envolvidos na aquisição, fornecimento, desenvolvimento, operação, manutenção e suporte de sistemas de software.

Para tanto abordagens e experiências para a melhoria de processo de software baseadas em modelos têm sido utilizadas com sucesso pelas organizações de software. Os modelos mais utilizados têm sido o SW-CMM® - *Capability Maturity Model*, ISO/IEC 12207, ISO/IEC 15504 e CMMI-SE/SW.

Sommerville (2003) diz que melhorar o processo de software de uma organização não significa simplesmente adotar métodos ou ferramentas específicas ou algum modelo de processo que tenha sido utilizado em outro lugar, pois os processos de software são inerentemente complexos e envolvem um número muito grande de atividades. Assim como os produtos, os processos também têm atributos ou características (ver Tabela 2.1).

Tabela 2.1 - Características do processo

Característica do processo	Descrição
Facilidade de compreensão	Até que ponto o processo está explicitamente definido e com que facilidade se pode compreender a definição do processo?
Visibilidade	As atividades de processo culminam em resultados nítidos, de modo que o progresso do processo seja externamente visível?
Facilidade de suporte	Até que ponto as atividades do processo podem ser apoiadas por ferramentas CASE - *Computer-Aided Software Engineering*?
Aceitabilidade	O processo definido é aceitável e utilizável pelos engenheiros responsáveis pela produção do produto de software?

Confiabilidade	O processo está projetado de tal maneira que seus erros sejam evitados ou identificados antes que resultem em erros no produto?
Robustez	O processo pode continuar, mesmo que surjam problemas inesperados?
Facilidade de manutenção	O processo pode evoluir para refletir os requisitos mutáveis da organização ou melhorias de processo identificadas?
Rapidez	Com que rapidez pode ser concluído o processo de entrega de um sistema, a partir de uma determinada especificação?

Fonte: Sommerville (2003)

Para que alcançar a melhoria do processo, é necessário que se promova a execução das seguintes atividades denominadas de estágios:

- Análise de processo: examina os processos existentes e produz um modelo específico para documentar e compreender o processo;
- Identificação de melhoria: ocupa-se de utilizar os resultados coletados da fase de análise de processo para identificar gargalos relativos à qualidade, ao prazo e ao custo, em que os fatores de processo influenciam adversamente a qualidade do produto. A melhoria de processo deve enfocar as eliminações desses gargalos, propondo novos procedimentos, métodos e ferramentas para corrigir os problemas;
- Introdução de mudança de processo: implantam-se novos procedimentos, métodos, ferramentas e integra-os com outras atividades de processo. É importante dar tempo suficiente para introduzir as mudanças e garantir que elas sejam compatíveis com outras atividades de processo e com os procedimentos e padrões organizacionais;
- Treinamento em mudanças de processo: essa fase é muito importante, pois sem ela não é possível obter os plenos benefícios das mudanças de processo. Quando as mudanças de processo são impostas sem treinamento adequado,

os efeitos dessa mudança resultam na deterioração e não na melhoria da qualidade do produto;

- Ajuste de mudanças: As mudanças de processo propostas nunca serão inteiramente eficazes assim que forem introduzidas. Há a necessidade de uma fase de ajuste, em que problemas menores sejam descobertos e modificações no processo sejam propostas e introduzidas.

O sucesso da melhoria de processo depende do cumprimento organizacional com as metas estabelecidas, da disponibilidade de recursos e do apoio e participação de todos os colaboradores da organização.

Os diversos modelos identificam processos fundamentais para a engenharia de software e, em todos eles, tem-se a gerência de configuração como sendo um desses. A gerência de configuração é essencial para a avaliação do software desenvolvido.

Uma das atividades consideradas por Pressman (2002) como "atividade guarda-chuva aplicada ao longo de todo o processo de software" é a gerência de configuração de software, conforme ilustrado na Figura 2.2.

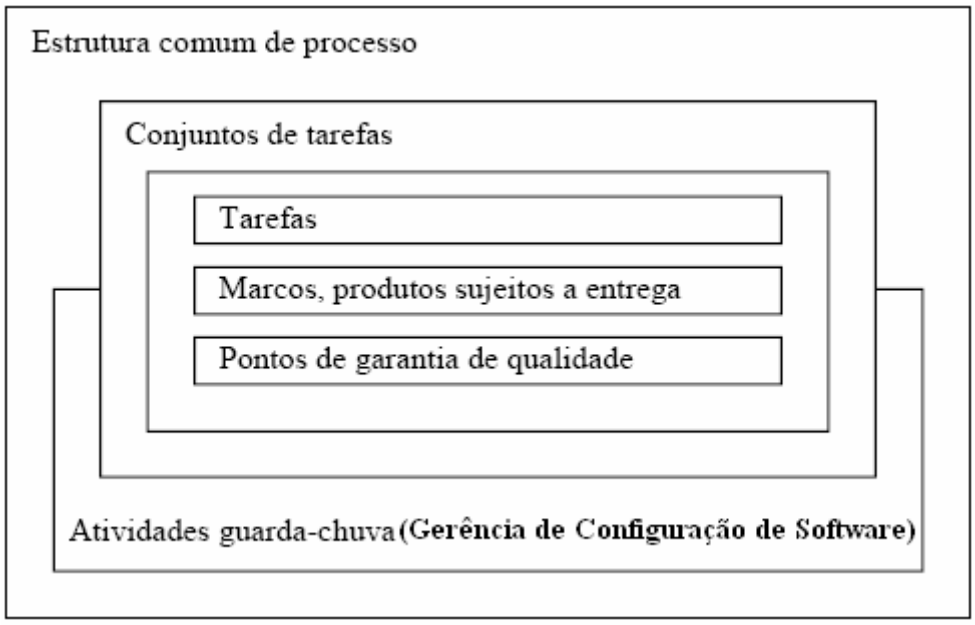


Figura 2.2 - O Processo de Software
Fonte: Adaptado de Pressman (2002)

Entretanto, gerenciar a configuração de um software não é uma atividade trivial, e exige conhecimentos, habilidades e infra-estrutura específicos.

2.3. Fábrica de Software

Nos últimos anos pôde-se perceber uma movimentação crescente no mercado de desenvolvimento de sistemas em direção ao modelo denominado fábrica de software.

O termo Fábrica de Software surgiu no mercado, como uma solução para alcançar maior produtividade e menor custo na produção de sistemas de software. Esta afinidade com o mercado é possivelmente o principal fator que concorre para a baixa quantidade de materiais acadêmicos abordando o tema. O que existe na verdade são iniciativas, partindo de empresas de diferentes portes, para pôr em prática os conceitos de produção provenientes de fábricas industriais. Sendo assim, uma das principais características deste modelo é a adoção de técnicas utilizadas na engenharia industrial de produção em série, para a criação de um ambiente produtivo de desenvolvimento de software com qualidade e baixo custo.

Conforme Siy *et al.* (2001), uma fábrica de software é uma organização que provê serviços de desenvolvimento de sistemas com alta qualidade, a baixo custo e de forma rápida, utilizando um processo de desenvolvimento de software bem definido e tecnologia de ponta, além de algumas formas de *feedback*[1] para reconhecer e lidar com oportunidades de melhoria do processo.

De acordo com Aaen (1997), há mais de trinta anos a idéia de fábrica de software vem sendo continuamente moldada. As primeiras fábricas surgiram no final da década de 60 e, ainda hoje, o termo fábrica possui uma interpretação controversa quando o desenvolvimento de software é comparado à produção em massa de produtos industriais.

É importante frisar que, esta comparação com produção industrial de massa, deve ser realizada com ressalvas, principalmente pelo fato de um produto de software ser único, ou seja, cada nova demanda de produção da fábrica possuirá características específicas, o que descaracteriza o conceito de produção em massa que se tem para os produtos industriais. Apesar das controvérsias e das diferentes visões com relação ao termo, em sua forma mais trivial, todas as definições possuem pontos em comum. As diferenças surgem ao se detalhar os conceitos e nos aproximando dos aspectos estratégicos, funcionais e operacionais, quando então se percebe uma série de diferentes opções para se estruturar uma fábrica de software.

[1] *Feedback*, nesse contexto, significa retroinformação, ou seja, comentários e informações sobre algo que já foi feito com o objetivo de avaliação

O termo Fábrica de Software deve ser interpretado como uma organização projetada de maneira particular e sistematizada, composta por pessoas envolvidas em um esforço comum, onde as tarefas são organizadas e a padronização é utilizada com o objetivo de auxiliar na coordenação e formalização do processo, segundo Aaen (1997).

Apesar de ser um modelo antigo, surgido no início da década de 60, fábrica de software nunca foi um modelo adotado intensivamente pelo mercado, onde as principais iniciativas partiam de grandes corporações como HP - *Hewlett-Packard Development Company*, IBM - *International Business Machines Corporation* e Unisys, associações entre institutos de pesquisa ou projetos governamentais. Durante os últimos trinta anos, existiram várias iniciativas para criação de fábricas de software em diferentes partes do mundo, cada uma delas adotando diferentes estratégias para organização e execução de suas atividades, enquanto algumas tinham o foco na adoção de processos, outras priorizavam a adoção de ferramentas de automação.

Com a constante evolução da engenharia de software e das tecnologias envolvidas no desenvolvimento de sistemas, as fábricas serão cada vez mais efetivas dentro de seu objetivo de produzir softwares de qualidade, em pouco tempo e por um baixo custo espera-se, portanto, um crescimento ainda maior na adoção deste modelo para o desenvolvimento de sistemas.

Através das novas facilidades tornaram possíveis também que empresas de médio e até de pequeno porte, pudessem montar suas fábricas de software para prestar serviços de desenvolvimento de sistemas à crescente terceirização do mercado, resultando numa proliferação deste novo modelo de fábrica pelo mundo.

Neste contexto, as fábricas de software podem constituir núcleos de desenvolvimento dentro de grandes empresas ou serem empresas independentes. Além disso, o escopo do processo dentro do ciclo de desenvolvimento de software varia conforme a fábrica. Existem fábricas de software que prestam serviços desde a análise de sistemas e outras que trabalham a partir da fase de implementação.

Desta forma, as fábricas de software podem se tornar estruturas complementares à organização do cliente, ampliando de forma eficaz e qualificada a capacidade de atendimento à demanda de serviços de software.

Fábricas localizadas na Índia exportam muitos serviços de programação de aplicações para os EUA – Estados Unidos da América e países da Europa, cujos mercados de tecnologia da informação são os maiores do mundo. Entretanto, grandes investimentos

na criação de fábricas de software vêm sendo realizados no Brasil, China e Rússia, conforme Computerworld (2006).

Ainda em Computerworld (2006), são listadas cinco razões para esse crescimento das fábricas de software no Brasil:

1. O Brasil se tornou uma opção interessante para exportação de programação devido à desvalorização cambial em relação aos EUA e ao baixo custo de homem/hora;
2. Grande parte dos trabalhos de fábrica de software é decorrente de revisão de processos ou projetos de integração como, por exemplo, serviços de customização de produtos produzidos por multinacionais para o mercado brasileiro;
3. As arquiteturas de sistemas são projetadas fragmentadas em camadas, o que possibilita o desenvolvimento remoto de partes desses fragmentos;
4. O crescimento de fábricas que possuem conhecimentos de negócios e prestam serviços de análise de sistemas, e não apenas implementação;
5. A tendência existente de terceirização de serviços por parte de empresas que se concentram em suas atividades principais e transferem para parceiros as atividades que não estão diretamente ligadas ao seu negócio principal.

O conceito de fábrica de software está baseado na idéia de ter uma linha de produção de sistemas (software) a partir de requisitos levantados por um "cliente" da fábrica. Essa produção deve ser realizada de preferência sem a necessidade de comunicação dos profissionais da linha de produção (desenvolvedores) com os usuários, analistas e projetistas e de acordo com um escopo, cronogramas e padrões pré-estabelecidos de qualidade e de projeto.

Para que a fábrica funcione com sucesso e os resultados atendam às expectativas do cliente é fundamental a adoção de um processo que considere todo o ciclo de desenvolvimento, assim como auxilie no gerenciamento (planejamento e controle) de todas as atividades e recursos envolvidos nos projetos. Dessa forma é possível medir e controlar a produtividade da equipe envolvida.

2.4. Gerência de Configuração de Software (GCS)

A Gerência de Configuração (GC) surgiu nos anos 50 devido à necessidade da indústria aeroespacial norte-americana controlar as modificações na documentação

referente à produção de aviões de guerra e naves espaciais, segundo Leon (2000), Estublier *et al.* (2002), Hass (2003). Posteriormente, nos anos 60 e 70, a GC passou a abranger artefatos de software, indo além dos artefatos de hardware já estabelecidos e desencadeando o surgimento da Gerência de Configuração de Software (GCS), como pode ser visto em Christensen *et al*.(2002).

Apesar do surgimento da GCS nos anos 70, o seu foco era muito restrito às aplicações militares e aeroespaciais, e somente no final dos anos 80, com o surgimento do padrão IEEE Std 1042 (IEEE, 1987), e em meados dos anos 90, com o surgimento da norma ISO 10007 (ISO, 1995), a GCS foi finalmente assimilada no processo de desenvolvimento de software de organizações não militares, segundo Leon (2000).

Segundo Bersoff (1984), a Gerência de Configuração de Software (GCS) nasceu como resposta aos erros cometidos durante os anos 70, quando computadores começaram a ser utilizados de maneira geral para a solução dos mais diversos e complexos problemas para os quais o gerenciamento tradicional mostrou-se inadequado. Principalmente, quando se descobriu que a complexidade de gerenciamento de um sistema não variava de maneira linear com o número de linhas de código, mas de maneira exponencial.

Como toda nova área de pesquisa, existem diversas definições para GCS.

Babich (1986) define "a arte de coordenar o desenvolvimento de software para minimizar ... confusão é chamada de gerência de configuração. A gerência de configuração é a arte de identificar, organizar e controlar modificações no software que está sendo construído por uma equipe de programação. O objetivo é maximizar a produtividade pela minimização dos erros."

Estublier (2000) diz que GC é a disciplina que permite evoluir produtos de software de forma controlada, e, desta forma, contribui na satisfação de restrições de qualidade e de tempo.

Contudo, a definição mais aceita e utilizada, atualmente, caracteriza a GCS como uma disciplina que aplica procedimentos técnicos e administrativos para identificar e documentar as características físicas e funcionais de itens de configuração, controlar as alterações nessas características, armazenar e relatar o processamento das modificações e verificar a compatibilidade com os requisitos especificados e garantir que foi feito o que deveria ter sido feito. (IEEE, 1990).

Desta forma, a GCS não se propõe a definir quando e como devem ser executadas as modificações nos artefatos de software, papel este reservado ao próprio processo de

desenvolvimento de software. A sua atuação ocorre como processo auxiliar de controle e acompanhamento dessas atividades.

Como observa Babich (1986), ela é uma disciplina útil para a gerência do projeto, pois lhe permite executar tarefas complexas de uma maneira mais organizada, o que proporciona aumento de produtividade e redução de erros.

No processo de produção de software, um objeto em desenvolvimento não tem um comportamento estático, ele evolui com o tempo. Chama-se de versão ao estado particular de um objeto e, por configuração, entende-se a relação entre as versões de um objeto composto, ou seja, configuração é uma instância do sistema composta da união de uma versão específica de cada objeto componente, segundo Victorelli (1990).

De acordo com Capretz (2002), a Gerência de Configuração de Software não fornece um método de projeto, nem um modelo de ciclo de vida, nem, tampouco, define como a qualidade dos itens deve ser julgada; ela fornece um fundamento sólido para todas as outras atividades de engenharia de software. Pode-se defini-la como uma disciplina para gerenciamento efetivo da produção de software. Cabe a ela identificar a configuração de um sistema em relação ao tempo com a finalidade de, sistematicamente, controlar as mudanças desta configuração, manter sua integridade e, desta forma, possibilitar seu rastreamento através do ciclo de vida do sistema, segundo Babich (1986).

Inicialmente, a preocupação da GC estava voltada basicamente para a equipe de programação, o que não ocorre mais hoje.

A gerência de configuração atualmente abrange todas as pessoas envolvidas com o projeto de software, sejam elas analistas de requisitos, implementadores, testadores, gestores da qualidade, dentre outros.

Uma concepção errônea, mas bastante freqüente em algumas organizações, consiste em julgar que a gerência de configuração é garantida pelo simples fato da empresa utilizar uma ferramenta de controle de versões. Na realidade, a GCS envolve todo um conjunto de regras e procedimentos, que são apenas parcialmente automatizáveis.

Em relação à confusão, de acordo com Babich (1986), ela surge quando as modificações não são analisadas antes de serem feitas, não são registradas antes de serem implementadas, não são relatadas àqueles que têm necessidade de saber delas ou não são controladas no sentido de melhorar a qualidade e reduzir os erros.

As modificações em um produto de software são inevitáveis durante seu ciclo de vida. Elas podem ocorrer pelo surgimento de novas necessidades dos clientes, que exigem

modificações nos dados produzidos por sistemas de informação ou até mesmo por restrições de orçamento ou cronograma, que causam redefinições no sistema ou produto. Como as mudanças nos produtos de software são intrínsecas, se não forem bem controladas, podem levar à perda de controle do produto e, como conseqüência, comprometer sua integridade e qualidade.

Para entender melhor a definição de GCS e, também, suas atividades, são necessárias algumas definições sobre o tema.

Villas Boas (2003) assume que configuração são características físicas e funcionais de um produto, conforme definidas na documentação técnica obtidas no produto. E diz ainda, que documentos de configuração são documentos (em qualquer tipo de mídia) que definem os requisitos, projeto, construção/produção e verificação para um item de configuração.

Durante o processo de desenvolvimento de um produto, é produzida uma grande quantidade de itens de informação que podem ser alterados durante o processo, a esses itens dá-se o nome de item de configuração (IC). Segundo Villas Boas (2003), IC é o conjunto de materiais e equipamentos, informações, materiais processados, serviços ou qualquer de suas partes distintas, que é designado para a gerência de configuração e tratado como entidade única no processo de GCS. É todo e qualquer ente passível de manipulação, ou seja, que possa ser univocamente identificado e gerenciado. São normalmente considerados ICs, documentos, modelos e código (seja fonte, executável, *etc*.).

Para que cada item de configuração possa ser efetivamente gerenciado é necessário o estabelecimento de pontos bem definidos dentro do processo de desenvolvimento: as *baselines,* que também são chamadas na literatura como linhas de referência ou linhas de base ou configuração-base.

Segundo Villas Boas (2003), *baseline* é uma especificação ou produto que tenha sido formalmente revisto e acordado e que, a partir de então, serve como base para desenvolvimento futuro, podendo ser alterado apenas com o uso de um procedimento formal de alteração.

Esses pontos podem ocorrer ao final de cada uma das fases do processo de desenvolvimento ou de algum outro modo definido pela gerência. Nos pontos estabelecidos pelas *baselines*, os itens de configuração devem ser identificados, analisados, corrigidos, aprovados e armazenados em um local sob controle de acesso denominado repositório dos itens de configuração.

Quando um projeto de software não utiliza gerência de configuração podem ocorrer problemas como: perda de código-fonte, indisponibilidade e/ou dificuldade na determinação de versões de software a serem mantidas, impossibilidade de se gerenciar as bibliotecas de software necessárias ao funcionamento de um sistema, falta de sincronismo entre as atividades realizadas, dentre outros.

A inadequação no ambiente de produção para a implantação do produto, a ausência de controle sobre alterações efetuadas em documentos de apoio e no próprio código-fonte e de informações sobre a composição de uma determinada versão de sistema são outros problemas freqüentes ocasionados pela falta de GCS.

Segundo Pressman (2002), é importante salientar que gerência de configuração de software é um conjunto de atividades de acompanhamento e controle que começa quando o projeto de engenharia de software tem início e só termina quando o software é retirado de operação. Por este motivo é considerada uma atividade "guarda-chuva", visto que pode estar sendo executada durante todo o projeto, pois mudanças podem ocorrer a qualquer momento.

Dentre os benefícios que podem ser obtidos com a implementação adequada de GCS destacam-se: o controle efetivo e sistematizado do que é produzido, a geração de registro documental sobre o andamento evolutivo dos sistemas e suas alterações, facilidade de distribuição e nivelamento do conhecimento absorvido ao longo de todo o desenvolvimento realizado, tornando possível a rastreabilidade dos componentes que se encontram sob controle.

2.4.1. Funções da Gerência de Configuração

Conforme IEEE (1990), norma NBR ISO 10007 da ABNT (1996), Bersoff (1984) e outros autores, a GCS é dividida em quatro funções: identificação da configuração (que itens constituem uma configuração), controle da configuração (que passos no processo de alteração afetam uma configuração), auditoria da configuração (quais são as diferenças entre as versões) e contabilização da situação de configuração (que modificações foram feitas por determinado programador).

2.4.1.1. Identificação da Configuração

Esta atividade consiste em nomear de maneira única os componentes de uma configuração. Estes componentes básicos são os itens de configuração. Um IC pode ser

tanto um trecho de código quanto um documento, e deve ficar a cargo do projeto de controle de configuração definir quais serão os ICs do projeto que serão controlados.

A forma de nomeação dos ICs é completamente livre, mas deve ser padronizada dentro do projeto. Pode-se aproveitar a estrutura do produto (quando ela for hierárquica) ou uma outra forma qualquer. Para a indicação numérica, pode-se adotar, por exemplo, o uso de dois numerais, sendo o primeiro a indicação da configuração e o segundo a indicação da versão do IC. No entanto, hoje em dia esta informação é fornecida automaticamente pelas ferramentas de controle.

O padrão da ABNT/ISO para Gerência de Configuração, conforme visto em ABNT (1996), prevê as seguintes funções dentro dessa atividade:

1. **Seleção dos itens de configuração**: Os ICs devem ser escolhidos pelo processo de decomposição.

2. **Determinação da estrutura da configuração**: Esta estrutura deve descrever a posição de cada IC que compõe o projeto. O uso de uma estrutura hierárquica é recomendável.

3. **Documentação dos itens de configuração**: Todas as características físicas e funcionais de um item (interfaces, alterações, desvios e concessões) devem ser descritas em documentos bem identificados.

4. **Sistema de numeração**: Um sistema de identificação deve ser escolhido para garantir unicidade na identificação dos ICs.

5. **Estabelecimento de configurações básicas**: Configurações básicas (*baselines*) devem ser estabelecidas através de um acordo formal em determinados pontos do ciclo de vida do projeto e utilizadas como ponto de partida para o controle formal da configuração. Estes pontos podem ser as mudanças de fase no ciclo de vida, bem como, marcos estabelecidos no cronograma do projeto ou até mesmo necessidades emergenciais.

2.4.1.2. Controle de Configuração

A função de controle de configuração é designada para o acompanhamento da evolução dos ICs selecionados e descritos pela função de identificação.

Segundo Bersof (1984), o conceito de *baseline* é um dos fundamentos da Gerência de Configuração. A geração de uma *baseline* é o momento no qual é acordado que um ou mais IC está em conformidade com os requisitos do projeto e deve ser protegido contra alterações não autorizadas. Após o estabelecimento de uma configuração-base (*baseline*),

toda e qualquer alteração sobre os itens dessa configuração deve ser formalmente controlada.

A *baseline* é uma referência para desenvolvimento futuro, permite a verificação e recuperação das informações, representa um marco definido no ciclo de desenvolvimento do projeto e é uma referência para auditar o projeto.

O impacto da alteração, os requisitos do cliente e a configuração-base afetada influenciam o grau de formalidade do processo de alteração e podem servir de base para qualquer sistema de classificação utilizado para a classificação/categorização de alterações.

As seguintes atividades devem ser documentadas em detalhes no processo de controle: documentação e justificação da alteração, avaliação das conseqüências da alteração, aprovação/recusa dos pedidos de alteração, implementação e verificação das alterações, desvios e concessões.

A fim de proteger a integridade da configuração e fornecer a base para o controle de alteração, é essencial que os ICs sejam mantidos em um ambiente que: satisfaça as condições ambientais necessárias (por exemplo, para hardware, software, dados, documentos, diagramas.); proteja-os de alterações não autorizadas e destruição acidental; forneça mecanismos para recuperação; permita recuperação controlada de todos os dados armazenados; proporcione consistência entre a configuração construída e a especificada.

Dois controles básicos são instituídos no processo de gerência de configuração: controle de versão e controle de mudança.

Um item, ao ser desenvolvido, evolui até que atinja um estado que atenda os propósitos para o qual foi criado. Isso implica em diversas alterações, gerando uma versão (revisão) do item a cada estado. Para estabelecer o controle sobre as diversas versões, todas devem ser armazenadas e identificadas e, hoje, diversas ferramentas apóiam essa atividade.

A importância do controle de mudanças pode ser facilmente visualizada durante o processo de desenvolvimento, onde mudanças descontroladas podem levar rapidamente ao caos. Assim, deve-se instituir na organização um processo que combine procedimentos humanos e ferramentas automatizadas para proporcionar um mecanismo de controle de mudanças eficaz.

Algumas das ferramentas de apoio às atividades de controle de versão e de mudanças são apresentadas na seção 2.4.2 deste trabalho.

2.4.1.3. Contabilização da Satisfação de Configuração

A contabilização da situação da configuração consiste no registro e relato formais das informações necessárias para gerenciar a configuração de maneira efetiva. As informações englobam a listagem da identificação da configuração aprovada, o *status* das mudanças propostas à configuração e o *status* de implementação das mudanças aprovadas. É chamada, também, de contabilização do *status* da configuração, relato da configuração, entre outros.

De forma geral, a obtenção de informações sobre o *status* da configuração deve começar assim que os ICs são gerados. Estas informações devem permitir o rastreamento de todas as alterações efetuadas sobre os ICs. Desta maneira, pode-se saber quem efetuou uma alteração, quando ela foi executada e o que foi feito, por exemplo.

