Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Carlos Eduardo Domingues Barros

Gerência de configuração de software: um estudo de caso

São Paulo

2007

Carlos Eduardo Domingues Barros	Gerência de configuração de software: um estudo de caso

Carlos Eduardo Domingues Barros

Gerência de configuração de software: um estudo de caso

Dissertação apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, para obtenção do título de mestre em: Engenharia de Computação.

Área de concentração: Engenharia de Software.

Orientador:
Prof. Dr. Mauro de Mesquita Spinola

São Paulo
12 / 2007

RESUMO

A partir do momento em que um programa de computador entra em funcionamento, é iniciado um fluxo contínuo e repetitivo de interações para que suas funcionalidades sejam corrigidas, aprimoradas ou ampliadas. A Gerência de Configuração de *Software* (GCS) engloba o planejamento, coordenação e monitoração das alterações de um *software*, desde sua concepção inicial até a sua implementação em ambiente produtivo. O objetivo deste trabalho é o de criar um roteiro para a GCS, abstraindo a etapa de desenvolvimento. Busca-se estabelecer, neste roteiro, as necessidades básicas para que o contratante do *software* possa, de forma segura, conduzí-lo em ambiente produtivo durante o seu ciclo de vida, que inclui a implantação de novas versões. O roteiro foi criado a partir da comparação de informações coletadas, embasadas em fundamentação teórica e em estudo de caso aplicado em uma instituição financeira, com objetivo de compreender, com maior clareza, o processo de GCS. O resultado deste trabalho foi a criação de um roteiro para Gerência de Configuração de Software. O trabalho conclui que a GCS, quando utilizada adequadamente, pode contribuir efetivamente para a melhoria da integridade do *software*, diminuindo os riscos das implementações realizadas no ambiente produtivo.

Palavras-chave: Gerência de Configuração de *Software*, melhoria de processos.

ABSTRACT

Software Configuration Management: a case study

From the moment the computer starts operating, a continuous stream of interactions is started, the function of which is to correct, improve and amplify. The Software Configuration Management (SCM) encompasses the monitoring, planning and coordination of changes made to the software, from its initial conception to its implementation in a production environment. The objective of this research is to create a blueprint for SCM, extracting the development stage. Looking to include the basic software requirements of the costumer, so that the software can be controlled in the production environment during its life cycle, which includes version updates. This blueprint was created from comparisons of collected data, based on fundamental theory and on a case study in a financial institution, with the objective of understanding, with greater clarity, the SCM process. The result of this research was the creation of a blueprint for SCM. This research concluded that SCM, when appropriately used, can effectively contribute to an improvement in the software integrity, reducing the risks involved in the implementation of software in the production environment.

Key words: Software configuration management, process improvement.

Lista de figuras

Lista de quadros

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
ANSI	American National Standards Institute
BC	Banco Central
Backout	Plano de contingência para restaurar o ambiente na última versão disponível.
Backup	Processo de restaurar uma cópia realizada anteriormente.
CCM	Conselho de Controle de Modificações
CMMI	*Capability Maturity Model Integration*
COB	*Continuity of Business*
DA	*Data administration*
DBA	*Data base administration*
DoD	*Department of Defense*
DR	Desastre e Recuperação
GCS	Gerência de configuração de *software*
GEIA	*Government Electronics and Information Technology Association*
IEEE	*Institute of Electrical and Electronics Engineers*
IPT	Instituto de Pesquisas Tecnológicas do Estado de São Paulo
ISO	*International Organization for Standardization*
ITIL	*Information Technology Infrastructure Library*
PQA	*Process quality assurance*
QA	*Quality Assurance*
Regional Options	Componente do sistema operacional Windows para configuração de formato de data, números, entre outros.
SCM	*Software configuration management*
SEI	*Software Engineering Institute*
SPB	Sistema de Pagamento Brasileiro
TI	Tecnologia da informação

Sumário

1 Introdução

A GCS (Gerência de Configuração de *Software*) visa estabelecer e manter a integridade dos itens de *software* desenvolvidos durante o ciclo de vida do sistema, através de: controle de versão, controle sobre as mudanças e integração contínua (SEI-CMMI, 2006, p.114).

Babich (1986, p.16) cita que “todo o produto existe em formas alternativas, versões sucessivas ou customizações para diferentes ambientes”. Pressman (2005, p.740) fortalece tal idéia, citando que “Uma alteração é inevitável quando um *software* é produzido” e descreve que as principais motivações para alterações de um *software* são: novas funcionalidades, alterações legais ou regras novas para um produto já existente, modificações específicas de um cliente, crescimento da empresa criando novas demandas, alterações orçamentárias ou de cronograma redefinindo o sistema ou produto.

Sommerville (2007, p.44) explica que sistemas de missões críticas são aqueles que em caso de falha podem resultar em uma grande perda financeira, danos físicos ou perda de vidas humanas.

Para contextualizar o risco de uma alteração, é relatada abaixo uma experiência sobre a implantação de uma nova versão de um sistema de missão crítica.

1.1 Um exemplo real

Um *software,* desenvolvido para uma organização financeira, tinha a responsabilidade de manter a comunicação com o BC (Banco Central) através da rede do SPB (Sistema de Pagamento Brasileiro). Esta comunicação permitia transferências eletrônicas entre as instituições financeiras e, por questões legais, o ambiente deveria ficar disponível no horário compreendido as 6h30 e 18h30, com uma zona de conforto de 30 minutos antes do horário de abertura e 30 minutos após o horário de fechamento. Somando o horário de disponibilidade do *software* e o tempo necessário para backup, o tempo para alterações é de 7 horas, sendo que deste período, 4 horas são reservadas para restaurar a versão anterior do sistema em caso de necessidade. Dessa forma, das 24 horas de um dia, restavam apenas 3 horas para realizar todas as atividades de alteração do *software*.

O *software* já estava em operação por mais de quatro anos e, após várias alterações implantadas, foi iniciado mais um processo de alteração. Os procedimentos para instalação da nova versão foram executados com sucesso, ou seja, a versão com todas as características, funções e parâmetros especificados foi implementada em produção. Contudo, durante a

validação inicial, o *software* apresentou falhas para o envio de mensagens, que impedia seu funcionamento. Imediatamente, para correção dessa falha, teve início um processo de análise da causa da falha, apesar de o tempo para realizar as correções do *software* ser curtíssimo. Praticamente tudo foi verificado e, após análise, o fabricante de *software* informou que seria necessário gerar uma nova versão para contornar o problema manifestado naquele momento. Embora o teste da nova versão no ambiente de homologação ser viável, o tempo para realizar os testes e implementar a solução em produção não era suficiente. Para que não se corresse maiores riscos, tomou-se a decisão de voltar para a versão anterior do *software*.

O processo para implementação da versão anterior iniciou-se às 3 horas da manhã e quase todas as atividades foram concluídas rapidamente, porém, a restauração do banco de dados demorou além do tempo estimado e atrasou a liberação do ambiente.

A comunicação com o SPB ficou disponível às 6h27 da manhã, ultrapassando a zona de conforto de funcionamento do sistema e criando um mal estar entre os gestores da instituição financeira e o Banco Central.

A primeira questão levantada foi sobre o ambiente de homologação, o qual não apresentou a mesma instabilidade de produção. Após uma investigação detalhada nos ambientes, foi descoberta uma irregularidade: um arquivo de banco de dados, que continha algumas instruções, não foi executado em produção, causando a falha no *software*.

Durante a manipulação dos arquivos entre os ambientes de homologação e produção, um dos arquivos não foi submetido no fluxo correto. Embora o problema tenha ocorrido por falha humana, o risco nas atividades foi potencializado pela falta de ferramentas e processos adequados para gerenciar o procedimento de implementação.

O risco para a instituição financeira foi enorme, pois poderia gerar prejuízos financeiros diretos, em virtude de multas do BC, e indiretos, pela perda de credibilidade perante o mercado.

Babich (1986, p.4) relata problemas similares, como o de um desenvolvedor que corrigiu e testou um defeito no sistema, mas o resultado dos testes realizado pela área de QA (*Quality Assurance*) era negativo. A falha aconteceu porque a versão utilizada pelo departamento de QA era diferente daquela que teve a falha corrigida.

1.2 Contexto

Na perspectiva da engenharia de *software*, a GCS facilita o desenvolvimento e as atividades necessárias para que seja realizada uma alteração (IEEE-SWEBOK, 2004, p.7-2). Sendo

assim, os problemas relatados no item 1.1 poderiam ser rastreados e solucionados com maior facilidade, ou até mesmo não ocorrer se houvesse GCS implementada.

A Figura 1.1 ilustra o fluxo de uma alteração em um sistema já implementado em produção. No exemplo, a alteração parte do surgimento de uma nova idéia. Durante a pré-especificação é realizado o levantamento dos requisitos necessários, para alterar o *software*, através de conferências, e-mail e reuniões entre os integrantes dos grupos envolvidos. Após os estudos necessários, a especificação pronta é encaminhada para a área de desenvolvimento. Os ciclos de desenvolvimento do *software* realizados durante o desenvolvimento não serão abordados nesse trabalho. Após a etapa de desenvolvimento, o *software* e a documentação são entregues para o cliente, que iniciará o processo de homologação. A homologação da nova versão é realizada em um ambiente que contenha as mesmas características do ambiente produtivo. Nesse ambiente, o usuário efetuará a verificação no *software*, ou seja, checará se o *software* cumpre com suas especificações. Somente após a aprovação do usuário, a nova versão será implementada no ambiente produtivo.

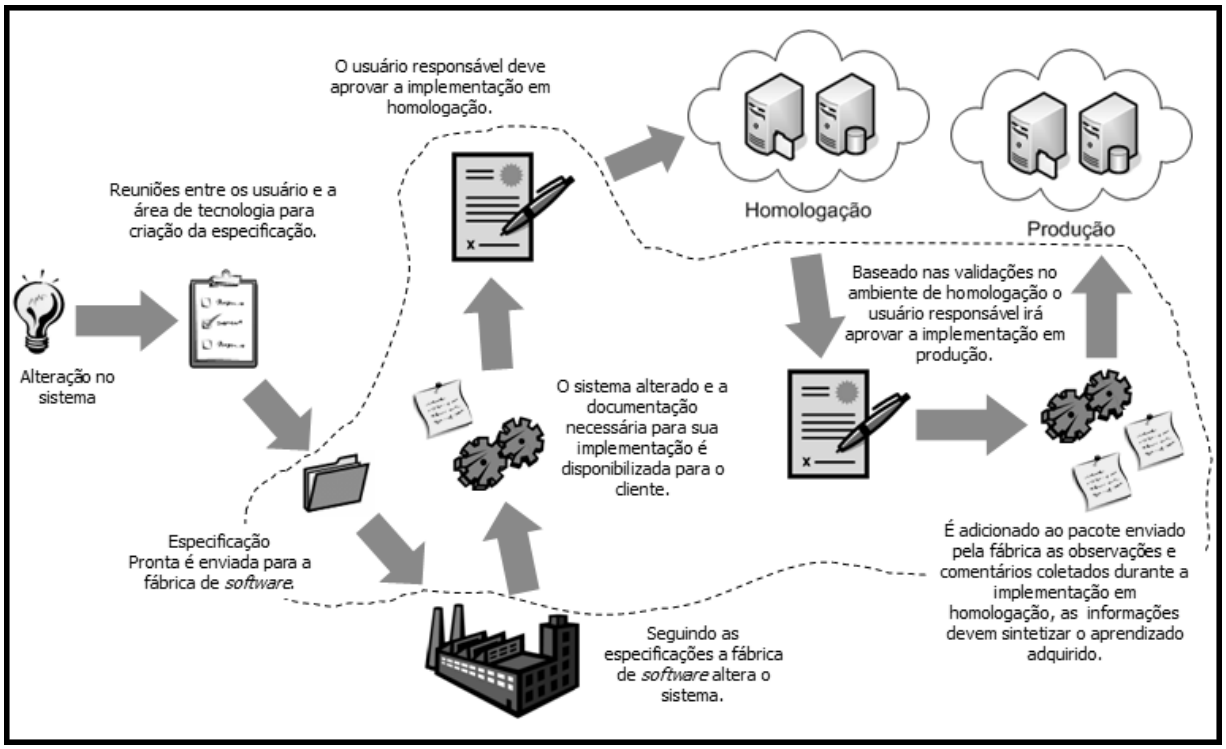

Figura 1.1 - Representação gráfica do contexto do trabalho.

Babich (1986, p.9) define três problemas típicos para justificar a utilização da Gerência de Configuração de *Software*:

(1) Quando existem duas cópias do mesmo *software* e as duas devem ser conduzidas dentro da empresa. O processo de GCS deve gerenciar as cópias e quando um defeito for encontrado, as duas versões devem ser atualizadas.

Ex.: a empresa possui dois ambientes: o primeiro é utilizado para produção e o segundo é utilizado para contingência em caso de catástrofe. Para que o ambiente de contingência seja útil, é necessário que ele acompanhe a mesma versão do *software* de produção.

(2) Quando existem problemas de dados compartilhados.

Ex.: essa situação é aplicável principalmente no processo de desenvolvimento e ocorre quando dois desenvolvedores alteram a mesma parte do código de forma conflitante.

(3) Quando existem atualizações simultâneas. Elas podem ser conflitantes e, por isso, deixar um dos sistemas indisponíveis.

Ex.: dois *softwares* diferentes, que utilizam o mesmo componente para conexão com o banco de dados, mas com versões diferentes, podem ser conflitantes entre si. Uma atualização nesse componente pode afetar o funcionamento do outro *software* que não está envolvido na alteração. Dessa maneira, a alteração não ocorrerá até que o conflito seja resolvido.

Dentre os problemas mencionados, os mais comuns, em ambientes corporativos, são os do tipo um e três, porque em uma instituição com diversos departamentos geograficamente separados, a distribuição de um mesmo *software* pode acontecer várias vezes, aumentando o risco de uma atualização. Outro fator de risco é o número elevado de sistemas rodando simultaneamente em ambiente produtivo, pois a incompatibilidade e o conflito entre eles serão potencializados.

1.3 Objetivo

O objetivo deste trabalho é o de desenvolver um roteiro para Gerência de Configuração de *Software*.

1.4 Resultados esperados e contribuições

O resultado esperado é o da criação de um roteiro elaborado por meio da comparação entre os métodos observados nos fundamentos teóricos da GCS (Gerência de Configuração de *Software*) e pelos resultados obtidos no estudo de caso.

A principal contribuição que se busca é da criação de um roteiro de Gerência de Configuração de *Software*.

1.5 Metodologia do trabalho

O estudo de caso foi escolhido como técnica para demonstrar o processo de GCS aplicado em uma instituição. O presente trabalho propõe a comparação das práticas encontradas a partir desse estudo de caso com os métodos levantados com base em fundamentação teórica para criação do roteiro.

Marconi e Lakatos (2006, p.157) sintetizam as fases necessárias para uma investigação científica nos seguintes passos:

- *Seleção do tópico ou do problema para investigação.* A solução do problema deve preencher as lacunas do sistema, a partir de um conjunto de conhecimentos, ou propondo melhorias de forma clara e precisa.
- *Definição e diferenciação do problema.* Existe a possibilidade de recolocação de um velho problema à luz de novos conhecimentos (empíricos ou teóricos, substantivos ou metodológicos).
- *Levantamento de hipóteses de trabalho.* Criação de uma ou mais proposições que se fazem na tentativa de verificar a validade de uma resposta existente para o problema.
- *Coleta, sistematização e classificação dos dados.* Existem vários procedimentos para coleta dos dados, as quais variam de acordo com a investigação ex.: coleta documental, observação, entrevista, questionário, entre outros... Após a coleta, os dados são classificados de forma sistemática, seguindo os passos, respectivamente, de seleção, codificação e tabulação.
- *Análise e interpretação dos dados.* A análise e a interpretação dos dados são duas atividades distintas, mas estreitamente relacionadas. Na análise, o pesquisador entra em maiores detalhes sobre os dados decorrentes do trabalho estatístico. Durante o processo de interpretação é feita a atividade intelectual que procura dar um significado mais amplo às respostas.
- *Relatório do resultado da pesquisa.* Tabelas ou quadros são um método estatístico sistemático, de apresentar os dados. O propósito mais importante é o de ajudar o investigador na distinção de diferenças nos dados coletados, sempre fazendo uso da maior clareza possível.

Para criação do roteiro são utilizados elementos de pesquisa bibliográfica, obtida por meio de levantamento de literatura específica, bem como do resultado do estudo de caso aplicado em uma instituição financeira, conforme demonstra a Figura 1.2.

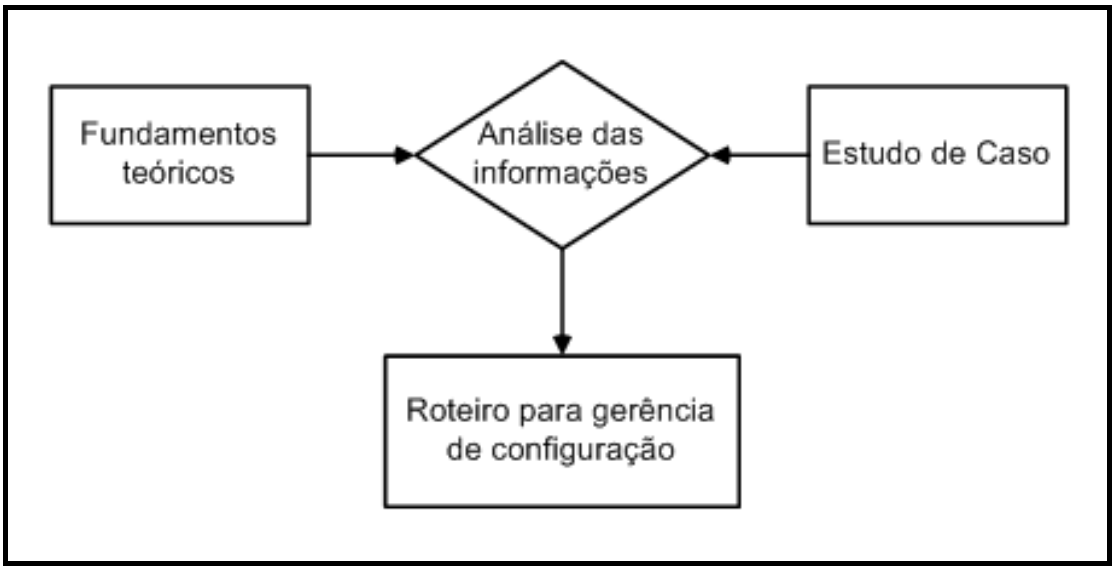

Figura 1.2 – A metodologia de pesquisa (Adaptado de TONINI, 2006, p.12).

A análise das informações - a partir da comparação entre as técnicas dos fundamentos teóricos e os resultados encontrados no estudo de caso - busca identificar e explicar as diferenças percebidas entre as duas abordagens.

1.6 Organização do trabalho

O trabalho está organizado em nove capítulos. Abaixo segue um resumo de sua organização:

O capítulo um, *Introdução**Erro! Fonte de referência não encontrada.***, expõe o problema tratado neste trabalho. Nesse capítulo também são explicados o objetivo, os resultados esperados, expectativas e a metodologia utilizada.

O capítulo dois, *Fundamentos teóricos*, apresenta os fundamentos da Gerência de Configuração de *Software*.

O capítulo três, *Estudo de caso: aspectos metodológicos*, enfoca os fundamentos da metodologia de pesquisa. Os aspectos apresentados nesse capítulo foram utilizados como base para a realização do estudo de caso.

No capítulo quatro, *Planejamento do estudo de caso*, é apresentado o plano para a realização do estudo de caso.

No capítulo cinco, *Aplicação do estudo de caso*, são apresentados os dados coletados no estudo de caso, bem como sua classificação e interpretação. Nesse capítulo também é apresentado o relatório do resultado sobre a pesquisa realizada.

O capítulo seis, *Proposta do roteiro* , apresenta o roteiro criado neste trabalho. O roteiro foi baseado em conhecimentos adquiridos durante os capítulos *Fundamentos teóricos* e *Aplicação do estudo de caso*.

O capítulo sete, *Conclusões e considerações finais*, discute as diferenças entre o que dita a fundamentação teórica e os aspectos práticos levantados com base no estudo de caso. O capítulo também apresenta as considerações finais e os planos para eventuais trabalhos futuros.

Os capítulos oito e nove, *Referências* e *Bibliografia complementar*, apresentam as fontes de material de pesquisa utilizadas no trabalho.

O capítulo dez, *Apêndices*, apresenta os materiais de suporte julgados oportunos para documentar e enriquecer o trabalho, ilustrando e, por esse motivo, facilitando o entendimento de alguns pontos da dissertação.

2 Fundamentos teóricos

Este capítulo apresenta os fundamentos da gerência de configurações existentes na atualidade.

2.1 Gerência de configuração de software

Gerência de Configuração de *Software* é a arte de identificar, organizar e controlar modificações de *softwares* criados pela equipe de desenvolvimento (BABICH, 1986, p.8). A GCS prevê a integridade do *software*, usando identificação e controle das configurações, provendo *status* e informações para métricas e auditoria (SEI-CMMI, 2006, p.114) (IEEE Std 828, 2005, p.iii) (ESTUBLIER, 2005, p.2). O objetivo da Gerência de Configuração de *Software* é o de maximizar a produtividade e minimizar as possíveis falhas. (LEON, 2005, p.xix)

GEA (2004, Cap.5) define as principais funções da configuração de *software*: (1) planejamento e gerenciamento, (2) Identificação dos itens de configuração, (3) Gerenciamento da configuração de *software*, (4) Métricas e (5) Verificação e auditoria.

A Figura 2.1 apresenta um esquema com base nas funções de configuração de *software*, articulando cada uma delas ao conjunto para implementação da GCS. Vale ressaltar que para alcançar o benefício da GCS, é necessário que todas as cinco funções colaborem em todas as partes do ciclo de vida do produto.