2.4.1.4. Auditoria da Configuração

SEI (2002) define a auditoria da configuração como a auditoria conduzida para verificar se um item de configuração está em conformidade com um padrão ou requisito especificados.

Segundo Bryan (1982) e o próprio IEEE (1993), auditoria é o processo de examinar um produto, sendo que esse processo é executado por um grupo independente do grupo que o produziu. Bryan também sustenta que quanto mais crítico é o projeto em termos de tamanho, custo ou cronograma, maior é a necessidade de se ter um processo eficiente de auditoria.

Pode-se dizer que auditoria é o preço da qualidade no projeto, no sentido de que através dela é assegurado ao usuário que o produto (no caso o software) está de acordo com os requisitos de desempenho e operacionais, ao cliente que será entregue no prazo especificado e dentro dos limites orçamentários e, por fim, ao fornecedor que seu produto evoluiu de uma maneira rastreável (e, portanto, manutenível) e de acordo com as expectativas do usuário final.

A auditoria de configuração deve ser executada antes que uma *baseline* seja definida, para certificar que o produto está de acordo com os requisitos (de especificação e contratuais) e também para garantir que está precisamente descrito em seus documentos técnicos.

Toda alteração também precisa ser auditada para garantir a integridade do produto que está sendo produzido. Ou seja, ela é utilizada para tornar visível ao gerenciamento o *status* do software ao longo do ciclo de vida do projeto e também para revelar se os

requisitos iniciais estão sendo satisfeitos e se a passagem de uma configuração a outra não descaracteriza o produto.

Duas atividades são executadas neste processo: verificação e validação.

A verificação é a atividade que assegura a coerência entre as configurações, ou seja, toda configuração candidata a se tornar uma nova configuração-base deve ser verificada com relação à configuração-base anterior.

A validação é a atividade que determina a coerência entre as várias configurações-base e a especificação de requisitos, ou seja, ela garante que cada configuração-base, em cada estágio do ciclo de vida, está de acordo com os requisitos especificados.

2.4.2. Ferramentas de Apoio

Segundo Dias (2006), do ponto de vista das ferramentas existentes, a GCS é apoiada pelas mesmas nas atividades: Controle de Versão, Controle de Mudança e Integração Contínua.

O controle de versão é a espinha dorsal de toda a gerência de configuração, apoiando as atividades de controle de mudança e integração contínua. O sistema de controle de versão rastreia e controla todos os artefatos do projeto (código-fonte, arquivos de configuração, documentação, *etc*.) e desse modo consegue coordenar o trabalho paralelo de desenvolvedores.

Essas ferramentas oferecem serviços como: identificação, armazenamento e gerenciamento dos itens de configuração e de suas versões durante todo o ciclo de vida do software; histórico de todas as alterações efetuadas nos itens de configuração; criação de rótulos e ramificações no projeto; recuperação de uma configuração em um determinado momento desejado do tempo.

O controle de versão de software é uma prática de engenharia de software comprovadamente eficaz. Por isso, faz parte das exigências para melhorias do processo de desenvolvimento de certificações tais como CMMI-SE/SW e MPS.BR.

O controle de mudanças fornece um serviço complementar ao oferecido pelo sistema de controle de versão. O foco desse tipo de ferramenta é nos procedimentos pelos quais as mudanças de um ou mais itens de configuração são propostas, avaliadas, aceitas e aplicadas. Oferece serviços para identificar, rastrear, analisar e controlar as mudanças nos itens de configuração.

Para as necessidades da GCS, bastaria um controle de construção de software que cuidasse da identificação, empacotamento e preparação de uma *baseline* para a entrega a um cliente externo ou interno, tornando-a uma *release*[2] ou uma *build*[3], respectivamente.

A idéia de utilizar uma integração contínua, entretanto, vai um pouco mais além. O objetivo é garantir que as mudanças no projeto são construídas, testadas e relatadas tão logo quanto possível depois de serem introduzidas.

Em projetos de software, a construção do software é feita pela recuperação da configuração correta no sistema de controle de versão e a construção dos arquivos executáveis e de instalação do produto. Este processo é executado geralmente após cada mudança publicada no sistema de controle de versão ou em intervalos de tempo pré-definidos.

Geralmente, são combinadas duas ferramentas separadas: uma que faz a construção do software e outra que monitora alterações no controle de versão e dispara a primeira para a construção.

Existem diversas ferramentas disponíveis para apoiar atividades de GCS, como: para controle de versão, Subversion, CVS, ClearCase, StarTeam; para controle de mudança, Trac, Mantis, Bugzilla, Jira, FogBugz e para integração mútua, Scons, Bitten, Ant, AntHill Pro, FinalBuilder.

A quantidade de funcionalidades, a maturidade, a documentação e o suporte disponíveis, e a popularidade de cada ferramenta variam bastante. Embora as ferramentas comerciais geralmente apresentem mais funcionalidades e um maior grau de integração, o custo de licenciamento muitas vezes torna o seu uso proibitivo principalmente para micro e pequenas empresas de software.

As ferramentas *open source*[4], por outro lado, possuem diversas vantagens além do custo mais baixo de aquisição tais como qualidade, segurança, independência de fornecedor, possibilidade de adequação a necessidades específicas, estabilidade e suporte técnico. Essas características tornam a escolha por ferramentas *open source* uma solução extremamente interessante não só para micro e pequenas empresas.

---

[2] *Release*, que também é chamada de liberação, é o nome dado a uma versão disponibilizada para um propósito específico. Por exemplo, *release* para testes, liberação para entrega ao cliente.
[3] *Build* é a construção do sistema a partir dos itens fonte para uma configuração alvo.
[4] Ferramenta *open source*, ou em português, código aberto, é um tipo de ferramenta cujo código-fonte é público.

2.5. Considerações Finais do Capítulo

Neste capítulo foram apresentados conceitos relacionados à Engenharia e Qualidade de Software, Melhoria do Processo de Software, Fábrica de Software e Gerência de Configuração de Software.

Depois de elaborado o estudo sobre as funções e as ferramentas de apoio à gerência de configuração de software, foi possível definir as que seriam utilizadas na implantação no processo de desenvolvimento da empresa Beta.

3. METODOLOGIA

É apresentada neste capítulo a metodologia que foi aplicada no desenvolvimento deste projeto e a descrição de como o estudo de caso foi realizado.

3.1. Tipo de Pesquisa

O método de pesquisa utilizado neste trabalho foi a pesquisa-ação, de natureza tecnológica, com objetivos de caráter exploratório, utilizando procedimentos de estudo de caso fundamentada em referências bibliográfica e documental (Jung, 2004).

A pesquisa-ação é quando os pesquisadores e participantes representativos da situação ou do problema estão envolvidos de modo cooperativo ou participativo.

Uma pesquisa que utiliza conhecimentos básicos, tecnologias existentes, conhecimentos tecnológicos, e que tenha como objeto um novo produto ou processo é caracterizada como tecnológica.

Pesquisa de caráter exploratório visa o aprimoramento de idéias ou a descoberta de intuições, ou seja, fornecer ao pesquisador um maior conhecimento sobre o tema ou problema de pesquisa em questão. Este tipo de pesquisa proporciona maior familiaridade com o problema, tornando-o mais explícito.

Segundo Jung (2004), normalmente, a pesquisa exploratória não exige grandes teorizações e, sim, a experimentação para coleta de dados que servirão de base para a formulação de modelos inovadores ou explicativos.

Quanto aos procedimentos o tipo de pesquisa adotado foi o estudo de caso realizado em campo, ou seja, a pesquisa foi realizada dentro da fábrica de software da empresa “Beta”.

O estudo de caso é um estudo aprofundado e exaustivo de um ou de poucos objetos, de maneira a permitir o seu conhecimento amplo e detalhado. É adequado para explorar situações da vida real, descrever a situação do contexto em que está sendo feita determinada investigação e explicar as variáveis causais de determinado fenômeno em situações muito complexas.

Através do estudo de caso foi possível explicar e descrever um objeto dentro do contexto local e real e correlacioná-lo ao que existe na literatura sobre o tema, nesse trabalho o objeto de estudo é o processo de gerência de configuração na organização “Beta”.

O embasamento teórico da pesquisa é imprescindível e nesse trabalho as referências adquiridas foram através de estudo e referencial bibliográfico e documental.

Conforme Jung (2004), a pesquisa bibliográfica tem por finalidade principal formar uma consistente base "mental" a partir daquilo que é existente, e oportunizar uma ampla aquisição de conhecimentos para o entendimento substancial do assunto, viabilizando ao pesquisador "ousar" ao propor novos argumentos que justifiquem as descobertas.

Ainda de acordo Jung (2004), assim como a pesquisa bibliográfica, a documental visa formular uma base consistente de conhecimentos ao pesquisador, fornecendo a estas fontes subsidiárias para importantes aplicações referenciais. A diferença existente entre uma e outra consiste no tipo e estado de tratamento do material textual. As informações existentes em documentos são muitas vezes de caráter inédito e, por isto, esta fonte é indispensável ao processo de pesquisa e pode representar um diferencial na formulação dos argumentos.

Os documentos analisados na empresa "Beta" foram o seu processo de desenvolvimento de software, seus *templates*, políticas e manuais.

3.2. Procedimentos Metodológicos

O trabalho iniciou-se com o interesse pessoal da autora em aprofundar seus conhecimentos em Engenharia de Software, mais especificamente na disciplina Gerência de Configuração e tornou-se possível com o apoio da empresa "Beta", estudo de caso.

A pesquisa foi realizada no período de abril a agosto de 2006, no ambiente de desenvolvimento da fábrica de software da empresa Beta.

A primeira etapa foi definir o tema da pesquisa a ser realizada, bem como, o problema a ser investigado.

Em seguida, foi realizada uma revisão bibliográfica sobre os aspectos da engenharia e qualidade de software, melhoria do processo de software, fábrica de software e gerência de configuração de software, a fim de se obter maior conhecimento sobre os temas abordados.

Na seqüência, buscou-se conhecer a empresa, através de um diagnóstico da situação atual da empresa no que se refere à gerência de configuração, que foi realizado pela pesquisadora desse trabalho no ambiente de desenvolvimento da organização "Beta".

A partir desse diagnóstico, foi planejada a implantação e definidas as atividades relacionadas às práticas de gerência de configuração que seriam incorporadas ao processo

de desenvolvimento de software da organização. Também foram selecionados projetos pilotos para a implantação de tais práticas possibilitando, assim, uma avaliação da implantação.

As práticas de gerência de configuração implantadas no processo de desenvolvimento foram adaptadas à realidade da empresa estudada, levando em consideração o fato se tratar de uma empresa de pequeno porte, jovem e com poucos recursos humano e financeiro para o desenvolvimento desse trabalho.

No processo de execução da implantação foram institucionalizadas as práticas através de treinamentos realizados com todos os colaboradores da empresa, principalmente, os envolvidos nos projetos pilotos.

Após a implantação nos projetos pilotos, ocorreu um processo de avaliação da implantação onde foram identificadas inconsistências, melhorias às práticas de GCS definidas.

Ao final, o consenso da adoção das práticas de GCS em todos os projetos da fábrica de software da empresa Beta foi estabelecido.

4. RESULTADOS E DISCUSSÃO

Neste capítulo apresenta-se o estudo de caso realizado na empresa desenvolvedora de software Beta e as principais atividades que foram executadas para a implantação de práticas de gerência de configuração no processo de desenvolvimento da empresa.

4.1. Caracterização da Empresa

A empresa utilizada como estudo de caso, está situada na cidade de Lavras – MG. É uma empresa da área de Tecnologia da Informação (TI) que tem como principais áreas de atuação:

- Fábrica de Software: atua no mercado de TI, desenvolvendo soluções eficientes, que trazem grandes transformações, gerando retornos financeiros e operacionais aos seus clientes;
- Ensino a Distância: oferece suporte técnico e operacional para cursos ministrados a distância (ambiente virtual de ensino a distância).

A empresa conta atualmente em seu quadro de funcionários com 30 (trinta) colaboradores, sendo 14 (quatorze) profissionais graduados em Ciência da Computação e 16 (dezesseis) estagiários do curso de Ciência da Computação da Universidade Federal de Lavras – UFLA.

Apesar de ser uma empresa jovem – apenas 4 (quatro) anos – e formada por profissionais recém-formados e estagiários, o comprometimento com a qualidade de seus produtos sempre foi tomada como uma premissa dentro da organização.

4.2. Diagnóstico das Práticas de Gerência de Configuração Atuais

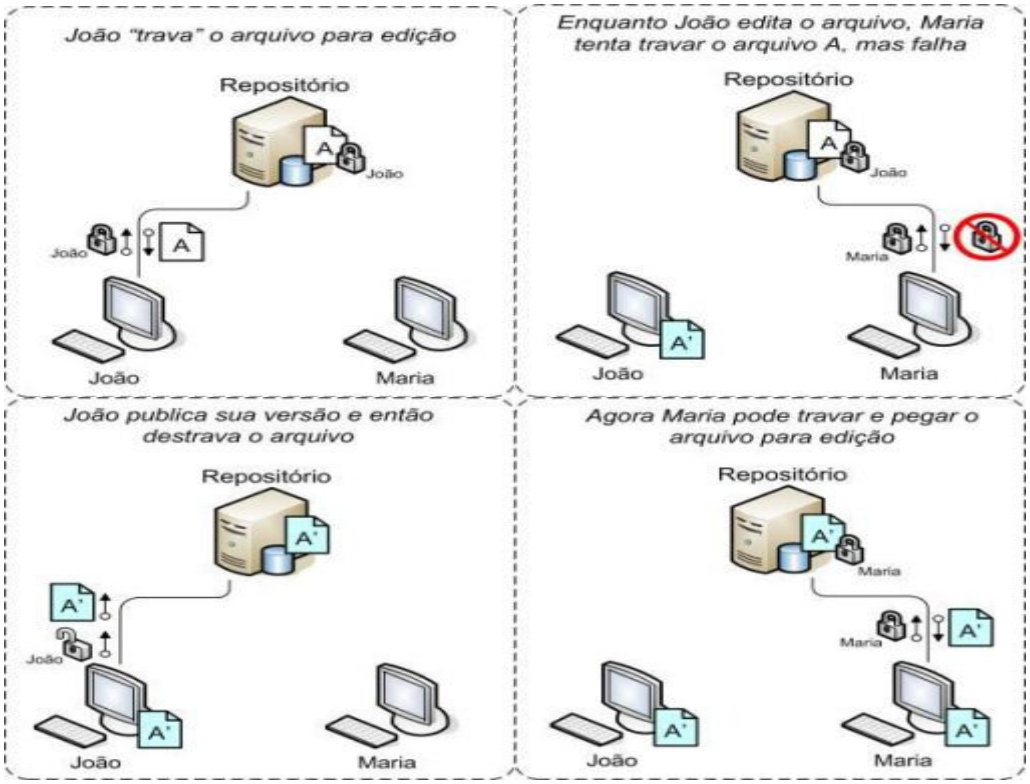
Com o intuito de planejar as atividades de gerência de configuração de software e saber o estado atual da empresa, com relação às práticas desta disciplina, foi feito um diagnóstico na empresa. Esse diagnóstico foi realizado com a participação dos colaboradores que possuíam visões técnicas, gerenciais e administrativas dos projetos executados pela empresa.

O diagnóstico foi executado através de entrevista com os colaboradores, análise do processo definido da empresa, bem como, participação ativa da pesquisadora em projetos da empresa, onde se evidenciou toda a situação da empresa no contexto da GCS.

Apesar de a empresa possuir um processo de desenvolvimento de software definido, esse não tratava a gerência de configuração formalmente. Não existia institucionalização do uso das práticas de gerência de configuração, como também não havia documentação específica a ser aplicada ao processo de desenvolvimento do produto da empresa.

As funções da gerência de configuração não estavam presentes nem mesmo informalmente no processo da empresa. Não existiam atividades que fizessem a identificação da configuração, o controle da configuração, a contabilização da situação da configuração e a auditoria da configuração. Nem mesmo, parte dessas atividades eram realizadas.

A única iniciativa que existia na empresa era a utilização da ferramenta *FreeVCS* para o controle de versão. Porém, na prática, essa ferramenta auxiliava apenas o controle de concorrência entre os desenvolvedores, cuja política é "trava-modifca-destrava" (ver Figura 4.1) e a cópia da versão mais atual de cada um dos arquivos que se encontravam no repositório para a local de trabalho dos desenvolvedores (ver Figura 4.2).



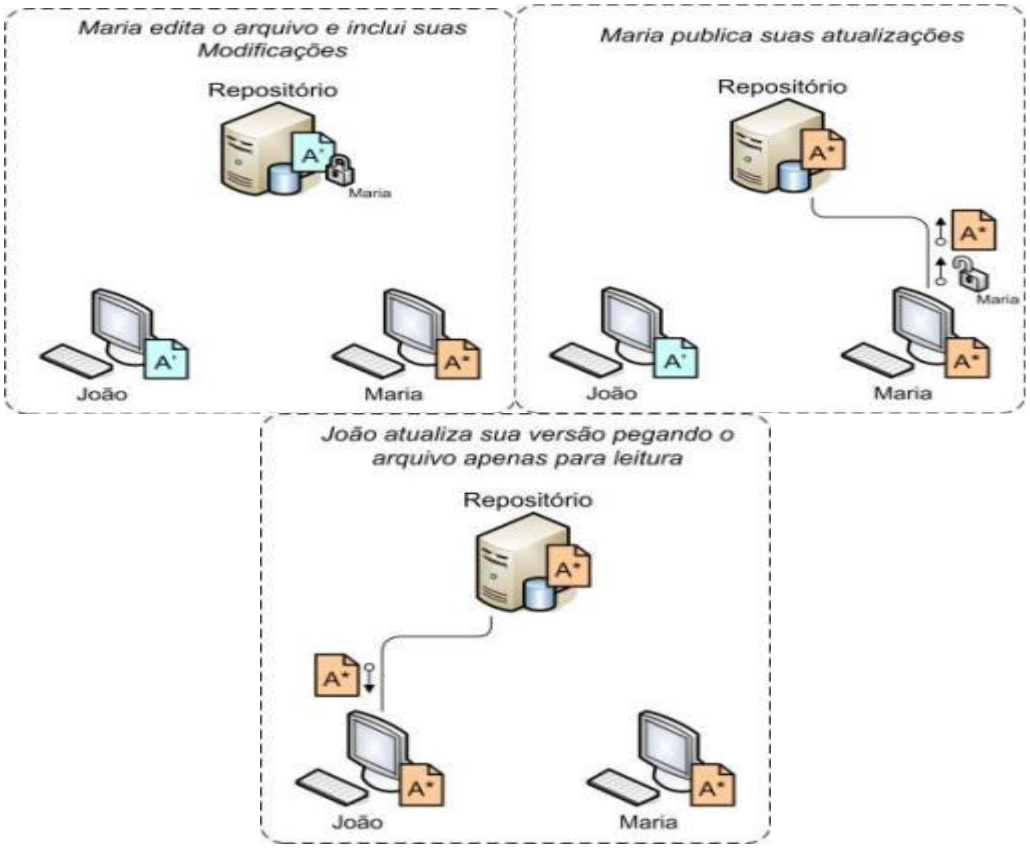


Figura 4.1 - Política "trava-modifica-destrava"
Fonte: Dias (2006)

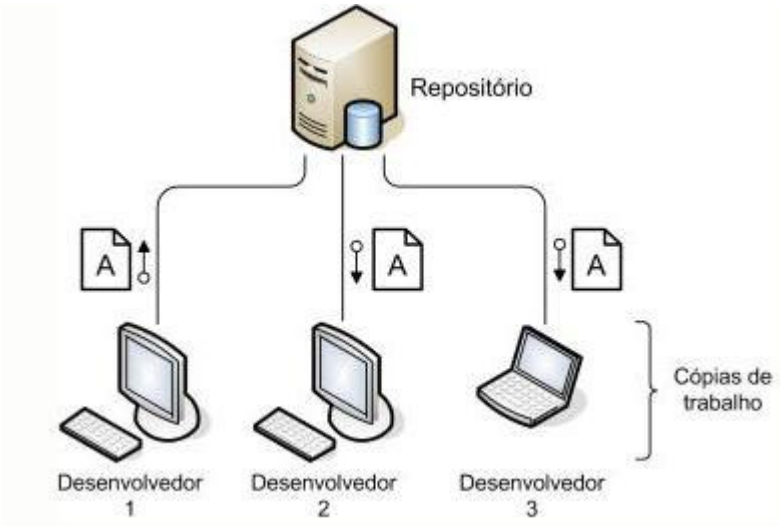


Figura 4.2 - Cópia dos arquivos do repositório para o local de trabalho
Fonte: Dias (2006)

A imaturidade com relação à disciplina GCS era notória na empresa, pois muitos colaboradores ligados diretamente ao desenvolvimento de software nem mesmo conheciam

termos e conceitos importantes dessa disciplina como: *baseline*, *branch*[5], *tag*[6], *release*, contabilização da situação, entre outros. O fato de serem termos em inglês poderia levar à errônea conclusão de que as pessoas não conheciam tais termos por serem estrangeiros; porém quando apresentados os seus mais diversos sinônimos, como: linha de base, ramificação, rótulo, liberação – esses também não eram conhecidos.

As especificações dos recursos de hardware e software a serem utilizados por um projeto não eram documentadas, eram informações que as pessoas não possuíam dentro da organização. Por exemplo, não existia documento que informasse em um projeto desenvolvido (ou a ser desenvolvido) quais eram as configurações necessárias para as máquinas da equipe do projeto (processador, memória), software a serem utilizados pela equipe de desenvolvimento (sistema operacional, servidor *web*, sistema gerenciador de banco de dados), políticas de backup.

A empresa não possuía um responsável pela GCS, nem tampouco, alguma pessoa que possuía um conhecimento mais aprofundado nas funções e atividades da GCS. A empresa ainda não estava ciente da necessidade e suma importância de se institucionalizar as práticas de GCS no processo de desenvolvimento dentro da organização e definir responsáveis para desempenhar as atividades de GCS.

Não havia, no histórico da organização, tentativa de implantação de práticas de GCS, nem mesmo, projetos que aplicavam essas práticas de forma isolada; porém a necessidade dessa implantação apresentava-se cada vez mais nos projetos onde diversos problemas ocorriam pela falta de GCS.

Problemas que aconteciam comumente durante o desenvolvimento dos projetos da empresa eram: perda de código-fonte; impossibilidade de determinar o que aconteceu com um programa, ou parte dele; e, também, de determinar quem, por que e quando foram efetuadas modificações; desaparecimento de requisitos documentados e implementados; programa em execução e seu código-fonte em diferentes versões.

Esses foram os principais pontos fracos da organização no que se refere à GCS, porém foram levantados pontos fortes também.

Através do diagnóstico foi possível perceber que como pontos fortes a organização possuía o fato de ter se submetido à implantação de melhorias do processo de

[5] *Branch*, também conhecida como ramo ou ramificação, é uma versão que não segue a linha principal de desenvolvimento.
[6] *Tag*, também chamada de rótulo ou etiqueta, é o mecanismo usado para identificar uma configuração.

desenvolvimento recentemente e, com isso, existir uma conscientização da necessidade de melhorias contínuas e buscar alternativas de se contornar os seus pontos francos.

Os colaboradores sentiam que mudanças deveriam acontecer para evitar diversos problemas, porém não sabiam que a implantação de práticas de GCS remediaria boa parte destes problemas.

Com o diagnóstico, a organização percebeu que se implantadas e seguidas essas novas práticas poderiam minimizar um número muito grande de problemas que ocorriam durante o desenvolvimento, bem como, produzir softwares com mais qualidade.

A empresa motivou-se a buscar a identificação e o detalhamento dessas práticas, para que as mesmas fossem adotadas de fato. Achou-se conveniente que a implantação ocorresse de maneira mais suave, evitando alterações bruscas na cultura da organização.

4.3. Planejamento da Implantação das Práticas de GCS na Empresa

A implantação das práticas de GCS no processo de desenvolvimento de software da empresa foi planejada e separada por 5 (cinco) etapas que são apresentadas na Figura 4.3:

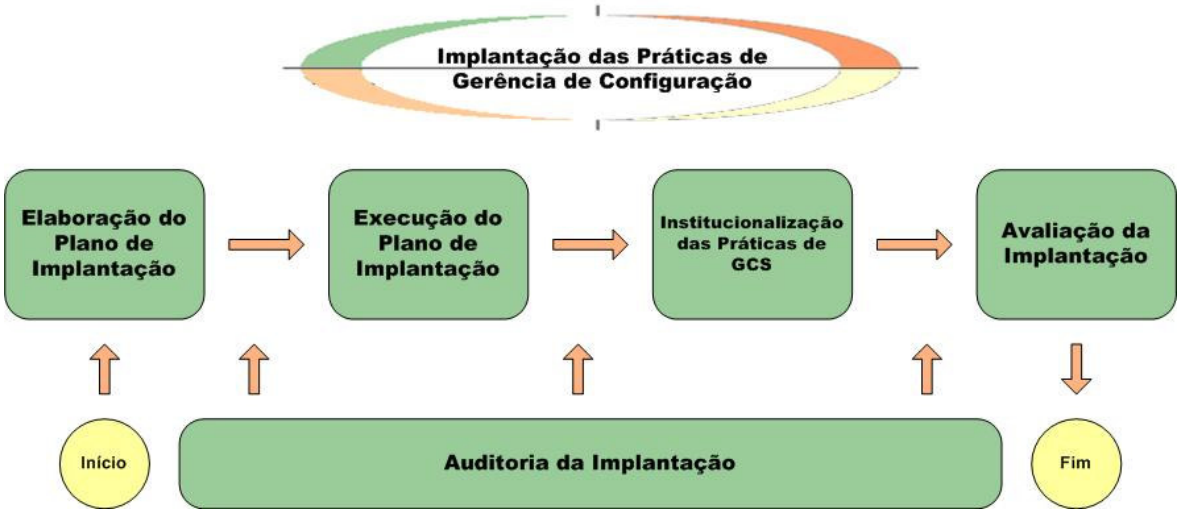


Figura 4.3 - Etapas da Implantação das Práticas de GCS
Fonte: Elaborado pela autora

4.3.1. Auditoria da Implantação

Para a implantação das práticas de gerência de configuração de software foram realizadas auditorias durante toda a execução deste projeto. Dentro de cada uma das etapas, as atividades foram submetidas à auditoria. Pensou-se inicialmente em realizar a auditoria somente na transição de cada uma das etapas, porém essa proposta mostrou-se inviável. A

complexidade e o alto impacto de algumas atividades dentro de algumas etapas deveriam ser acompanhadas com um pouco mais de cuidado para não afetassem de forma negativa os projetos e a organização como um todo.