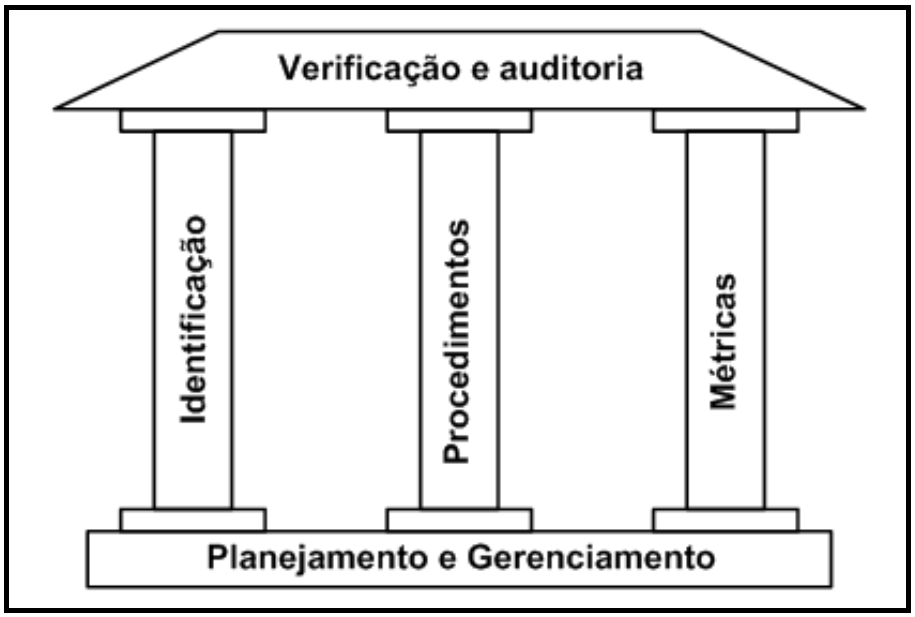

Figura 2.1 – Adaptação da representação gráfica das funções de gerência de configuração. (GEIA, 2004, cap.5)

O modelo ITIL (2004, p.79) define os principais objetivos da gerência de configuração. São eles:

- Prover uma gerência com grande controle sobre os itens de configuração;
- Prover as informações corretas para os processos ligados, criar e manter um banco de dados (repositório) confiável.

O ITIL (2004, p.86 e 98) descreve os benefícios da GCS:

- Informações sobre os itens de configurações para um melhor gerenciamento dos recursos de TI;
- Maior eficácia para efetuar alterações, para o oferecimento de suporte para problemas de segurança e consistência no processo de alteração;
- Minimizar os problemas de sincronismo com as diversas versões; redução de falhas através de um ambiente controlado;
- Elaboração de um registro completo sobre as alterações realizadas no ambiente.

2.1.1 Histórico

A Gerência de Configuração de *Software* teve sua origem na indústria bélica. Em 1962, a Força Aérea de Estados Unidos, reagindo a problemas de projeto de um avião a jato, publicou um padrão para gerência de configuração. Este foi o primeiro padrão para gerência de configuração, seguido por muitos outros padrões, a maioria deles com origem nos Militares e no Departamento de Defesa dos Estados Unidos (DoD).

Com a popularização dos computadores, a importância da indústria de desenvolvimento de *software* começou a aumentar. No início, os *softwares* serviam para facilitar atividades triviais, automatizando tarefas que eram realizadas manualmente. Com o tempo, as tarefas executadas foram se tornando complexas e sofisticadas, fazendo com que o número de pessoas envolvidas no processo de desenvolvimento aumentasse. Os *softwares* foram se tornando cada vez mais complexos e a dificuldade de gerenciar os projetos de *software* acompanhou essa jornada.

O Departamento de Defesa dos Estados Unidos, bem como várias organizações internacionais, como IEEE (*Institute of Electrical and Electronics Engineers*), ANSI (*American National Standards Institute*) e ISO (*International Organization for Standardization*), iniciaram pesquisas sobre a Gerência de Configuração, cada um definindo seus próprios padrões. Dentre eles, os padrões mais largamente utilizados são os padrões ANSI/IEEE.

Atualmente, a Gerência de Configuração de *Software* é aceita como uma disciplina e é praticada pela maior parte das organizações e a consciência sobre a necessidade e a importância de GCS também está aumentando. (LEON, 2005, p.8) (KEYES, 2004, p.xvii e p.45) (ESTUBLIER, 2002, p.1)

2.1.2 Planejamento

O primeiro passo da GCS é o seu planejamento, normalmente, de responsabilidade do gerente do projeto ou líder da equipe (KEYES, 2004, p.26). Durante o planejamento, são definidos os objetivos e os fatores de sucesso, chave do processo de GCS. Exemplos desses itens e atividades são tarefas, funções, linhas de baseline, políticas, entre outros. (ITIL, 2004, p.82)

Durante a GCS deverão ser inclusas especificações e documentos para definição das necessidades do produto. Alguns documentos, como resultados de testes, por exemplo, deverão ser criados e armazenados de acordo com a criticidade do produto (SEI-CMMI, 2006, p.117). Durante o desenvolvimento do *software*, vários documentos são criados, mas alguns deles não possuem relevância para a GCS. Para tratar disso, Sommerville (2007, p.693) sugere que, durante o planejamento, seja decidido quais documentos ou tipos de documentos devem ser controlados e quais não.

GEIA (2004, cap.5.1) define o planejamento e gerenciamento da seguinte forma:

- Definir as responsabilidades por cada elemento da GCS.
- Treinar e capacitar as pessoas que possuem responsabilidades de GCS.
- Determinar e aplicar os recursos necessários (incluindo ferramentas de GCS) para facilitar a implementação de um produto.
- Medir e indicar o desempenho para uma melhora contínua.
- Disponibilizar processos de GCS para os fornecedores e subcontratados.
- Integrar e organizar as informações dos processos de configuração.

Além das definições apresentadas acima, uma sugestão registrada no PMBOK (2004, p.90) é de que o planejamento seja revisado por especialistas, para que possam desenvolver detalhes técnicos que posteriormente poderiam ser inclusos no plano de projeto.

Abaixo, apresenta-se a definição dos itens de configuração e *baseline* que deverão ser utilizados durante a GCS.

2.1.3 Itens de configuração

Parte do planejamento é destinado para identificar os itens que deverão ser controlados na GCS. (SOMMERVILLE, 2007, p.693) Para tanto, deve-se compreender o contexto de configuração do *software*, selecionando os itens apropriados, descrevendo o seu relacionamento e identificando os *baselines* a serem usados:

- **Definição** – Item de configuração é um grupo de objetos indicado para gerenciamento que será tratado como uma entidade única no processo de GCS. Os itens de *software* com potencial para serem incluídos como itens de configuração são: documentos de especificação, material de teste, código fonte, executáveis compilados, documento para instalação, entre outros. Um item de configuração também pode ser um *hardware* ou a combinação de *hardware* e *software* (KEYES, 2004, p.78).
- **Relacionamento** – O relacionamento estrutural dos itens de configuração pode afetar outras atividades ou tarefas do GCS como a construção do *software* ou a análise de impacto na proposta de alteração. Mei (2001, p.2) exemplifica esse relacionamento através da dependência entre itens de configuração. Nesse caso necessariamente deverá existir um relacionamento entre os itens.

O rastreamento desses relacionamentos também é importante para suportar a rastreabilidade das alterações. GEIA (2004, cap.5.2.1) mostra na Figura 2.2 que o item de configuração consiste na junção das informações operacionais e na definição do produto.

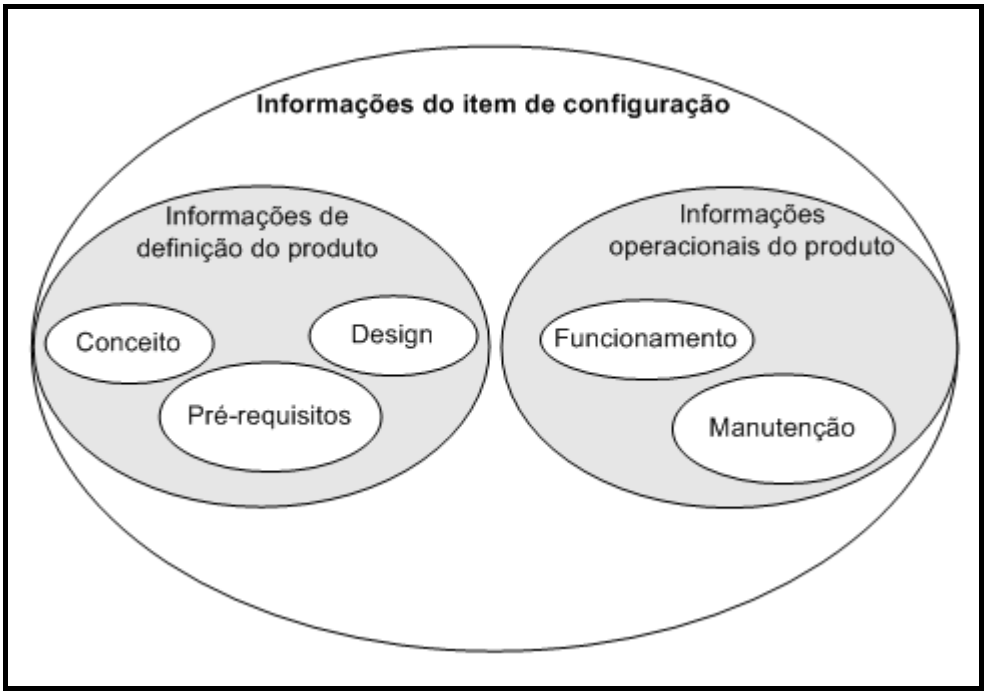

Figura 2.2 - Representação gráfica das funções dos itens de configuração e seus relacionamentos. (GEIA, 2004, cap.5.2.1)

Keyes (2004, p.85) explica que não existe nenhuma regra estipulando o número exato de itens de configuração necessários para uma implementação, mas quanto menor o número melhor. Selecionando um grande número de itens de configuração, o custo de desenvolvimento e suporte será maior.

Abaixo, seguem as conseqüências de se possuir um número alto de itens de configuração:

- Inúmeros relacionamentos entre os itens de configuração para serem definidos e documentados.
- Baixo nível de definição das funcionalidades dos itens de configuração, obrigando criação de documentos desnecessários.
- Aumento do número geral de requisições de documentos.
- Criação de uma excessiva fragmentação, piorando o entendimento e complicando a revisão de documentos, aprovação e controle de processos.

A seguir, as conseqüências, como contraponto, de se possuir um número baixo de itens de configuração:

- Aumento da complexidade de cada item de configuração.
- Diminuição do esforço para gerenciamento dos itens de configuração.
- Diminuição da possibilidade de reutilização dos itens de configuração.
- Testes formais de funcionalidades críticas são realizados com maior dificuldade.
- Dificuldade em endereçar as alterações efetivas, particularmente quando existe uma quantidade diferente de componentes entregues em cada alteração.
- Aumento da complexidade para gerenciar componentes comuns.

2.1.4 Baseline

"*Baseline* é um termo utilizado para definir um item de configuração designado e fixo em um determinado ponto no ciclo de vida do sistema." (IEEE-SWEBOK, 2004, p.7-6). A função do *baseline* varia de acordo com a fase em que o sistema está, mas seu objetivo é o de marcar pontos de transição durante o projeto. A entrega do sistema para o cliente, bem como, sua instalação em produção seriam pontos de transição chamados de *baseline*. Bryan, Chadbourne e Siegel (1980, p.26) relatam que cada *baseline* é uma oportunidade para validar o *software*, examinando sua integridade e verificando se as requisições foram atendidas. A Figura 2.3 demonstra a associação do ciclo de vida e seus respectivos *baselines*.

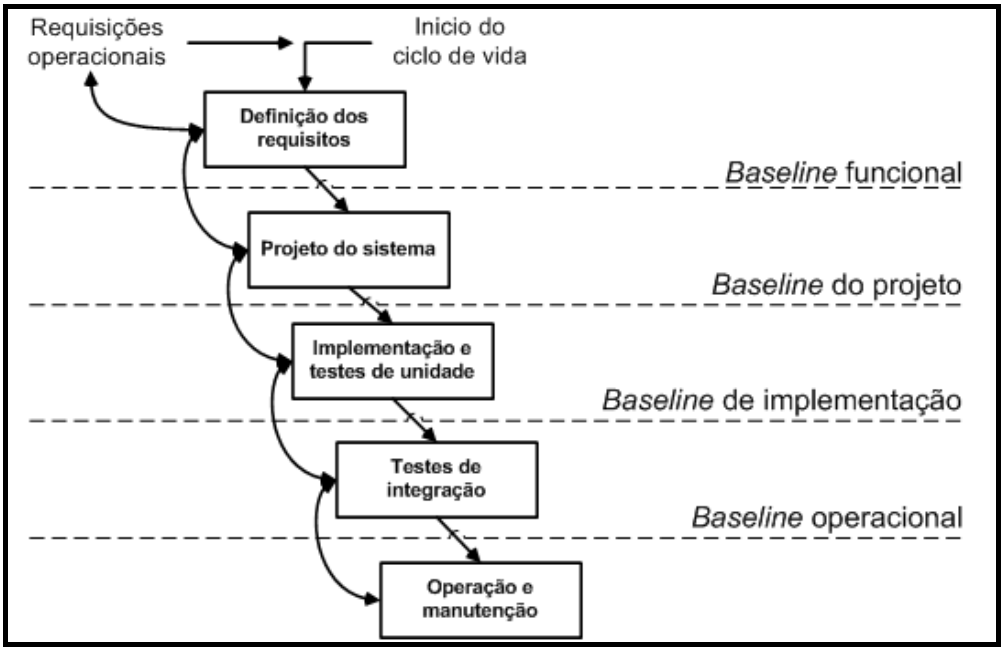

Figura 2.3 - Adaptação da representação gráfica do ciclo de vida associado com os *baselines*. (BRYAN; CHADBOURNE; SIEGEL, 1980, p.26)

Pressman (2005, p.743) define que um item de configuração, antes de passar por um baseline, pode ser alterado rapidamente e informalmente. No entanto, o baseline é como um marco para o item de configuração. Novas alterações podem ser realizadas, mas especificações e procedimentos formais devem ser aplicados na evolução e verificação de cada alteração.

2.1.5 Papéis e responsabilidades dos profissionais

O ITIL (2004, p.84 e 96) define os principais personagens para a GCS.

- O **gerente de configuração** ou **CCM** (Conselho de Controle de Modificações) determinará o escopo e grau de detalhamento necessários para a GCS, implementará os procedimentos para interação com os outros processos e times e garantirá que somente requisições aprovadas sejam implementadas.
- O ***librarian* da configuração** é a pessoa que controla o acesso às cópias do *software* e da documentação. Como ocorre com qualquer outro *librarian*, o foco está nos itens físicos que deverão ser armazenados na DSL (*definitive software library*).
- O **gerente de versões** é responsável pela definição e manutenção da política de versões. Ele efetua o processo de certificação para evitar conflitos entre sistemas no ambiente produtivo. Deve, portanto, ter um perfil técnico, com um bom conhecimento nas ferramentas de suporte que serão utilizadas. A equipe responsável por gerenciar as

versões deve receber treinamento adequado para manutenção de *software* englobando desenvolvimento e manutenção de software e *hardware*.

Outros perfis encontrados no processo de GCS:

- O **solicitante da alteração** é o responsável pela solicitação inicial. A solicitação pode ser gerada por qualquer pessoa que tenha necessidade de alteração, na grande maioria das vezes, o solicitante será o próprio usuário do software. O solicitante também será a pessoa responsável pela aprovação dos testes no ambiente de homologação e produção.
- Os **grupos envolvidos (*Assignee*)** são compostos de personagens que fazem parte do processo de alteração e, em algum momento do processo, serão envolvidos para executar uma atividade. O gerente de configuração é o responsável por determinar quais serão os grupos necessários para a realização da alteração. Cada grupo envolvido deve aprovar a alteração antes do seu início.
- O **CCM** (Conselho de Controle de Modificações) pode ser composto por um analista ou por um grupo deles, responsáveis por analisarem as requisições e, dependendo do risco e complexidade da alteração, envolverem os grupos necessários. (IEEE Std 828, 2005, p.8) (SOMMERVILLE, 2007, p.697) (SEI-CMMI, 2006, p.ii). O objetivo do CCM é o de estabelecer mecanismos que ajudarão a garantir que cada versão tenha todos os elementos necessários para implementação e que todos os grupos irão trabalhar juntos. (KEYES, 2004, p.11). Após analisar as informações de uma requisição, o CCM pode aprovar ou rejeitar uma alteração. (IEEE-SWEBOK, 2004, p.7-7) As atividades do CCM incluem: Centralizar as informações das implementações para que não aconteça concorrência entre grupos envolvidos, validar se a alteração está aprovada antes de sua implementação e determinar o fluxo da GCS e de seus *baselines*.

2.1.6 Repositório

O repositório é usado para armazenar qualquer informação relevante à configuração do *software*. (SOMMERVILLE, 2007, p.694) e (LEON, 2005, p.69) As principais informações que devem ser registradas no repositório, segundo ITIL, (2004, p.88) são:

- Informações sobre os itens de configuração. Conforme definido no capítulo *Itens de configuração*, as principais informações que devem ser guardadas são as definições do

produto com as informações operacionais e o relacionamento com outros itens de configuração.

- A lista de itens de configuração afetados por uma determinada alteração. Com este relacionamento, é possível evitar problemas com o andamento do fluxo. Por exemplo: Se a versão do *software* for alterada, as aprovações informadas até aquele momento serão desconsideradas, forçando o envio de novas aprovações.
- Todas as requisições de alterações relacionadas a um item de configuração. Serve como forma de controle para que atividades como aprovações sejam exigidas novamente.
- Lista de alterações e problemas associados a um item de configuração. Servirá de base para a geração de métricas.
- Lista dos itens de configurações afetados por um problema. Assim como no tópico anterior, as informações, aqui, também servirão de base para a geração de métricas.

Em ambientes modernos, o repositório da GCS é parte de um sistema e os detalhes dos itens de configuração são gravados automaticamente. Caso o processo seja manual, os detalhes dos itens de configuração devem ser inseridos manualmente no sistema de GCS. Os passos para inserção de um item de configuração não devem ser ignorados, mas se um item de configuração for acrescentado sem uma entrada no repositório, então a integridade e a utilidade do repositório estarão perdidas. (LEON, 2005, p.69)

Pressman (2005, p.746) define as diretrizes para o repositório:

- Integridade das funções: valida os dados de entrada, garante consistência no relacionamento dos objetos e, automaticamente, promove modificações em cascata quando uma alteração de um objeto acontece.
- Informação compartilhada: permite que múltiplos desenvolvedores com múltiplas ferramentas gerenciem e controlem múltiplos acessos aos dados, e bloqueando ou desbloqueando os objetos de uma manutenção, sem interferir em outras manutenções.
- Ferramenta de integração: estabelece o modelo de dados que pode ser acessado por ferramentas externas de engenharia de *software*.
- Integração dos dados: permite que várias tarefas possam ser conduzidas por um ou mais itens de configuração.
- Metodologia de execução: define o relacionamento entre os objetos gravados no repositório, implicando um processo específico de engenharia de *software*.

- Documentos padrões: permitem que definições criadas no repositório estejam diretamente ligadas aos padrões de documentos da engenharia de *software*.

Uma característica que o repositório deve possuir é a de controlar entrada e saída de arquivos na forma de *check-in* e *checkout*. Os métodos de *check-in* e *checkout* é a maneira de manter os dados corretos e íntegros (SEI-CMMI, 2006, p.122). Abaixo, a Figura 2.4 representa graficamente os métodos de *check-in* e *checkout*.

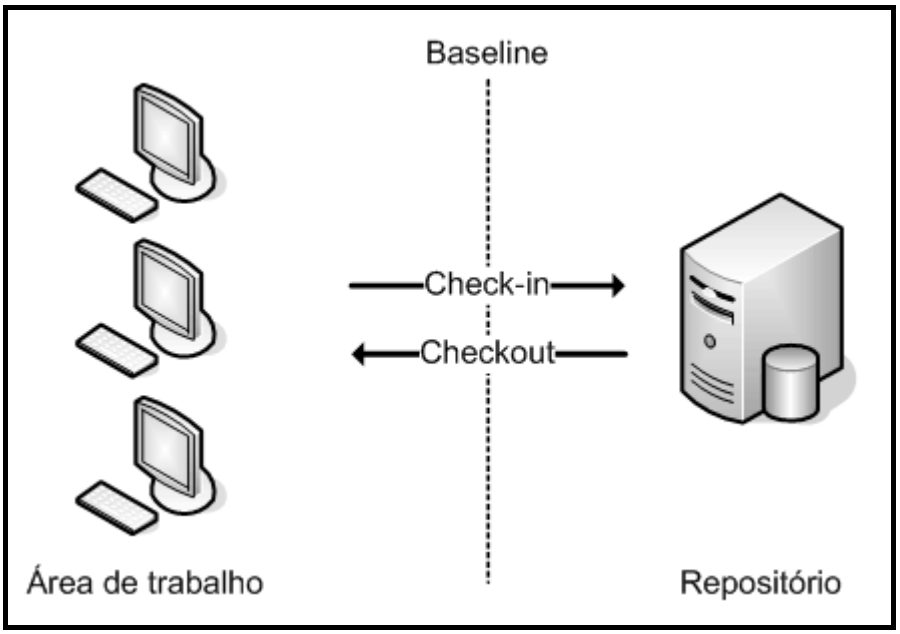

Figura 2.4 - Conceito de *check-in* e *checkout*. (PACHECO e SANCHES, 1997, p.7)

Usando o conceito de *check-in* e *checkout* no repositório, é possível garantir que as revisões e atualizações sejam registradas corretamente e que o arquivamento e a recuperação de *baselines* sejam feitos de forma adequada. Para realizar uma alteração é necessário realizar o *checkout* e após a alteração, faz-se necessário realizar o *check-in*, gerando assim uma versão nova. (MEI, 2001, p.2)

As informações armazenadas no repositório devem disponibilizar:

- Quais usuários utilizam o *software*;
- Qual é o *hardware* e o sistema operacional necessário para executar determinada versão;
- Qual é o número de versões com suas respectivas datas de criação;
- Quais são os componentes alterados e quantos problemas reportados existem em cada versão.

ITIL (2004, p.91) define DHS (*Definitive Hardware Store*) e DSL (*Definitive Software Library*) como ambientes físicos e seguros para o armazenamento do *hardware* e *software* aprovados.

Preferencialmente, o repositório com as informações das configurações poderia ser integrado ao sistema de gerenciamento de versões, para guardar e manipular todos os documentos e versões do sistema.

2.1.7 Ferramentas de GCS

Com o número de projetos crescendo e se tornando mais complexos, as ferramentas automatizadas para sustentação do GCS tornam-se cada vez mais importantes. (IEEE-SWEBOK, 2004, p.7-4).

Baseado no IEEE-SWEBOK (2004, p.7-4), as características principais de uma ferramenta de GCS são:

- Biblioteca de GCS. A biblioteca do GCS, também chamada de repositório, é o local onde ficarão as versões do *software*.
- Procedimentos para requisições e aprovações. Trata-se de uma forma para controlar as novas requisições e suas respectivas aprovações. Funciona coma um procedimento formal, controlado pela ferramenta de GCS.
- Relatórios, métricas e *status*. Como suporte para as atividades gerenciais é necessário que a ferramenta disponibilize relatórios, retratando o andamento das alterações.
- Relatórios e procedimentos de auditoria. Essa ferramenta será a principal fonte de informações para o processo de auditoria. Também é de responsabilidade dessa ferramenta garantir que procedimentos críticos estejam devidamente preenchidos antes de prosseguir com o fluxo de alteração.
- Rastreabilidade e controle de versões: Essa ferramenta de GCS deve permitir que todos os itens de configurações (*scripts, softwares*, documentos, entre outros) sejam passíveis de rastreabilidade.

2.1.8 Procedimentos

Os procedimentos da GCS são os responsáveis por garantir que os custos e benefícios de uma alteração sejam propriamente analisados e produzidos em um ambiente controlado.

GEIA (2004, cap.5.3) define os procedimentos da GCS como:

- Identificar a necessidade de uma alteração. Tal atividade serve para criar uma ligação entre as necessidades de negócio e a alteração que está sendo conduzida.
- Definir a alteração. Uma vez identificada a necessidade de negócio, o próximo passo é o de explicar o que foi feito para atender àquela necessidade. Esta informação, portanto, é uma síntese do que foi realizado na alteração.
- Documentar o impacto da alteração. Com base nas alterações realizadas no *software* e no risco calculado, é registrado o impacto da alteração para a instituição.
- Evoluir e coordenar a proposta de alteração (incluindo as aprovações e reprovações). Envolver os times necessários para execução da implementação, registrar as aprovações e garantir que o fluxo da GCS esteja sendo seguido é um dos pré-requisitos para uma boa implementação de *software*.
- Incorporar as alterações aprovadas em um produto e relacioná-las com as informações de GCS. O objetivo deste procedimento é unificar diversas ações relacionadas à alteração, transformando tudo isso em uma única ação.
- Verificar a incorporação da alteração e continuar com os testes de acordo com as informações definidas para a GCS. É vital garantir o correto funcionamento do *software*, portanto este procedimento visa registrar os testes realizados durante a homologação.
- Identificar, documentar, aprovar e implementar variações oriundas do *baseline* e requisições do produto.

GEIA (2004, cap.5.3) demonstra que as definições dos procedimentos acima servem para garantir que:

- Os *baselines* de configurações sejam mantidos e controlados.
- O produto e as informações de configuração do produto estejam consistentes.
- As informações de configuração estejam apresentadas de maneira ordenada.
- Os custos, economias e as modificações sejam avaliados.
- As decisões para as alterações devem ser baseadas no conhecimento de impacto.
- Usuários afetados devem ser considerados nas alterações.
- Interfaces de produtos sejam controladas.
- Variações sejam documentadas, controladas e que os produtos sejam suportados após a alteração.

O ITIL (2004, p.93) descreve as atividades ligadas à GCS conforme Figura 2.5.

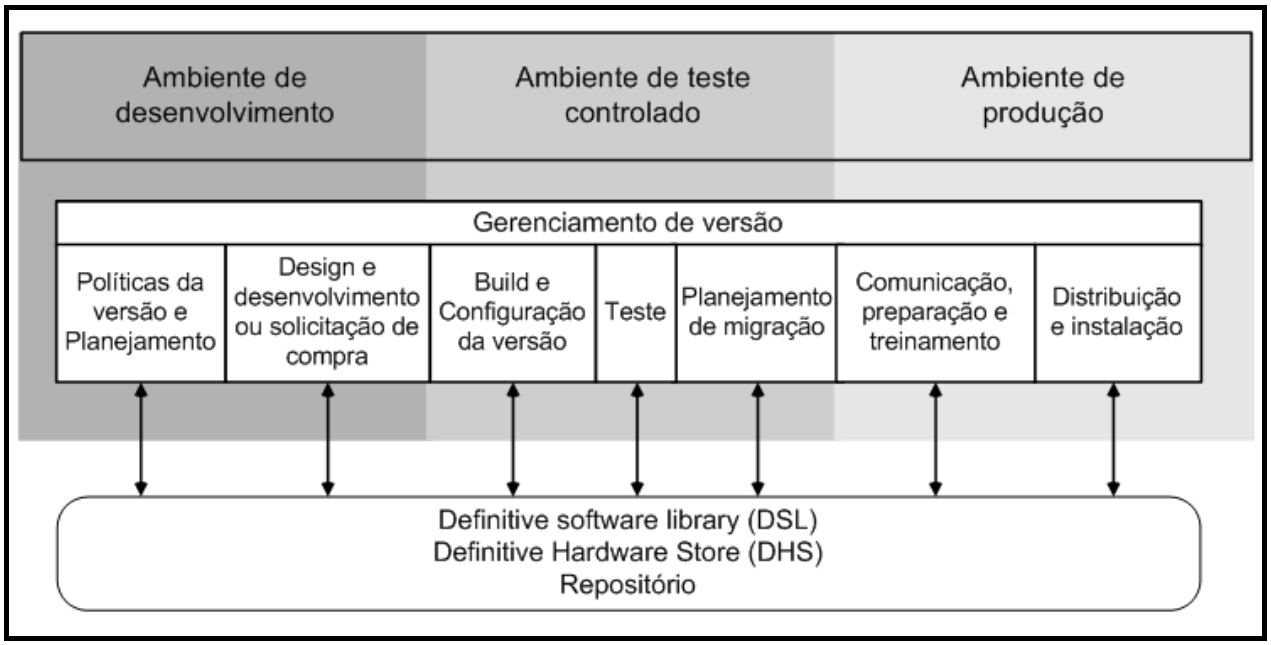

Figura 2.5 - Representação gráfica das atividades da GCS. (ITIL, 2004, p.93)

- **Política da versão** é um documento que descreve como a organização utilizará a nova versão de *hardware* ou *software* na infra-estrutura. Esta política deve conter, também, a freqüência em que as versões serão aprovadas pela área de negócio, a maneira como proceder com as versões emergenciais, a política de teste, qual o grau de controle necessário e nomes padrões para os *releases*. O **planejamento** é a chave para qualquer melhora no processo. As principais informações do planejamento são: conteúdo da versão, cronograma, pré-requisitos da versão, regras e responsabilidades, definição dos componentes do projeto, plano de contingência, plano de aceite.
- ***Design, build* e configuração de versões**. Esta fase é extramente técnica e todas as atividades neste estágio são conduzidas por um time de especialistas em um ambiente controlado. No final desta fase, o plano de *backout*, responsável por demonstrar como voltar à versão anterior após a instalação do *software*, deve estar pronto. Além do plano de *backout* nesta fase, também são criadas as instruções de instalação e plano de testes.
- **Teste**. A falta de testes adequados pode permitir que falhas no sistema aconteçam em ambiente produtivo. O representante do negócio deveria executar os testes nas funcionalidades esperadas, deixando as funções básicas para a equipe de testes.
- O **Planejamento de migração** consiste em adicionar informações no plano de instalação do *software* tais como: lista de recursos necessários para a execução de cada

tarefa; lista dos itens de configuração instalados; a documentação deve refletir as diferenças entre os ambientes utilizados, comunicar todos os envolvidos, realizar reuniões com a gerência e os grupos envolvidos com a nova versão. Caso tenha acontecido a aquisição de *software* ou *hardware* o plano de instalação incluirá os procedimentos a serem seguidos para a correta utilização.

- **Comunicação, preparação e treinamento**. É importante que haja a comunicação entre as partes envolvidas para o sucesso da nova versão. Isto implica várias reuniões e treinamentos para os times de tecnologia envolvidos. Os treinamentos devem ser planejados de acordo com a data de migração para o ambiente produtivo. A área de suporte será informada sobre a nova versão, para que, durante os testes, sejam estabelecidas as necessidades de suporte.
- **Distribuição e instalação**. O gerente de versões é responsável pela compra, transporte, entrega e guarda (licença de uso) da nova versão. **Distribuição e instalação** são atividades diferentes. A distribuição (quando *software*) pode ser feita de maneira *off-line* e a instalação é somente a ativação da versão.

2.1.9 Fluxo da GCS

O fluxo da Gerência de Configuração de *Software* apresenta o caminho que uma alteração percorre até a sua finalização. A seguir, é explicado cada passo encontrado na Figura 2.6.

Requisição para alteração: O fluxo da GCS é iniciado quando existe a necessidade de uma alteração.

É uma alteração emergencial?: Em função da criticidade, alterações emergenciais devem ser implementadas imediatamente. Em virtude disso, algumas atividades no fluxo da GCS serão desconsideradas.

Registrar requisição de alteração: Uma alteração não emergencial será registrada na ferramenta de GCS. As principais informações neste momento serão: o risco, impacto e complexidade da alteração.

Acionar os grupos envolvidos: Baseado nas informações de registro os grupos necessários são envolvidos. O envolvimento de novos grupos poderá ser feito a qualquer momento, sempre respeitando as necessidades das aprovações.

Preparação para alteração: Esta etapa consiste em uma preparação para as próximas etapas, pois existe uma interação entre todos os grupos para o andamento do fluxo.

Avaliação técnica: Baseado nas informações técnicas registradas na documentação e no resultados do testes técnicos é feita uma avaliação.

Avaliação de negócio: Baseada nas informações funcionais e nos testes realizados é feita uma avaliação sobre a alteração.

Data para implementação: Os times envolvidos na alteração devem analisar a data de implantação para evitar conflitos de agendas.

A alteração está aprovada?: Com base nas avaliações técnicas e de negócio, decide-se se a alteração será aprovada ou não.

Retorna a alteração?: Em caso positivo, a alteração irá retornará para o ponto de Preparação da alteração, caso contrário a requisição será fechada.

Aprovado verbalmente?: Esta etapa está ligada à decisão de implementar ou não uma alteração emergencial.

Implementação da alteração: Esta etapa se refere ao ato da alteração.

Alteração efetuada com sucesso?: Etapa para averiguar quais alterações foram executadas e se estão funcionando corretamente.

É necessário *Backout*?: Esta etapa serve para validar a necessidade de execução do plano de contingência e para voltar à versão anterior do software.

Executar o plano de *Backout*: Esta etapa se refere ao ato de voltar à versão anterior.

Existiu impacto para o usuário?: Esta pergunta irá definir a execução ou não do processo de Gerenciamento de problemas.

Foi uma alteração emergencial?: Se a alteração for emergencial, é necessário executar a etapa de Registrar alteração emergencial que deve acontecer até 24 horas após a sua execução. Tal registro na ferramenta de GCS serve para formalizar o processo e formalizar as aprovações fornecidas verbalmente.

Fechar requisição de alteração: Com a alteração executada e funcionando, o último passo é fechar o registro na ferramenta de GCS.

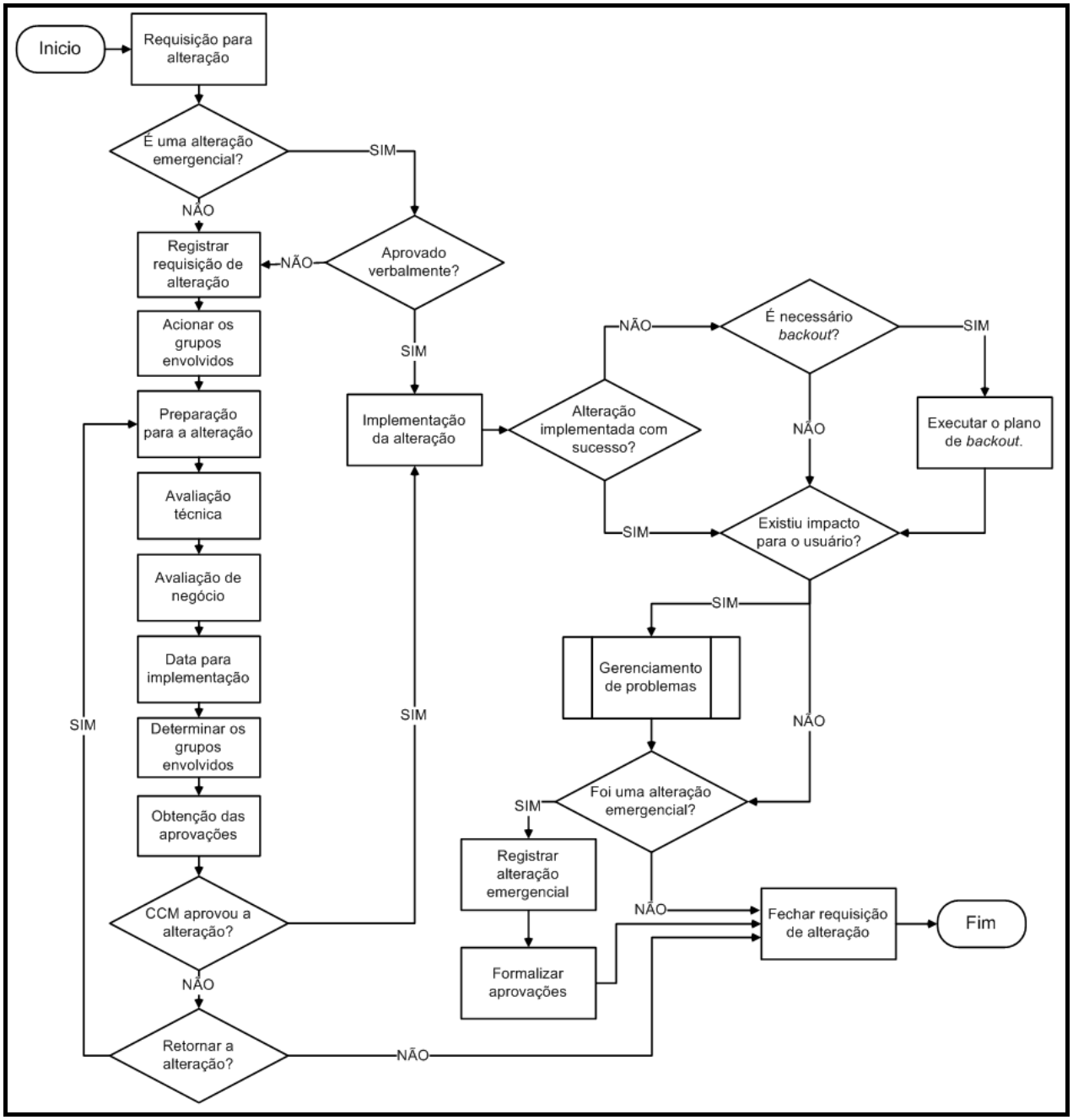

Figura 2.6 - Representação gráfica do fluxo de uma alteração. CMPE (2007).

2.1.10 Rastreabilidade e Controle de versões

Rastreabilidade é a atividade de registrar o histórico dos estados de um item de configuração. Qualquer alteração deve ser passível de rastreamento (ITIL, 2004, p.83). Todas as alterações de um sistema que estão sobre a GCS devem ser passíveis de controle e rastreamento (SEI-CMMI, 2006, p.121) (PING, 2007, p.2). A “versão” representa um estado de um item que deve ser registrado e restaurado, mantendo a coerência entre suas diferentes versões. (WESTFECHTEL, 2001, p.1)

O controle de uma alteração começa na sua requisição, momento em que é feita uma análise para determinar o impacto no produto, orçamento e cronograma. Todas as requisições devem ser inicialmente registradas no repositório. Durante a evolução da requisição, ela deve ser analisada e aprovada pelos responsáveis do projeto. Os documentos devem ser registrados no repositório, na medida em que são criados ou alterados. Além do registro no repositório, a rastreabilidade sobre os documentos, aprovações devem ser possíveis também.

Novas versões de um *software* podem possuir funcionalidades, melhora de desempenho ou correções. Algumas das alterações podem funcionar com um *hardware* ou *software* específico. Portanto, um controle sobre as versões facilitará a GCS. Existem três tipos básicos para identificar uma versão:

- A numérica: é aquela em que o componente recebe explicitamente um número único.
- O versionamento através de data: ocorre quando o *software* recebe a data para identificação. Uma forma usual de utilização é a de data invertida, onde o dia, mês a ano é invertido para ano, mês e dia. Por exemplo: 11/03/1979 transforma-se em 19790311. A troca para ano, mês e dia é comum para facilitar a ordenação dos arquivos quando necessário.
- A baseada em identificação: é aquela em que a versão recebe um nome, associado ao conjunto de alterações.
- A identificação orientada à requisição: é aquela atrelada a um sistema que atribui o nome baseado na identificação associado a uma ou mais requisições de alterações.

O ITIL (2004, p.99) divide as versões em três tipos:

- Versão delta. Incluem somente os itens de configuração que realmente foram alterados.
- Versão completa. Na versão completa, todos os componentes são atualizados.
- Pacote completo. O pacote completo inclui todas as versões deltas e completas. Este tipo de atualização geralmente é implementada com menor freqüência, portanto levam em conta períodos mais longos de estabilidade no ambiente produtivo.

2.1.11 Tipos de manutenções

Segundo IEEE Std 610.12-1990 (1990, p.46), manutenção é: A modificação de um produto de software, depois de liberado, para corrigir falhas, melhorar o desempenho ou outros atributos, ou para adaptar o produto a um ambiente alterado.

Categorizar uma manutenção é útil para organizar e priorizar as requisições dos usuários (KEYES, 2004, p.186). Ao priorizar e padronizar as requisições do *software*, é possível seguir fluxos diferentes na GCS, fazendo com que o processo esteja de acordo com as necessidades do sistema.

2.1.11.1 Manutenção corretiva

Manutenção corretiva é a correção do *software* no momento em que ocorre uma falha ou no momento em que o sistema não faz o que está originalmente especificado na requisição (PARIKH, 1986, p.51). A correção corretiva é o tipo mais familiar para os usuários, pois geralmente são as mais graves para eles (KEYES, 2004, p.186). Correções corretivas geralmente recebem prioridade máxima, podendo paralisar a organização, se não forem identificadas e corrigidas. Manutenções corretivas consomem 14% do esforço geral das alterações de um sistema (BASILLI, 1996, p.469). As principais habilidades requeridas para manutenções corretivas são:

- Boa habilidade de diagnóstico.
- Boa habilidade para testes.
- Boa habilidade para documentação.

2.1.11.2 Manutenção adaptativa

Manutenções adaptativas são alterações de adaptação a alterações externas, como alterações de sistema operacional, regulamentações do governo, alterações de *hardware*, alterações de compiladores entre outros (PARIKH, 1986, p.51). Manutenções adaptativas são, tipicamente, parte de uma nova versão, as quais integram um grande esforço de desenvolvimento. Aproximadamente 5% das alterações são destinadas às manutenções adaptativas (BASILLI, 1996, p.469).

2.1.11.3 Manutenção evolutiva

Manutenção evolutiva é o ato de melhorar as funcionalidades do software conforme requisição dos usuários, melhorando, assim, a eficiência e produtividade de suas atividades (PARIKH, 1986, p.51; KEYES, 2004, p.187). Alterações evolutivas incluem:

- Adição de funcionalidades.

- Melhora de desempenho.
- Melhora na manutenção do *software*.

Manutenções evolutivas são as grandes consumidoras do esforço de alterações. Aproximadamente 61% do tempo das alterações são gastos em alterações evolutivas (BASILLI, 1996, p.469).

2.1.11.4 Manutenção preventiva

Manutenções preventivas são alterações para prevenir problemas futuros no *software* (KEYES, 2004, p.187). A diferença entre manutenções preventivas e manutenções corretivas é que, nas manutenções preventivas, a falha foi detectada, mas ainda não se manifesta.

2.1.11.5 Manutenção emergencial

Se não for possível executar a manutenção no tempo mínimo, conforme demonstrado no capítulo *Calculando o risco de uma alteração*, ela será considerada como uma manutenção excepcional, podendo ser classificada como emergencial.

Uma manutenção emergencial ocorre quando a alteração é vital para a continuidade dos negócios da empresa, mas não pode seguir os caminhos normais de uma alteração. Exemplo disso ocorre quando existe um problema e suas conseqüências podem acarretar perdas financeiras ou de imagem da empresa perante o mercado. Nestes casos, a prioridade para solução do problema inibirá algumas atividades do fluxo da GCS. O processo de aprovação é o único que nunca poderá ser substituído, mas no caso de manutenções emergenciais, as aprovações podem ser dadas verbalmente, a fim de se resolverem os problemas rapidamente.