Na empresa *Beta* existe um grupo chamado SEPG (*Software Engineering Process Group*) que é responsável pela definição, manutenção e melhoria do processo de software da organização. A execução da auditoria da implantação ficou sob sua responsabilidade. Essa escolha fundamentou-se no fato desse grupo ter total conhecimento do processo definido na empresa; ter um compromisso com a busca contínua de melhorias e qualidade; o mesmo já possuir experiência em auditorias de outras naturezas, entre outros.

A função da auditoria se resumiu em acompanhar cada uma das etapas e verificar se as atividades propostas dentro das mesmas estavam sendo executadas corretamente e no tempo estimado. Caso isso não estivesse acontecendo, analisavam-se as causas, os impactos e propunham-se soluções alternativas para contornar os problemas.

4.3.2. Elaboração do Plano de Implantação

A partir dos resultados encontrados no diagnóstico e visando uma melhor condução das atividades relacionadas ao projeto, foi elaborado um plano de ação denominado *Plano de Implantação*.

O *Plano de Implantação* é um plano de ação que identifica os objetivos do projeto, quais os impactos do mesmo sobre a organização, o método de trabalho que será utilizado, as ações específicas e os recursos necessários para a implantação das práticas de GCS no processo de desenvolvimento de software.

Assim, nessa etapa de implantação, foi elaborada a documentação de ações que deveriam ser executadas dentro do propósito de implantar as práticas de GCS na organização atendendo principalmente o que foi detectado no diagnóstico realizado.

4.3.3. Execução do Plano de Implantação

As atividades relacionadas à implantação das práticas de gerência de configuração foram descritas no *template*[7] denominado *Plano de Gerência de Configuração*. Nesse plano é apresentado como e quando as atividades serão efetuadas, quem serão os responsáveis por elas e quais recursos serão necessários. O plano de gerência de

[7] *Template* é um modelo de documento definido como padrão e que é utilizado no processo de desenvolvimento de software.

configuração foi desenvolvido de acordo com as peculiaridades da empresa e também as atividades que compõem a GCS, segundo NBR ISO 10007, ABNT (2006).

Foi recomendado que o plano de gerência de configuração adotasse padrões que tinham maior compatibilidade com o projeto para o qual o plano estivesse sendo escrito, ou seja, que sempre fosse adaptado para a realidade do projeto, levando-se em consideração tecnologia, escopo do projeto, enfim, às necessidades do mesmo. No Apêndice A deste trabalho, é apresentado o *Plano de Gerência de Configuração*.

Foram apresentadas as responsabilidades na execução das práticas para os diversos papéis dos colaboradores dentro da organização. Criou-se um grupo de gerência de configuração (CMG – *Configuration Management Group*) na empresa, que seria o responsável por aplicar melhorias constantes nas diversas atividades definidas no processo de GCS. Esse grupo foi formado por um gerente de configuração, um membro do grupo SEPG, o gerente do projeto e um membro da equipe de infra-estrutura da empresa, podendo ter mais de um colaborador na mesma área, porém não menos que esta estrutura de quatro integrantes e das áreas descritas.

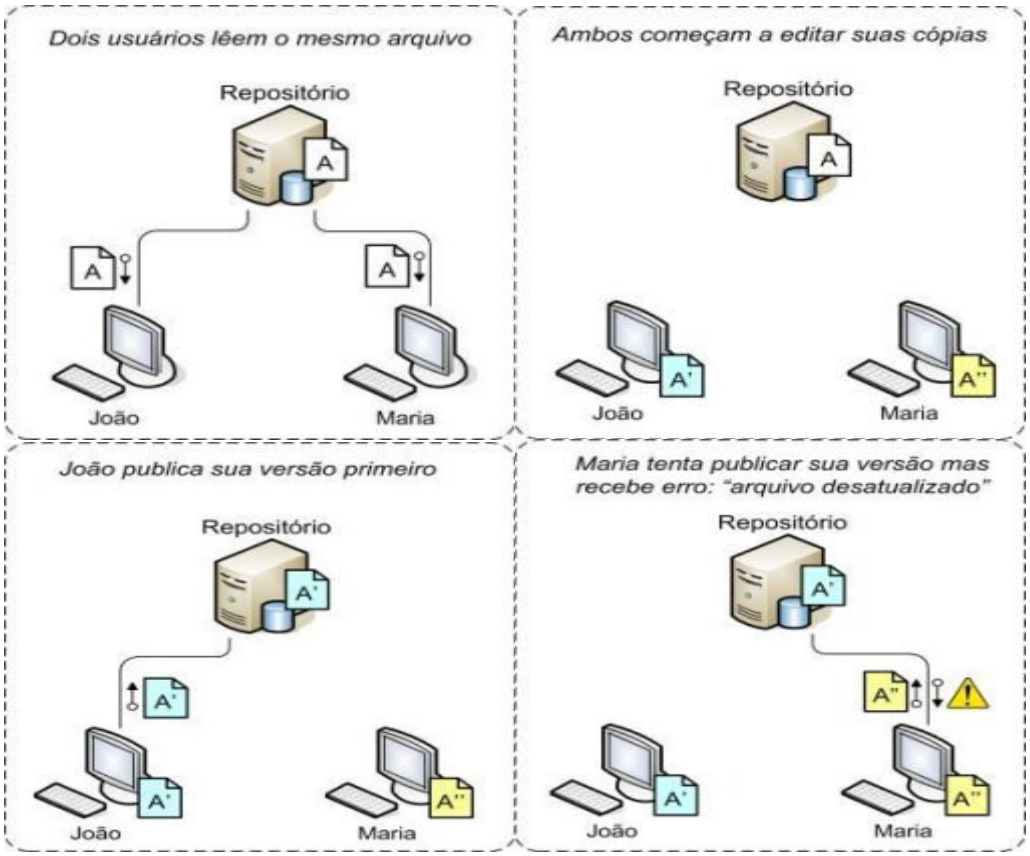
Como a empresa não possuía condições para alocar uma grande quantidade de colaboradores para integrarem o CMG, nem mesmo, condições de permitir que os integrantes se dedicassem exclusivamente ao grupo, decidiu-se que as funções desempenhadas pelo grupo seriam realizadas paralelamente às suas demais funções dentro da organização.

Houve a elaboração do *Plano de Ambiente*, cujo objetivo é definir formalmente os recursos de hardware e software a serem utilizados por um projeto. Por exemplo, todo projeto desenvolvido (ou a ser desenvolvido) possuirá um documento ao qual serão encontradas informação como: quais foram as configurações necessárias para as máquinas da equipe do projeto (processador, memória), softwares e suas versões utilizados pela equipe de desenvolvimento (sistema operacional, servidor web, sistema gerenciador de banco de dados), políticas de backup. O *Plano de Ambiente* encontra-se no Apêndice B.

Foi desenvolvido um *Glossário de Gerência de Configuração* para que todos os envolvidos no desenvolvimento passassem a conhecer os conceitos acerca das novas práticas implantadas no processo de desenvolvimento do software, uma vez que, o diagnóstico apontou como fato o desconhecimento da disciplina GCS e dos conceitos a ela vinculados. Termos da língua estrangeira foram apresentados, neste glossário, também com sua tradução para o português para facilitar a familiarização com os conceitos

contidos nas práticas de GCS. O *Glossário de Gerência de Configuração* é apresentado no Apêndice C.

Decidiu-se migrar da ferramenta *FreeVCS* para o *Subversion* para a realização de do controle de versão dos itens de configuração. O pouco suporte oferecido pelo *FreeVCS* às funções de GCS, motivou essa migração. A escolha do *Subversion* baseou-se em diversos fatores, como: ser uma ferramenta *open source*; oferecer suporte a funções importantes para o controle da versão como *branch*, *tag*; utilizar a política "copia-modifica-resolve" (ver Figura 4.2) para o controle de concorrência e ainda permitir o travamento de arquivos que não podem ser mesclados com facilidade (*merge*[8]); armazenamento através de versionamento de diferenças[9] (delta) no repositório, reduzindo o espaço requerido em disco; entre outros.



[8] *Merge*, nesse contexto, significa mesclar os arquivos modificados simultaneamente.
[9] Versionamento de diferenças, também chamado de versionamento delta, é a forma pela qual os ICs são armazenados no repositório. Nesse caso, o repositório armazena apenas as modificações sofridas pelo IC, ou seja, não salva tudo novamente, apenas o que foi alterado.

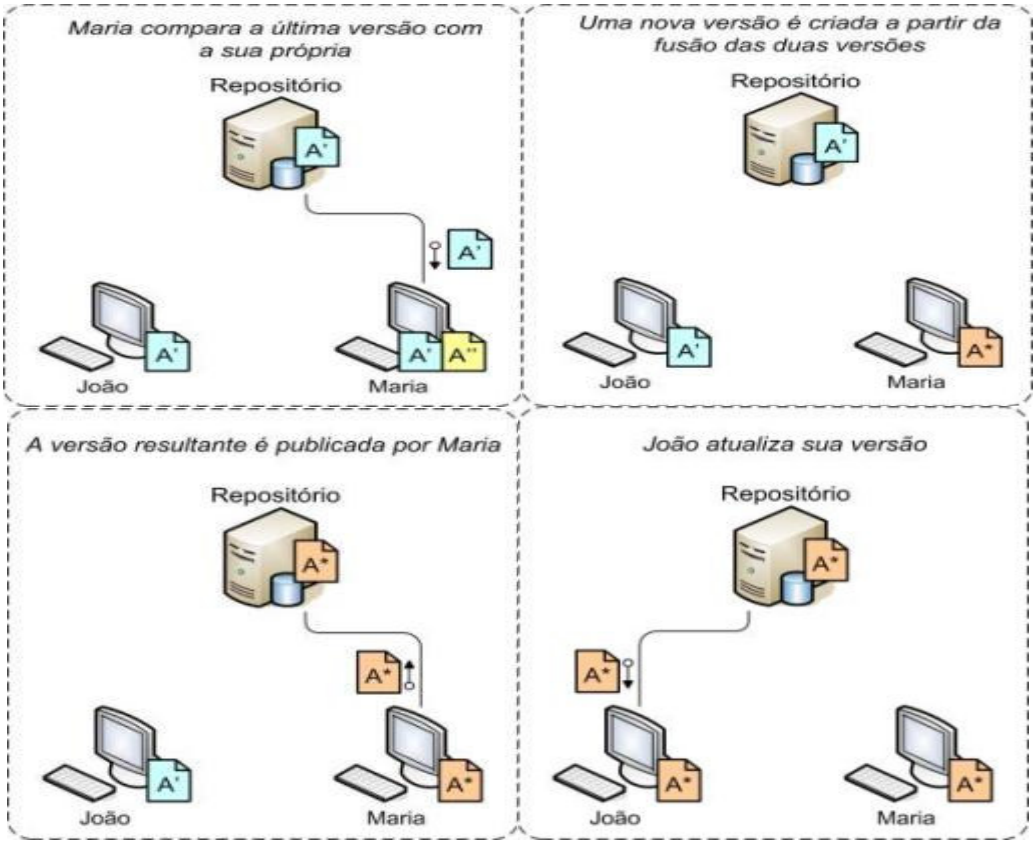


Figura 4.4 - Política "copia-modifica-resolve"
Fonte: Dias (2006)

Foi realizada uma adaptação dos fluxos de *bugs*[10] e requisitos do processo de teste para que os mesmos auxiliassem e atendessem às práticas relacionadas ao controle de mudanças. Os fluxos foram desenhados baseados na utilização da ferramenta *Mantis*.

Essa ferramenta já é utilizada na organização para o processo de relatos de erros e inconsistências encontradas e para atribuição de atividades como a implementação de requisitos para os desenvolvedores.

Com isso, viabilizou-se o uso também dessa ferramenta para se aplicar o controle de mudanças, assim, a execução dessa atividade tornou-se menos complicada e mais atrativa para os envolvidos, uma vez que foram adicionadas atividades às que os colaboradores já desempenhavam.

Após reuniões com os membros do SEPG, as práticas de GCS foram definidas, os *templates* gerados, planos e documentos, a ferramenta *Subversion* foi adotada e melhorias foram aplicadas baseando-se nos procedimentos já existentes na empresa.

[10] Um *bug*, nesse contexto, é qualquer falha encontrada no software que o impede de funcionar como esperado.

4.3.4. Institucionalização das Práticas de GCS

Para a institucionalização das práticas implantadas no processo de desenvolvimento da empresa foi realizado treinamento com todos os colaboradores envolvidos no processo de desenvolvimento de software, onde foi exposto: em qual momento do desenvolvimento as novas práticas seriam aplicadas; a utilidade de cada um dos documentos, *templates* e planos e como esses deveriam ser utilizados e, principalmente, a importância de se adotar essas práticas definidas.

Também foram apresentadas palestras a fim de familiarizar cada vez mais os colaboradores com novas atividades por eles desempenhadas e, também, incentivá-los a utilizar o que foi definido da forma desejada.

Os temas dessas palestras foram: "Introdução à Gerência de Configuração de Software", onde foram introduzidos os conceitos iniciais (apresentação formal do *Glossário de Gerência de Configuração*), as principais atividades dentro da disciplina GCS e o fluxo sintético da gerência de configuração; "Utilização do *Subversion*", apresentando as principais funções da ferramenta e o fluxo de operação desta ferramenta; "Utilização e preenchimento dos *Templates* – Planos, Documentos e Outros Artefatos", exposição de todos os artefatos relacionados às práticas de GCS mostrando em cada um desses artefatos os seus objetivos, suas seções e subseções, simulando o preenchimento dos mesmos tanto e conscientizando sobre a importância do correto preenchimento tanto para o projeto como para a organização.

Foram selecionados dois projetos pilotos da organização para a implantação das práticas de GCS: Sistema de Administração Escolar e Sistema Integrado de Venda de Imóveis. O primeiro é um sistema de administração de escolas de ensinos fundamental e médio; conta com cadastros de alunos, professores, turmas além de lançar alunos em uma série, disciplinas para um professor, notas e faltas de um aluno e, também, gerar boletim, histórico escolar, entre outros. O segundo trata-se de um sistema que automatiza a venda de imóveis e é formando por cinco módulos: contábil, financeiro, administrativo, comercial e relatórios, englobando em seus módulos funcionalidades como cadastros desde o loteamento até a forma de pagamento das vendas de lotes, geração de boletos bancários, controle financeiro dos diversos setores da empresa, além de emitir relatórios de lotes à venda, contas em atraso, projetos dos loteamentos, entre outros.

Nesses projetos foi possível verificar o uso das práticas, bem como, suas inconsistências, erros existentes e procedimentos desnecessários que somente estavam burocratizando o processo de desenvolvimento.

4.3.5. Avaliação da Implantação

Ao final da implantação das práticas de gerência de configuração de software foi realizada uma avaliação, levando em conta todos os envolvidos neste projeto e os resultados desta avaliação são apresentados ao final da seção 4.5 deste trabalho.

4.4. Implantação das Práticas de GCS

Após a realização de todas as atividades citadas acima e uma auto-avaliação, as práticas de GCS inseridas ao novo processo de desenvolvimento de software da empresa mostraram-se como uma forma viável e aplicável, não somente aos dois projetos pilotos, mas a todos os outros projetos da organização.

As práticas de gerência de configuração definidas foram: desenvolvimento do plano de gerência de configuração, definição do plano de ambiente, identificação da configuração, controle da configuração, contabilização da situação da configuração e auditoria da configuração, conforme ilustra a figura 4.5.

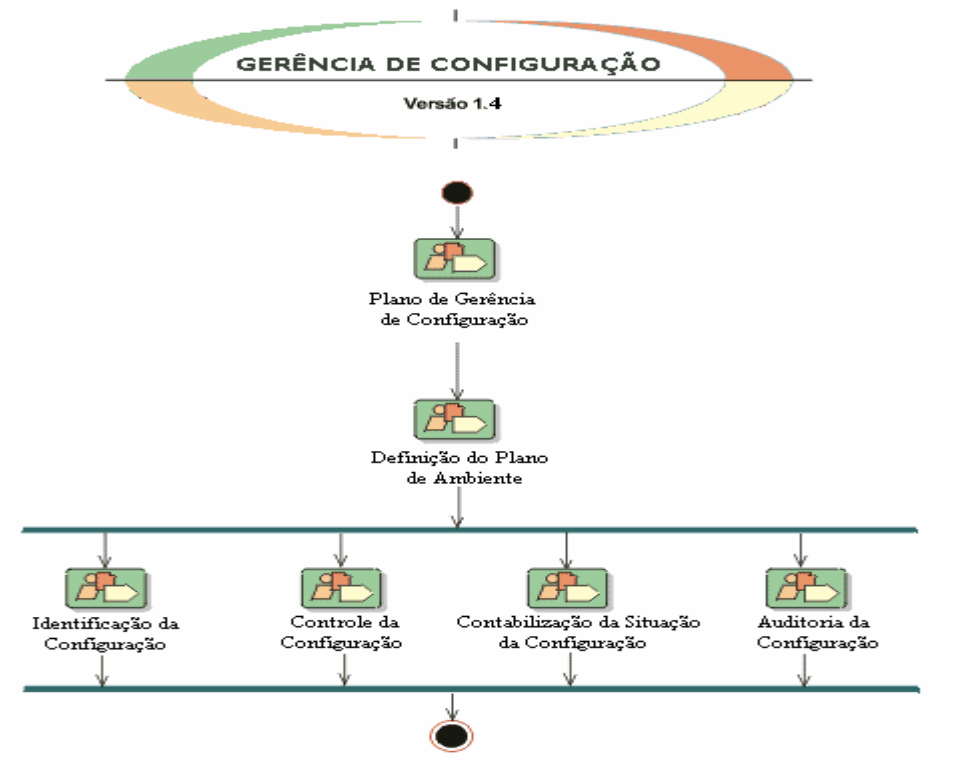


Figura 4.5 - Práticas de Gerência de Configuração
Fonte: Elaborado pela autora

Para cada uma das macro-atividades das práticas de GCS foram definidas outras atividades para que fossem gerados os produtos de trabalho que, nesse contexto, são os artefatos que devem ser produzidos ao longo do processo de gerência de configuração de software.

As próximas subseções descrevem as práticas que foram definidas para serem aplicadas ao longo do processo de desenvolvimento da empresa Beta.

4.4.1. Plano de Gerência de Configuração

O propósito do *Plano de Gerência de Configuração* é definir formalmente as ferramentas a serem utilizadas e os procedimentos a serem seguidos durante a evolução do software em desenvolvimento.

Os leitores alvo deste documento são os novos membros da equipe de um projeto. Também, pode ser utilizado por membros experientes quando houver a necessidade da aplicação de algum procedimento para gerência de configuração.

Um plano de gerência de configuração é elaborado contendo as diretrizes que todo projeto dentro da empresa deve seguir, além das suas diretrizes específicas.

4.4.2. Plano de Ambiente

O *Plano de Ambiente* define os recursos de *hardware* e *software* a serem utilizados por um projeto. Por exemplo, todo projeto desenvolvido (ou a ser desenvolvido) possuirá um documento ao qual serão encontradas informação como: quais foram as configurações necessárias para as máquinas da equipe do projeto (processador, memória), softwares e suas versões utilizados pela equipe de desenvolvimento (sistema operacional, servidor web, sistema gerenciador de banco de dados), políticas de backup.

Este plano tem como alvo novos membros do time de um projeto. Também, pode ser usado por membros experientes quando houver a necessidade de realizar alguma mudança no ambiente.

4.4.3. Identificação da Configuração

Nessa atividade são definidas as seguintes funções: seleção dos itens de configuração; determinação da estrutura da configuração; documentação dos itens de configuração; sistema de numeração; estabelecimento de *baselines*.

Sendo assim, inicialmente são selecionados os itens de configuração a serem gerenciados. É importante que sejam selecionados itens relevantes, porque uma superdocumentação onera muito a gerência de configuração. Geralmente, os itens mais usados no ciclo de vida, os mais genéricos, os mais importantes para a segurança, os projetados para reutilização e os que podem ser modificados ao mesmo tempo, sofrem gerenciamento de configuração. Somente os itens selecionados serão controlados, sendo que os outros itens poderão ser alterados livremente.

Após a seleção, determina-se a estrutura da configuração, ou seja, descreve-se como esses itens se relacionam. A identificação dos relacionamentos é muito importante para a manutenção, pois permite que se localizem rapidamente os itens afetados decorrentes de cada alteração.

Depois de escolhidos os itens e estabelecidos os relacionamentos, são descritas todas as características físicas e funcionais de um item (interfaces, alterações, desvios e concessões) em documentos bem identificados.

Depois é definido um esquema de identificação dos itens, com a atribuição de nomes exclusivos (garantindo a sua unicidade) a cada um dos componentes, para que seja possível reconhecer a evolução da cada uma das versões dos componentes e a hierarquia existente entre eles a partir de seus nomes. Um esquema de identificação simples utiliza a combinação de nome do projeto, tipo de item, nome do item e versão.

Após o estabelecimento do esquema de identificação, são planejadas as *baselines* dentro do ciclo de vida do projeto. Geralmente, cria-se uma linha de referência ao final de cada fase do ciclo de vida do projeto e, periodicamente, depois de cada manutenção. São especificados quais itens serão revisados e armazenados em cada uma das *baselines* planejadas.

A última etapa da identificação é descrita a maneira como os itens serão arquivados e recuperados do repositório.

4.4.4. Controle da Configuração

Dois controles básicos são instituídos no processo de gerência de configuração: controle de versão e controle de mudança.

4.4.4.1. Controle de Versão

Um item, ao ser desenvolvido, evolui até que atinja um estado que atenda os propósitos para o qual foi criado. Isso implica diversas alterações, gerando uma versão

(revisão) do item a cada estado. Para estabelecer o controle sobre as diversas versões, todas devem ser armazenadas e identificadas e para tal processo foi adotada a ferramenta *Subversion*.

O *Subversion* é uma ferramenta *open source* e suas principais funções pertinentes ao controle de versões podem ser resumidas em: recuperação de versões anteriores; auditoria das modificações realizadas: quem, quando, o quê; automatização do rastreamento de arquivos; estabelecimento de meios para obter a situação de um projeto em determinado ponto do tempo; permite o desenvolvimento paralelo; resolve conflitos entre desenvolvedores.

O *Subversion* armazena em seu repositório as modificações realizadas num arquivo ao longo do tempo; cada modificação é identificada por um número chamado de revisão. Toda revisão armazena as modificações realizadas, quem realizou essas modificações, quando foram realizadas, entre outras informações. Qualquer revisão poderá ser rastreada para fins de consulta, comparação ou união com outras revisões.

Conta, ainda, com um mecanismo capaz de controlar os acessos simultâneos e as modificações paralelas, garantindo a integridade das modificações e a atomicidade das operações.

A Figura 4.6 apresenta o fluxo sintético de gerência de configuração e a Figura 4.7 mostra o fluxo de operação do *Subversion*, ambas apresentarão de forma resumida o funcionamento do controle de versão.