2.1.12 Métricas de Gerência de Configuração de Software

As métricas de GCS podem ser designadas para uma informação específica envolvendo um produto ou para proporcionar uma remodelagem no processo interno de GCS elaboradas durante o planejamento (IEEE-SWEBOK, 2004, p.7-5). Elas permitem entender e gerenciar o processo, bem como medir o impacto de quaisquer de suas alterações. Ex: novos métodos, treinamentos, entre outros (KEYES, 2004, p.240).

Métricas são usadas para medir o progresso do projeto, a qualidade do produto, o esforço necessário e para disponibilizar uma base de conhecimento para comparações futuras. ITIL

(2004, p.57) define as principais métricas para o processo de *software* como: número de incidentes por período, número de incidentes por categoria, número de incidentes por nível de prioridade, tempo gasto para solução, número de incidentes fechados por período.
Em ITIL (2004, p.87), também são definidas as principais métricas para a GCS.

- Número de itens de configuração não autorizados e não utilizados.
- Número de alterações com informações incorretas que causaram incidentes ou problemas.
- Solicitações de alterações não completadas com sucesso, em função de um controle de versão pobre ou informações incorretas.
- Tempo necessário gasto nas alterações do início ao fim.
- Licença de uso para *software* que foi perdida ou que não foi utilizada.

Se a recorrência das falhas é alta, o processo de definição da requisição deve ser avaliado para aumentar a clareza e a precisão dos requisitos. Com a existência do desenvolvimento de *software* de forma terceirizada, outra métrica possível é a recorrência de falhas, por produtos, de uma mesma fábrica de *software*. Desta forma, é possível avaliar a qualidade de desenvolvimento, por fábricas, que possa auxiliar no momento de uma nova contratação.

2.1.13 Riscos de alteração

As informações deste item foram baseadas no curso CMPE (2007). Existem cinco principais níveis para classificar o risco de uma alteração: crítica, alta, média, baixa e habitual. O risco de uma alteração é determinado por diversos fatores, entre eles: complexidade na implementação da alteração, dificuldade no processo de *backout*, número de usuários afetados, a quantidade de documentação que deve ser atualizada, o número de aplicativos afetados e quais deles são críticos para a instituição. O risco da alteração também determinará o tempo de que a requisição necessita ficar no repositório antes de sua execução. A seguir, segue uma descrição dos níveis de risco para uma alteração:

- **Crítica** Alterações críticas têm um grande fator de risco e também impactos críticos para a instituição, em caso de falhas. Estas alterações, normalmente, requerem: planejamento, cronograma e coordenação das atividades entre os grupos. Adicionalmente, essas alterações devem ser programadas, sempre que possível, em

períodos de tempo prolongados: janelas técnicas, finais de semana, feriados, entre outros.

- **Alta** Alterações com risco alto têm um fator significativo para a instituição em caso de falhas. Tal alteração também exige planejamento, cronograma e coordenação com os grupos envolvidos. É sugerido que alterações de alto risco sejam executadas em período de tempo prolongado.
- **Média** Alterações de médio risco têm um impacto mínimo para a instituição em caso de falhas. Exigem, contudo, coordenação de planejamento, cronograma e coordenação com os grupos envolvidos, a fim de que se evitem transtornos quaisquer na rotina da empresa.
- **Baixa** Alterações de risco baixo não apresentam risco para a instituição em caso de falhas. Planejamento, cronograma e coordenação da atividade dentro de um grupo podem assegurar a implementação correta da alteração.
- **Habitual** Alterações com este nível de risco podem fazer parte de atividades em andamento que acontecem dia após dia. A solicitação deve ser avaliada e aprovada para que ela se torne uma alteração habitual. Por exemplo: Atividades de backup ou do CPD (Centro de processamento de dados) que são diárias podem ser classificadas como habitual.

2.1.13.1 Calculando o risco de uma alteração

Neste capítulo, será apresentado um método para calcular o risco de uma alteração. Cada tabela contém os valores pré-determinados que depois de classificados serão utilizados para calcular o risco. As informações deste item foram baseadas no curso CMPE (2007).

1. O primeiro passo é determinar o número de usuários impactados pela nova implantação. Por exemplo: Uma alteração no editor de textos da empresa vai afetar um grande número de usuários, pois se trata de um software padrão. Uma alteração em um sistema utilizado apenas pelo departamento de recursos humanos, afetará um número menor de usuários. De acordo com os números levantados, ir-se-á classificar a alteração, conforme Quadro 2.1.

Número de usuários impactados	Classificação
> 150	5
50 – 150	4

Número de usuários impactados	Classificação
2 – 49	3
0 – 1	2
0	1

Quadro 2.1 – Número de usuários impactados versus classificação.

2. Neste passo, será determinado o número de grupos de suporte envolvidos para implementação da alteração. De acordo com o número de grupos, é feita a classificação, conforme Quadro 2.2.

Número de grupos envolvidos	Classificação
3 ou mais grupos	4
2 grupos de suporte	3
Mais de 1 pessoa, mesmo grupo	2
1 pessoa	1

Quadro 2.2 – Número de grupos envolvidos na implementação.

3. Neste passo, será determinado o esforço de preparação. O critério utilizado é do tempo necessário para preparação da implementação, conforme demonstrado no Quadro 2.3.

Dias de esforço	Classificação
6 dias ou mais	5
2 a 5 dias	4
1 dia	3
Menor que 1 dia	2
1 hora ou menos	1

Quadro 2.3 – Tempo necessário para preparação da implementação.

4. Duração da implementação. Este passo classifica o tempo necessário para execução da implementação. O Quadro 2.3 demonstra esta classificação.

Tempo necessário para realizar a implementação.	Classificação
> 2 horas ou que solicite um horário especial de janela de	5

Tempo necessário para realizar a implementação.	Classificação
manutenção.	
1 a 2 horas (Em janela de manutenção já prevista)	4
< 1 hora (Em janela de manutenção já prevista)	3
< 1 hora	2
Implementação imediata	1

Quadro 2.4 – Tempo necessário para executar a implementação.

5. O quinto passo é classificar o tempo necessário para execução do plano de contingência, voltando a versão anterior do software em caso de necessidade. O Quadro 2.5 demonstra esta classificação.

Tempo necessário para realizar *backout* da versão.	Classificação
> 2 horas	5
entre 1 ou 2 horas	4
1 hora	3
< 1 hora	2
Volta imediata	1

Quadro 2.5 – Tempo necessário para efetuar o *backout* da versão.

A somatória de todas as classificações será novamente classificada, determinando o risco da alteração, conforme é demonstrado no Quadro 2.6.

Total	Total
Crítica	> 20
Alta	Entre 17 e 19
Média	Entre 13 e 16
Baixa	Entre 9 – 12
Habitual	Entre 5 e 8

Quadro 2.6 – Criticidade de uma implementação.

Com base na criticidade de uma alteração, é possível, também, determinar o tempo mínimo para retenção dos seus dados.

Risco	Dias de retenção
Crítica	14
Alta	8
Média	2
Baixa	1
Habitual	0

Quadro 2.7 – Dias para retenção dos dados de uma implementação.

Os dias mínimos para retenção dos dados são estabelecidos para permitir que todas as partes envolvidas no planejamento e a revisão da alteração tenham tempo suficiente para assegurar o sucesso da implementação, minimizando o impacto inesperado em serviços fornecidos pela instituição. O tempo de retenção deverá ser negociado e comunicado a todas as partes envolvidas na alteração.

2.1.14 Auditoria

A auditoria de *software* é uma atividade realizada para avaliar, de forma independente, a conformidade dos produtos de *software* e processos, para aplicar padrões, planos, guias e procedimentos (IEEE Std 1028-1997). As auditorias na GCS devem confirmar que os registros de gerência e os itens de configuração estejam completos, consistentes e exatos (SEI-CMMI, 2006, p.124) (LEON, 2005, p.147).

ITIL (2004, p.84) define as principais atividades de auditoria para o processo de configuração:

- Em uma auditoria regular, deve ser possível verificar todos os itens de configuração registrados corretamente no repositório;
- A primeira auditoria deve ser conduzida após a implementação do repositório, a fim de garantir a correta representação da infra-estrutura;
- O processo de auditoria pode ser conduzido após desastres e/ou grandes alterações, seguindo um horário pré-determinado.
- Há que se avaliar o que será atingido com cada auditoria. Auditorias parciais ou verificações específicas são válidas para extrair as informações do processo de GCS.

Auditorias de configuração fornecem meios para verificar que o esforço de desenvolvimento realizou todos os requisitos especificados nos *baselines*.

Leon (2005, p.149) divide o processo de auditoria em três partes: auditoria de configuração funcional, auditoria de configuração física, auditoria do sistema de GCS.

2.1.14.1 Auditoria de configuração funcional

O objetivo da auditoria de configuração funcional é o de verificar se os requisitos definidos na documentação dos itens de configuração foram atendidos. IEEE-SWEBOK (2004, p.7-2) define a auditoria funcional como uma auditoria para verificar se o item de configuração foi concluído de forma satisfatória e se seus documentos de suporte estão completos e satisfatórios.

A equipe de auditoria, durante a auditoria funcional, deve revisar os planos de testes, verificar a massa de teste utilizada e a metodologia usada, a fim de garantir que os parâmetros testados e seus resultados são satisfatórios.

2.1.14.2 Auditoria de configuração física

IEEE-SWEBOK (2004, p.7-2) define que a auditoria de configuração física deva garantir que o item de configuração foi construído conforme a documentação técnica que o define. A auditoria de configuração física, geralmente, é executada após a configuração funcional. A auditoria física demonstrará que os documentos de especificação, descrição da versão e características físicas estão consistentes e de acordo com o produto entregue. O time de auditoria, durante a auditoria física, examinará o documento do projeto com código fonte ou qualquer outro item que acompanhe a versão final do produto.

2.1.14.3 Auditoria do sistema de GCS

Leon (2005, p.150) define que a auditoria do sistema de GCS é realizada para assegurar que a implementação da GCS permaneça compatível com a política e procedimentos estabelecida pela instituição. A auditoria no sistema de GCS é essencial para garantir que os processos definidos sejam propriamente aplicados e controlados.

Os aspectos gerais considerados durante a auditoria de sistema são: o controle operacional do processo; a rastreabilidade das aprovações para a especificação e requisitos originais; a disponibilidade das aprovações dos documentos de especificação, descrição da versão e características físicas do produto.

2.2 Considerações finais

Neste capítulo, foram apresentados uma introdução da engenharia de *software* e os fundamentos teóricos da Gerência de Configuração de *Software*. Um estudo detalhado foi realizado para definir as atividades de gerência de configuração que constam nos modelos e normas descritos atualmente. O capítulo *Estudo de caso: aspectos metodológicos* apresenta o planejamento para execução do estudo de caso.

3 Estudo de caso: aspectos metodológicos

Neste capítulo, serão apresentados os fundamentos para realização do estudo. Em princípio, será apresentada a teoria da modalidade do estudo de caso, e, no capítulo seguinte, será detalhado o planejamento para execução dessa modalidade.

3.1 Estratégias de pesquisa

Yin (2005, p.23) indica três condições para se decidir a estratégia de pesquisa a ser utilizada:

- No tipo de questão de pesquisa proposta;
- Na extensão de controle que o pesquisador tem sobre os eventos comportamentais atuais;
- No grau de enfoque em acontecimentos contemporâneos em oposição a acontecimentos históricos;

Definir as questões é, provavelmente, o passo mais importante a ser considerado em um estudo de pesquisa (YIN, 2005, p.26). Com as perguntas definidas, com as condições sobre o controle dos eventos comportamentais e o grau de enfoque sobre acontecimentos contemporâneos é possível visualizar a estratégia de pesquisa a ser executada. O Quadro 3.1 demonstra cinco estratégias: experimento, levantamento, análise de arquivos, pesquisa histórica e estudo de caso.

Estratégia de pesquisa	**Forma de questão**	**Exige controle sobre eventos comportamentais**	**Focaliza acontecimentos contemporâneos**
Experimento	como, por quê	sim	sim
Levantamento	quem, o que, onde , quantos, quanto	não	sim
Análise de arquivos	quem, o que, onde quantos, quanto	não	sim / não
Pesquisa histórica	como, por quê	não	não
Estudo de caso	como, por quê	não	sim

Quadro 3.1 - Situações relevantes para diferentes estratégias de pesquisa. (YIN, 2005, p.24)

A coluna “Forma de questão” da tabela acima demonstra um esquema básico de categorização para os tipos de questões, representado pela série: “quem”, “o que”, “onde”, “como” e “por

quê". Após a categorização pela forma de questão são verificadas duas condições: grau de controle sobre os eventos e foco temporal (eventos contemporâneos versus fenômenos históricos).

3.2 O método de estudo de caso

O estudo de caso, segundo Yin (2005, p.20), é utilizado como estratégia de pesquisa para contribuir com o conhecimento que se tem dos fenômenos organizacionais, sociais, políticos, individuais e de grupo. No presente trabalho, o estudo de caso terá o objetivo de extrair informações para compreender o processo de Gerência de Configuração de *Software* em uma instituição financeira.

3.2.1 Tipos de estudo de caso

Yin (2005, p.24 e 37) considera que há tipos diferentes de estudos de caso. São eles: exploratório, descritivo e explanatório (causal). Abaixo, segue uma tabela, associando os tipos de estudo com os tipos de questões.

Tipo de estudo	**Forma de questão aplicável**
Causais ou explanatórias	Como, por quê.
Descritivos	Quem, onde, quantos, quanto.
Exploratórios	O que.

Quadro 3.2 - Tipos de estudo versus tipo de questão.

3.2.2 Estudos de casos únicos e múltiplos

Uma distinção básica que existe ao se projetarem estudos de caso são as diferenças entre estudos de caso único e de casos múltiplos (YIN, 2005, p.61).

O estudo de caso único é um projeto apropriado em várias circunstâncias. Cinco fundamentos lógicos sobre estudo de casos únicos são apresentados seguir.

- **Decisivo** - Encontra-se um fundamento lógico para um caso único, quando ele representa o caso decisivo ao testar uma teoria bem formulada. A teoria especificou um conjunto claro de proposições, assim como as circunstâncias nas quais se acredita que as proposições sejam verdadeiras. Para confirmar, contestar ou estender a teoria, deve existir um caso único que satisfaça todas as condições para comprovar a teoria.

- **Raro ou extremo** – É um acontecimento ou distúrbios tão raros que vale a pena documentar e analisar qualquer caso único.
- **Representativo ou típico** – O objetivo é capturar as circunstâncias e condições de uma situação lugar-comum ou dia-a-dia. Parte-se do princípio de que as lições que se aprendem desses casos fornecem muitas informações sobre as experiências da pessoa ou instituição usual.
- **Revelador** – Essa situação ocorre quando o pesquisador tem a oportunidade de observar e analisar um fenômeno previamente inacessível à investigação científica. Quando outros pesquisadores têm oportunidades semelhantes e podem desvendar alguns fenômenos predominantes previamente inacessíveis aos cientistas, as condições justificam a utilização de um estudo de caso único, tendo como base sua natureza reveladora.
- **Longitudinal** – Este fundamento lógico é baseado no estudo do mesmo caso único em dois ou mais pontos diferentes no tempo. A teoria de interesse provavelmente especificaria que certas condições mudam com o tempo, e os intervalos desejados de tempo a serem selecionados refletiriam os estágios presumidos nos quais as alterações devem se revelar.

As diferenças metodológicas entre os estudos de caso único e múltiplos são reveladas pelos diferentes fundamentos lógicos que subjazem à replicação, em oposição à lógica da amostragem.

- **Replicação** – A lógica da replicação é similar àquela utilizada em experimentos múltiplos. Por exemplo, após revelar uma descoberta significativa a partir de um experimento único, o objetivo imediato da pesquisa seria o de replicar essa descoberta, conduzindo um segundo, um terceiro, ou até mais experimentos. Algumas dessas replicações, talvez, tenham tentado duplicar as condições exatas do experimento original. Outras replicações podem ter alterado uma ou duas condições experimentais consideradas irrelevantes à descoberta original, para constatar se ela ainda poderia ser considerada. Somente com estas replicações é que a descoberta original seria considerada forte e digna de investigações adicionais.

Os estudos de casos podem ainda ser classificados como *holísticos* ou *incorporados*. Na *holística*, a *importância* é o fenômeno como um todo e não o destaque de cada um de seus

componentes, porém os casos *incorporados* enfatizam a importância de cada componente, processo ou etapa de um determinado fenômeno.

3.2.3 O protocolo do estudo de caso

Segundo Yin (2005, p.92), o protocolo é uma das táticas principais para aumentar a confiabilidade da pesquisa de estudo de caso e destina-se a orientar o pesquisador ao realizar a coleta dos dados.

Os componentes do protocolo são os seguintes:

- Uma visão geral do projeto do estudo de caso (objetivos e patrocínios do projeto, questões do estudo de caso e leituras importantes sobre o tópico que está sendo investigado);
- Procedimentos de campo (apresentação de credencias, acesso aos "locais" de estudo de caso, fontes gerais de informações e advertências de procedimento);
- Questões de estudo de caso (as questões específicas que o pesquisador do estudo de caso deve manter em mente ao coletar os dados, planilhas para disposição específica de dados e as fontes em potencial para informações ao se responder cada questão);
- Guia para o relatório do estudo de caso (esboço, formato para os dados, uso e apresentação de outras documentações e informações bibliográficas).

3.2.4 Fontes para coleta dos dados

Uma lista completa de fontes possíveis pode ser bastante extensa – incluindo filmes, fotografias e videoclipes, Contudo, Yin (2005, p.111) cita que as fontes mais comumente utilizadas ao se realizarem estudos de caso são documentações, registros em arquivos, entrevistas, observação direta, observação participante e artefatos físicos. O Quadro 3.3 demonstra uma visão geral sobre as seis fontes principais citadas por Yin (2005, p.113) com seus pontos fortes e fracos de forma comparativa.

Fonte de evidências	**Pontos fortes**	**Pontos fracos**
Documentação	• Estável. Os documentos podem ser revisados inúmeras vezes. • Discreta. Não foi criada como resultado do estudo de caso.	• Capacidade de recuperação pode ser baixa. • Seletividade tendenciosa, se a coleta não estiver completa.

Fonte de evidências	**Pontos fortes**	**Pontos fracos**
	• Exata. Contém nomes, referências e detalhes exatos de um evento. • Ampla cobertura. Longo espaço de tempo, muitos eventos e muitos ambientes distintos.	• Relato de vieses reflete as idéias preconcebidas (desconhecidas) do autor. • Acesso pode ser deliberadamente negado.
Registro em arquivos	• [Os mesmos mencionados para documentação]. • Precisos e quantitativos.	• [Os mesmos mencionados para documentação]. • Acessibilidade aos locais devido a razões particulares.
Entrevistas	• Direcionada – enfocam diretamente o tópico do estudo de caso. • Perceptivas – fornecem inferências causais percebidas.	• Direcionamento incorreto devido a questões mal-elaboradas. • Respostas indiretas. • Ocorrem imprecisões devido à memória fraca do entrevistado. • Reflexibilidade. O entrevistado dá ao entrevistador o que ele quer ouvir.
Observações diretas	• Realidade. Tratam de acontecimentos em tempo real. • Contextuais. Tratam do contexto do evento.	• Consomem muito tempo. • Seletividade. Salvo ampla cobertura. • Reflexibilidade. O acontecimento pode ocorrer de forma diferenciada porque está sendo observado. • Custo. Horas necessárias pelos observadores humanos.
Observação participante	• [Os mesmos mencionados para observação direta]. • Perceptiva em relação a comportamentos e razões interpessoais.	• [Os mesmos mencionados para observação direta]. • Direcionamento incorreto devido à manipulação dos eventos por parte do pesquisador.
Artefatos físicos	• Capacidade de percepção em relação a aspectos culturais. • Capacidade de percepção em relação a operações técnicas.	• Seletividade. • Disponibilidade.

Quadro 3.3 – Seis fontes de evidências: pontos fortes e pontos fracos. (YIN, 2005, p.113)

Os benefícios que se podem obter a partir dessas seis fontes de evidências podem ser maximizados se três princípios forem seguidos: utilizarem-se várias fontes de evidência, criar-se um banco de dados para o estudo de caso, manter-se o encadeamento de evidências (YIN, 2005, p.124).

- Várias fontes de evidências. Evidências provenientes de duas ou mais fontes devem ser utilizadas, desde que se dirigem ao mesmo conjunto de fatos ou descobertas.
- Um banco de dados para o estudo de caso. Uma reunião formal de evidências distintas a partir do relatório final do estudo de caso é pertinente.
- Um encadeamento de evidências. Para maior comprovação de lógica e veracidade, há que se estabelecer ligações explícitas entre os dados coletados e as conclusões a que se chegou.

3.2.5 Critérios de qualidade para os dados obtidos

Como se supõe que um projeto de pesquisa represente um conjunto lógico de proposições, também é possível julgar a qualidade de qualquer projeto com certos testes lógicos. Yin (2005, p.55) estabelece alguns critérios para garantir a qualidade dos dados obtidos conforme Quadro 3.4.