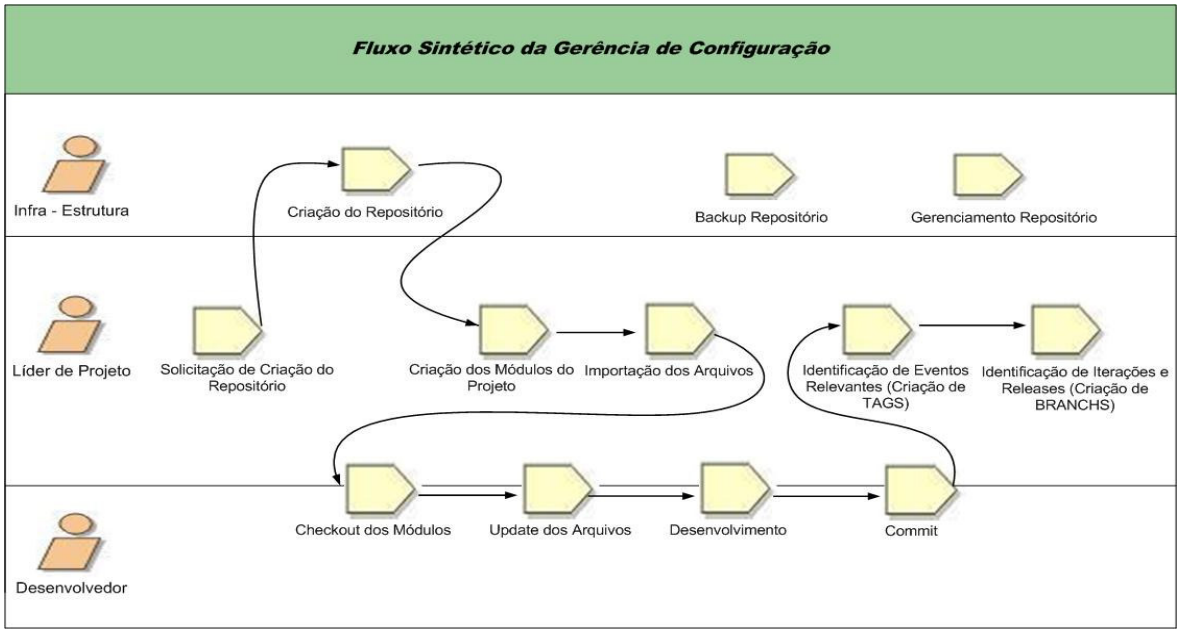


Figura 4.6 - Fluxo Sintético de Gerência de Configuração
Fonte: Elaborado pela autora

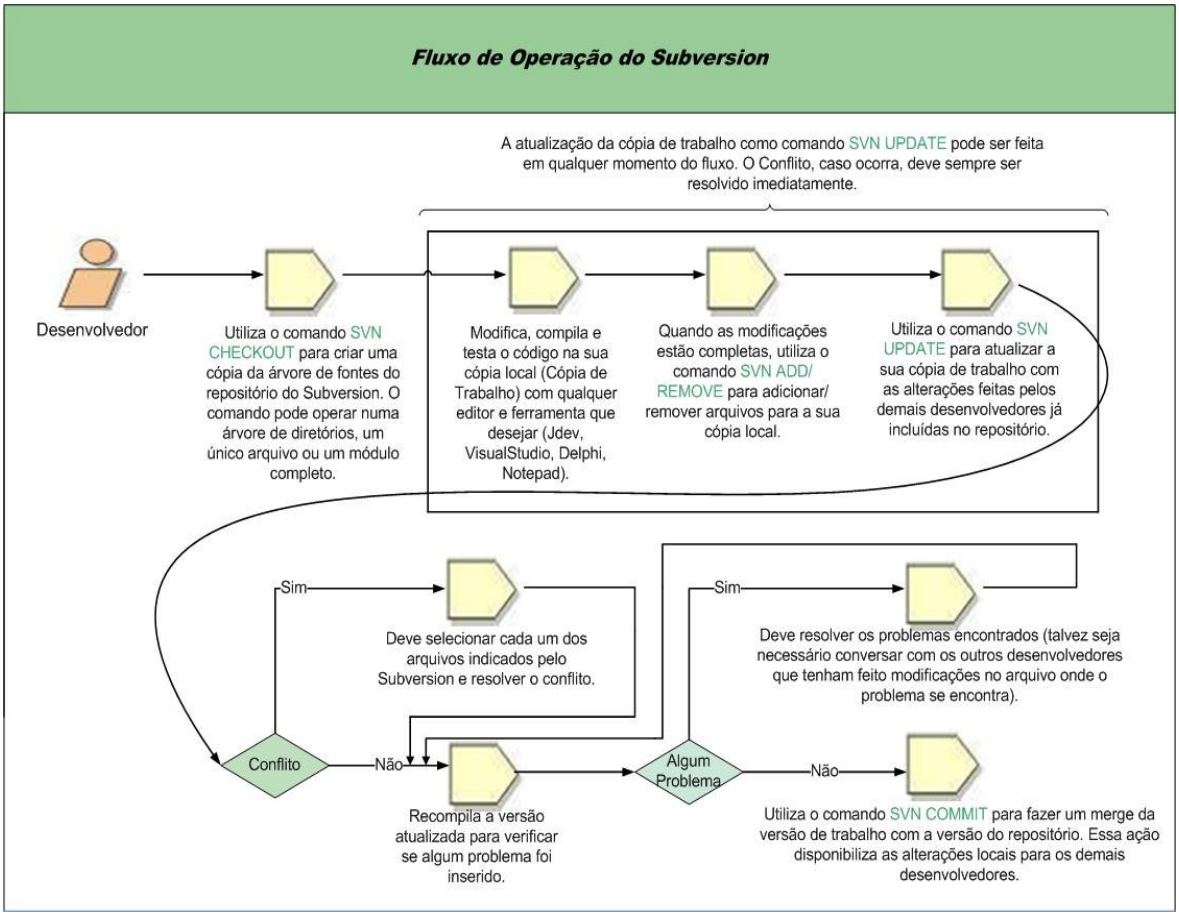


Figura 4.7 - Fluxo de Operação do *Subversion*
Fonte: Elaborado pela autora

Todas as atividades envolvidas no processo de controle de versão foram apresentadas aos colaboradores através dos treinamentos realizados.

4.4.4.2. Controle de Mudanças

Durante o processo de desenvolvimento, mudanças descontroladas podem levar rapidamente ao caos. Assim, foi instituído na organização um processo que combine procedimentos humanos e ferramentas automatizadas para proporcionar um mecanismo de controle de mudanças.

Esse processo deve ser implementado depois que uma *baseline* for estabelecida; antes disso, somente um controle de mudanças informal[11] precisa ser aplicado, ou seja, até que um item de configuração faça parte de uma *baseline*, o mesmo não precisará passar por controle de mudanças formal.

[11] O termo informal, nesse contexto, refere-se a qualquer mudança que não necessite da aprovação e procedimentos realizados pelo gerente de configuração da organização.

A Figura 4.8 apresenta o fluxo de controle de mudanças formal a ser aplicado para os itens que compõem uma *baseline*.

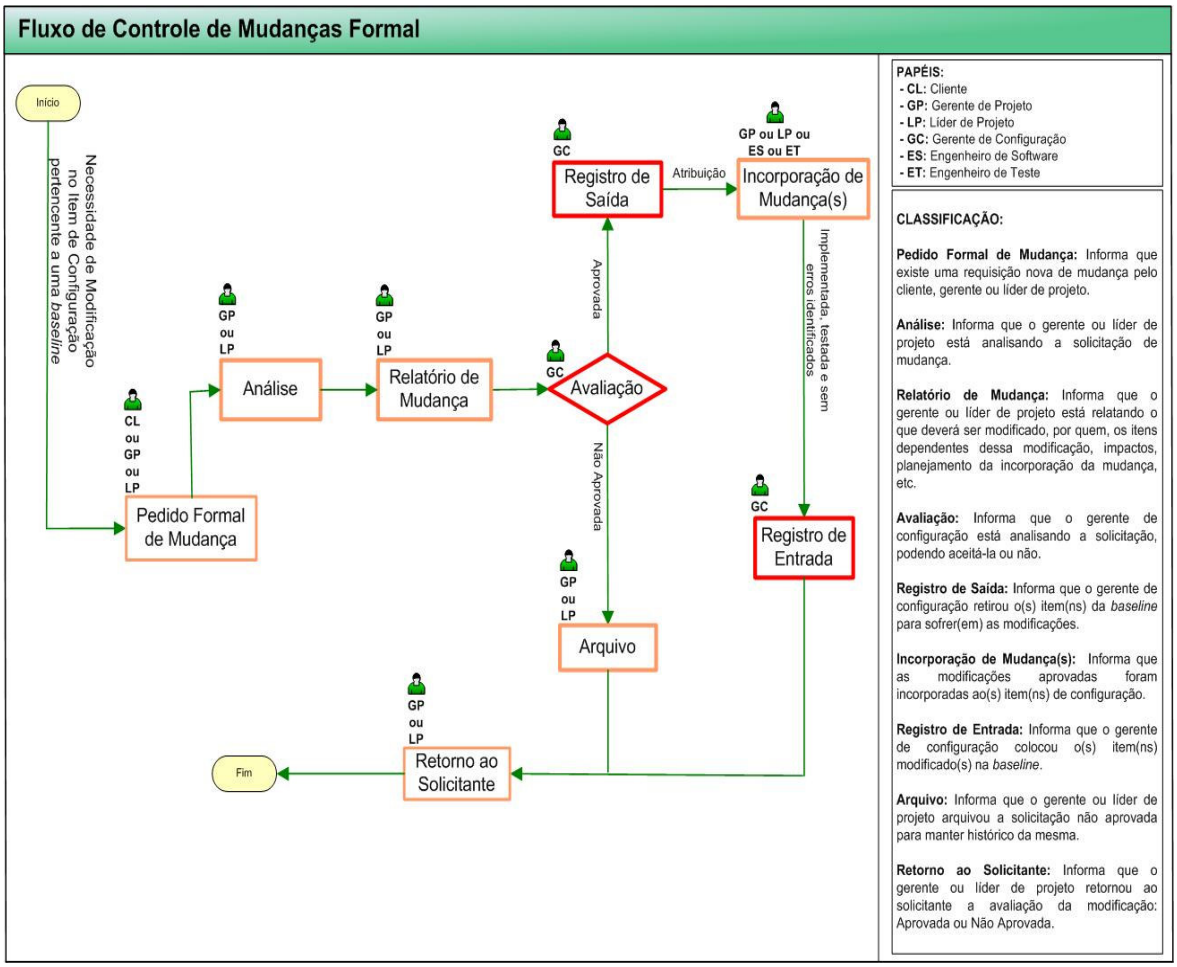


Figura 4.8 - Fluxo de Controle de Mudanças Formal
Fonte: Elaborado pela autora

Para o controle de mudanças dos itens que não compõem ainda uma *baseline*, ou seja, controle de mudanças informal decidiu-se utilizar a ferramenta *Mantis*. Com isso, toda e qualquer modificação realizada no desenvolvimento, antes do estabelecimento de uma *baseline,* poderia ser verificada e monitorada.

Houve uma adaptação dos fluxos do processo de teste da empresa para o controle de mudança informal. No fluxo de *bugs* são tratadas as modificações referentes a erros detectados (não conformidades) e no fluxo de requisitos, desenvolvimento de novos requisitos.

Na Figura 4.9 e na Figura 4.10 são apresentados o fluxo de *bugs* adaptado e o fluxo de requisitos adaptado, respectivamente.

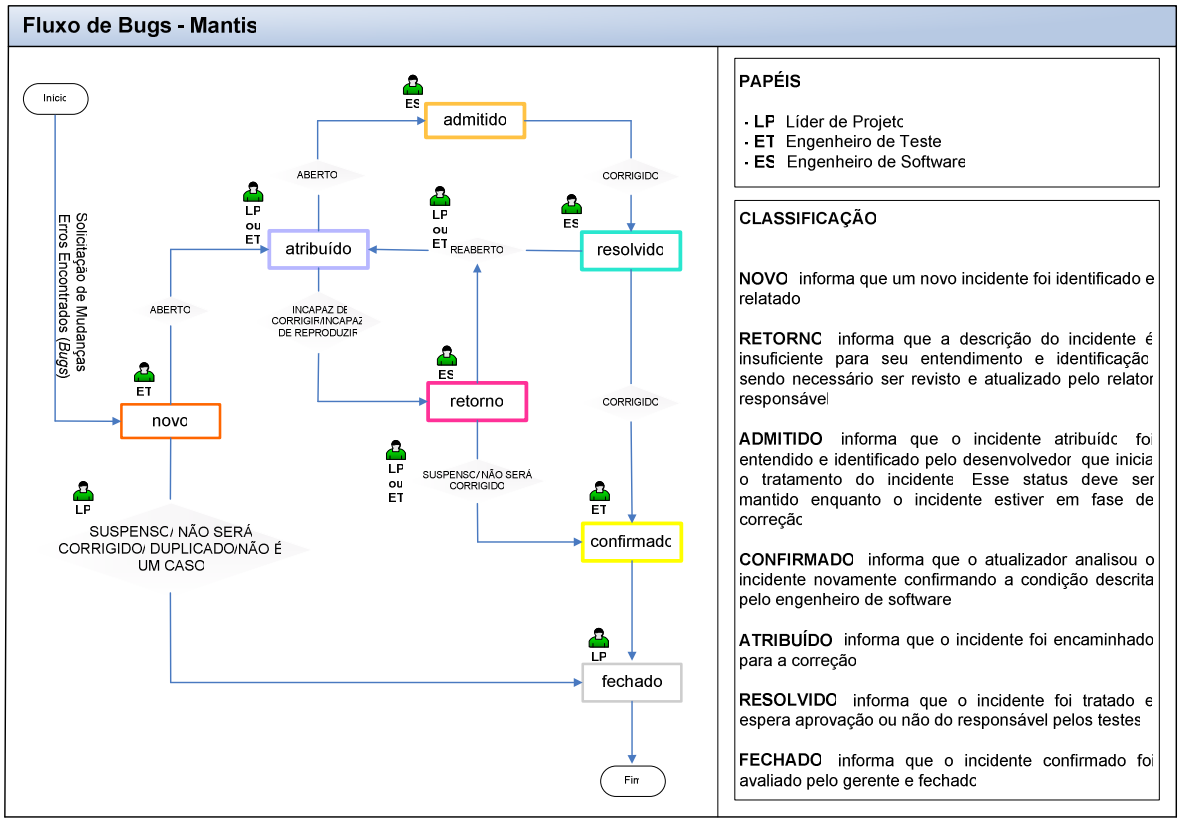


Figura 4.9 - Fluxo de *bugs* adotado para a Ferramenta *Mantis*
Fonte: Empresa Beta

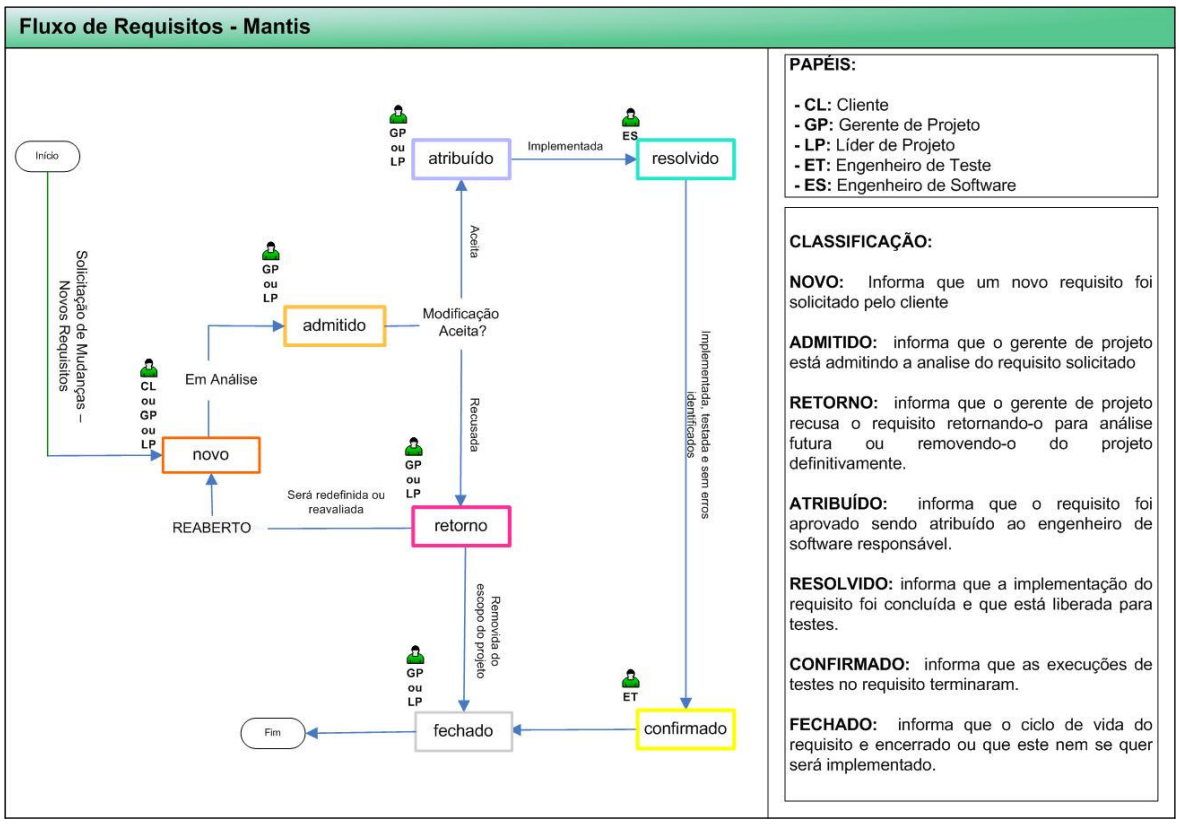


Figura 4.10 - Fluxo de requisitos adotado para a Ferramenta *Mantis*
Fonte: Empresa Beta

A estruturação do uso das ferramentas *Subversion* e *Mantis* foi definida visando facilitar e auxiliar o processo de contabilização da situação da configuração e, também, a auditoria da configuração.

Todas as atividades envolvidas no processo de controle de mudanças foram apresentadas aos colaboradores através dos treinamentos realizados.

4.4.5. Contabilização da Situação da Configuração

O objetivo dessa atividade é relatar a todas as pessoas envolvidas no desenvolvimento e na manutenção do produto as seguintes informações sobre as alterações na configuração do produto: *"O que aconteceu?"*; *"Quem o fez?"*; *"Quando aconteceu?"* e *"O que mais será afetado?"*.

As ferramentas *Subversion* e *Mantis* foram adotadas para registrar, armazenar e permitir a busca por essas informações. Na primeira são disponibilizadas a todos os colaboradores que possuem acesso ao projeto informações como: quem modificou um item, quando, a qual solicitação está vinculada a modificação, entre outros. Na segunda são encontradas informações de o que foi alterado, o porquê da alteração, o que mais será afetado, quantas modificações sofreu um determinado item de configuração entre outros; essas informações visualizadas através de permissões de acesso

O acesso rápido às informações sobre a gerência de configuração agiliza o processo de desenvolvimento e melhora a comunicação entre as pessoas, o que é uma maneira de eliminar muitos problemas relativos à modificação do mesmo item de informação com intenções diferentes e conflitantes.

4.4.6. Auditoria da Configuração

A auditoria de configuração é executada antes que uma *baseline* seja definida e tem como finalidade certificar que o produto está de acordo com os requisitos (de especificação e contratuais) e, também, garantir que está precisamente descrito em seus documentos técnicos.

As alterações também são auditadas para garantir a integridade do produto que está sendo produzido. Ou seja, ela auxilia a visualização da situação da configuração durante todo o desenvolvimento e revela se os requisitos iniciais estão sendo satisfeitos e se a passagem de uma configuração a outra não descaracteriza o produto.

A auditoria será realizada por um grupo independente do grupo que produziu a configuração. Tanto a atividade de verificação como de validação será realizada pela equipe de teste e a gerente de configuração da organização.

4.5. Avaliação da Implantação das Práticas de GCS

Após a implantação das práticas descritas acima, constatou-se fatores positivos e negativos de toda a execução deste projeto.

Como pontos positivos podem ser citados:

- Diminuição de perda de código-fonte, de requisitos já documentados e implementados, evitando o retrabalho;
- Aumento da produtividade;
- Aumento da memória organizacional da empresa, pois agora tudo era armazenado e controlado;
- Maior controle sobre o desenvolvimento, tornou-se possível identificar quais mudanças ocorreram, quem e quando as realizou;
- Existência de documentação sobre a evolução do sistema;
- Desenvolvimento dependente do processo e, não de pessoas;
- Diminuição dos custos do processo de desenvolvimento;
- Maior confiabilidade nos prazos estabelecidos;
- Melhor comunicação entre a equipe de desenvolvimento;
- Maior compatibilidade de versões.

Os pontos negativos encontrados foram:

- Resistência ao uso de novas ferramentas e procedimentos no processo de desenvolvimento;
- Dificuldade de conscientização de todos os colaboradores.

5. CONCLUSÕES

A preocupação com a melhoria do processo de desenvolvimento de software vem sendo impulsionada por exigências do mercado por mais qualidade e produtividade. Muitas empresas têm revisto seus processos e procurado se capacitar no mercado cada vez mais competitivo.

Os estudos realizados sobre Engenharia e Qualidade de Software, Melhoria do Processo de Software e Gerência de Configuração de Software revelaram que em processo de desenvolvimento de software é de suma importância o controle e gerenciamento das mudanças a que são submetidos os itens de configuração.

O elevado número de problemas encontrados na empresa como: perda de código-fonte, requisitos já documentados e requisitos implementados, falta de controle sobre o desenvolvimento, constantes retrabalhos; indicaram que a empresa de software descrita no estudo de caso necessitava de aplicar ao seu processo de desenvolvimento práticas de Gerência de Configuração de Software.

A definição das práticas a serem adotadas levou em consideração principalmente o perfil da empresa, ou seja, empresa de pequeno porte, jovem, de recursos moderados (humano, tempo e capital). Sendo assim, foram adotadas ferramentas *open source* para o suporte ao controle de versão e mudanças, atividades relacionadas à GCS foram atribuídas aos próprios líderes de projeto. O pouco tempo para a execução deste projeto também foi fator determinante para o pouco treinamento oferecido.

A implantação das práticas proporcionou melhorias no processo; permitiu um controle sobre o desenvolvimento; diminuição das perdas de código; uniformidade do trabalho; mudança na cultura organizacional; aumento da memória organizacional da empresa; maior organização dos processos e projetos; melhor adequação dos prazos estipulados para os projetos e melhoria nas relações entre os clientes e a empresa.

As principais dificuldades verificadas para a implantação das práticas de gerência de configuração foram: as deficiências na alocação de recursos para a execução deste projeto; a cultura organizacional, a falta de consenso em relação ao estabelecimento das melhores práticas a serem implantadas, falta de conhecimento mais amplo de gerência de configuração dos colaboradores da empresa.

Contudo, os treinamentos realizados para a implantação das práticas juntamente com a escolha dos projetos pilotos viabilizaram este trabalho. As práticas ajudaram

também, através dos seus planos, *templates*, processos, nova ferramenta adotada, a reduzir falhas na comunicação entre os colaboradores da empresa e aumentar a produtividades.

Percebeu-se que uma implantação pode utilizar um processo já existente adicionando-se apenas algumas atividades a esse processo, ou seja, implantação pode representar apenas adaptações ao invés de criações de processos que "comecem do zero".

As vantagens da implantação na empresa Beta foram muito superiores às desvantagens, o que permitiu a conclusão desse trabalho com êxito. Porém, não se pode acreditar que esse é o melhor processo a ser adotado, deve-se buscar melhorias contínuas a fim de se ter produtos e processos com a alta qualidade.

Toda modificação requer um período de adaptação, um treinamento adequado, comprometimento dos envolvidos; caso isso não exista pode ser definido o insucesso do projeto. Vale ressaltar que toda implantação, ao final, deve ser submetida a uma avaliação, pois assim se tem base para projetos futuros e se realiza um levantamento de melhorias a serem aplicadas.

As práticas de GCS mostraram-se um importante elemento para a melhoria da qualidade dos produtos, principalmente devido ao perfil da organização, que emprega profissionais que muitas vezes nunca tiveram uma experiência prática com desenvolvimento de software para o mercado. Os colaboradores da empresa se familiarizaram com os conceitos e atividades relacionadas à disciplina da engenharia de software – gerência de configuração – fundamental ao processo de desenvolvimento de software.

Como trabalho futuro, pretende-se implantar melhorias nas práticas já institucionalizadas de gerência de configuração e buscar continuamente qualidade no processo e nos produtos da empresa.

6. REFERENCIAL BIBLIOGRÁFICO

ABNT, **Gestão da Qualidade - Diretrizes para a Gestão da Configuração. NBR ISO 10007**. ABNT, 1996.

Aaen, I., Bottcher, P., Mathiassen, L., **The Software Factory: Contributions and Illusions**, In the Proceedings of the Twentieth Information Systems Research Seminar, Scandinavia, Oslo, 1997.

Asklund, U., Bendix, L. **A Study of Configuration Management in Open Source Software**. In: IEE Proceedings - Software, v. 149, pp. 40-46, 2002.

Babich, W.A. **Software Configuration Management - Coordination For Team Productivity.** Addison-Wesley, 1986.

Bersoff, E.H. **Elements Of Software Configuration Management**. IEEE Transactions on Software Engineering, v. 10, n. 1, p. 79-87, 1984.

Bryan, W. L., Siegel, S. G., Whiteleather, G. L. **Auditing Through The Software Life Cycle: A Primer**. Computer, v. 15, n. 3, p. 57-67, 1982.

Capretz, M.A.M. **A Software Maintenance Method Based On The Software Configuration Management Discipline.** Tese de Doutorado. University of Durham, Inglaterra, 1992.

Castor, E. M. **Fábrica de Software: Passado, Presente e Futuro**. Pós-Graduação Lato Sensu em Tecnologia da Informação - UNIBRATEC - União dos Institutos Brasileiros de Tecnologia, 2004.

Christensen, M. J., Thayer, R. H. **The Project Manager's Guide to Software Engineering's Best Practices**. 1 ed., IEEE Computer Society Press and John Wiley & Sons, 2002.

Computerworld (2006), **Crescimento das Fábricas de Software no Brasil.** Disponível em, http://computerworld.uol.com.br/mercado/2005/09/13/idgnoticia.2006-05-15.329843166/IDGNoticia_view. Acessado no dia 20 de maio de 2006.

Dias, A.F. **O que é Gerência de Configuração?,** Disponível em http://www.pronus.eng.br/artigos_tutoriais/gerencia_configuracao/gerencia_configuracao.php?pagNum=0. Acessado dia 18 de março de 2006.

Estublier, J. **Software Configuration Management: a Roadmap**. In: Proceedings of 22nd International Conference on Software Engineering, The Future of Software Engineering, pp. 279-289, Limerick, Ireland, 2000.

Estublier, J., Leblang, D., Clemm G., *et al*. **Impact of the Research Community On the Field of Software Configuration Management**. In: Software Engineering Notes (ACM), v. 27, pp. 31-39, Orlando, Florida, 2002.

Gil, A. C. **Como Elaborar Projetos de Pesquisa**. 3. ed. São Paulo: Atlas, 1991. 159 p.

Hass, A. M. J. **Configuration Management Principles and Practices**, Boston, MA, Pearson Education, Inc, 2003.

Herbsleb, James D.; Mockus, Audris; Finholt, Thomas A.;Grinter, Rebecca E. **Distance, Dependencies, and Delay in a Global Collaboration**. Conference On Computer Supported Cooperative Work, Philadelphia, Pennsylvania. Proceedings... New York: ACM Press, 2000. p. 319-328.