Testes de caso	**Tática do estudo**	**Fase da pesquisa na qual a tática deve ser aplicada.**
Validade do constructo	• Utiliza fontes múltiplas de evidências. • Estabelece encadeamento de evidências. • O rascunho do relatório do estudo de caso é revisado por um informante chave.	Coleta de dados Coleta de dados Composição
Validade interna	• Faz adequação ao padrão. • Faz construção da explanação. • Estuda explanações concorrentes. • Utiliza modelos lógicos.	Análise dos dados
Validade externa	• Utiliza teoria em estudos de caso único. • Utiliza lógica de replicação em estudos de casos múltiplos.	Projeto de pesquisa
Confiabilidade	• Utiliza protocolo de estudo de caso. • Desenvolve banco de dados para o estudo de caso.	Coleta de dados

Quadro 3.4 – Táticas do estudo de caso para quatro testes de projeto. (YIN, 2005, p.55)

- Validade do constructo: estabelece medidas operacionais corretas para os conceitos que estão sob estudo.
- Validade interna: (apenas para estudos explanatórios ou causais e não para estudos descritivos ou exploratórios): visa garantir a total abrangência e consistência da argumentação e da inferência concluída pelo pesquisador.
- Validade externa: estabelece o domínio até aos quais as descobertas de um estudo podem ser generalizadas.
- Confiabilidade: preza pelos cuidados com a repetição, fidedignidade e confirmabilidade, isto é, se outro pesquisador realizar o mesmo estudo e seguir os mesmos procedimentos, deve chegar às mesmas conclusões que o primeiro pesquisador.

4 Planejamento do estudo de caso

A aplicação do estudo de caso seguiu as etapas discutidas no capítulo anterior. A metodologia pode ser resumida conforme mostra o Quadro 4.1.

Atributo	**Especificação adotada**
Estratégias de pesquisa	Pesquisa histórica e estudo de caso
Tipo de pesquisa	Explanatória
Tipo de estudo de caso	Único incorporado com lógica representativa
Fontes de evidências	Documentação e entrevistas

Quadro 4.1 - Resumo das especificações metodológicas.

4.1 Protocolo do estudo de caso

A finalidade do protocolo é a de formalizar as atividades necessárias para coleta dos dados, garantindo a consistência das informações e rastreabilidade das atividades. Abaixo os seguintes itens que compõem o protocolo do estudo de caso:

- Objetivo do estudo de caso;
- Questão do estudo de caso;
- Proposições do estudo de caso;
- Os critérios para seleção das unidades de análise;
- Procedimentos de campo;
    - As regras da pesquisa na unidade de análise;
    - Questionário de pesquisa;
- Levantamento dos dados;
- Análise dos dados;
- Finalização.

4.2 Objetivo do estudo de caso

O objetivo do trabalho, conforme detalhado no capítulo um, é o de desenvolver um roteiro para Gerência de Configuração de *Software*. Para isso, os objetivos específicos do estudo de caso são: identificar os benefícios e as restrições do processo de GCS, implementado atualmente em uma instituição financeira; explorar os problemas encontrados pela unidade de análise e analisar quais foram as soluções adotadas para cada um deles.

4.3 Questão do estudo de caso

A pergunta elaborada para orientar o estudo de caso é a seguinte:

"Como é o processo de Gerência de Configuração de Software em uma instituição financeira?"

O experimento é explanatório, com pesquisas históricas, devido ao fato de que as informações lidam com questões operacionais que necessitam ser analisadas ao longo do tempo.
Ao definir a questão de pesquisa foram definidas também as limitações e a abrangência sobre o tema:

- Em relação ao processo de GCS, o estudo de caso foi o mais abrangente possível para contemplar todos os métodos, técnicas e atividades utilizadas.
- A unidade de análise foi escolhida, pois tem uma dependência tecnológica para realização de seus negócios.
- A organização deve depender de tecnologia para suprir as necessidades da área de negócio, mas o processo de desenvolvimento deve ser realizado fora de sua organização caracterizando o modelo da empresa que utiliza uma fábrica de *software* externa.

4.4 Proposições do estudo de caso

As proposições idealizam a tentativa de verificar a validade de resposta existente para o problema. As proposições elaboradas para este estudo de caso foram:

- *Os usuários de negócio estão envolvidos no processo de GCS e possuem responsabilidades.*

Esta proposição diz respeito ao grau de envolvimento dos usuários de negócio ao processo de GCS. Algumas diretrizes do SEI-CMMI (2006, p.127) definem que os *stakeholders* têm o objetivo de estabelecer *baselines*, revisar os relatórios gerados, avaliar o impacto das mudanças geradas pelos itens de configuração e revisar os resultados das auditorias do GCS.

Os usuários de negócio também fazem parte do grupo de *stakeholders*, pois são responsáveis pela criação de novas requisições e pelas aprovações das alterações até o ambiente produtivo, portanto, a consciência dessa responsabilidade é vital para o bom funcionamento do processo.

- *O software e os documentos criados estão armazenados em suas diversas versões.*

Uma das definições do relatório de GCS sugere que sejam guardadas todas as versões do *software* e os documentos pertinentes ao processo, para permitir a eventual consulta dos arquivos no futuro. O gerenciamento de versões permite a recuperação do sistema quando necessário (SOMMERVILLE, 2007, p.699). Além de permitir a consulta de versões anteriores dos documentos registrados, também é necessário identificar os responsáveis por cada alteração.

- *O processo de GCS fornece informações gerenciais sobre os documentos, aprovações e métricas registradas.*

As informações e métricas são necessárias para suportar o processo de GCS, fornecendo o status das necessidades gerenciais, de programação e atividades relacionadas (IEEE-SWEBOK, 2004, p.7-8).

4.5 Os critérios para seleção da unidade de análise

As unidades de análise de interesse neste estudo de caso são organizações que tenham um departamento de TI, mas cujo processo de desenvolvimento seja realizado por entidades externas.

Uma característica da unidade de análise usada na seleção é a de que o principal negócio da organização (*core business*) não esteja ligado à tecnologia, mas que a dependência de tecnologia seja vital para a sobrevivência da organização.

Outro critério para seleção da unidade de análise é o interesse da organização em um processo formal de GCS como forma de melhoria nos serviços prestados pela área de tecnologia.

Em todo o estudo de caso, garantiu-se o sigilo da identidade da organização e dos entrevistados.

4.6 Procedimentos de campo

Os procedimentos de campo detalham as ações e atividades do pesquisador para a execução do estudo de caso.

4.6.1 As regras da pesquisa na unidade de análise

Para facilitar as atividades na unidade de análise foram elaboradas algumas regras a serem executadas durante a aplicação da entrevista, revisão dos dados e confidencialidade das informações.

As entrevistas foram feitas após um contato formal com os entrevistados e foram realizadas com apenas uma pessoa por vez.

A forma de comunicação foi entrevista *in loco*.

O tempo estimado para cada entrevista foi de 30 minutos, com até 15 minutos de tolerância.

As observações e comentários adicionais foram anotados e entregues para revisão dos entrevistados, antes de serem registrados.

Qualquer opinião expressada pelos entrevistados será definida como opinião própria e não da unidade de análise.

4.6.2 Questionário de pesquisa

O modelo do instrumento de pesquisa utilizado, contendo os detalhes e comentários de cada informação, está descrito no *Apêndices*, deste trabalho. O questionário utilizado na entrevista conta com os seguintes grupos de questões:

- Apresentação do documento para os colaboradores da instituição, com a finalidade de demonstrar o escopo do trabalho e os termos de confidencialidade.

- Levantamento dos dados da empresa com a descrição dos negócios, atividade produtiva, organograma, perfil dos usuários, política de segurança e características inerentes ao processo de qualidade.

- Levantamento dos dados sobre a GCS e a coleta da documentação exigida pela instituição, elaboração dos fluxos do GCS para cada tipo de manutenção, apontamento dos processos de aprovação durante o fluxo, média do tempo necessário para

implantação de uma alteração, número de falhas por alteração, número de falhas por sistemas e por fábrica de *software*.

4.7 Levantamento dos dados

As estratégias utilizadas no estudo de caso para levantamento dos dados foram: consulta de documentação e entrevistas com os integrantes da unidade de análise.

Conforme demonstrado no capítulo anterior, a vantagem da pesquisa em documentos é a possibilidade da cobertura de eventos que já aconteceram, possibilitando a consulta dos nomes, referências e detalhes sobre cada evento. Referente às entrevistas, a vantagem é a possibilidade de direcionar o assunto diretamente ao tópico do estudo de caso, fornecendo conhecimentos empíricos sobre os eventos.

5 Aplicação do estudo de caso

Neste capítulo é apresentado o estudo de caso do processo de GCS desenvolvido em uma instituição financeira, seguindo as diretrizes definidas no capítulo anterior.

5.1 Unidade de análise

A unidade de análise de estudo é sede brasileira de uma instituição financeira, presente em mais de 100 países e com cerca de 200 milhões de clientes. No Brasil, sua participação é de 1,6 milhões de clientes, com sete mil colaboradores entre funcionários, estagiários e contratados.

Embora a base de clientes seja formada por pessoas físicas, sua especialidade é o gerenciamento de contas corporativas, mantendo uma fidelidade global em suas negociações com mega corporações.

O parque tecnológico segue padrões globais de ferramentas. O desenvolvimento de sistemas para gerenciamento dos negócios possui desenvolvimento local, regional e global.

- Sistemas locais são aqueles desenvolvidos por fábricas de *software* localizadas no país de origem.
- Sistemas regionais são os desenvolvidos para atender países do mesmo continente ou conglomerado financeiro.
- Sistemas globais atendem países em diferentes continentes ou conglomerados financeiros.

Os padrões globais também são aplicáveis na metodologia utilizada para garantir padrões mínimos de segurança, integridade e confiabilidade. O levantamento de dados para a dissertação foi feito no escritório de São Paulo com o gerente de configuração, com o *librarian* da configuração, com um usuário e um responsável pelo processo de alteração.

5.1.1 Políticas de segurança

A unidade de análise possui diversas políticas de segurança para manter a integridade, disponibilidade e confidencialidade dos dados da corporação.

- **Confidencialidade**: controlar o acesso às informações e manter a privacidade ou confidencialidade das informações, quando necessário.

- **Integridade**: serve para proteger as informações e os sistemas contra alterações não autorizadas, danos ou destruição.
- **Disponibilidade**: assegura que as informações estão continuamente disponíveis às pessoas que estão autorizadas a acessarem as mesmas.

Abaixo alguns métodos utilizados na instituição financeira para políticas de segurança:

- Rotinas regulares de backup. A realização periódica de backup fornece proteção contra a destruição inesperada de sistemas ou dados.
- Utilização de *software* antivírus. Os *softwares* de antivírus aprovados pela instituição detectam e removem vírus de computador antes que os mesmos sejam disseminados.
- Controle de alterações. Os procedimentos de alterações ajudam a instituição a controlar quais alterações são realizadas nos sistemas e por quem.
- Plano de continuidade de negócios. Existe um planejamento para recuperação de desastre, de forma que a instituição seja capaz de continuar os negócios, normalmente, em caso de falha grave dos sistemas ou de outros desastres.
- Segurança 24 horas. Um edifício com segurança 24 horas significa que apenas pessoas autorizadas a transitarem pelo espaço têm acesso às instalações de trabalho em todos os momentos, sendo restrito o acesso para quaisquer outras pessoas.
- Classificação da informação. Os sistemas de classificação ajudam a assegurar que todas as informações estejam classificadas de acordo com sua “sensibilidade” e que o acesso seja controlado conforme necessário.
- Crachá de acesso. O crachá de acesso permite o acesso de pessoas autorizadas para às áreas protegidas, tornando restrito o acesso das pessoas não autorizadas.
- O controle de senha. O controle de senha para acesso aos sistemas ajuda a gerenciar quem pode acessar e modificar os dados contidos no sistema.

Baseado nos métodos de segurança da unidade de análise, é possível observar que a Gerência de Configuração de *Software* contribuirá para o controle de alterações e integridade do plano de continuidade de negócios.

5.2 Ambientes

A unidade de análise tem como prática disponibilizar, para cada aplicação, quatro ambientes: desenvolvimento, homologação, produção e COB. Durante o estudo teórico foram observados

3 ambientes, mas a nova abordagem encontrada na unidade de análise com o ambiente de COB refletirá mudanças no fluxo final.

Para sistemas com desenvolvimento local, o ambiente de desenvolvimento não é disponibilizado, visto que os testes iniciais são realizados pela própria empresa responsável pelo desenvolvimento do *software*. Para sistemas regionais ou globais, existe um ambiente de desenvolvimento disponível na corporação, mas ele não é suportado pela equipe de TI da subsidiária brasileira.

O ambiente de homologação é disponibilizado para todos os sistemas e possui as mesmas regras e procedimentos já utilizados no ambiente de produção.

Uma prática observada na unidade de análise é a fusão de dois ou mais sistemas com as mesmas características no ambiente de homologação. Apesar de afetar diretamente a integridade do ambiente, esta prática é utilizada para diminuir custos de manutenção do ambiente.

O ambiente de produção é controlado e as alterações são testadas no ambiente de homologação antes de serem implementadas. Ligado ao ambiente produtivo, existe o ambiente de COB (*Continuity of Business*). O ambiente de COB é preparado e mantido para servir de contingência em caso de desastres no ambiente produtivo. O ambiente de COB é geograficamente separado do ambiente de produção e as alterações são executadas de forma equivalente. Após 24 horas úteis da implementação em produção, é realizada a implementação das alterações no ambiente do COB. Os testes realizados, após a implementação no COB, são realizados pela equipe de TI e duas vezes por ano é realizado o exercício de DR (Desastre e Recuperação) com a participação do usuário. O exercício de DR consistente em migrar todo o ambiente produtivo para o ambiente de COB dentro de um prazo pré-estabelecido.

5.3 Ferramentas

A unidade de análise, buscando maior segurança nos processos ligados a TI, no final de 2002, iniciou o processo de implantação da GCS. A implantação aconteceu em toda a América Latina e Caribe, como estratégia para a região. As ferramentas e procedimentos utilizados eram padronizados, mas o país de origem poderia incluir novas funcionalidades caso fosse necessário.

- A ferramenta de GCS utilizada pela unidade de análise foi implementada em toda a América Latina e Caribe. A estratégia da corporação era dar maior controle e eficiência às atividades realizadas nestes paises. O desenvolvimento da ferramenta foi interno e a implementação foi acompanhada de treinamento e palestras para que os procedimentos da ferramenta fossem utilizados corretamente. Os relatórios são listas das implementações realizadas, podendo incluir filtros como: nome de abertura, nome do sistema, por área, data de fechamento, entre outros.

- Para gerenciar as versões dos arquivos compilados do *software* e da documentação do *software*, a unidade de análise utiliza uma ferramenta denominada PVCS Gerenciador de versões. Desenvolvida pela empresa Serena, o objetivo da ferramenta é o de gerenciar a versão de qualquer arquivo, desde o ambiente de desenvolvimento. Embora a ferramenta tenha uma integração com ferramentas de desenvolvimento como Visual Studio.NET e o ambiente Eclipse, ela é exclusivamente utilizada para repositório e controle de versões.

A integração entre as ferramentas de GCS e O versionamento é feita de forma manual pelo *librarian* da configuração. O *librarian* realiza o *download* diretamente do site da empresa, a qual desenvolve o *software* ou pode recebê-la através de correio eletrônico. Os meios físicos como CD, DVD são pouco utilizados ficando quase que exclusivamente para versões inicias na fase de projeto, momento em que o número de objetos é maior. Nesta pesquisa não será abordado quais são as melhores ferramentas para Gerência de Configuração de *Software*, mas processos definidos na ferramenta da unidade de analise serão incluídos como passos a serem cumpridos no fluxo proposto.

5.4 Fluxo de GCS

O fluxo da GCS observado na unidade análise é seguido pela ferramenta de GCS. O fluxo está dividido entre requisições emergenciais e normais, sendo que nas requisições normais existe mais uma divisão entre sistemas locais ou regionais/globais. Abaixo é apresentado o fluxo emergencial registrado na unidade de análise, representado pela Figura 5.1.

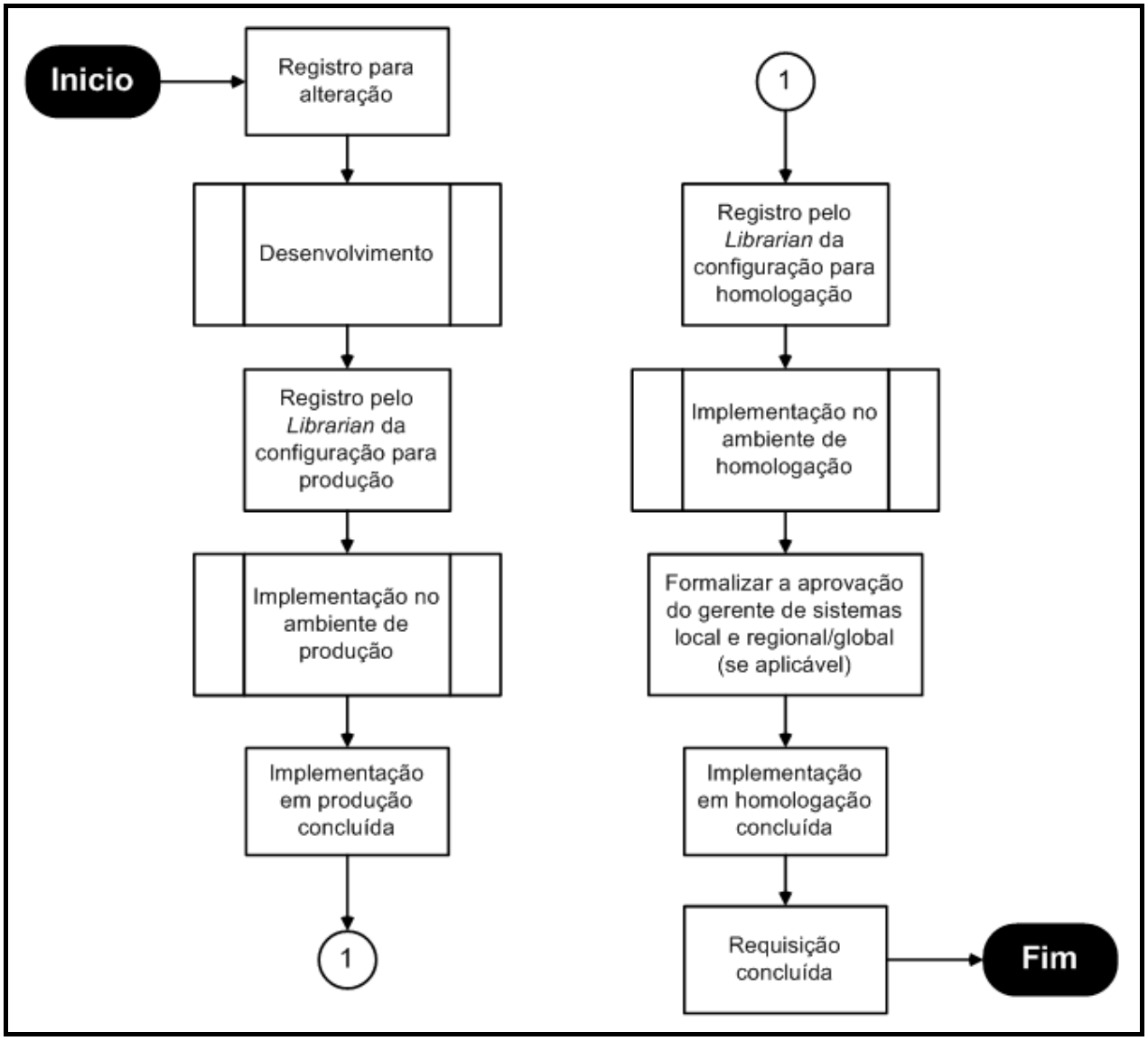

Figura 5.1 - Fluxo emergencial de GCS para sistemas locais, regionais ou globais registrado na ferramenta de GCS.

A Figura 5.1 demonstra o fluxo emergencial registrado na unidade de análise e agora terá comentado sobre cada quadro desse fluxo.

Registro para alteração – Esse é o primeiro passo na formalização de uma manutenção. Os dados de entrada são as especificações, registro de erros ou qualquer informação relevante para que os desenvolvedores entendam os requisitos corretamente.

Desenvolvimento – Tal passo está ligado às etapas necessárias ao desenvolvimento da manutenção. Os registros coletados da falha serão os dados de entrada e o *software* corrigido será a saída.

Registro pelo *Librarian* da configuração para produção – A ação, nesse passo, é a de registrar na ferramenta de versionamento, os arquivos de configuração, os documentos e os arquivos compilados do *software*.

Implementação no ambiente de produção – Por se tratar de uma manutenção emergencial, a implementação é realizada, primeiramente em ambiente produtivo. Esse passo refere-se à ação dos grupos envolvidos com a implementação.

Implementação em produção concluída – Nesse momento, os grupos envolvidos com a implementação devem formalizar o resultado obtido. Além da formalização, qualquer arquivo de registro gerado durante a implementação deve ser anexado à ferramenta de GCS para registro.

Registro pelo *Librarian* da configuração para homologação – Por se tratar de uma manutenção emergencial, o registro dos arquivos relacionados à manutenção já foram gravados na ferramenta de versionamento, portanto, o que deve ser feita é a promoção dos arquivos para o ambiente de homologação.

Implementação no ambiente de homologação – Os arquivos de configuração, os documentos e os arquivos compilados do *software*. que estão registrados na ferramenta de versionamento devem ser implementados no ambiente de homologação para que a sincronia não seja perdida.

Formalizar a aprovação do gerente de sistemas local e regional/global – Após a implementação em homologação, é necessário formalizar a aprovação dos gerentes envolvidos. O fluxo emergencial atende sistemas locais, regionais ou globais, portanto será necessária a formalização da aprovação do gerente de sistemas local e, quando for ele de um sistema regional ou global, também será necessária a formalização dele.

Implementação em homologação concluída – Nesse momento, o resultado obtido deve ser formalizado pelos grupos envolvidos com a implementação. Qualquer arquivo de registro gerado durante a implementação deve ser anexado na ferramenta de GCS para registro.

Requisição concluída – Esse é o passo de encerramento da requisição de implementação.