IEEE. **IEEE Standards Collection - Software Engineering**. 1993.

IEEE, Std 610.12 - **IEEE Standard Glossary of Software Engineering Terminology**. 1990.

IEEE, Std 1042 - **IEEE Guide to Software Configuration Management**. 1987.

IEEE. **SWEBOK - Guide to the software engineering body of knowledge**: trial version (version 1.00) / executive editors, Alain Abran, James W. Moore; editors, Pierre Bourque, Robert Dupuis, Leonard L. Tripp. 2001.

ISO. **ISO - International organization for Standardization / International Electrotechanical Commission. Information Technology - Software Product evaluation – Quality characteristics and guidelines for use**. - ISO/IEC 9126, Genève 1991.

ISO. **ISO 10007, Quality Management - Guidelines for Configuration Management**. 1995.

Jung, C. **Metodologia Para Pesquisa & Desenvolvimento – Aplicada a Novas Tecnologias, Produtos e Processos**. Rio de Janeiro: Axcel Books, 2004. 162 p.

Leon, A. **A Guide to Software Configuration Management**, Norwood, MA, Artech House Publishers, 2000.

Paulk, M.C.; Weber, C.V.; Curtis, B.; Chrissis, M.B.; *et al.* **The Capability Maturity Model: Guidelines for Improving the Software Process**. Estados Unidos: Addison-Wesley, 1995.

Pessoa, M. S. de P. **Introdução ao CMM – Modelo de Maturidade da Capacidade de Processo de Software** – Lavras: UFLA/FAEPE, 2003.

Pressman, Roger S. **Engenharia de Software**. 5. ed. Rio de Janeiro: McGraw-Hill, 2002. 843p.

SEI, Software Engineering Institute. **Capability Maturity Model® Integration (CMMI^{SM})**, Version 1.1. CMMI^{SM} for Software Engineering (CMMI-SW, V1.1), Staged Representation. Pittsburgh, PA, EUA: Carnegie Mellon University 2002.

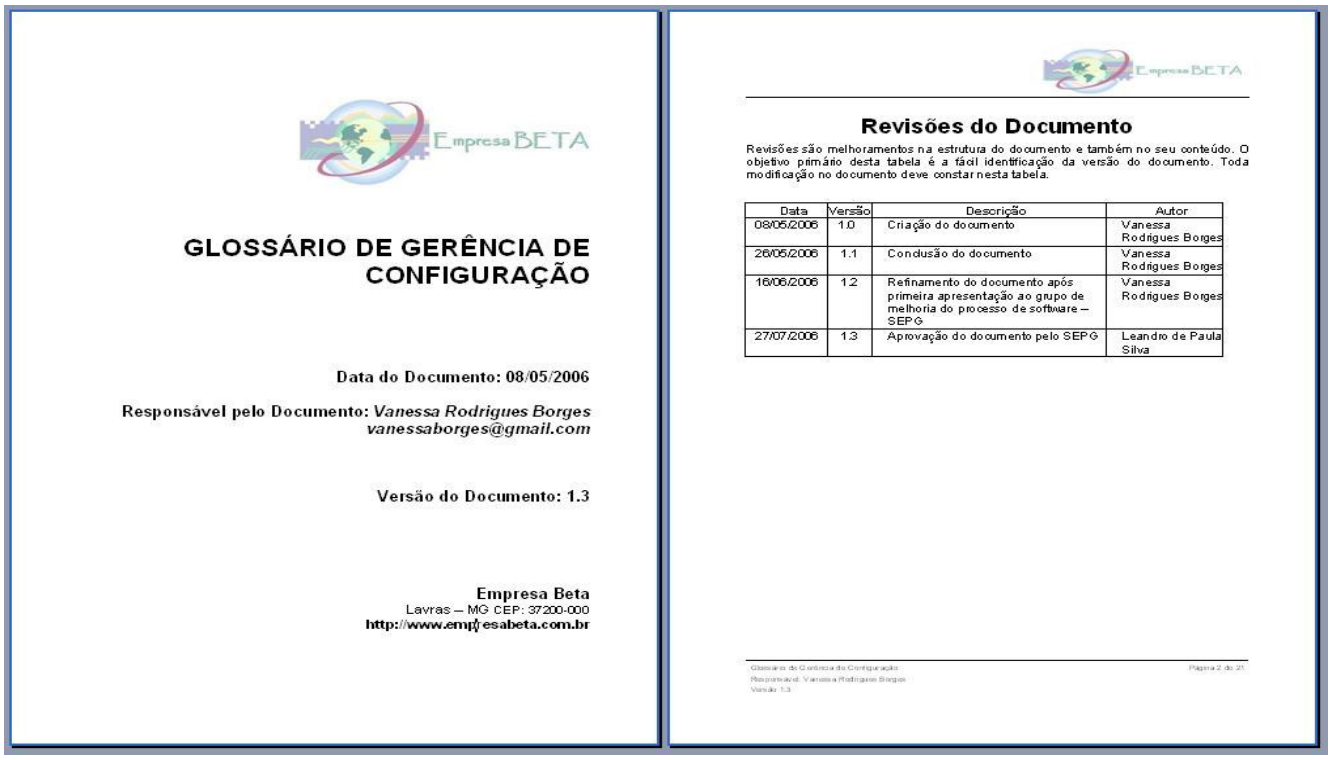
Siy, H, P., Mockus, A., Herbsleb, J, D., Krishnan, M., Tucker, G, T., **Making The Software Factory Work: Lessons from a decade of experience**, In the Seventh International Symposium on Software Metrics, London, England, 2001.

Sommerville, I. **Engenharia de Software**. 6 ª ed. São Paulo, Addison Wesley, 2003.

Victorelli, E.Z. **Controle de Versões e Configurações em Ambientes de Desenvolvimento de Software.** Dissertação de Mestrado. DCC/IMECC/UNICAMP, 1990.

Villas Boas, A. L. C. **Gestão de Configuração para Teste de Software**, Dissertação de mestrado apresentada na Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas. Campinas, SP: [s.n.], 2003.

Apêndice A – Glossário de Gerência de Configuração



GLOSSÁRIO DE GERÊNCIA DE CONFIGURAÇÃO

Data do Documento: 08/05/2006

Responsável pelo Documento: *Vanessa Rodrigues Borges*
vanessaborges@gmail.com

Versão do Documento: 1.3

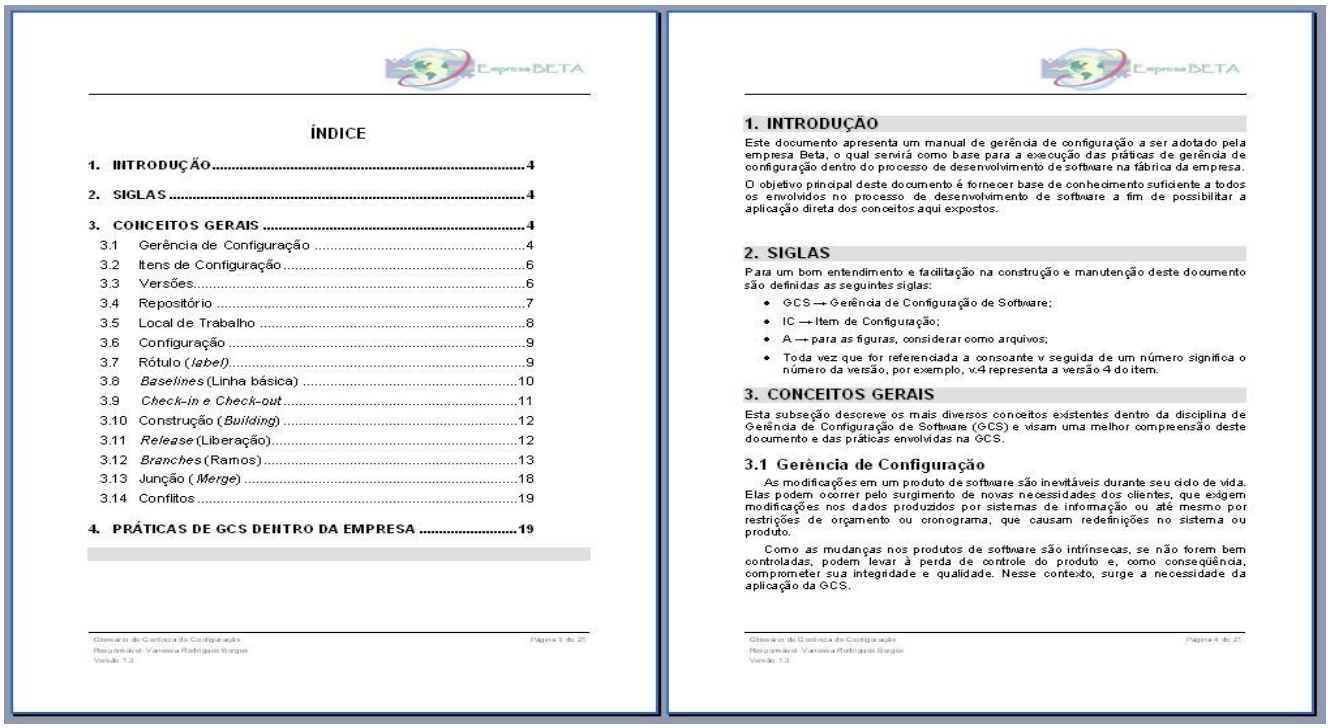
Empresa Beta
Lavras – MG CEP: 37200-000
http://www.empresabeta.com.br

Revisões do Documento

Revisões são melhoramentos na estrutura do documento e também no seu conteúdo. O objetivo primário desta tabela é a fácil identificação da versão do documento. Toda modificação no documento deve constar nesta tabela.

Data	Versão	Descrição	Autor
08/05/2006	1.0	Criação do documento	Vanessa Rodrigues Borges
26/05/2006	1.1	Conclusão do documento	Vanessa Rodrigues Borges
16/06/2006	1.2	Refinamento do documento após primeira apresentação ao grupo de melhoria do processo de software – SEPG	Vanessa Rodrigues Borges
27/07/2006	1.3	Aprovação do documento pelo SEPG	Leandro de Paula Silva

Figura A.1 - Páginas 1 e 2 do Glossário de Gerência de Configuração
Fonte: Elaborado pela autora



ÍNDICE

1. INTRODUÇÃO 4
2. SIGLAS 4
3. CONCEITOS GERAIS 4
3.1 Gerência de Configuração 4
3.2 Itens de Configuração 6
3.3 Versões 6
3.4 Repositório 7
3.5 Local de Trabalho 8
3.6 Configuração 9
3.7 Rótulo (*label*) 9
3.8 *Baselines* (Linha básica) 10
3.9 *Check-in* e *Check-out* 11
3.10 Construção (*Building*) 12
3.11 *Release* (Liberação) 12
3.12 *Branches* (Ramos) 13
3.13 Junção (*Merge*) 18
3.14 Conflitos 19
4. PRÁTICAS DE GCS DENTRO DA EMPRESA 19

1. INTRODUÇÃO

Este documento apresenta um manual de gerência de configuração a ser adotado pela empresa Beta, o qual servirá como base para a execução das práticas de gerência de configuração dentro do processo de desenvolvimento de software na fábrica da empresa.

O objetivo principal deste documento é fornecer base de conhecimento suficiente a todos os envolvidos no processo de desenvolvimento de software a fim de possibilitar a aplicação direta dos conceitos aqui expostos.

2. SIGLAS

Para um bom entendimento e facilitação na construção e manutenção deste documento são definidas as seguintes siglas:

- GCS → Gerência de Configuração de Software;
- IC → Item de Configuração;
- A → para as figuras, considerar como arquivos;
- Toda vez que for referenciada a consoante v seguida de um número significa o número da versão, por exemplo, v.4 representa a versão 4 do item.

3. CONCEITOS GERAIS

Esta subseção descreve os mais diversos conceitos existentes dentro da disciplina de Gerência de Configuração de Software (GCS) e visam uma melhor compreensão deste documento e das práticas envolvidas na GCS.

3.1 Gerência de Configuração

As modificações em um produto de software são inevitáveis durante seu ciclo de vida. Elas podem ocorrer pelo surgimento de novas necessidades dos clientes, que exigem modificações nos dados produzidos por sistemas de informação ou até mesmo por restrições de orçamento ou cronograma, que causam redefinições no sistema ou produto.

Como as mudanças nos produtos de software são intrínsecas, se não forem bem controladas, podem levar à perda de controle do produto e, como conseqüência, comprometer sua integridade e qualidade. Nesse contexto, surge a necessidade da aplicação da GCS.

Figura A.2 - Páginas 3 e 4 do Glossário de Gerência de Configuração
Fonte: Elaborado pela autora

A Gerência de Configuração de Software pode ser definida como a arte de identificar, organizar e controlar modificações no software que está sendo construído por uma equipe de programação. O objetivo é maximizar a produtividade e minimizar os erros.

Em outras palavras, a GCS é uma disciplina, dentro da Engenharia de Software, que gerencia e controla a evolução de um software através, basicamente, de controle formal de versão e solicitação de mudanças.

Pensando em contra-definição, pode-se dizer que a GCS:

- Não é backup;
- Não é somente controle de versão;
- Não é simples;
- Não é impossível;
- Não é opcional ou modismo;
- Não evita que ocorram modificações;
- Não é somente para sistemas grandes e complexos e/ou para equipes geograficamente distribuídas.

O foco de atuação da GCS é:

- Controle sobre a evolução de produtos de trabalho, tanto produtos de trabalho do projeto como dos processos;
- Fonte de informações para outros processos, por exemplo, dados históricos para medição e análise;
- Garantia de que foi feito o que deveria ter sido feito.

A falta de GCS pode ocasionar problemas como:

- Perda de código-fonte, perda no sentido de desaparecimento e não sobreposição;
- Bibliotecas inesperadamente não funcionam mais;
- Impossibilidade de determinar o que aconteceu com um programa, ou parte dele;
- Impossibilidade de determinar quem, por que e quando foram efetuadas modificações;
- Requisitos já documentados desaparecem;
- Requisitos implementados desaparecem do código;
- O programa em execução e o seu código fonte estão em diferentes versões;

- Entre outros.

Os benefícios gerados com o uso da GCS podem atingir vários níveis dentro da organização como:

- Aumento da "memória" organizacional da empresa;
- Desenvolvimento dependente do processo e não das pessoas, ou seja, transforma-se em conhecimento explícito e não só tácito;
- Controle sobre o desenvolvimento;
- Rastreabilidade entre os diferentes níveis de abstração de um mesmo componente (requisito, análise, projeto, código, programa executável);
- Documentação sobre a evolução do sistema.

Esses entre outros motivos fazem da GCS a espinha dorsal do desenvolvimento de software.

3.2 Itens de Configuração

Durante o processo de desenvolvimento de um produto, é produzida uma grande quantidade de itens de informação que podem ser alterados durante o processo. Para que não haja inconsistência nos itens de informação importantes para o projeto, a criação e alterações desses itens devem ser acompanhadas e controladas pelo gerente do projeto.

Item de configuração é o nome dado ao item de informação produzido durante o processo de desenvolvimento do produto para o qual é importante que seja realizado o controle das alterações. Um conjunto de itens de configuração compõe uma configuração do produto.

3.3 Versões

Versões são instâncias de um mesmo IC que diferem entre em algo, também podem ser chamadas de revisões apesar de ser pouco usual.

Dentro do conceito de versões vale ressaltar o conceito de variantes, que são versões funcionalmente equivalentes, mas projetadas para ambientes de hardware ou software distintos, por exemplo, uma aplicação que "roda" no Firefox e outra no Internet Explorer.

A figura abaixo ilustra um pouco este conceito:

Figura A.3 - Páginas 5 e 6 do Glossário de Gerência de Configuração
Fonte: Elaborado pela autora

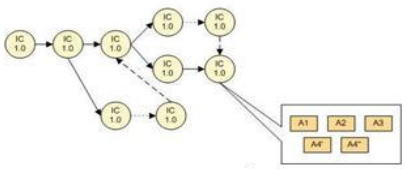


Figura 1: ICs, versões e variantes
Fonte: Adaptado de Pressman, 1997.

3.4 Repositório

Repositório é o local onde os itens de configuração são armazenados, existindo poucos por projeto, normalmente somente um.

À medida que o projeto evolui, o repositório passa a guardar múltiplas versões dos arquivos que compõem o projeto.

Existem tipos diferentes de armazenamento num repositório tais como: versionamento completo ou versionamento de diferenças (delta).

O versionamento completo como o próprio nome diz, qualquer modificação sofrida por um IC é inteiramente armazenada, ou seja, se um artefato for modificado por alguém, o repositório armazena o artefato inteiro que foi alterado. Esse tipo de armazenamento demanda grande espaço em disco, porém permite uma rápida recuperação dos ICs.

Já o versionamento de diferenças (delta), também é intuitivo e armazena apenas as modificações sofridas pelo IC, ou seja, não salva tudo novamente, apenas o que foi alterado. Esse tipo de armazenamento reduz o espaço requerido em disco, porém pode demandar grande carga de processamento para recuperar ICs.

As formas de armazenamento podem ser vistas na figura abaixo:

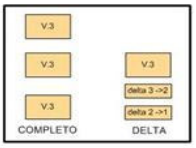


Figura 2: Tipo de armazenamento completo e por delta do repositório
Fonte: Elaborado por Murta, 2006.

Também existem diferentes mecanismos de controle de concorrência são eles: pessimista, otimista ou misto.

No modo de controle de concorrência pessimista somente um desenvolvedor pode modificar um IC em um dado momento. Não existe neste mecanismo custo de junção de trabalho, uma vez que cada desenvolvedor bloqueia o arquivo para que outros não o modifiquem, com isso inexiste paralelismo de desenvolvimento.

No modo otimista, vários desenvolvedores podem modificar um mesmo IC ao mesmo tempo, com isso há um alto custo de junção de trabalho no caso de ICs complexos (por exemplo: IC binário).

No modo de controle de concorrência misto, que também é chamado de controle de concorrência otimista com notificação, é permitido que qualquer desenvolvedor saiba quem mais está modificando o IC. Possui um bom custo/benefício entre o controle otimista e pessimista.

3.5 Local de Trabalho

Os desenvolvedores não trabalham diretamente nos arquivos do repositório. Cada desenvolvedor possui um local de trabalho que espelha os arquivos do repositório para trabalhar com liberdade enquanto termina suas tarefas, isolado dos outros desenvolvedores. Também chamado de espaço de trabalho, cópia de trabalho, caixa de areia ou *sandbox*.

Nele o usuário armazena um momento específico do projeto, trabalha com autonomia e depois, se necessárias e aprovadas, envia suas modificações, inclusões ou remoções para o repositório.

Figura A.4 - Páginas 7 e 8 do Glossário de Gerência de Configuração
Fonte: Elaborado pela autora

A imagem abaixo dá uma visão do que representaria o repositório juntamente com o local de trabalho e os ICs:

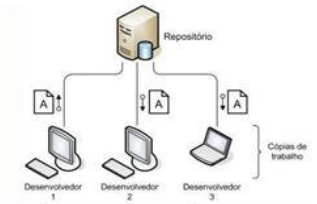


Figura 3: Relação entre repositório e as cópias de trabalho dos arquivos
Fonte: Elaborado por Dias, 2006.

3.6 Configuração

Ao conjunto de versões de ICs, onde existe somente uma versão selecionada para cada IC do conjunto dá-se o nome de configuração.

Outra forma de se definir configuração é que se trata de características físicas e funcionais de um produto, conforme definidas na documentação técnica obtidas no produto.

Uma configuração pode ser vista como um IC composto de outros ICs, por exemplo, configuração do sistema, do processo, do módulo *X*, do código fonte, etc.

3.7 Rótulo (*label*)

Rótulo é o mecanismo usado para identificar uma configuração, sendo assim, as diversas versões de ICs marcados com um rótulo constituem uma configuração.

A criação de rótulos permite identificar níveis de qualidade dos ICs.

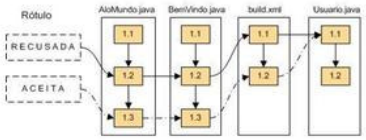


Figura 4: Exemplo de rótulos
Fonte: Elaborado por Murta, 2006.

O rótulo também é conhecido como etiqueta ou *tag*.

3.8 *Baselines* (Linha básica)

Para que cada item de configuração possa ser efetivamente gerenciado é necessário o estabelecimento de pontos bem definidos dentro do processo de desenvolvimento: as *baselines*. Esses pontos podem ocorrer ao final de cada uma das fases do processo de desenvolvimento ou de algum outro modo definido pela gerência.

As *baselines* além de linhas básicas também são conhecidas como linhas de base, linhas de referência, configuração-base.

Nos pontos estabelecidos pelas *baselines*, os itens de configuração devem ser identificados, analisados, corrigidos, aprovados e armazenados no repositório dos itens de configuração. Esses itens só poderão ser alterados após uma solicitação de alteração formalmente aprovada pelo gerente de configuração ou gerente do projeto. Essa é uma forma de controlar a situação de cada um dos itens de configuração, evitando assim inconsistências.

Podem ser estabelecidas ao final de cada fase do desenvolvimento: análise (*functional*), projeto (*allocated*) e implementação (*product*).

Vale ressaltar neste contexto de *baseline* a diferença entre a mesma e versão. Genericamente: "Um sistema *S* é composto pelos arquivos *X*, *Y* e *Z*". Concretamente: "A *baseline* 5 do sistema *S* é composta pela versão *2* do arquivo *X*, versão *4* do arquivo *Y* e versão 6 do arquivo *Z*.".

Figura A.5 - Páginas 9 e 10 do Glossário de Gerência de Configuração
Fonte: Elaborado pela autora

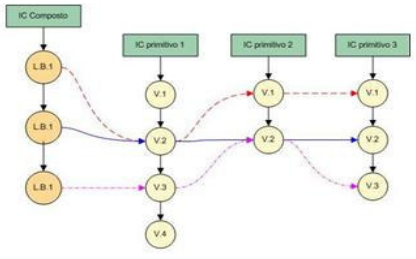


Figura 5: Exemplo de *baselines*
Fonte: Elaborado por Murta, 2006.

3.9 *Check-in* e *Check-out*

O método utilizado para trabalhar com itens de configuração que já estão no repositório é chamado de *check-in*/*check-out*, ou seja, conferência na entrada e conferência na saída.

Quando for desejada uma alteração em algum item de configuração do repositório, uma cópia do item a ser alterado será colocada em no local de trabalho do solicitante (*check-out*). Dentro de sua área o solicitante tem total liberdade de trabalho.

Após terminar as alterações no item de configuração, ele será revisado e recolocado no repositório (*check-in*). Nesse momento, uma nova *baseline* deverá ser traçada, de modo que uma nova configuração do item alterado seja formada e congelado no repositório. Depois do congelamento, o acesso será liberado para que outros solicitantes também executem alterações neste item de configuração.

Sendo assim podemos definir:

Check-out é o processo de requisição de modificações, aprovação e cópia de um item de configuração do repositório.

Check-in é o processo de revisão, aprovação e cópia de um item de configuração para o repositório.

A figura abaixo ilustra o processo de atualização de uma *baseline* descrita acima pelos métodos de *check-in*/*check-out*:

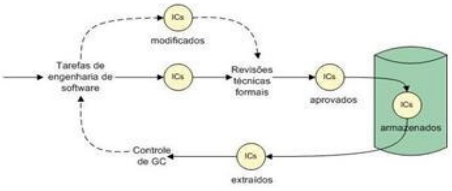


Figura 6: Processo de atualização de *baselines*
Fonte: Pressman, 1997.

3.10 Construção (*Building*)

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

Por exemplo: *makefile*, *build.xml*

Os arquivos de comandos também devem ser considerados itens de configuração.

3.11 *Release* (Liberação)

Uma *release* pode ser definida como uma versão disponibilizada para um propósito específico. Ou também pode ser entendida como uma notificação formal e distribuição de uma versão aprovada.

Entretanto se deve ter claro que:

- Toda *release* é uma versão;

Figura A.6 - Páginas 11 e 12 do Glossário de Gerência de Configuração
Fonte: Elaborado pela autora

- Mas nem toda versão é uma *release*.

Em alguns casos *releases* podem ser desenvolvidas em paralelo (*time to market*).

Exemplos de *releases*:

- *Release* para testes de sistema;
- *Release* para homologação;
- *Release* para entrega ao diente.

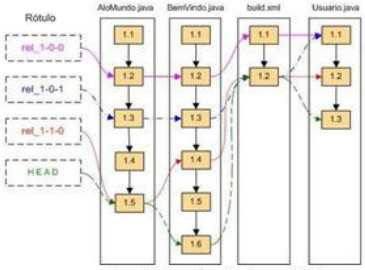


Figura 7: Exemplos de *releases* definidas.

Fonte: Adaptado de Murta, 2006

3.12 *Branches* (Ramos)

Branches são versões que não seguem a linha principal de desenvolvimento. Em outras palavras, é o termo usado para descrever o processo de divisão dos arquivos de um projeto em linhas de desenvolvimento independentes.

São responsáveis por fornecerem isolamento para o processo de desenvolvimento. Os ramos são migrados à linha principal de desenvolvimento e essa migração pode ser complicada no caso de isolamento longo.

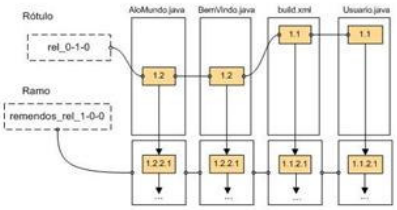


Figura 8: Exemplo de ramo

Fonte: Elaborado por Murta, 2006.

Existem diferentes estratégias de criação de *branches*: estratégias de manutenção, de organização e de verificação e essas podem ser combinadas.

Para a estratégia de manutenção existem a caótica, em série e em cascata.

A manutenção caótica, praticada por empresas imaturas, apresenta como ramo principal a evolução e correções; impossibilitando separa o que é manutenção corretiva de evolutiva.

Figura 9: Estratégia de manutenção caótica

Fonte: Elaborado por Murta, 2006.

Figura A.7 - Páginas 13 e 14 do Glossário de Gerência de Configuração
Fonte: Elaborado pela autora

A manutenção em série, vista como a mais simples, define a evolução no ramo principal e nos ramos auxiliares as correções; dificulta-se a manutenção de várias liberações em paralelo.

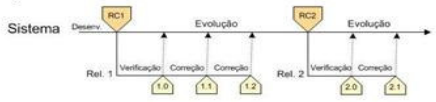


Figura 10: Estratégia de manutenção em série

Fonte: Elaborado por Murta, 2006.

Já a em cascata, mais complicada, define as correções no ramo principal e a evolução no ramo auxiliar; dificulta-se a coordenação de *checkouts* evolutivos em andamento.

Para a estratégia de organização encontram-se a organização por desenvolvedores, por subprojetos, por requisições e por customizações.

A organização por desenvolvedor define o ramo principal com a integração e cada desenvolvedor tendo um ramo no seu espaço de trabalho seriam os ramos auxiliares, assim; permite que cada desenvolvedor realize *checkins* intermediários sem que os outros desenvolvedores sejam afetados.

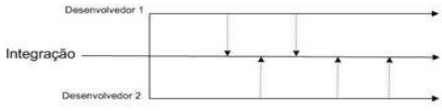


Figura 11: Estratégia de organização por desenvolvedores

Fonte: Elaborado por Murta, 2006.

Já a organização por subprojetos define a integração para o ramo principal e os projetos para os ramos auxiliares; fornece baixo isolamento entre os membros de um mesmo subprojeto, porém isola a equipe do subprojeto das demais.

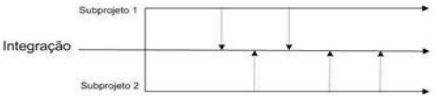


Figura 12: Estratégia de organização por subprojetos

Fonte: Elaborado por Murta, 2006.

A organização por requisitos, por sua vez, define no ramo principal a integração e nos ramos auxiliares as requisições; permite que cada uma das requisições seja identificada e torna possível a remoção de uma requisição do produto.

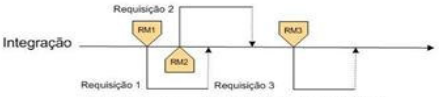


Figura 13: Estratégia de organização por requisições

Fonte: Elaborado por Murta, 2006.

A por customização define também que no ramo principal esteja contida a integração, porém nos ramos auxiliares estejam as customizações; permite habilitar e desabilitar funcionalidades no software em funcionamento, ou seja, seleção de variabilidade em tempo de construção usualmente é mais apropriado.

Figura A.8 - Páginas 15 e 16 do Glossário de Configuração
Fonte: Elaborado pela autora

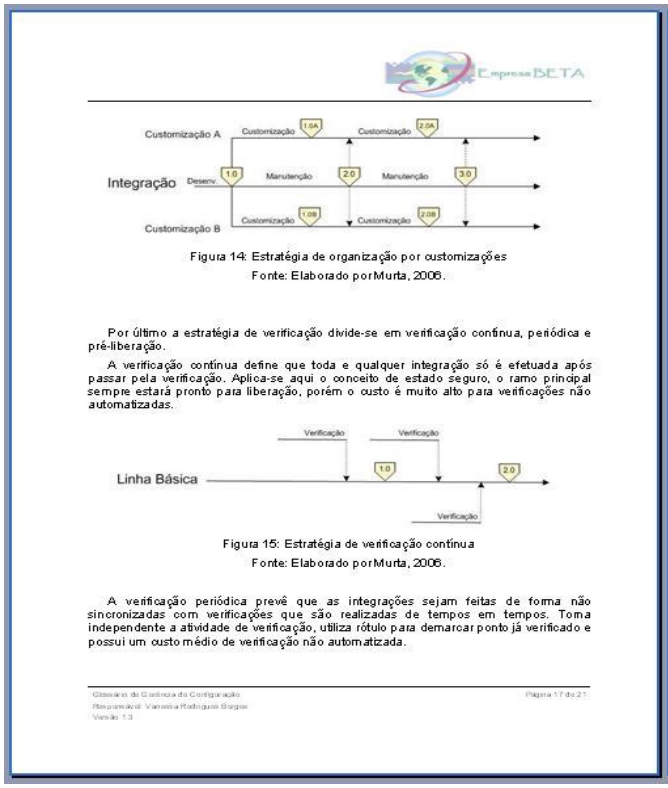


Figura 14: Estratégia de organização por customizações
Fonte: Elaborado por Murta, 2006.

Por último a estratégia de verificação divide-se em verificação contínua, periódica e pré-liberação.

A verificação contínua define que toda e qualquer integração só é efetuada após passar pela verificação. Aplica-se aqui o conceito de estado seguro, o ramo principal sempre estará pronto para liberação, porém o custo é muito alto para verificações não automatizadas.

Figura 15: Estratégia de verificação contínua
Fonte: Elaborado por Murta, 2006.

A verificação periódica prevê que as integrações sejam feitas de forma não sincronizadas com verificações que são realizadas de tempos em tempos. Torna independente a atividade de verificação, utiliza rótulo para demarcar ponto já verificado e possui um custo médio de verificação não automatizada.

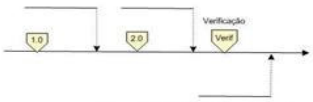


Figura 16: Estratégia de verificação periódica
Fonte: Elaborado por Murta, 2006.

Verificação pré-liberação é executa somente antes da liberação. Congela-se o ramo principal e aceitam-se somente correções neste momento de congelamento, possui um baixo de verificação não automatizada.

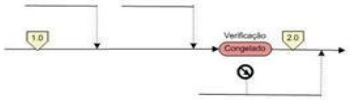


Figura 17: Estratégia de verificação pré-liberação
Fonte: Elaborado por Murta, 2006.

3.13 Junção (*Merge*)

É o processo de junção de espaços de trabalho com o repositório e entre os ramos principal e auxiliares. O processo de migração entre espaço de trabalho e repositório é quando é dado *checkin* do arquivo e entre os ramos quando é realizado o *merge* entre linha principal e linha auxiliar de desenvolvimento.

Figura A.9 - Páginas 17 e 18 do Glossário de Configuração
Fonte: Elaborado pela autora

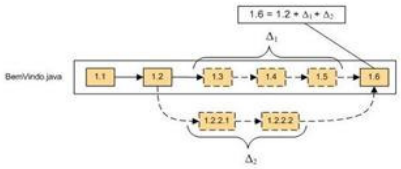


Figura 18: Exemplo de junção
Fonte: Elaborado por Murta, 2006.

Existem ferramentas que dão suporte a essa atividade de se juntar cada artefato do ramo e, para essa junção, são levados em consideração todas as modificações desde o ancestral em comum.

3.14 Conflitos

É a situação onde não é possível executar a junção de forma automática. Os tipos de conflitos são: conflito físico, lógico e semântico.

O conflito físico é o que acontece em linha do arquivo, para esse tipo de conflito as algumas ferramentas de GCS detectam tal conflito. Por exemplo: modificação em paralelo de uma mesma linha, seja inserindo, removendo ou alterando e alteração na mesma região do arquivo.

O lógico ocorre na sintaxe do arquivo e o semântico no conteúdo do arquivo, ambos não são detectados por ferramentas.

4. PRÁTICAS DE GCS DENTRO DA EMPRESA

O propósito do *Plano de Gerência de Configuração* é definir formalmente as ferramentas a serem utilizadas e os procedimentos a serem seguidos durante a evolução do software em desenvolvimento.

O *Plano de Ambiente* define os recursos de *hardware* e *software* a serem utilizados por um projeto. Por exemplo, todo projeto desenvolvido (ou a ser desenvolvido) possuirá um documento ao qual serão encontradas informação como: quais foram as configurações necessárias para as máquinas da equipe do projeto (processador,

memória), softwares e suas versões utilizados pela equipe de desenvolvimento (sistema operacional, servidor web, sistema gerenciador de banco de dados), políticas de backup.

Nessa atividade são definidas as seguintes funções: seleção dos itens de configuração; determinação da estrutura da configuração; documentação dos itens de configuração; sistema de numeração, estabelecimento de *baselines*.

Dois controles básicos são instituídos no processo de gerência de configuração: controle de versão e controle de mudança. Foi instituído na organização um processo que combine procedimentos humanos e ferramentas automatizadas para proporcionar um mecanismo de controle da configuração.

O objetivo dessa atividade é relatar a todas as pessoas envolvidas no desenvolvimento e na manutenção do produto as seguintes informações sobre as alterações na configuração do produto: *"O que aconteceu?"*; *"Quem o fez?"*; *"Quando aconteceu?"* e *"O que mais será afetado?"*.

A auditoria de configuração é executada antes que uma *baseline* seja definida e tem como finalidade certificar que o produto está de acordo com os requisitos (de especificação e contratuais) e, também, garantir que está precisamente descrito em seus documentos técnicos.

Figura A.10 - Páginas 19 e 20 do Glossário de Configuração
Fonte: Elaborado pela autora

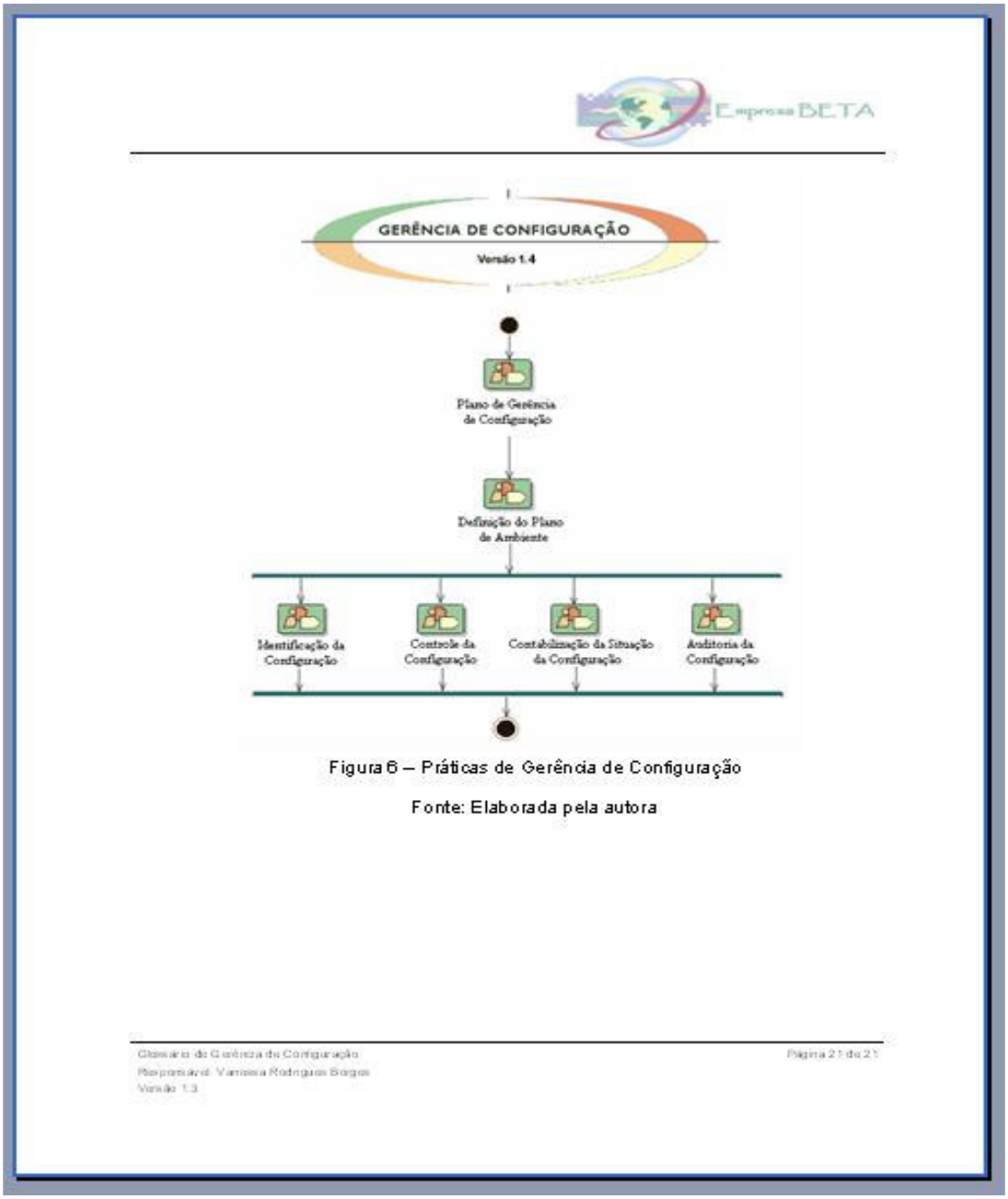


Figura 6 – Práticas de Gerência de Configuração

Fonte: Elaborada pela autora

Figura A.11 - Página 21 do Glossário de Configuração
Fonte: Elaborado pela autora

Apêndice B – Plano de Gerência de Configuração

PLANO DE GERÊNCIA DE CONFIGURAÇÃO

Data do Documento: 17/05/2006

Responsável pelo Documento: *Vanessa Rodrigues Borges*
vanessaborges@gmail.com

Versão do Documento: 1.4

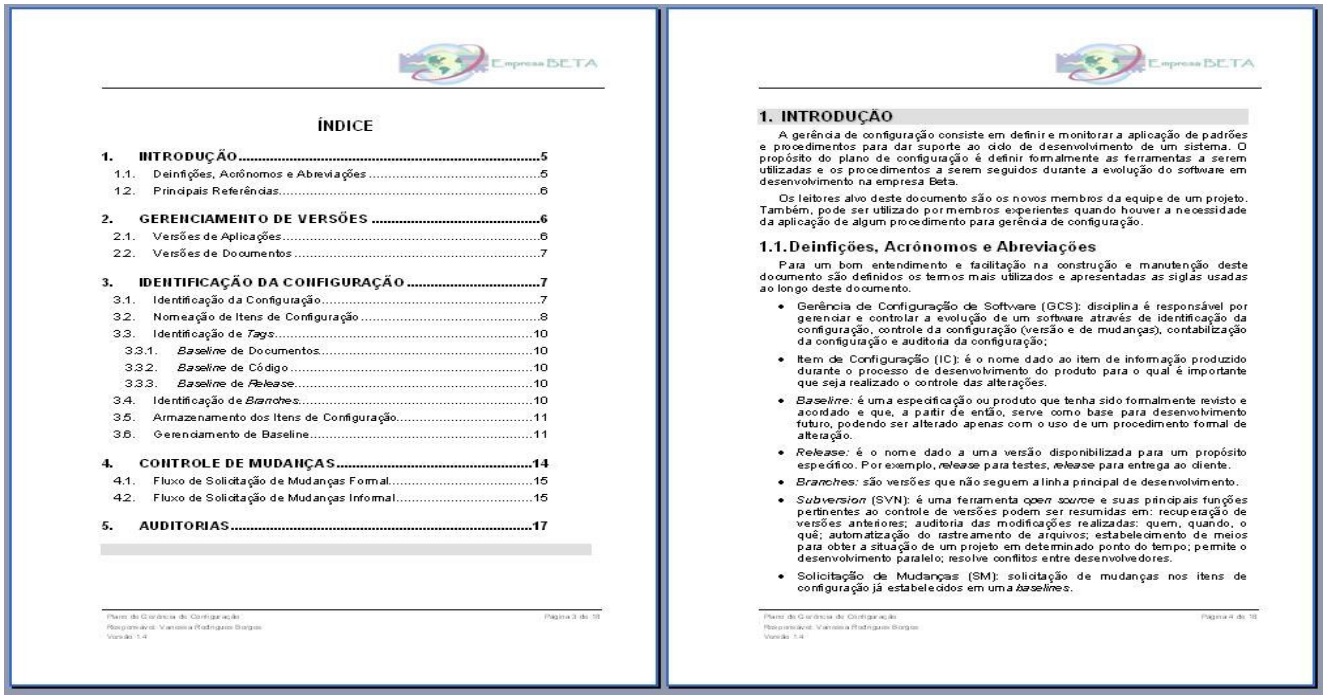
Empresa Beta
Lavras – MG CEP: 37200-000
http://www.empresabeta.com.br

Revisões do Documento

Revisões são melhoramentos na estrutura do documento e também no seu conteúdo. O objetivo primário desta tabela é a fácil identificação da versão do documento. Toda modificação no documento deve constar nesta tabela.

Data	Versão	Descrição	Autor
17/05/2006	1.0	Criação do documento	Vanessa Rodrigues Borges
12/06/2006	1.1	Conclusão do documento	Vanessa Rodrigues Borges
26/06/2006	1.2	Refinamento do documento após primeira apresentação ao grupo de melhoria do processo de software – SEPG	Vanessa Rodrigues Borges
03/07/2006	1.3	Refinamento do documento da segunda apresentação ao grupo de melhoria do processo de software – SEPG	Vanessa Rodrigues Borges
31/07/2006	1.4	Aprovação do documento pelo SEPG	Leandro de Paula Silva

Figura B.1 - Páginas 1 e 2 do Plano de Gerência de Configuração
Fonte: Elaborado pela autora



ÍNDICE

1. INTRODUÇÃO 5
1.1. Deinfições, Acrônomos e Abreviações 5
1.2. Principais Referências 6
2. GERENCIAMENTO DE VERSÕES 6
2.1. Versões de Aplicações 6
2.2. Versões de Documentos 7
3. IDENTIFICAÇÃO DA CONFIGURAÇÃO 7
3.1. Identificação da Configuração 7
3.2. Nomeação de Itens de Configuração 8
3.3. Identificação de *Tags* 10
3.3.1. *Baseline* de Documentos 10
3.3.2. *Baseline* de Código 10
3.3.3. *Baseline* de *Release* 10
3.4. Identificação de *Branches* 10
3.5. Armazenamento dos Itens de Configuração 11
3.6. Gerenciamento de Baseline 11
4. CONTROLE DE MUDANÇAS 14
4.1. Fluxo de Solicitação de Mudanças Formal 15
4.2. Fluxo de Solicitação de Mudanças Informal 15
5. AUDITORIAS 17

1. INTRODUÇÃO

A gerência de configuração consiste em definir e monitorar a aplicação de padrões e procedimentos para dar suporte ao ciclo de desenvolvimento de um sistema. O propósito do plano de configuração é definir formalmente as ferramentas a serem utilizadas e os procedimentos a serem seguidos durante a evolução do software em desenvolvimento na empresa Beta.

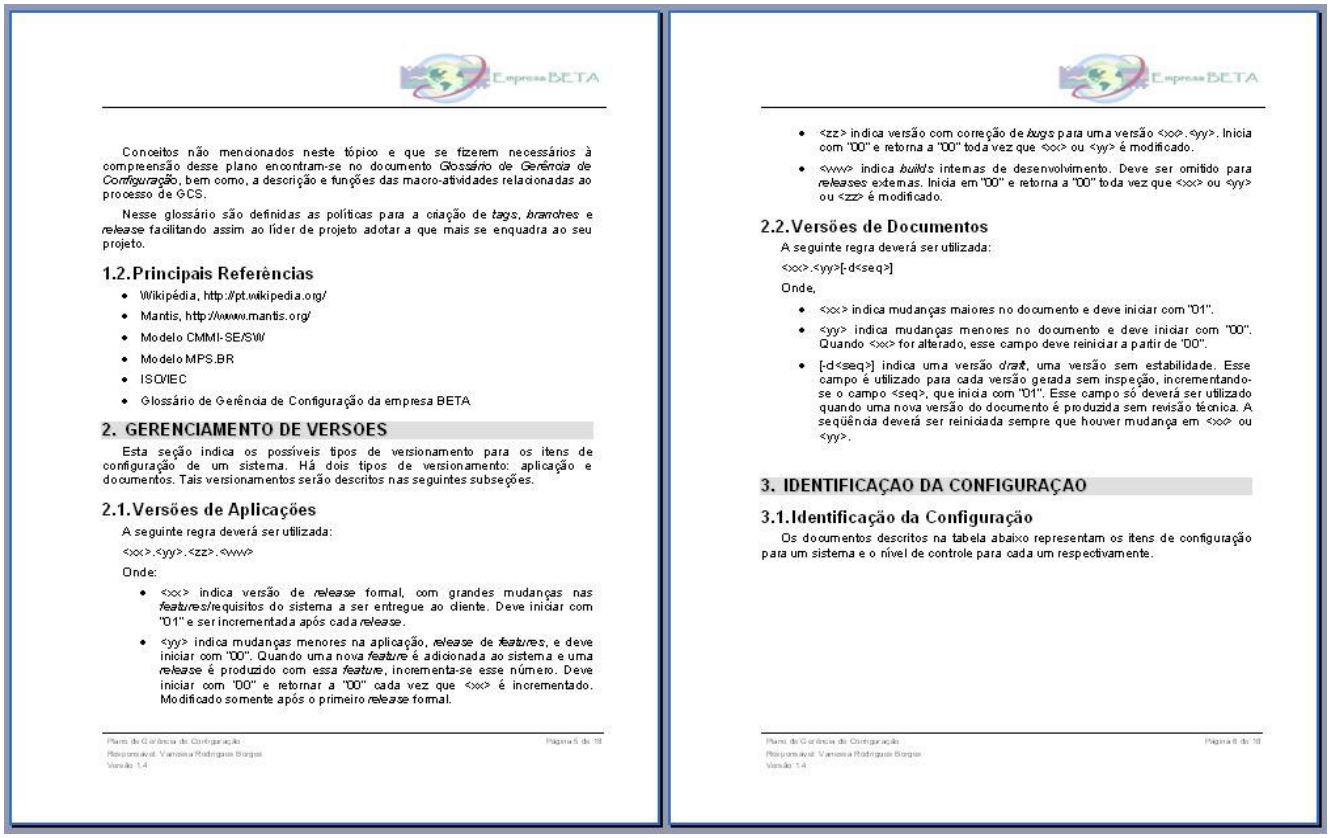
Os leitores alvo deste documento são os novos membros da equipe de um projeto. Também, pode ser utilizado por membros experientes quando houver a necessidade da aplicação de algum procedimento para gerência de configuração.

1.1. Deinfições, Acrônomos e Abreviações

Para um bom entendimento e facilitação na construção e manutenção deste documento são definidos os termos mais utilizados e apresentadas as siglas usadas ao longo deste documento.

- Gerência de Configuração de Software (GCS): disciplina é responsável por gerenciar e controlar a evolução de um software através de identificação da configuração, controle da configuração (versão e de mudanças), contabilização da configuração e auditoria da configuração;
- Item de Configuração (IC): é o nome dado ao item de informação produzido durante o processo de desenvolvimento do produto para o qual é importante que seja realizado o controle das alterações.
- *Baseline:* é uma especificação ou produto que tenha sido formalmente revisto e acordado e que, a partir de então, serve como base para desenvolvimento futuro, podendo ser alterado apenas com o uso de um procedimento formal de alteração.
- *Release:* é o nome dado a uma versão disponibilizada para um propósito específico. Por exemplo, *release* para testes, *release* para entrega ao cliente.
- *Branches:* são versões que não seguem a linha principal de desenvolvimento.
- *Subversion* (SVN): é uma ferramenta *open source* e suas principais funções pertinentes ao controle de versões podem ser resumidas em: recuperação de versões anteriores; auditoria das modificações realizadas: quem, quando, o quê; automatização do rastreamento de arquivos; estabelecimento de meios para obter a situação de um projeto em determinado ponto do tempo; permite o desenvolvimento paralelo; resolve conflitos entre desenvolvedores.
- Solicitação de Mudanças (SM): solicitação de mudanças nos itens de configuração já estabelecidos em uma *baselines*.

Figura B.2 - Páginas 3 e 4 do Plano de Gerência de Configuração
Fonte: Elaborado pela autora



Conceitos não mencionados neste tópico e que se fizerem necessários à compreensão desse plano encontram-se no documento *Glossário de Gerência de Configuração*, bem como, a descrição e funções das macro-atividades relacionadas ao processo de GCS.

Nesse glossário são definidas as políticas para a criação de *tags*, *branches* e *release* facilitando assim ao líder de projeto adotar a que mais se enquadra ao seu projeto.

1.2. Principais Referências

- Wikipédia, http://pt.wikipedia.org/
- Mantis, http://www.mantis.org/
- Modelo CMMI-SE/SW
- Modelo MPS.BR
- ISO/IEC
- Glossário de Gerência de Configuração da empresa BETA

2. GERENCIAMENTO DE VERSOES

Esta seção indica os possíveis tipos de versionamento para os itens de configuração de um sistema. Há dois tipos de versionamento: aplicação e documentos. Tais versionamentos serão descritos nas seguintes subseções.

2.1. Versões de Aplicações

A seguinte regra deverá ser utilizada:

<xx>.<yy>.<zz>.<www>

Onde:

- <xx> indica versão de *release* formal, com grandes mudanças nas *features*/requisitos do sistema a ser entregue ao cliente. Deve iniciar com "01" e ser incrementada após cada *release*.
- <yy> indica mudanças menores na aplicação, *release* de *features*, e deve iniciar com "00". Quando uma nova *feature* é adicionada ao sistema e uma *release* é produzido com essa *feature*, incrementa-se esse número. Deve iniciar com "00" e retornar a "00" cada vez que <xx> é incrementado. Modificado somente após o primeiro *release* formal.