A ferramenta utilizada pela unidade de analise é regional, portanto as alterações para atender as necessidade locais nem sempre são realizadas. O fluxo registrado e suportado pela ferramenta de GCS apresentado na Figura 5.1 não representa o fluxo realizado pelos analistas em casos emergenciais. Embora a ferramenta e a documentação não apresentem os passos adicionais, eles são executados, visto que há que se preencher e atender requisitos de auditoria e do CCM (Conselho de Controle de Modificações).

Os passos em cinza apresentados na Figura 5.2 não são suportadas pela ferramenta de GCS. Dessa forma, todas as formalizações necessárias são realizadas por meio de mensagens eletrônicas ou documentos anexados à ferramenta. Dos seis passos adicionados à Figura 5.2 cinco deles possuem a participação exclusiva do usuário. Portanto, é evidente a necessidade do usuário no fluxo de GCS.

Os passos adicionais no fluxo efetivamente realizado são:

Usuário faz uma requisição de alteração – Qualquer solicitação para alteração deve ser associada a uma requisição do usuário e, nos casos em que a requisição é da própria área de tecnologia é necessária uma ciência do usuário responsável. A formalização do pedido é necessária para impedir que o processo seja criado sem a participação do usuário. No fluxo emergencial, a formalização da requisição poderia ser feita após a implementação em produção, para não impactar ainda mais o ambiente produtivo.

Aprovação verbal do usuário responsável e do gerente de sistemas local e regional/global para realizar a implementação em produção – No fluxo emergencial, o tempo para implementação é crítico, portanto as aprovações são dadas verbalmente. Os gerentes de sistemas e o usuário responsável são requisitos mínimos para aprovar a implementação em produção.

A falha foi corrigida? – Depois da implementação, é necessário que o usuário verifique se a falha foi corrigida. Caso a falha não tenha sido corrigida, os desenvolvedores serão acionados novamente.

Implementação no ambiente de COB – Após 24 horas úteis da implementação em produção é realizada a implementação do *software* no ambiente de COB. O período de 24 horas serve para garantir que a nova versão não apresente falhas durante o primeiro dia de expediente.

Aprovação do usuário responsável para realizar a implementação em homologação – Após a implementação em produção, o fluxo continua como emergencial, mas com maior formalismo. Para implementar a manutenção em homologação, é necessário que o usuário responsável formalize a ação antes.

A Figura 5.2 apresenta o fluxo realizado pelos analistas em uma requisição emergencial.

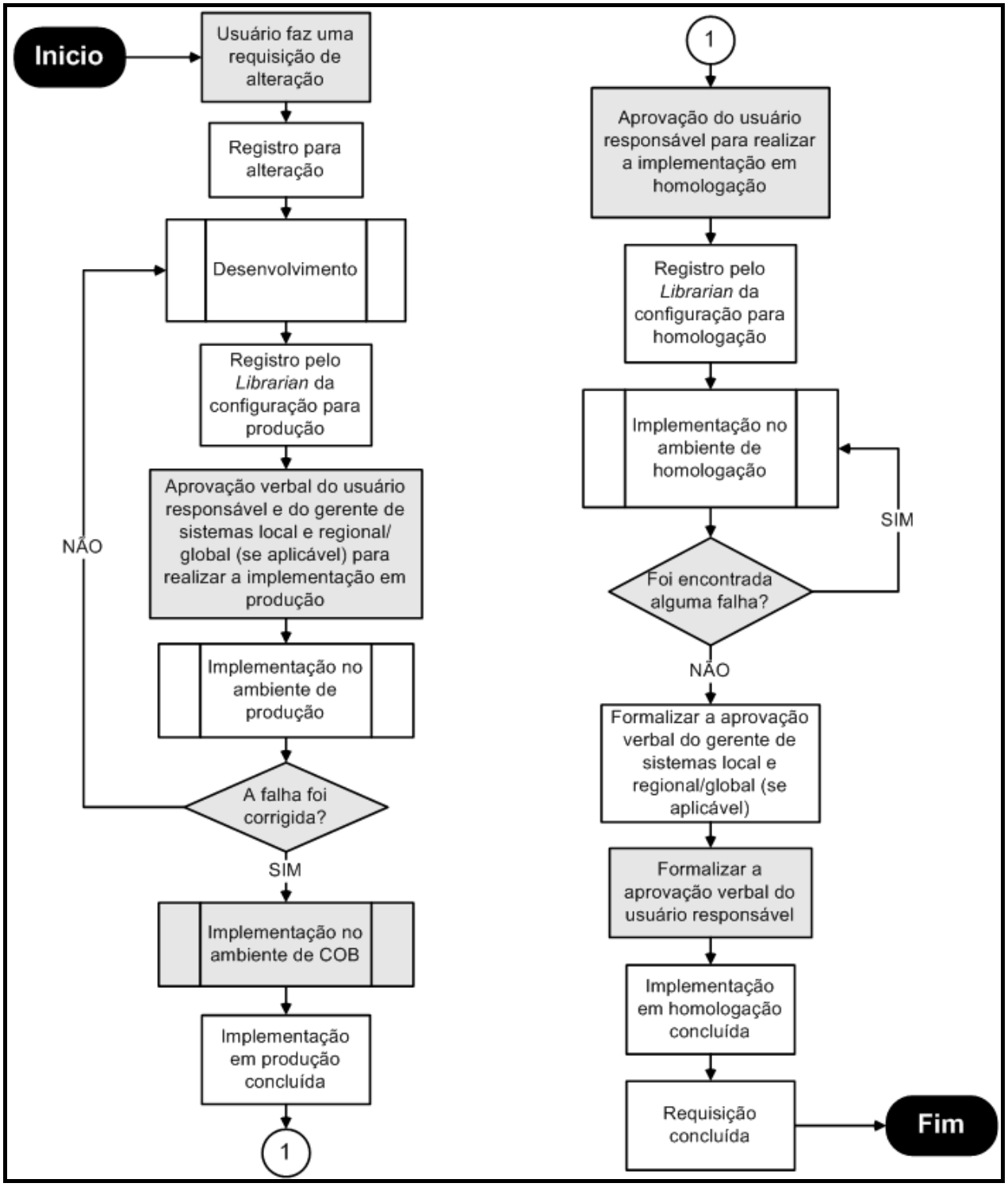

Figura 5.2 - Fluxo emergencial efetivamente realizado pela unidade de análise.

Foi encontrada alguma falha? – Após a implementação, o usuário irá validará o ambiente, procurando falhas no *software* em decorrência de erros durante a execução da implementação. Se alguma falha for encontrada, a implementação no ambiente será revisada.

Formalizar a aprovação verbal do usuário responsável – As aprovações verbais fornecidas durante o fluxo emergencial devem ser formalizadas. Esse é, portanto, o momento dedicado à formalização da aprovação do usuário responsável.

A Figura 5.3 apresenta o fluxo de implementação para sistemas locais.

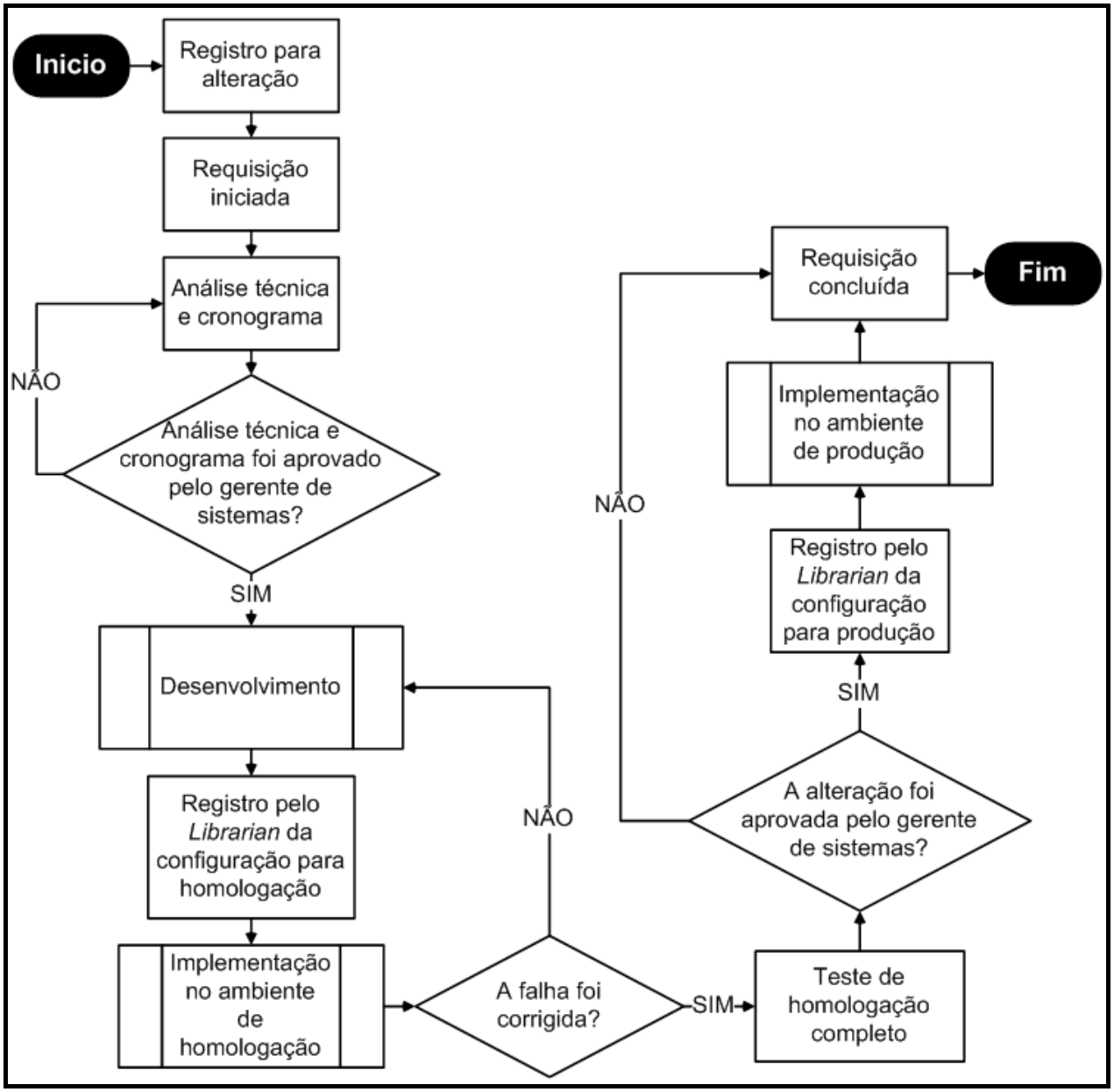

Figura 5.3 - Fluxo de GCS para sistemas locais registrado na ferramenta de GCS.

Alguns passos demonstrados na Figura 5.3 são os mesmos realizados no fluxo emergencial. Os passos iguais, portanto, não serão comentados.

Requisição iniciada – Neste momento, a requisição já foi oficializada, mas é passo intermediário, pois nesse período é realizada a preparação da análise técnica e do cronograma.

Análise técnica e cronograma – Nesse passo, a análise técnica e o cronograma são criados e anexados à ferramenta de GCS. O objetivo da análise técnica é apresentar as alterações necessárias ao *software* que resultarão em mudanças na infra-estrutura ou no processo da unidade de análise. O cronograma contém as datas para implementação e as horas previstas para utilização.

Análise técnica e cronograma foi aprovado pelo gerente de sistemas? – Esse passo indica que as alterações realizadas no *software*, as datas para implementação e as horas previstas foram aprovadas. Caso as alterações técnicas ou o cronograma não tenham sido aprovados, o fluxo volta para o passo de "Análise técnica e cronograma".

Teste de homologação completo – Nesse momento, os testes dos usuários são formalizados, indicando que não foi encontrada nenhuma falha que pudesse trazer risco para o ambiente produtivo.

A alteração foi aprovada pelo gerente de sistemas? – Tal passo está relacionado à necessidade do gerente de sistemas aprovar a implementação para o ambiente produtivo. Caso a manutenção não seja aprovada, ela será fechada indo para o passo de "Requisição concluída".

Assim como aconteceu no fluxo anterior, alguns passos foram incluídos para atender requisitos de auditoria e do CCM (Conselho de Controle de Modificações). A Figura 5.4 apresenta o fluxo da GCS para sistemas locais.

Análise técnica e cronograma foi aprovado pelo usuário responsável? – A análise técnica sobre as alterações e principalmente o cronograma devem ser aprovados pelo usuário responsável. Caso a análise técnica ou o cronograma não seja aprovado pelo usuário responsável, o fluxo voltará para "Análise técnica e cronograma", até que as necessidades do usuário responsável sejam atendidas.

É necessário certificação? – Este passo indica se a implementação requer o processo de certificação. O processo de certificação será necessário quando forem incluídos ou atualizados novos componentes no *software* que possam ser conflitantes no ambiente produtivo. Configurações no sistema operacional, principalmente na estação do cliente, também são analisadas. Por exemplo: O sistema **A** trabalha com o *Regional Options* do Windows no formato inglês, mas o sistema **B** trabalha do formato português. Neste caso, o processo de certificação deve ser envolvido para evitar qualquer tipo de conflito entre os *softwares*. A decisão de envolver ou não a equipe de certificação é feita pelo gerente de sistemas, baseado na documentação enviada pela equipe de desenvolvimento.

Aprovado pela certificação? – Esse passo indica que o processo de certificação foi realizado. Caso seja aprovado, o *software* será encaminhado para a implementação em homologação. Em caso negativo, o fluxo voltará para o desenvolvimento onde serão feitas as alterações necessárias.

Aprovado pelo DA? – A maioria dos *softwares* possui um banco de dados para guardar as informações necessárias ao processamento. A equipe de DA (*Data administration*) analisará o modelo da base de dados para evitar dados duplicados dentro da unidade de análise. Por exemplo: O cadastro de clientes deve ser único, portanto, quando um *software* precisa consultar informações de clientes, é necessário acessar tabelas corporativas. Se o modelo do banco de dados não for aprovado pela equipe de DA, o fluxo volta para o desenvolvimento onde os desenvolvedores devem efetuar as alterações necessárias.

Usuário responsável aprovou a implementação em homologação? – Esse passo serve para que o usuário responsável aprove a implementação no ambiente de homologação. Desta forma é possível garantir que o usuário responsável esteja ciente de todas as implementações que são executadas no ambiente.

Usuário responsável aprovou a implementação em produção? – Nesse passo, o usuário responsável, assim como aconteceu em homologação, aprova a implementação em produção.

A alteração foi aprovada pelos grupos envolvidos? – Os grupos envolvidos devem aprovar a manutenção para garantir que não haverá conflito entre as atividades programadas. As aprovações dos grupos envolvidas são fornecidas durante a reunião do CCM de forma verbal, portanto, nenhuma aprovação dos grupos envolvidos é registrada na ferramenta de GCS.

Aprovado pelo CCM? – Esse passo é para que o CCM (Conselho de Controle de Modificações) tenha tempo de revisar e aprovar todas as manutenções. A função do CCM, aqui, é garantir que as aprovações sejam dadas corretamente e que o fluxo foi seguido. O CCM da unidade de análise realiza reuniões semanais para aprovar as manutenções programadas.

Foi encontrada alguma falha? – Esse passo serve para que o usuário indique alguma falha na nova versão do *software*. Caso seja encontrada alguma falha, o fluxo voltará para “Implementação no ambiente de produção”, para que os passos sejam revisados.

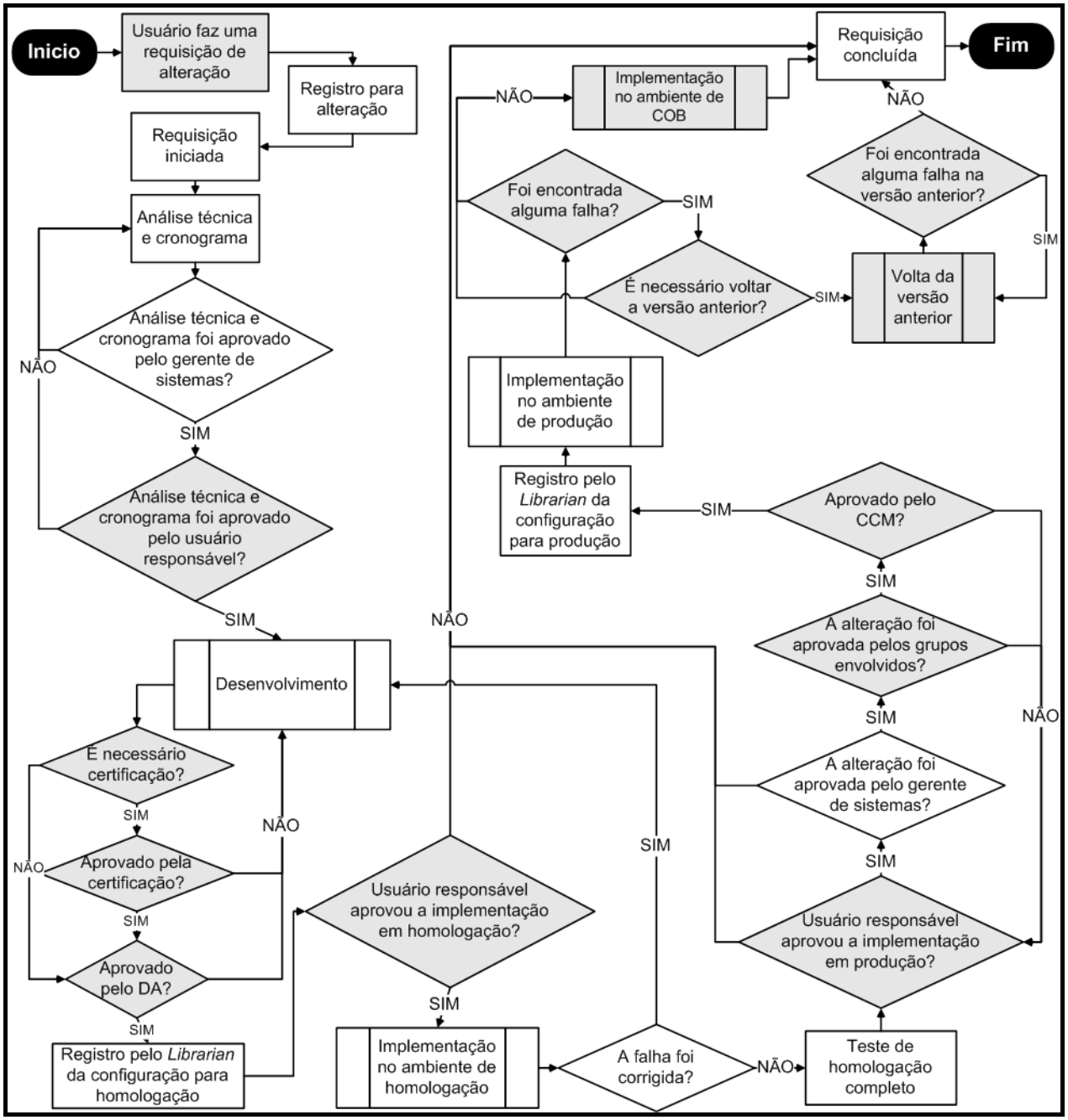

Figura 5.4 - Fluxo de GCS para sistemas locais efetivamente realizados pela unidade de análise.

<u>É necessário voltar a versão anterior?</u> – Este passo se refere à decisão de efetuar o *backout* da versão. Essa decisão é tomada de forma conjunta com o usuário responsável, o gerente de sistemas local e o gerente da regional/global (se aplicável).

Se não for necessário voltar à versão anterior, o fluxo seguirá para "Implementação no ambiente de produção". Se for necessário efetuar a volta de versão, o fluxo seguirá para "Volta da versão anterior".

<u>Volta da versão anterior</u> – Esse passo é o procedimento de efetuar o *backout* da versão.

Foi encontrada alguma falha na versão anterior? – Após o processo de *backout*, é necessário que o *software* seja testado novamente para garantir seu completo funcionamento. Em caso de falha, o fluxo voltará para o processo de "Volta da versão anterior" para revisão dos procedimentos executados. Caso não seja encontrada nenhuma falha o fluxo seguirá para "Requisição concluída".

O próximo fluxo de GCS é para sistemas regionais ou globais. A Figura 5.5 apresenta os passos completos e os passos adicionados são comentados abaixo:

Foi aprovado pela regional/global? – Esse passo se refere à aprovação da requisição pela primeira linha de gerentes regionais. O gerente da regional tem uma visão de todos os países suportados pelo *software*, portanto esse passo é necessário para que exista um controle sobre as manutenções e para que se evitem manutenções iguais ou conflitantes.

É necessário revisão da aprovação? – Esse passo serve para direcionar o fluxo para aprovação da segunda linha de gerentes da regional/global. A participação ou não dos gerentes de segundo nível é decidida pelo gerente de primeiro nível.

Foi aprovação pelo segundo nível da regional/global? – Nesse passo, o gerente de segundo nível aprovará ou não a requisição de manutenção. A principal função da aprovação de segundo nível é alinhar as requisições com as estratégias da corporação para região.

A análise técnica e o cronograma foram aprovados na regional/global e pelo gerente de sistemas local? – Tal passo indica que as alterações realizadas no *software* e as datas para implementação no cronograma foram aprovadas na regional/global e local. Caso as alterações técnicas ou o cronograma não sejam aprovados, o fluxo volta pra o passo de "Análise técnica e cronograma".