- <zz> indica versão com correção de *bugs* para uma versão <xx>.<yy>. Inicia com "00" e retorna a "00" toda vez que <xx> ou <yy> é modificado.
- <www> indica *build's* internas de desenvolvimento. Deve ser omitido para *releases* externas. Inicia em "00" e retorna a "00" toda vez que <xx> ou <yy> ou <zz> é modificado.

2.2. Versões de Documentos

A seguinte regra deverá ser utilizada:

<xx>.<yy>[-d<seq>]

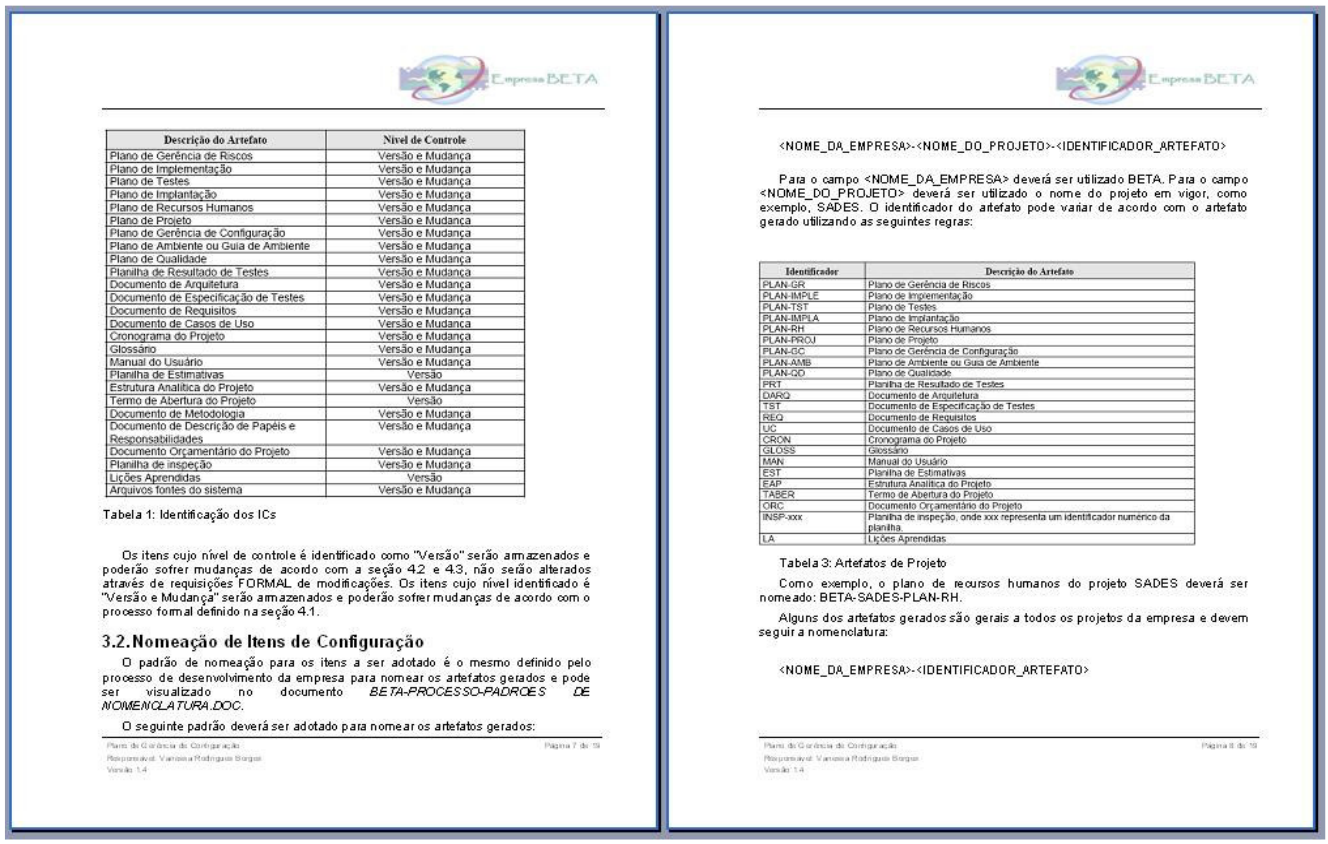
Onde,

- <xx> indica mudanças maiores no documento e deve iniciar com "01".
- <yy> indica mudanças menores no documento e deve iniciar com "00". Quando <xx> for alterado, esse campo deve reiniciar a partir de "00".
- [-d<seq>] indica uma versão *draft*, uma versão sem estabilidade. Esse campo é utilizado para cada versão gerada sem inspeção, incrementando-se o campo <seq>, que inicia com "01". Esse campo só deverá ser utilizado quando uma nova versão do documento é produzida sem revisão técnica. A seqüência deverá ser reiniciada sempre que houver mudança em <xx> ou <yy>.

3. IDENTIFICAÇAO DA CONFIGURAÇAO

3.1. Identificação da Configuração

Os documentos descritos na tabela abaixo representam os itens de configuração para um sistema e o nível de controle para cada um respectivamente.



Descrição do Artefato	Nível de Controle
Plano de Gerência de Riscos	Versão e Mudança
Plano de Implementação	Versão e Mudança
Plano de Testes	Versão e Mudança
Plano de Implantação	Versão e Mudança
Plano de Recursos Humanos	Versão e Mudança
Plano de Projeto	Versão e Mudança
Plano de Gerência de Configuração	Versão e Mudança
Plano de Ambiente ou Guia de Ambiente	Versão e Mudança
Plano de Qualidade	Versão e Mudança
Planilha de Resultado de Testes	Versão e Mudança
Documento de Arquitetura	Versão e Mudança
Documento de Especificação de Testes	Versão e Mudança
Documento de Requisitos	Versão e Mudança
Documento de Casos de Uso	Versão e Mudança
Cronograma do Projeto	Versão e Mudança
Glossário	Versão e Mudança
Manual do Usuário	Versão e Mudança
Planilha de Estimativas	Versão
Estrutura Analítica do Projeto	Versão e Mudança
Termo de Abertura do Projeto	Versão
Documento de Metodologia	Versão e Mudança
Documento de Descrição de Papéis e Responsabilidades	Versão e Mudança
Documento Orçamentário do Projeto	Versão e Mudança
Planilha de inspeção	Versão e Mudança
Lições Aprendidas	Versão
Arquivos fontes do sistema	Versão e Mudança

Tabela 1: Identificação dos ICs

Os itens cujo nível de controle é identificado como "Versão" serão armazenados e poderão sofrer mudanças de acordo com a seção 4.2 e 4.3, não serão alterados através de requisições FORMAL de modificações. Os itens cujo nível identificado é "Versão e Mudança" serão armazenados e poderão sofrer mudanças de acordo com o processo formal definido na seção 4.1.

3.2. Nomeação de Itens de Configuração

O padrão de nomeação para os itens a ser adotado é o mesmo definido pelo processo de desenvolvimento da empresa para nomear os artefatos gerados e pode ser visualizado no documento *BETA-PROCESSO-PADROES DE NOMENCLATURA.DOC*.

O seguinte padrão deverá ser adotado para nomear os artefatos gerados:

<NOME_DA_EMPRESA>-<NOME_DO_PROJETO>-<IDENTIFICADOR_ARTEFATO>

Para o campo <NOME_DA_EMPRESA> deverá ser utilizado BETA. Para o campo <NOME_DO_PROJETO> deverá ser utilizado o nome do projeto em vigor, como exemplo, SADES. O identificador do artefato pode variar de acordo com o artefato gerado utilizando as seguintes regras:

Identificador	Descrição do Artefato
PLAN-GR	Plano de Gerência de Riscos
PLAN-IMPLE	Plano de Implementação
PLAN-TST	Plano de Testes
PLAN-IMPLA	Plano de Implantação
PLAN-RH	Plano de Recursos Humanos
PLAN-PROJ	Plano de Projeto
PLAN-GC	Plano de Gerência de Configuração
PLAN-AMB	Plano de Ambiente ou Guia de Ambiente
PLAN-QD	Plano de Qualidade
PRT	Planilha de Resultado de Testes
DARQ	Documento de Arquitetura
TST	Documento de Especificação de Testes
REQ	Documento de Requisitos
UC	Documento de Casos de Uso
CRON	Cronograma do Projeto
GLOSS	Glossário
MAN	Manual do Usuário
EST	Planilha de Estimativas
EAP	Estrutura Analítica do Projeto
TABER	Termo de Abertura do Projeto
ORC	Documento Orçamentário do Projeto
INSP-xxx	Planilha de inspeção, onde xxx representa um identificador numérico da planilha
LA	Lições Aprendidas

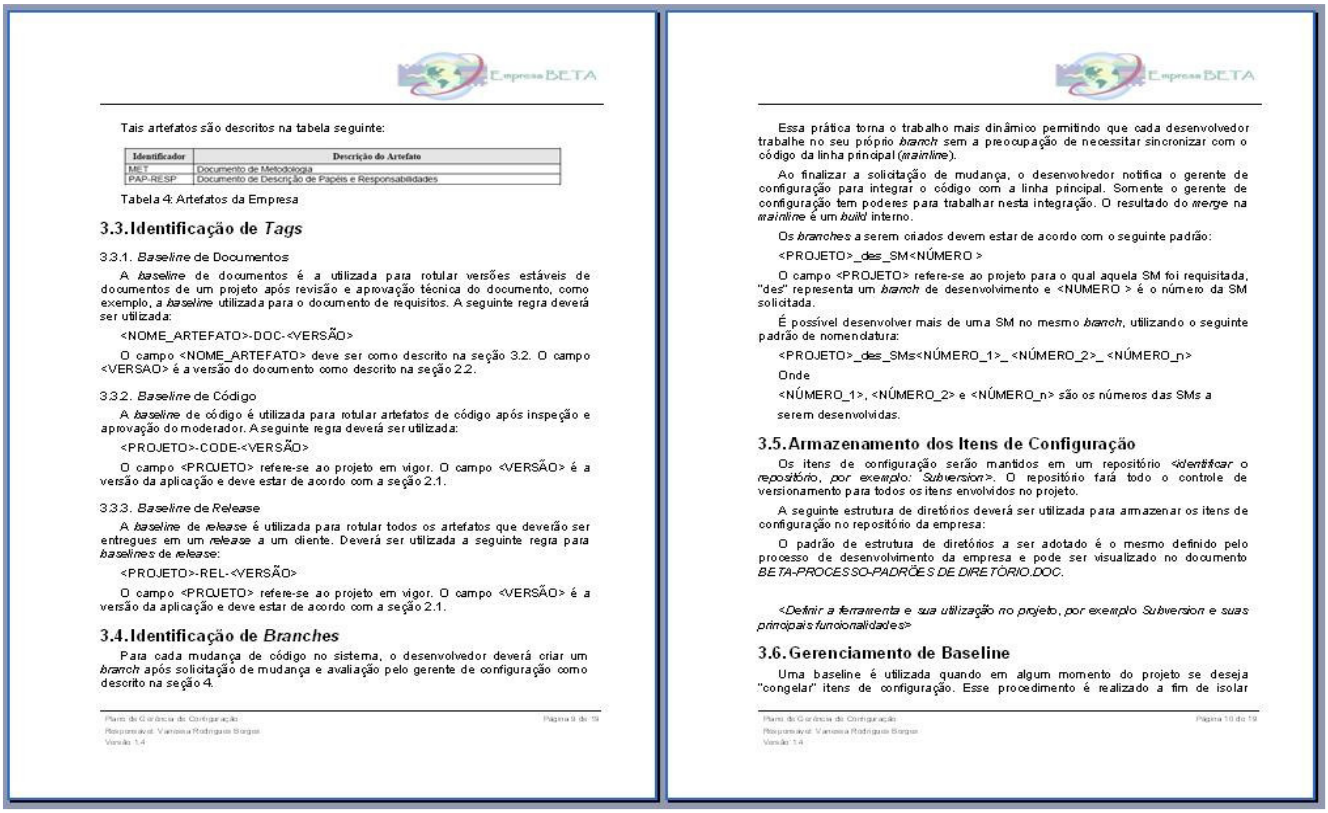
Tabela 3: Artefatos de Projeto

Como exemplo, o plano de recursos humanos do projeto SADES deverá ser nomeado: BETA-SADES-PLAN-RH.

Alguns dos artefatos gerados são gerais a todos os projetos da empresa e devem seguir a nomenclatura:

<NOME_DA_EMPRESA>-<IDENTIFICADOR_ARTEFATO>

Figura B.4 - Páginas 7 e 8 do Plano de Gerência de Configuração
Fonte: Elaborado pela autora



Tais artefatos são descritos na tabela seguinte:

Identificador	Descrição do Artefato
MET	Documento de Metodologia
PAP-RESP	Documento de Descrição de Papéis e Responsabilidades

Tabela 4: Artefatos da Empresa

3.3. Identificação de *Tags*

3.3.1. *Baseline* de Documentos

A *baseline* de documentos é a utilizada para rotular versões estáveis de documentos de um projeto após revisão e aprovação técnica do documento, como exemplo, a *baseline* utilizada para o documento de requisitos. A seguinte regra deverá ser utilizada:

<NOME_ARTEFATO>-DOC-<VERSÃO>

O campo <NOME_ARTEFATO> deve ser como descrito na seção 3.2. O campo <VERSAO> é a versão do documento como descrito na seção 2.2.

3.3.2. *Baseline* de Código

A *baseline* de código é utilizada para rotular artefatos de código após inspeção e aprovação do moderador. A seguinte regra deverá ser utilizada:

<PROJETO>-CODE-<VERSÃO>

O campo <PROJETO> refere-se ao projeto em vigor. O campo <VERSÃO> é a versão da aplicação e deve estar de acordo com a seção 2.1.

3.3.3. *Baseline* de *Release*

A *baseline* de *release* é utilizada para rotular todos os artefatos que deverão ser entregues em um *release* a um cliente. Deverá ser utilizada a seguinte regra para *baselines* de *release*:

<PROJETO>-REL-<VERSÃO>

O campo <PROJETO> refere-se ao projeto em vigor. O campo <VERSÃO> é a versão da aplicação e deve estar de acordo com a seção 2.1.

3.4. Identificação de *Branches*

Para cada mudança de código no sistema, o desenvolvedor deverá criar um *branch* após solicitação de mudança e avaliação pelo gerente de configuração como descrito na seção 4.

Essa prática torna o trabalho mais dinâmico permitindo que cada desenvolvedor trabalhe no seu próprio *branch* sem a preocupação de necessitar sincronizar com o código da linha principal (*mainline*).

Ao finalizar a solicitação de mudança, o desenvolvedor notifica o gerente de configuração para integrar o código com a linha principal. Somente o gerente de configuração tem poderes para trabalhar nesta integração. O resultado do *merge* na *mainline* é um *build* interno.

Os *branches* a serem criados devem estar de acordo com o seguinte padrão:

<PROJETO>_des_SM<NÚMERO >

O campo <PROJETO> refere-se ao projeto para o qual aquela SM foi requisitada, "des" representa um *branch* de desenvolvimento e <NUMERO > é o número da SM solicitada.

É possível desenvolver mais de uma SM no mesmo *branch*, utilizando o seguinte padrão de nomenclatura:

<PROJETO>_des_SMs<NÚMERO_1>_ <NÚMERO_2>_ <NÚMERO_n>

Onde

<NÚMERO_1>, <NÚMERO_2> e <NÚMERO_n> são os números das SMs a serem desenvolvidas.

3.5. Armazenamento dos Itens de Configuração

Os itens de configuração serão mantidos em um repositório *<identificar o repositório, por exemplo: Subversion>*. O repositório fará todo o controle de versionamento para todos os itens envolvidos no projeto.

A seguinte estrutura de diretórios deverá ser utilizada para armazenar os itens de configuração no repositório da empresa:

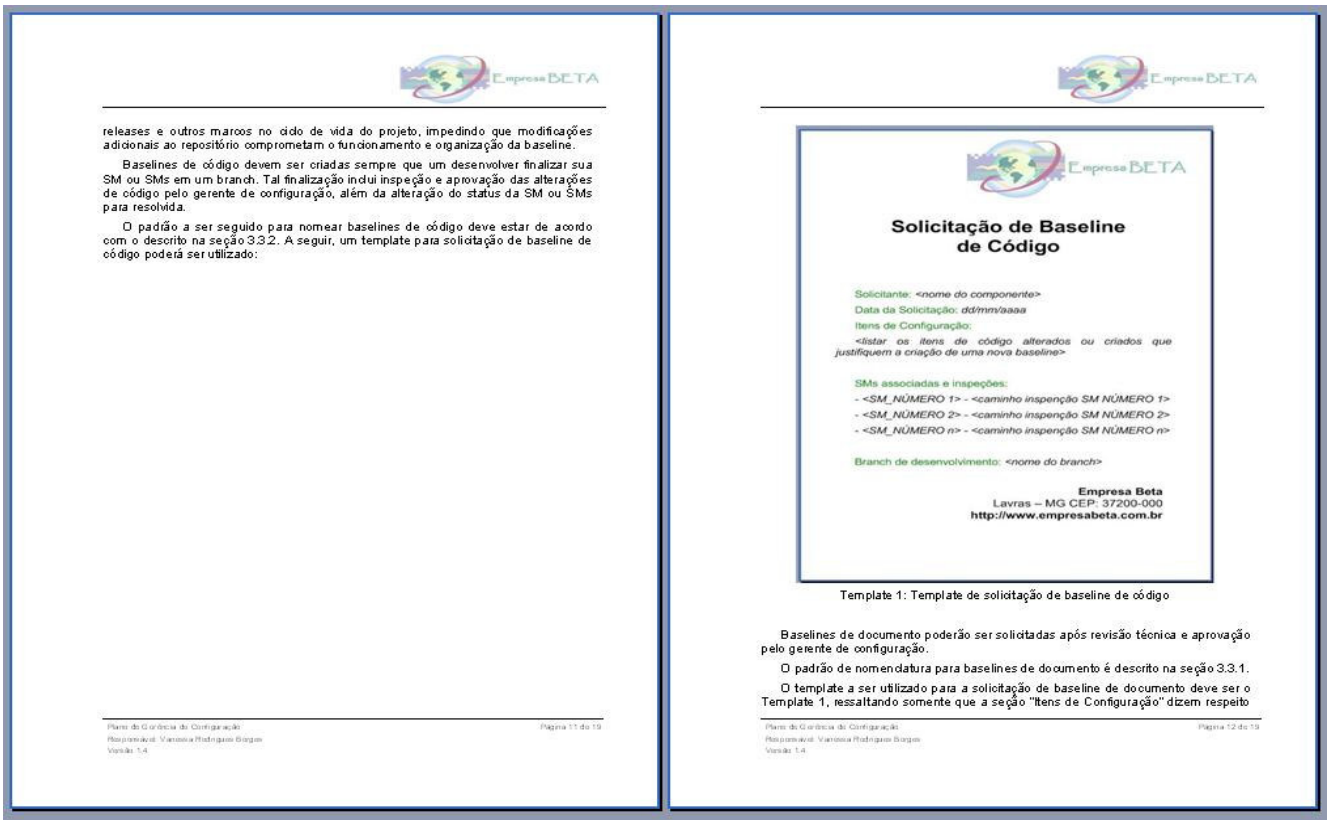
O padrão de estrutura de diretórios a ser adotado é o mesmo definido pelo processo de desenvolvimento da empresa e pode ser visualizado no documento *BETA-PROCESSO-PADRÕES DE DIRETÓRIO.DOC*.

<Definir a ferramenta e sua utilização no projeto, por exemplo Subversion e suas principais funcionalidades>

3.6. Gerenciamento de Baseline

Uma baseline é utilizada quando em algum momento do projeto se deseja "congelar" itens de configuração. Esse procedimento é realizado a fim de isolar

Figura B.5 - Páginas 9 e 10 do Plano de Gerência de Configuração
Fonte: Elaborado pela autora



releases e outros marcos no ciclo de vida do projeto, impedindo que modificações adicionais ao repositório comprometam o funcionamento e organização da baseline.

Baselines de código devem ser criadas sempre que um desenvolver finalizar sua SM ou SMs em um branch. Tal finalização inclui inspeção e aprovação das alterações de código pelo gerente de configuração, além da alteração do status da SM ou SMs para resolvida.

O padrão a ser seguido para nomear baselines de código deve estar de acordo com o descrito na seção 3.3.2. A seguir, um template para solicitação de baseline de código poderá ser utilizado:

Solicitação de Baseline de Código

Solicitante: *<nome do componente>*
Data da Solicitação: *dd/mm/aaaa*
Itens de Configuração:
<listar os itens de código alterados ou criados que justifiquem a criação de uma nova baseline>

SMs associadas e inspeções:
- *<SM_NÚMERO 1> - <caminho inspeção SM NÚMERO 1>*
- *<SM_NÚMERO 2> - <caminho inspeção SM NÚMERO 2>*
- *<SM_NÚMERO n> - <caminho inspeção SM NÚMERO n>*

Branch de desenvolvimento: *<nome do branch>*

Empresa Beta
Lavras – MG CEP: 37200-000
http://www.empresabeta.com.br

Template 1: Template de solicitação de baseline de código

Baselines de documento poderão ser solicitadas após revisão técnica e aprovação pelo gerente de configuração.

O padrão de nomenclatura para baselines de documento é descrito na seção 3.3.1.

O template a ser utilizado para a solicitação de baseline de documento deve ser o Template 1, ressaltando somente que a seção "Itens de Configuração" dizem respeito

Figura B.6 - Páginas 11 e 12 do Plano de Gerência de Configuração
Fonte: Elaborado pela autora

a documentos e não itens de código e também que o "Branch de desenvolvimento" não se aplica a esses casos, uma vez que as alterações de documentos são realizadas na mainline.

Baselines de release poderão ser solicitadas sempre que for necessário gerar uma versão para implantação no cliente acordada previamente.

O padrão de nomenclatura para baselines de release é descrito na seção 3.3.3. Uma baseline de release somente poderá ser solicitada após testes formais com planilha de resultados informando que todos os testes passaram.

Todos os documentos a serem entregues ao cliente sofrerão auditoria nesse momento, de acordo com o descrito na seção 5. O template a seguir deverá ser utilizado para a solicitação de um release:

Solicitação de Baseline de Release

Solicitante: *<nome do componente>*
Data da Solicitação: *dd/mm/aaaa*
SMs resolvidas e inspeções:
- *<SM_NÚMERO 1> - <caminho inspenção SM NÚMERO 1>*
- *<SM_NÚMERO 2> - <caminho inspenção SM NÚMERO 2>*
- *<SM_NÚMERO n> - <caminho inspenção SM NÚMERO n>*

Resultado de Testes: *<caminho da planilha contendo o resultado dos testes>*

Empresa Beta
Lavras – MG CEP: 37200-000
http://www.empresabeta.com.br

Template 2: Template de solicitação de baseline de release

4. CONTROLE DE MUDANÇAS

Durante o processo de desenvolvimento, mudanças descontroladas podem levar rapidamente ao caos. Assim, foi instituído na organização um processo que combine procedimentos humanos e ferramentas automatizadas para proporcionar um mecanismo de controle de mudanças.

Figura B.7 - Páginas 13 e 14 do Plano de Gerência de Configuração
Fonte: Elaborado pela autora

A seguir são descritos os procedimentos a serem seguidos para a gerência e controle de mudanças.

4.1. Fluxo de Solicitação de Mudanças Formal

Esse processo deve ser implementado depois que uma baseline for estabelecida; antes disso, somente um controle de mudanças informal precisa ser aplicado, ou seja, até que um item de configuração faça parte de uma baseline, o mesmo não precisará passar por controle de mudanças formal.

A Figura 1 apresenta o fluxo de controle de mudanças formal a ser aplicado para os itens que compõem uma baseline.

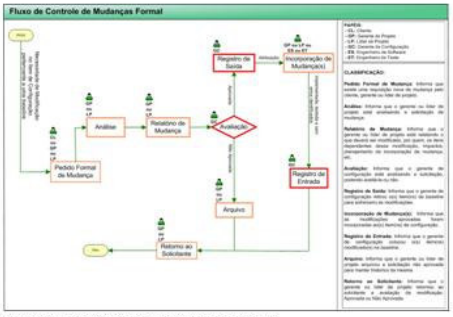


Figura 1: Fluxo de Controle de Mudanças Formal

4.2. Fluxo de Solicitação de Mudanças Informal

Para o controle de mudanças dos itens que não compõem ainda uma baseline, ou seja, controle de mudanças informal decidiu-se utilizar a ferramenta Mantis. Com isso,

toda e qualquer modificação realizada no desenvolvimento, antes do estabelecimento de uma baseline, poderia ser verificada e monitorada.

Na Figura 2 é apresentado o fluxo de requisitos.

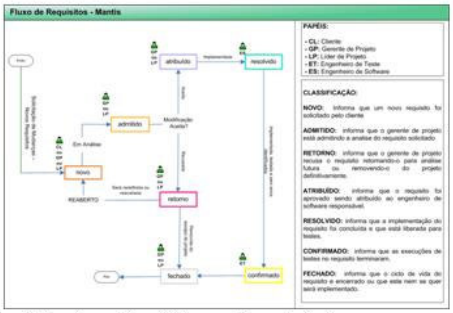


Figura 2: Fluxo de requisitos adotado para a Ferramenta Mantis

Na Figura 3 é apresentado o fluxo de bugs e o fluxo de requisitos.

Figura B.8 - Páginas 15 e 16 do Plano de Gerência de Configuração
Fonte: Elaborado pela autora

Figura 3: Fluxo de bugs adotado para a Ferramenta Mantis

5. AUDITORIAS

A auditoria da configuração tem como objetivo verificar se todas as mudanças implementadas foram feitas de uma forma correta. A auditoria é composta por duas partes: Funcional e Física.

A auditoria funcional é uma verificação técnica dos itens de configuração e verifica aspectos internos dos arquivos. Neste tipo de auditoria o foco é descobrir erros na configuração que não estão em conformidade com os padrões e procedimentos estabelecidos para os itens de configuração.

Para cada solicitação de baseline interna, quer seja de código ou documento, os itens de configuração relacionados deverão sofrer auditoria funcional, segundo o checklist apresentado abaixo: *<definir os itens que comporão o check list, abaixo é apresentado um exemplo de check list para auditoria funcional>*:

CHECK LIST –
AUDITORIA FUNCIONAL

√ Adequação a padrões:

□ Verificar caso haja a utilização de branches, o padrão de nomenclatura para nomeá-los;

□ Verificar padrão utilizado para solicitação de mudanças, que devem estar de acordo com a especificação nesse dovumento;

□ Itens de configuração referenciados na solicitação de baseline devem seguir o padrão de nomenclatura utilizada;

□ A estrutura de diretórios para os itens de configuração solicitados para a baseline devem estar de acordo com o previamente especificado;

□ Para baseline de documentos, devem ser verificados se o padrão de versionamento foi corretamente aplicado.

√ Status de SMs:

□ Todas as SMs presentes nessa solicitação de mudança, devem se encontrar no estado Fechado;

√ Planilhas de inspeção:

□ Devem ter sido aprovadas pelo gerente de configuração;

□ Todos os problemas encontrados em inspeção com status verificado;

□ Devem apresentar SM correspondente para a inspeção;

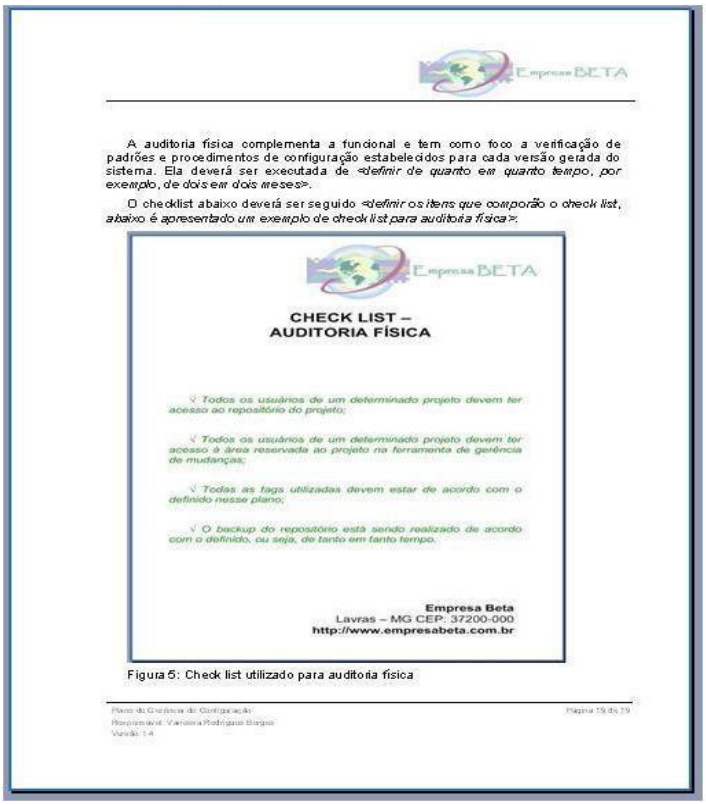
√ A baseline gerada a partir dessa solicitação deverá ser adicionada à ferramenta de controle de mudanças.

Empresa Beta
Lavras – MG CEP: 37200-000
http://www.empresabeta.com.br

Figura 4: Check list utilizado para auditoria funcional

Para cada release deverá ser realizada uma auditoria funcional.

Figura B.9 - Páginas 17 e 18 do Plano de Gerência de Configuração
Fonte: Elaborado pela autora



A auditoria física complementa a funcional e tem como foco a verificação de padrões e procedimentos de configuração estabelecidos para cada versão gerada do sistema. Ela deverá ser executada de *<definir de quanto em quanto tempo, por exemplo, de dois em dois meses>*.

O checklist abaixo deverá ser seguido *<definir os itens que comporão o check list, abaixo é apresentado um exemplo de check list para auditoria física>*:

CHECK LIST –
AUDITORIA FÍSICA

√ Todos os usuários de um determinado projeto devem ter acesso ao repositório do projeto;

√ Todos os usuários de um determinado projeto devem ter acesso à área reservada ao projeto na ferramenta de gerência de mudanças;

√ Todas as tags utilizadas devem estar de acordo com o definido nesse plano;

√ O backup do repositório está sendo realizado de acordo com o definido, ou seja, de tanto em tanto tempo.

Empresa Beta
Lavras – MG CEP: 37200-000
http://www.empresabeta.com.br

Figura 5: Check list utilizado para auditoria física

Figura B.10 - Página 19 do Plano de Gerência de Configuração
Fonte: Elaborado pela autora

Apêndice C – Plano de Ambiente

PLANO DE AMBIENTE

Data do Documento: 24/05/2006

Responsável pelo Documento: *Vanessa Rodrigues Borges*
vanessaborges@gmail.com

Versão do Documento: 1.2

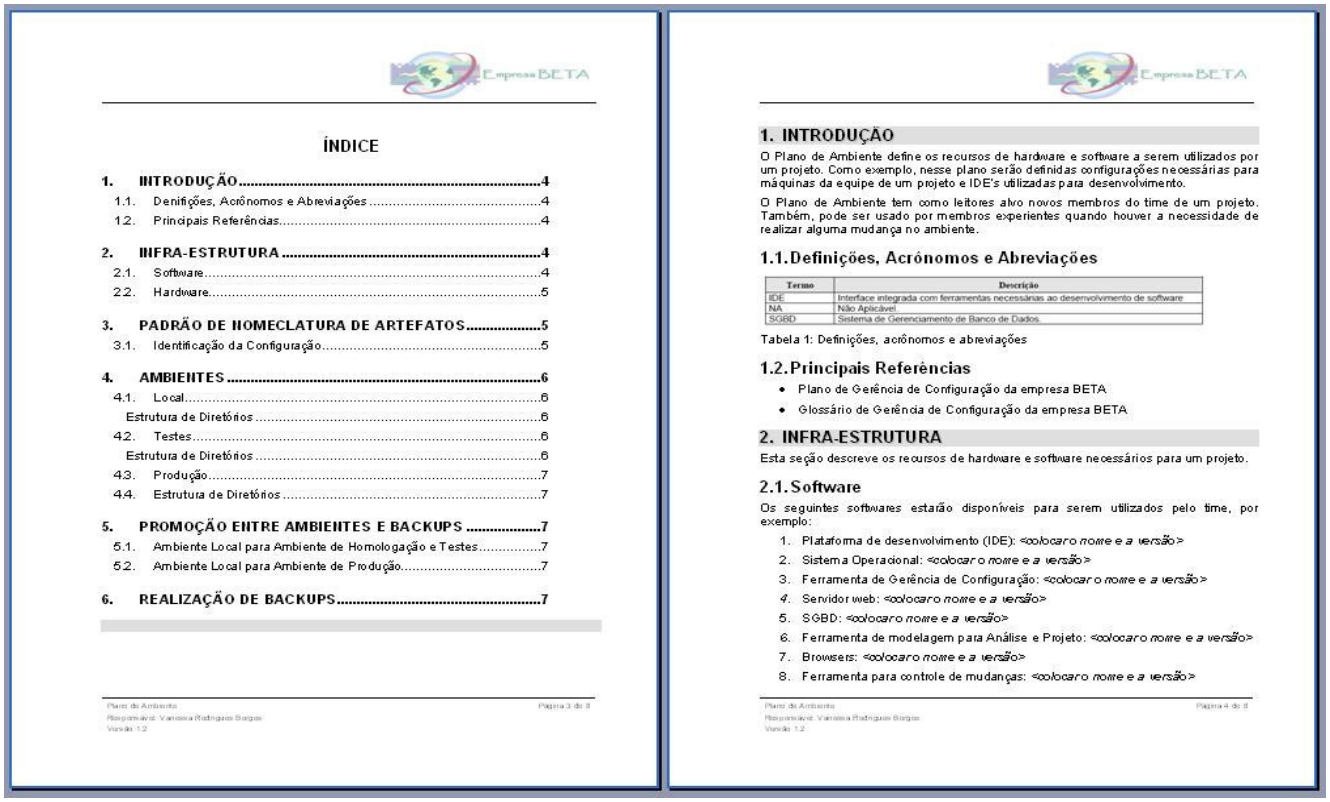
Empresa Beta
Lavras – MG CEP: 37200-000
http://www.empresabeta.com.br

Revisões do Documento

Revisões são melhoramentos na estrutura do documento e também no seu conteúdo. O objetivo primário desta tabela é a fácil identificação da versão do documento. Toda modificação no documento deve constar nesta tabela.

Data	Versão	Descrição	Autor
24/05/2006	1.0	Criação do documento	Vanessa Rodrigues Borges
19/06/2006	1.1	Conclusão do documento	Vanessa Rodrigues Borges
18/07/2006	1.2	Aprovação do documento pelo SEPG	Leandro de Paula Silva

Figura C.1 - Páginas 1 e 2 do Plano de Ambiente
Fonte: Elaborado pela autora



ÍNDICE

1. INTRODUÇÃO 4
1.1. Denifições, Acrônomos e Abreviações 4
1.2. Principais Referências 4
2. INFRA-ESTRUTURA 4
2.1. Software 4
2.2. Hardware 5
3. PADRÃO DE NOMECLATURA DE ARTEFATOS 5
3.1. Identificação da Configuração 5
4. AMBIENTES 6
4.1. Local 6
Estrutura de Diretórios 6
4.2. Testes 6
Estrutura de Diretórios 6
4.3. Produção 7
4.4. Estrutura de Diretórios 7
5. PROMOÇÃO ENTRE AMBIENTES E BACKUPS 7
5.1. Ambiente Local para Ambiente de Homologação e Testes 7
5.2. Ambiente Local para Ambiente de Produção 7
6. REALIZAÇÃO DE BACKUPS 7

1. INTRODUÇÃO

O Plano de Ambiente define os recursos de hardware e software a serem utilizados por um projeto. Como exemplo, nesse plano serão definidas configurações necessárias para máquinas da equipe de um projeto e IDE's utilizadas para desenvolvimento.

O Plano de Ambiente tem como leitores alvo novos membros do time de um projeto. Também, pode ser usado por membros experientes quando houver a necessidade de realizar alguma mudança no ambiente.

1.1. Definições, Acrônomos e Abreviações

Termo	Descrição
IDE	Interface integrada com ferramentas necessárias ao desenvolvimento de software
NA	Não Aplicável
SGBD	Sistema de Gerenciamento de Banco de Dados.

Tabela 1: Definições, acrônomos e abreviações

1.2. Principais Referências

- Plano de Gerência de Configuração da empresa BETA
- Glossário de Gerência de Configuração da empresa BETA

2. INFRA-ESTRUTURA

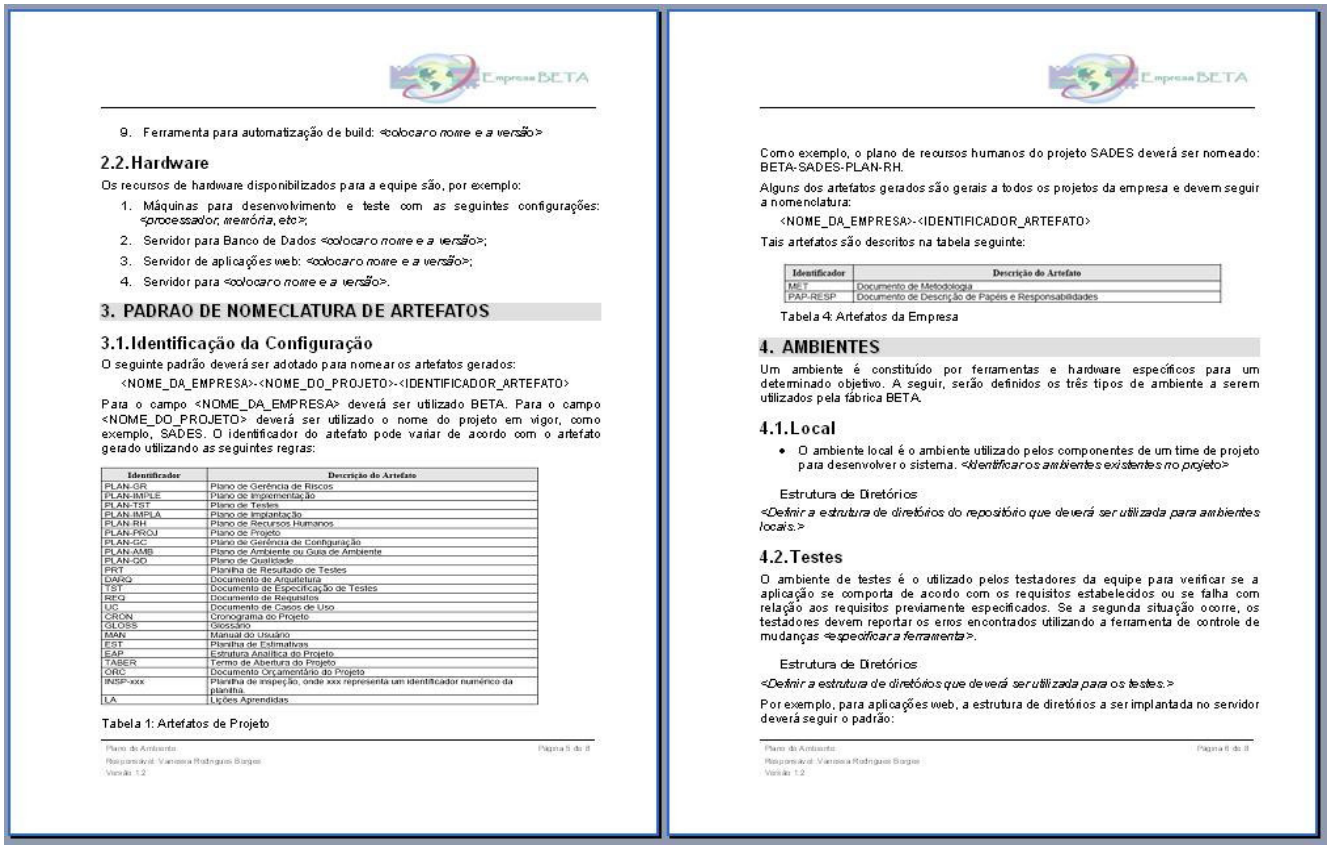
Esta seção descreve os recursos de hardware e software necessários para um projeto.

2.1. Software

Os seguintes softwares estarão disponíveis para serem utilizados pelo time, por exemplo:

1. Plataforma de desenvolvimento (IDE): *<colocar o nome e a versão>*
2. Sistema Operacional: *<colocar o nome e a versão>*
3. Ferramenta de Gerência de Configuração: *<colocar o nome e a versão>*
4. Servidor web: *<colocar o nome e a versão>*
5. SGBD: *<colocar o nome e a versão>*
6. Ferramenta de modelagem para Análise e Projeto: *<colocar o nome e a versão>*
7. Browsers: *<colocar o nome e a versão>*
8. Ferramenta para controle de mudanças: *<colocar o nome e a versão>*

Figura C.2 - Páginas 3 e 4 do Plano de Ambiente
Fonte: Elaborado pela autora



9. Ferramenta para automatização de build: *<colocar o nome e a versão>*

2.2. Hardware

Os recursos de hardware disponibilizados para a equipe são, por exemplo:

1. Máquinas para desenvolvimento e teste com as seguintes configurações: *<processador, memória, etc>*;
2. Servidor para Banco de Dados *<colocar o nome e a versão>*;
3. Servidor de aplicações web: *<colocar o nome e a versão>*;
4. Servidor para *<colocar o nome e a versão>*.

3. PADRAO DE NOMECLATURA DE ARTEFATOS

3.1. Identificação da Configuração

O seguinte padrão deverá ser adotado para nomear os artefatos gerados:

<NOME_DA_EMPRESA>-<NOME_DO_PROJETO>-<IDENTIFICADOR_ARTEFATO>

Para o campo <NOME_DA_EMPRESA> deverá ser utilizado BETA. Para o campo <NOME_DO_PROJETO> deverá ser utilizado o nome do projeto em vigor, como exemplo, SADES. O identificador do artefato pode variar de acordo com o artefato gerado utilizando as seguintes regras:

Identificador	Descrição do Artefato
PLAN-GR	Plano de Gerência de Riscos
PLAN-IMPLE	Plano de Implementação
PLAN-TST	Plano de Testes
PLAN-IMPLA	Plano de Implantação
PLAN-RH	Plano de Recursos Humanos
PLAN-PROJ	Plano de Projeto
PLAN-GC	Plano de Gerência de Configuração
PLAN-AMB	Plano de Ambiente ou Guia de Ambiente
PLAN-QO	Plano de Qualidade
PRT	Planilha de Resultado de Testes
DARQ	Documento de Arquitetura
TST	Documento de Especificação de Testes
REQ	Documento de Requisitos
UC	Documento de Casos de Uso
CRON	Cronograma do Projeto
GLOSS	Glossário
MAN	Manual do Usuário
EST	Planilha de Estimativas
EAP	Estrutura Analítica do Projeto
TABER	Termo de Abertura do Projeto
ORC	Documento Orçamentário do Projeto
INSP-xxx	Planilha de inspeção, onde xxx representa um identificador numérico da planilha
LA	Lições Aprendidas

Tabela 1: Artefatos de Projeto

Plano de Ambiente — Página 5 de 8
Responsável: Vanessa Rodrigues Borges
Versão 1.2

Como exemplo, o plano de recursos humanos do projeto SADES deverá ser nomeado: BETA-SADES-PLAN-RH.

Alguns dos artefatos gerados são gerais a todos os projetos da empresa e devem seguir a nomenclatura:

<NOME_DA_EMPRESA>-<IDENTIFICADOR_ARTEFATO>

Tais artefatos são descritos na tabela seguinte:

Identificador	Descrição do Artefato
MET	Documento de Metodologia
PAP-RESP	Documento de Descrição de Papéis e Responsabilidades

Tabela 4: Artefatos da Empresa

4. AMBIENTES

Um ambiente é constituído por ferramentas e hardware específicos para um determinado objetivo. A seguir, serão definidos os três tipos de ambiente a serem utilizados pela fábrica BETA.

4.1. Local

- O ambiente local é o ambiente utilizado pelos componentes de um time de projeto para desenvolver o sistema. *<Identificar os ambientes existentes no projeto>*

Estrutura de Diretórios

<Definir a estrutura de diretórios do repositório que deverá ser utilizada para ambientes locais.>

4.2. Testes

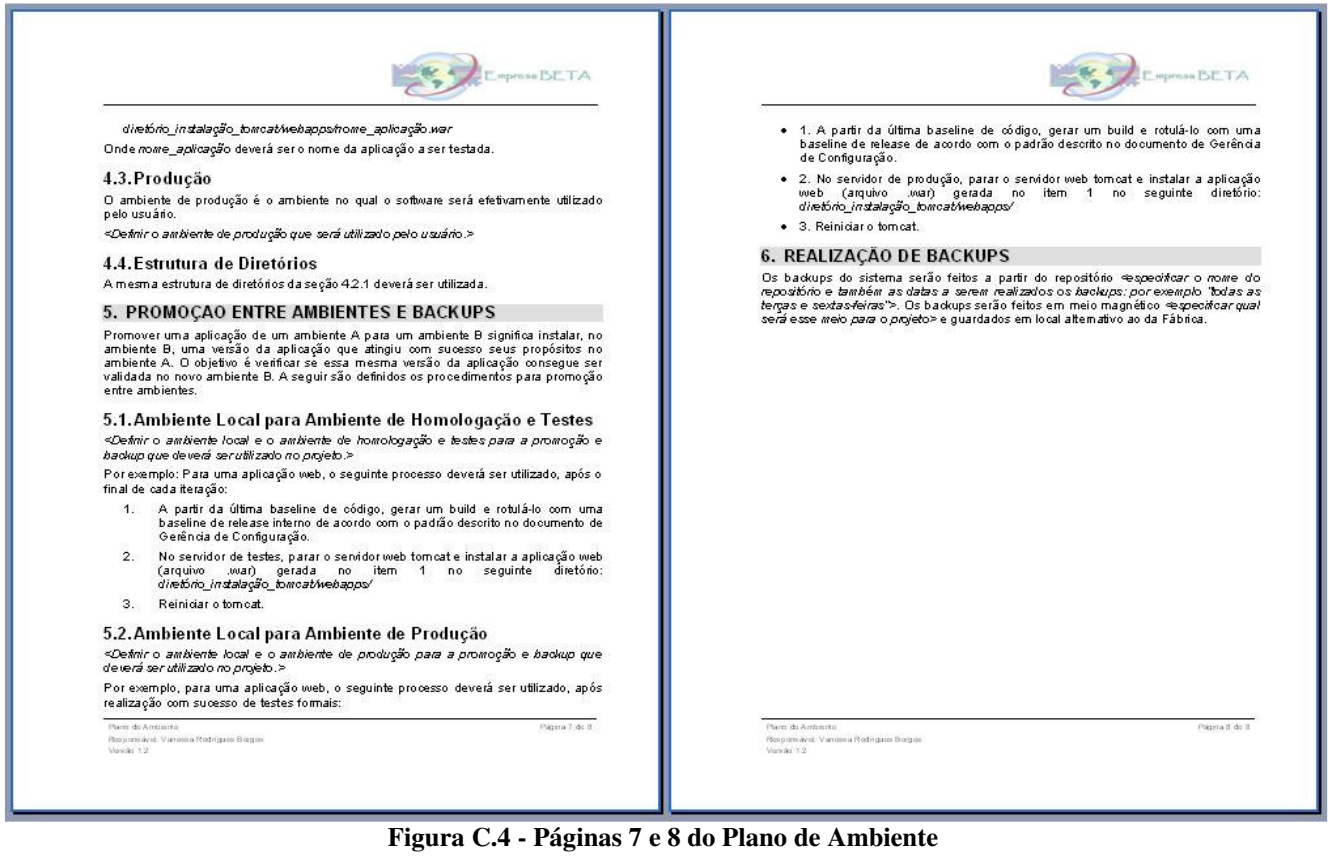
O ambiente de testes é o utilizado pelos testadores da equipe para verificar se a aplicação se comporta de acordo com os requisitos estabelecidos ou se falha com relação aos requisitos previamente especificados. Se a segunda situação ocorre, os testadores devem reportar os erros encontrados utilizando a ferramenta de controle de mudanças *<especificar a ferramenta>*.

Estrutura de Diretórios

<Definir a estrutura de diretórios que deverá ser utilizada para os testes.>

Por exemplo, para aplicações web, a estrutura de diretórios a ser implantada no servidor deverá seguir o padrão:

Plano de Ambiente — Página 6 de 8
Responsável: Vanessa Rodrigues Borges
Versão 1.2



diretório_instalação_tomcat/webapps/nome_aplicação.war

Onde *nome_aplicação* deverá ser o nome da aplicação a ser testada.

4.3. Produção

O ambiente de produção é o ambiente no qual o software será efetivamente utilizado pelo usuário.

<Definir o ambiente de produção que será utilizado pelo usuário.>

4.4. Estrutura de Diretórios

A mesma estrutura de diretórios da seção 4.2.1 deverá ser utilizada.

5. PROMOÇAO ENTRE AMBIENTES E BACKUPS

Promover uma aplicação de um ambiente A para um ambiente B significa instalar, no ambiente B, uma versão da aplicação que atingiu com sucesso seus propósitos no ambiente A. O objetivo é verificar se essa mesma versão da aplicação consegue ser validada no novo ambiente B. A seguir são definidos os procedimentos para promoção entre ambientes.

5.1. Ambiente Local para Ambiente de Homologação e Testes

<Definir o ambiente local e o ambiente de homologação e testes para a promoção e backup que deverá ser utilizado no projeto.>

Por exemplo: Para uma aplicação web, o seguinte processo deverá ser utilizado, após o final de cada iteração:

1. A partir da última baseline de código, gerar um build e rotulá-lo com uma baseline de release interno de acordo com o padrão descrito no documento de Gerência de Configuração.
2. No servidor de testes, parar o servidor web tomcat e instalar a aplicação web (arquivo .war) gerada no item 1 no seguinte diretório: *diretório_instalação_tomcat/webapps/*
3. Reiniciar o tomcat.

5.2. Ambiente Local para Ambiente de Produção

<Definir o ambiente local e o ambiente de produção para a promoção e backup que deverá ser utilizado no projeto.>

Por exemplo, para uma aplicação web, o seguinte processo deverá ser utilizado, após realização com sucesso de testes formais:

Plano de Ambiente — Página 7 de 8
Responsável: Vanessa Rodrigues Borges
Versão 1.2

- 1. A partir da última baseline de código, gerar um build e rotulá-lo com uma baseline de release de acordo com o padrão descrito no documento de Gerência de Configuração.
- 2. No servidor de produção, parar o servidor web tomcat e instalar a aplicação web (arquivo .war) gerada no item 1 no seguinte diretório: *diretório_instalação_tomcat/webapps/*
- 3. Reiniciar o tomcat.

6. REALIZAÇÃO DE BACKUPS

Os backups do sistema serão feitos a partir do repositório *<especificar o nome do repositório e também as datas a serem realizados os backups; por exemplo "todas as tergas e sextas-feiras">*. Os backups serão feitos em meio magnético *<especificar qual será esse meio para o projeto>* e guardados em local alternativo ao da Fábrica.

Plano de Ambiente — Página 8 de 8
Responsável: Vanessa Rodrigues Borges
Versão 1.2

Figura C.3 - Páginas 5 e 6 do Plano de Ambiente
Fonte: Elaborado pela autora

Figura C.4 - Páginas 7 e 8 do Plano de Ambiente
Fonte: Elaborado pela autora