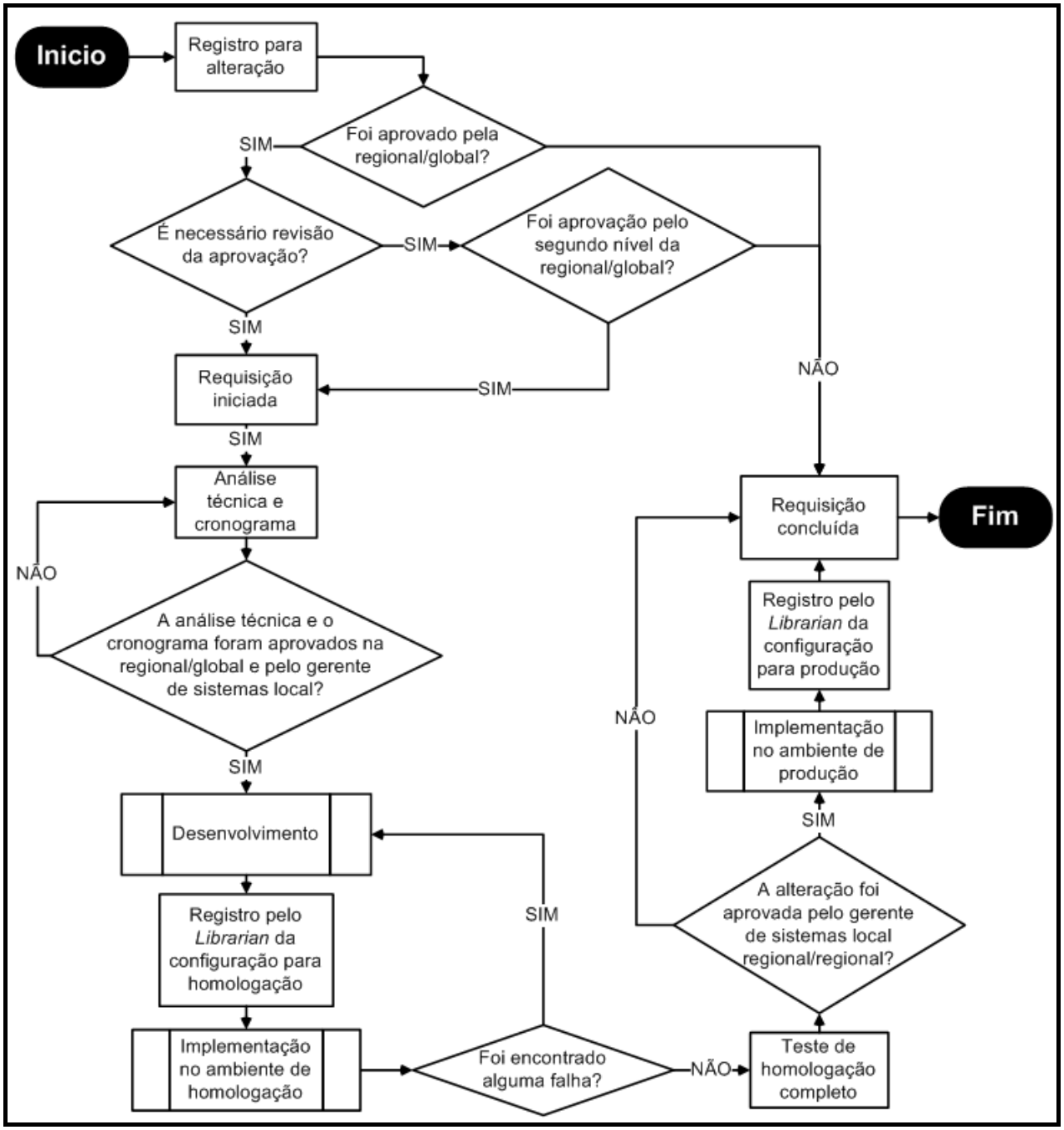

Figura 5.5 - Fluxo de GCS para requisição de sistemas regionais ou globais.

A Figura 5.6 e a Figura 5.7 apresentam o fluxo real, que é seguido pelos analistas durante uma manutenção de um sistema global/regional.

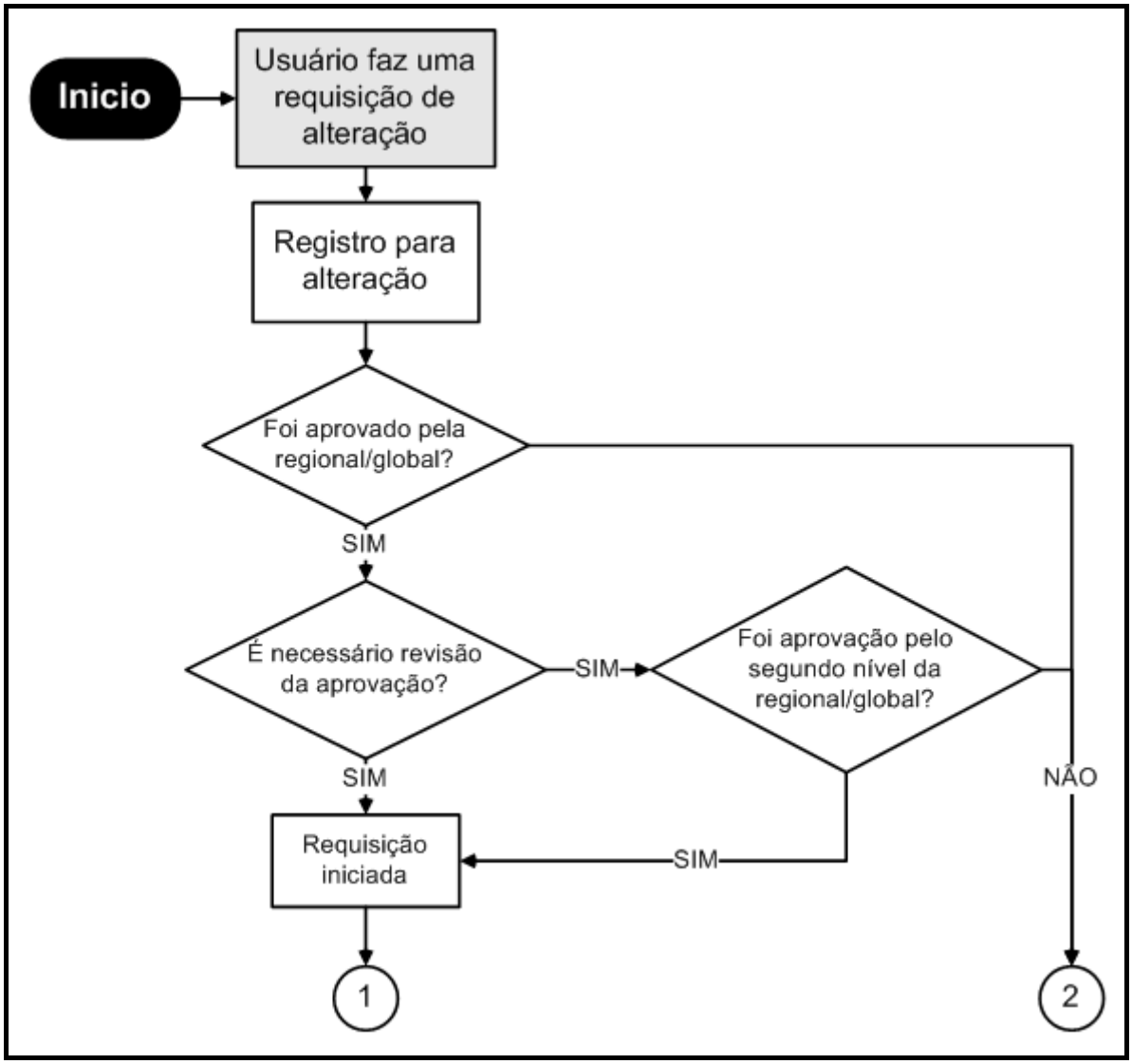

Figura 5.6 - Fluxo de GCS para sistemas regionais ou globais efetivamente realizados na unidade de análise.

O fluxo real para sistemas regionais ou globais realizado pela unidade de análise não apresentou nenhum passo novo em relação aos fluxos anteriores. A seguir, a continuação do fluxo através da Figura 5.7.

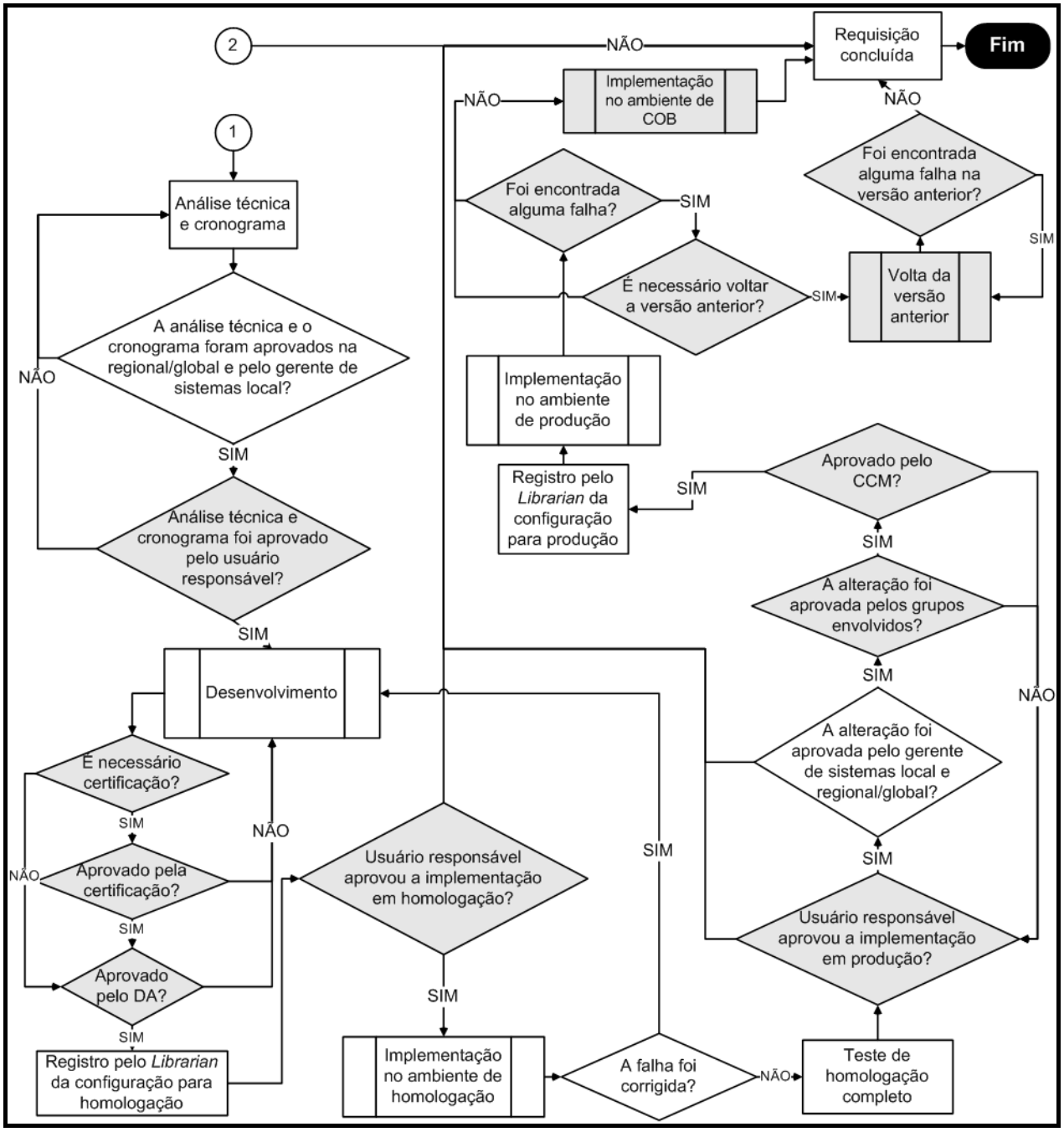

Figura 5.7 - Continuação do fluxo de GCS para sistemas regionais ou globais efetivamente realizados na unidade de análise.

Embora seja de conhecimento geral, as diferenças entre o fluxo realizado e o suportado pela ferramenta de GCS, as alterações estruturais são extremante raras, deixando as responsabilidades para as unidades locais de adaptações no fluxo.

5.5 Papéis e responsabilidades dos profissionais envolvidos

Abaixo, os papéis dos envolvidos no fluxo de GCS com suas respectivas responsabilidades encontrados na unidade de análise. Os papéis encontrados abaixo e conjunto com os papéis encontrados na pesquisa teórica foram a base para demonstrar os papéis dos profissionais envolvidos.

Gerente de configuração – O Gerente de configuração é formado por um analista ou grupo de analistas, responsável pelas alterações do fluxo de GCS, e manutenção e funcionamento da ferramenta de GCS. O gerente de configuração também é o representante da GCS da subsidiária local para o time regional/global. Ou seja, em qualquer assunto que esteja relacionado à GCS, o gerente de configuração estará envolvido.

CCM – O CCM (Conselho de Controle de Modificações) está ligado ao time de TI e é responsável por garantir que se dêem todas as aprovações. O CCM também coordenará as implementações entre os grupos envolvidos, a fim de evitar conflitos entre elas. Por exemplo: Uma parada nos servidores em função de ajustes elétricos não pode permitir que exista implementações programadas para o mesmo horário.

***Librarian* da configuração** – Na unidade de análise, o *librarian* é o responsável pela ligação entre a ferramenta de GCS e a ferramenta de versionamento. Existem várias formas para o *Librarian* receber a nova versão do *software*, mas a mais comum é o envio através de correio eletrônico, pois há a garantia comprovada para remetente e destinatário que se teve a versão correta enviada. O *Librarian* também é responsável em disponibilizar a versão do *software* para os times envolvidos na implementação.

Equipe de certificação – O processo de certificação faz parte das responsabilidades do *gerente de versões* definido no capitulo *Fundamentos teóricos*. A equipe da certificação na unidade de análise é uma equipe técnica, responsável por documentar os componentes e configurações necessárias para cada sistema. A cada necessidade de inclusão ou alteração de componentes é feita uma triagem de possíveis incompatibilidades no ambiente produtivo. Caso necessário, a equipe de certificação monta um ambiente similar ao de produção para reproduzir as possíveis incompatibilidades, fornecendo material necessário para a equipe de desenvolvimento realizar as alterações necessárias.

Equipe de DA – O time de DA (*Data administration*) é responsável por analisar o modelo de dados dos *softwares* a cada implementação, para garantir que as informações corporativas não sejam duplicadas. Outra função é a de analisar o modelo de forma técnica para evitar tabelas temporárias, sem chave primária ou situações que possam prejudicar a consistência dos dados.

Usuários e usuário responsável – Os usuários são os personagens principais de qualquer *software*, mas na unidade de análise eles não estão presentes explicitamente no fluxo oficial da GCS. Os usuários do *software* serão os responsáveis pela validação dos testes de negócio no ambiente de homologação e produção. O usuário responsável é alguém sênior designado para tomar as decisões referentes às estratégias do *software*. O usuário responsável está cadastrado na lista de inventário da unidade de análise, para que não seja possível burlar aprovações.

Gerente de sistemas local, regional ou Global – O gerente de sistemas é o responsável pela materialização das necessidades dos usuários. Conhecedor das regras de negócio e da tecnologia envolvida, ele caminha entre os dois universos para que as necessidades sejam cumpridas no menor tempo possível. O gerente também é o responsável pela previsão e controle das datas de implementação e horas previstas para implementação.
O fluxo oficial suportado pela ferramenta de GCS não prevê a participação do usuário. Sendo assim, todas as aprovações que o usuário responsável fornece são intermediadas pelo gerente de sistemas local. As aprovações são dadas por correio eletrônico e posteriormente anexadas à ferramenta de GCS sem qualquer manipulação.

PQA (Process quality assurance) – Criado pela área de tecnologia da unidade de análise, o PQA, cuja tradução do seu significado é “Garantia de Qualidade do Processo”, garantirá que o processo de alteração esteja sendo seguido e se necessário, promover alterações no fluxo da GCS para se adequar a novas necessidades. Outra função do PQA é realizar pré-auditorias internas com o objetivo de encontrar e resolver riscos antes da auditoria oficial.

Grupos envolvidos (Assignee) – Os grupos envolvidos são os times técnicos que realizarão a implementação no ambiente de homologação, produção e contingência. Na implementação em produção, os grupos envolvidos aprovam ou não a execução. A aprovação é dada verbalmente durante a reunião do CCM

Auditoria – A equipe de auditora é independente da subsidiária local, sendo controlada e gerenciada diretamente pela matriz. As fraudes, infrações ou qualquer ressalva são documentadas e informadas para a matriz. A principal função da auditoria, no processo ligado à GCS, é garantir que todas as implementações tenham sido realizadas com as aprovações necessárias. O processo de auditoria é dividido de acordo com os departamentos e acontece pelo menos uma vez ao ano em cada departamento.

5.6 Riscos de implementação

O risco de uma implementação na unidade de análise é medida através da criticidade do sistema e do tempo necessário para sua implementação.

Criticidade do sistema – A unidade de análise durante a fase inicial define a criticidade do sistema em: baixa, média ou alta, considerando algumas informações como: (1) Valor financeiro trafegado pelo sistema; (2) Número de usuários; (3) Interação com o cliente final da unidade de análise.

As informações são enviadas para a matriz, que analisa e retornam a criticidade para o sistema. Caso a equipe responsável pelo sistema no Brasil não esteja de acordo com classificação, é iniciado um processo entre os dois grupos para uma reclassificação do sistema.

Tempo para implementação – O tempo necessário para implementação determinará o "tipo" de manutenção. A previsão das horas necessárias é feita pelo gerente local, em conjunto com o gerente regional/global (quando aplicável) e são aprovadas durante o passo "Análise técnica e cronograma foi aprovado pelo gerente de sistemas?". Uma implementação com até 40 horas para execução é considerada nível um; implementações entre 41 e 200 horas é considerada nível dois; implementações com necessidades maiores que 200 horas são consideradas novos projetos.

A previsão inicial das horas é feita durante o passo "Análise técnica e cronograma", mas se no decorrer do fluxo o número de horas ultrapassar a previsão, a implementação será classificada novamente. O controle das horas é feito de forma automática, pois a ferramenta de GCS está ligada com a ferramenta de lançamentos de horas, facilitando o controle das horas. Por exemplo: Uma implementação do tipo um pode se tornar do tipo dois, se as horas utilizadas ultrapassarem 40 horas previstas. Nos casos em que a implementação ultrapassa o limite de

200 horas utilizadas, a requisição deve ser fechada e transformada em uma necessidade inicial de projeto.

Com base nas informações dadas, segue a classificação de uma implementação.

Criticidade do sistema	**Tempo para realizar a implementação**	**Risco de uma implementação**
Alta	< 40 HH	Alto
Alta	Entre 41 e 200 HH	Alto
Médio	< 40 HH	Baixo
Médio	Entre 41 e 200 HH	Médio
Baixo	< 40 HH	Baixo
Baixo	Entre 41 e 200 HH	Baixo

Quadro 5.1 – Risco de uma implementação.

As informações apresentam a classificação de uma implementação. Na unidade de análise, a classificação é feita de forma automática pela ferramenta de GCS. Com a criticidade do sistema e o tempo para realização, a ferramenta GCS classifica a implementação correspondente.

5.7 Aprovações

As aprovações são utilizadas largamente na unidade de análise. O objetivo é que nada seja feito sem a aprovação ou ciência (quando aplicável) dos envolvidos. A unidade de análise possui duas divisões para as aprovações:

- Aprovação ocorre quando o aprovador se responsabiliza pelas conseqüências que a ação pode desencadear. Por exemplo, quando o *software* troca de versão, o usuário responsável aprovará a alteração e se responsabilizará por problemas que ela eventualmente vier a apresentar.
- Ciência acontece quando é necessário envolver alguém ou alguma equipe, que deve ser informada em caso de problemas. Por exemplo, o *software* de conta corrente será atualizado. Em caso de problemas, pode afetar outros sistemas da unidade de análise. Portanto os usuários responsáveis devem ficar cientes desta implementação.

Em caso de falhas com perdas financeiras, o prejuízo é repassado para as áreas responsáveis. A diferença, portanto, entre ciência ou aprovação serve, também para determinar o “culpado”, quando necessário.

No fluxo oficial registrado na ferramenta de GCS, as aprovações necessárias são do gerente de sistemas local, regional ou global. No fluxo efetivamente realizado, é necessária a aprovação dos usuários responsáveis, dos grupos envolvidos e do CCM.

A responsabilidade de obter e registrar a aprovação ou ciência do usuário responsável é do gerente local. As aprovações dos grupos envolvidos e do CCM são dadas durante a reunião semanal e são registradas pelo próprio CCM.

Risco de uma implementação	**Aprovações necessárias**
Alta	Aprovação do usuário responsável. Aprovação dos usuários responsáveis afetados pela implementação. Aprovação do gerente local, regional/global (se aplicável). Aprovação dos grupos envolvidos. Aprovação dos gerentes imediatos dos grupos envolvidos. Ciência do CCM.
Médio	Aprovação do usuário responsável. Ciência dos usuários responsáveis afetados pela implementação. Ciência do gerente local, regional/global (se aplicável). Ciência dos grupos envolvidos. Ciência do CCM.
Baixo	Aprovação do usuário responsável. Ciência do gerente local, regional/global (se aplicável). Ciência dos grupos envolvidos. Ciência do CCM.

Quadro 5.2 – Risco de uma implementação.

O armazenamento das aprovações é realizado de duas formas: como arquivo anexado na ferramenta ou através do fluxo. Para as aprovações dos gerentes de sistemas local, regional ou global que participam do fluxo, a aprovação é feita diretamente na ferramenta de GCS. Para as demais aprovações, elas são armazenadas na ferramenta como arquivo anexado.

Essa característica da ferramenta, excluindo alguns participantes, compromete muito o controle sobre as aprovações, pois a responsabilidade de solicitar a gerência as aprovações necessárias é do gerente de sistemas local.

5.8 Auditoria

Por se tratar de uma instituição financeira em que a matéria prima é dinheiro, o processo de auditoria procura irregularidades fraudulentas ou falhas de processamento que possam gerar perdas financeiras.

O processo para realizar a auditoria em um departamento é realizado em três etapas:

- **Pré-auditoria**: É a fase inicial da auditoria em que os auditores conhecem os processos do departamento, solicitam documentos, relatórios ou quaisquer informações relevantes que posam apresentar as falhas do departamento. Esse período é considerado o período de investigação.
- **Auditoria**. É a fase em que os auditores apresentam todas as falhas encontradas para aos responsáveis pelo departamento. Os responsáveis recebem as informações e rebatem as falhas com explicações. Durante duas semanas são realizadas discussões sobre as falhas encontradas.
- **Pós-auditoria**: Baseado nas falhas encontradas durante a fase de pré-auditoria e nas explicações durante a auditoria, o time de auditores liberam um relatório final delatando as falhas encontradas. O relatório é enviado para a matriz, que, por sua vez, tomará as ações necessárias como investimentos em tecnologia, melhoria dos processos ou realocação de profissionais.

A GCS é de responsabilidade dos gerentes locais e dos grupos envolvidos e, por isso a auditoria referente à GCS é feita em duas instâncias: (1) na equipe de gerentes locais é feita a verificação das aprovações necessárias e a constatação de que a solicitação do usuário foi atendida; (2) na equipe dos grupos envolvidos é feita a constatação de que a implementação foi realmente realizada.

Nos dois casos, a equipe de auditoria verifica as evidências concretas através de arquivos anexados à ferramenta de GCS. Mensagens eletrônicas com as aprovações, mensagens eletrônicas com as solicitações do usuário, arquivos de registro do *software*, entre outros, são alguns exemplos para comprovar as ações realizadas.

O processo de auditoria contém vários agentes externos ao processo de GCS, mas o fluxo proposto deverá alimentar esse processo com relatórios e fluxos específicos para registrar as aprovações.

5.9 Conclusões finais

A aplicação do estudo de caso ajudou a visualizar os diferentes fluxos da unidade de analise, bem como as técnicas utilizadas para sua execução. Foi observado que a unidade de análise possui um quarto ambiente para contingência. Este ambiente possui características específicas no fluxo GCS e é uma alternativa para diminuir os riscos de uma implementação. Outra característica encontrada na GCS da unidade de análise é o programação de horários específicos para implementações com alto risco.

6 Proposta do roteiro

O roteiro proposto foi criado com base em definições teóricas apresentadas e o aprendizado que foi adquirido durante o estudo de caso.

6.1 Ambiente

O modelo ITIL (2004, p.93) sugere a criação de três ambientes distintos para a condução da GCS. Os ambientes de desenvolvimento, homologação e produção devem ser fisicamente ou virtualmente separados para garantir que instabilidades apresentadas em um ambiente não afetem os demais ambientes.

O ambiente de homologação deve ser considerado pré-produção. Portanto, os *softwares* básicos e todas as políticas de segurança devem ser idênticas às de produção. Qualquer alteração deve ser aplicada, testada e aprovada em homologação antes de ser aplicada em produção.

Durante o estudo de caso foi observado um quarto ambiente chamado de COB (*Continuity of Business*). A função deste ambiente é disponibilizar um ambiente físico em caso de problemas em produção, mas também permite um melhor gerenciamento durante as implementações de alto risco. Implementações com intervalo de 24 horas entre o ambiente produtivo e o COB podem garantir a continuidade dos negócios em caso de falhas no *software*.

A adoção de vários ambientes pode aumentar o custo geral de manutenção do *software*. Neste roteiro sugerimos a adição de quatro ambientes: desenvolvimento, homologação produção e pós implementação para obter todas as vantagens de cada um.

A seguir uma breve descrição de cada um. O ambiente de **desenvolvimento** é onde a equipe de desenvolvimento irá trabalhar. Este ambiente será usado para os primeiros testes dos desenvolvedores que poderão validar se as alterações estão de acordo com os requisitos. O ambiente de **homologação** é, em uma implementação normal, o próximo ambiente, após o desenvolvimento. Este ambiente é utilizado para que os usuários validem as alterações criadas pelos desenvolvedores, para uma melhor reprodução do ambiente produtivo este ambiente deve seguir as mesmas políticas de segurança implantada no ambiente produtivo. O ambiente **produtivo** deve ser inacessível para qualquer tipo de alteração não autorizada, portanto o controle de acesso e as versões devem ser controladas através de ferramentas adequadas.O ambiente de **pós-implantação** é destinado para necessidades emergências em que o ambiente de produção está indisponível ou inacessível.

6.2 Controle de versões

Uma exigência da CGS é a de que os itens de configuração (*software*, *scripts*, documentos, entre outros) sejam controlados e a de que as diversas versões sejam passiveis de rastreabilidade. (WESTFECHTEL, 2001, p.1) (SEI-CMMI, 2006, p.121).

Para atender a esta premissa, é necessário que exista um repositório para guardar as diversas versões do *software* ou documentos pertinentes. A unidade de análise utiliza uma ferramenta chamada PVCS para gerenciamento de verões que atende as premissas necessárias para o controle de versões.

Uma pratica na unidade de análise que deve ser evitada é a forma de interação com a ferramenta para controle de versões. Preferencialmente, a interação com a ferramenta de controle de versões deve ser automaticamente através de uma ferramenta de GCS, permitindo a rastreabilidade das implementações com as diversões versões do *software*.

6.3 Papéis e responsabilidades dos profissionais

O roteiro proposto sugere cinco papéis dentro do fluxo de GCS. São eles: usuário responsável, gerente de sistemas, *librarian*, grupos envolvidos, CCM.

Usuário responsável – O usuário responsável é composto de representantes de dois grupos: a primeira linha gerencial responsável pelo *software* e a primeira linha gerencial solicitante da implementação. O solicitante da implementação fornecerá as aprovações e o responsável pelo sistema dará o “de acordo”. A idéia principal é a de que o gerente responsável pelo sistema, que nem sempre é solicitante da implementação, esteja de acordo com as implementações que podem afetar o ambiente ou o *software* de sua responsabilidade.

O gerente solicitante tem a responsabilidade de realizar os testes sobre a implementação, garantindo que ela não afetará o ambiente.

Gerente de sistemas – O gerente de sistemas é a ligação entre o mundo de negócios e tecnologia. A idéia é a de que o gerente de sistemas direcione as implementações do sistema, visando sempre a qualidade do *software*. O gerente de sistemas também é responsável pela coordenação do fluxo de GCS porque, embora o fluxo proposto seja seqüencial, em grandes corporações é comum que implementações percam prioridade e sejam esquecidas. A idéia é que o gerente de sistemas não deixe que implementações em andamento sejam despriorizadas pelas áreas envolvidas.

Librarian – O *Librarian* é o responsável pela manipulação dos arquivos do *software*. Todos os arquivos compilados do software ou documentos pertinentes à implementação devem ser registrados na ferramenta de versionamento pelo *Librarian*. Os grupos envolvidos ou qualquer analista que irá manipular o ambiente para realizar a implementação só deve utilizar os arquivos fornecidos pelo *Librarian*.

Grupos envolvidos – São os grupos responsáveis pela realização da implementação. A responsabilidade dos grupos é executar a implementação, utilizando somente os arquivos compilados do *sofware* fornecidos pelo *Librarian* e evidenciando a conclusão da implementação. Outra responsabilidade das equipes envolvidas na implementação são as evidências da validação realizadas no ambiente de contingência. O ambiente de contingência é testado pelos usuários somente durante os exercícios de DR (Desastre e Recuperação). Portanto, os testes realizados após cada implementação serão executados pelos grupos envolvidos.

CCM – O CCM(Conselho de Controle de Modificações) é um grupo de analistas responsável por analisar todas as implementações agendadas para produção. A responsabilidade do CCM é garantir que somente implementações aprovadas pelos usuários responsáveis serão realizadas. Uma responsabilidade do CCM é garantir que implementações conflitantes não sejam executadas no mesmo horário, garantindo efetiva consistência sobre as atividades. O CCM também é responsável pela comunicação das implementações aprovadas e programadas para os grupos envolvidos.

Gerente de configuração – Embora não apareça no fluxo da GCS, esse profissional é o responsável pela manutenção do fluxo, quando necessário. Sua responsabilidade é garantir que nenhuma atividade de controle ou gerencial seja realizada fora do fluxo.
Quando existir a necessidade de incluir ou excluir qualquer passo no fluxo, o gerente de configuração deverá alterá-lo no fluxo oficial da GCS.

6.4 Aprovações

Conforme demonstrado no capitulo *Fundamentos teóricos*, para que a GCS seja aplicada de forma correta, é pré-requisito que qualquer implementação obtenha as aprovações necessárias durante o fluxo. (IEEE-SWEBOK, 2004, p.7-6)

Seawlho (2003, p.2) indica que um dos objetivos da GCS é garantir que os grupos envolvidos sejam informados sobre as implementações programadas. A aprovação é uma forma de formalizar que o analista ou o grupo de analistas envolvidos esteja ciente da implementação.

O método proposto possui dois tipos de aprovações:

- Aprovação acontece quando o aprovador se responsabiliza pelas conseqüências que a ação pode desencadear, neste caso o aprovador sempre será o solicitante da implementação;
- O “de acordo” acontece quando é necessário envolver alguém ou algum time que deve ser informado em caso de problemas. O usuário responsável pelo software, se não for o solicitante, sempre será envolvido para o “de acordo” na implementação.

O objetivo desta divisão é responsabilizar, em caso de problemas, o usuário que forneceu a aprovação. A aprovação é dada pelo usuário que realiza os testes necessários para garantir que a nova versão está funcionando corretamente.

6.5 Risco de alteração

A definição da criticidade da implementação pode seguir o padrão proposto no capítulo *Riscos de alteração.*

O risco de uma implementação pode ser calculado, utilizando o método demonstrado no capítulo *Calculando o risco de uma alteração* com algumas adaptações.

No estudo de caso, foi observado que os dados relativos ao processo de GCS são utilizados durante o processo de auditoria e que ocorre anualmente em cada departamento. Assim o Quadro 2.7 que indica os dias necessários para retenção dos dados deve ser substituído por outro de um período fixo, seguindo o calendário de execução da auditoria.

6.6 Relatório e Métricas para Gerência de Configuração de Software

As métricas da GCS fornecerão informações sobre as implementações que podem ser utilizadas para um melhoramento contínuo do fluxo de GCS. A seguir, alguns relatórios para serem extraídos da ferramenta de GCS:

- Número de implementações em andamento – O objetivo deste relatório é apresentar os dados das implementações que estão em andamento e a respectiva fase das que, estão paradas.
- Número de implementações em andamento divididas por sistema – A divisão do relatório por sistema auxiliará na gerência de GCS, apresentando os dados das implementações que ainda não foram finalizadas. Com estes dados, é possível identificar quais são as implementações pendentes de cada sistema.
- Número de implementações fechadas – Tal relatório apresentará os dados das implementações já fechadas.
- Número de implementações fechadas por sistema – Com a divisão do relatório por sistema é possível medir o número de implementações realizadas por sistema.
- Número de implementações fechadas, por sistema, com registro de erros – Nesse relatório estão presentes as implementações que estão fechadas, mas que, contudo, apresentaram erros durante os testes iniciais. O objetivo é que sistemas com alto índice de recorrência em problemas sejam investigados a fim de se descobrir a causa raiz dos erros, como por exemplo, falta de testes, especificações incompletas, baixa qualidade de codificação, entre outros.
- Número de implementações abertas ou fechadas divididas por departamento – O objetivo desse relatório é apresentar o número de implementações de um departamento específico, com esses dados será possível medir os departamentos com maior demanda de implementações.

Os relatórios sugeridos acima podem variar de acordo com as necessidades de cada instituição. Os relatórios sugeridos podem aumentar ou diminuir, mas algumas necessidades básicas para gerenciar a GCS e a realização do processo de auditoria podem ser menos traumáticos com os relatórios e métricas adequadas.

6.7 Auditoria

Durante o estudo de caso, foi observado que o processo de auditoria procura irregularidades fraudulentas ou falhas de processamento que possam gerar perdas financeiras. Embora o processo de auditoria tenha o mesmo princípio, cada processo seguirá as diretrizes e necessidades da empresa em que está em exercício.

No capitulo *Fundamentos teóricos* foi apresentada uma divisão entre: auditoria funcional, física e de GCS.

O objetivo da auditoria de configuração funcional é o de verificar se os requisitos definidos na documentação dos itens de configuração foram atendidos. A auditoria física deve garantir que o item de configuração foi construído conforme a documentação técnica que o define.

A auditoria do sistema de GCS é realizada para assegurar que a implementação da GCS permaneça compatível com a política e procedimentos estabelecida pela instituição. A auditoria no sistema de GCS é essencial para garantir que os processos definidos sejam propriamente aplicados e controlados.

6.8 Fluxo conhecido

Para respeitar os *Baselines* definidos e garantir que todas as áreas afetadas estejam envolvidas, é necessário que o fluxo da GCS seja conhecido e seguido. Para garantir a integridade do fluxo, é necessária uma ferramenta de GCS que suporte esse fluxo do inicio ao fim nos seus diversos passos. Uma característica positiva encontrada na unidade de análise é a criação de fluxos diferentes para implementações emergenciais e normais.

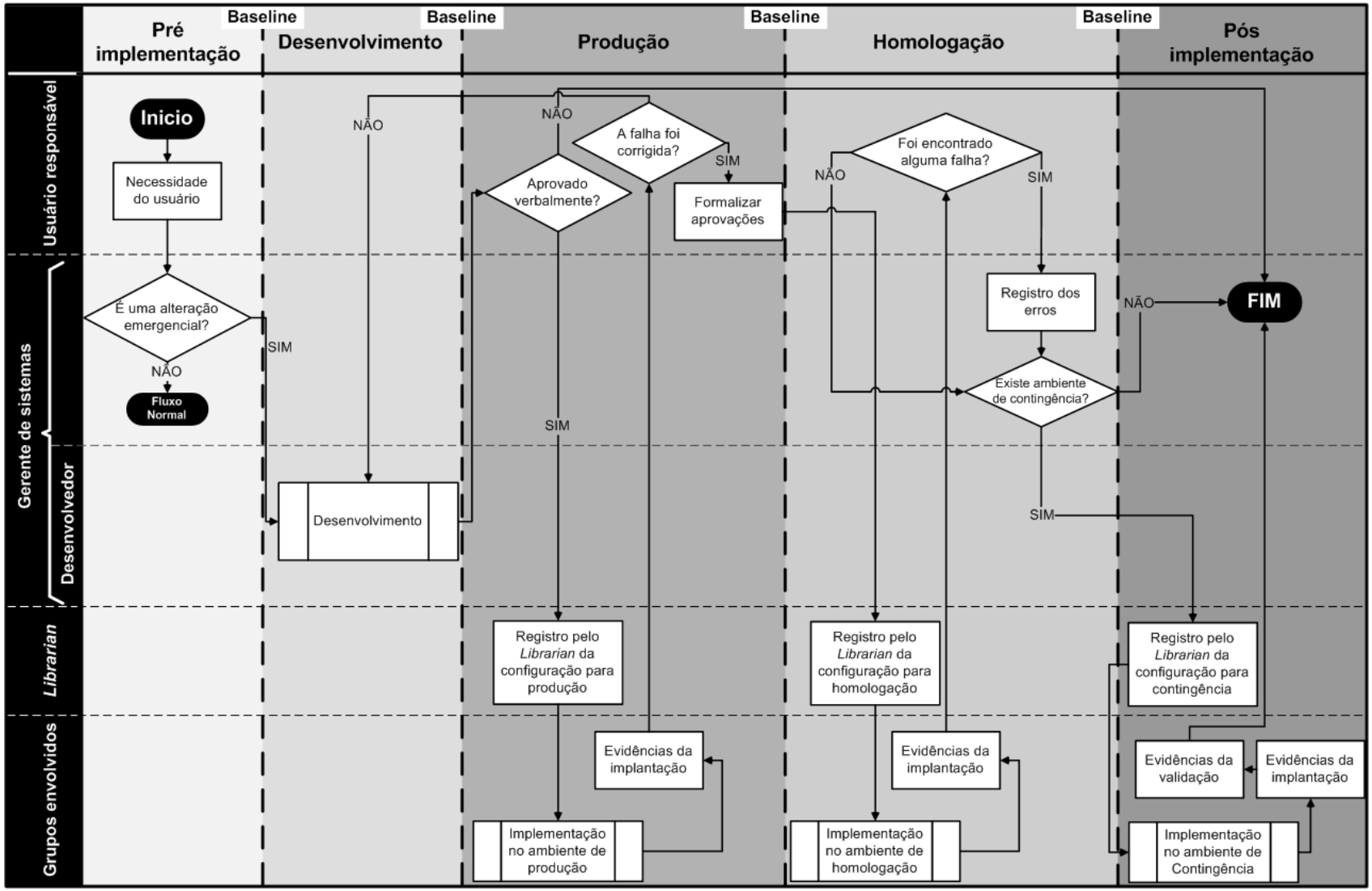

Figura 6.1 - Fluxo de GCS para implementações emergenciais.

A ferramenta direcionará as ações, conforme fluxo pré-definido e registrará os comentários e aprovações recolhidas durante o fluxo.

O fluxo apresentado na Figura 6.1 é o emergencial. Abaixo é apresentado o que cada passo significa no fluxo.

Necessidade do usuário – Todas as requisições são iniciadas pelo usuário responsável. O objetivo desse passo é determinar qual é a necessidade que irá justificará a implementação. A solicitação inicial deve ser feita com todos os dados disponíveis, a fim de permitir que o time de tecnologia tenha condições de alterar o sistema.

É uma alteração emergencial? – Nesse ponto, é necessário dividirem-se implementações emergenciais de implementações normais. Uma implementação emergencial, conforme definido no capitulo *Gerência de configuração de software*sugere que o ambiente produtivo seja normalizado o mais rápido possível, pois o bom andamento do sistema é vital para a continuidade dos negócios da empresa. O fluxo emergencial terá um caminho mais curto até o ambiente produtivo. Se a alteração for classificada como emergencial, o fluxo seguirá para o ambiente de desenvolvimento, caso contrário irá seguirá o fluxo normal de uma implementação. Para implementações emergenciais, é definido o risco de alteração.

Desenvolvimento – O processo de desenvolvimento realizado por uma entidade externa que não será controlada pelo fluxo proposto é representado nesse passo. A saída desse passo é o *software* e a documentação é necessária para executar a implementação.

Aprovado verbalmente? – Trata-se de uma implementação emergencial. Portanto, nesse momento, a formalidade é deixada de lado para que o fluxo seja ágil. As aprovações necessárias são fornecidas de forma verbal e, por se tratar de uma implementação emergencial, todos os usuários responsáveis ou usuários responsáveis suplentes devem aprovar a implementação.

Registro pelo *Librarian* da configuração para produção – Após a aprovação, o *software* e a documentação enviada devem ser registrados na ferramenta de versionamento. A ferramenta de versionamento garantirá que, caso necessário, a versão anterior seja disponibilizada novamente. Outra vantagem do versionamento é a de que, no momento de reinstalar um *software* em um ambiente novo, com as versões guardadas, é possível instalar o *software* novamente desde o início, sem a necessidade de pacotes especiais.

Implementação no ambiente de produção – Os procedimentos criados pelos desenvolvedores e registrados pelo *Librarian* são executados no ambiente produtivo.

Evidências da implantação – Exigir evidências da implementação é uma forma de dificultar alterações não autorizadas no ambiente. Após a implementação, os grupos envolvidos devem extrair as evidências do ambiente para garantir que a implementação tenha sido executada. As evidências podem ser registradas através de *print screen*[1], para evidenciar que os componentes do *software* tenham sido trocados. Estrutura das tabelas do banco de dados e registros de log da nova versão do *software* também são exemplos de evidências da implantação.

A falha foi corrigida? – O usuário responsável talvez não seja o usuário que testará o *software*, mas é vital para um bom controle que ele tenha ciência que a falha foi corrigida. Por isso, no fluxo, ele é o responsável pela aprovação. Caso a falha tenha sido corrigida, a emergência acabou. O fluxo seguirá para formalização das aprovações e a implementação para homologação. Se a falha não foi corrigida, o fluxo voltará para o desenvolvimento para que sejam realizados novos ajustes.

Formalizar aprovações – Por se tratar de uma implementação emergencial as aprovações foram dadas verbalmente no passo **Aprovado verbalmente?** , mas agora elas devem ser formalizadas na ferramenta de GCS. Não é necessário que as aprovações verbais e na ferramenta sejam das mesmas pessoas, basta que ambos tenham alçada para aprovação. O processo para formalizar a aprovação verbal deve acontecer em até 24 horas úteis.

Registro pelo *Librarian* da configuração para homologação – Os mesmos arquivos utilizados no ambiente produtivo devem ser promovidos no ambiente de homologação. O objetivo é o de que as versões entre os ambientes sejam iguais.

Implementação no ambiente de homologação – Os procedimentos criados pelos desenvolvedores e registrados pelo *Librarian* serão executados no ambiente de homologação.

Evidências da implantação – O objetivo das evidências na homologação, dificultar alterações não autorizadas no ambiente. Em ambiente produtivo existe o adicional de evitar fraudes e permitir uma fiscalização, se necessário.

Foi encontrada alguma falha? – Embora a implementação em produção já tenha acontecido é necessário que aconteça a verificação no ambiente. Caso não seja encontrada nenhuma falha, a implementação pode ser fechada, mas em caso de falhas, o fluxo irá para o passo "Registro dos erros".

[1] *Print Screen* significa tirar uma foto do que se está na tela do computador. Esta foto pode ser armazenada como um arquivo.

Registro dos erros – Os erros encontrados no ambiente de homologação devem ser registrados para futuras correções. Os erros não serão gerenciados pela GCS, mas esta informação é importante para métricas relacionadas ao desempenho de cada sistema.

Existe ambiente de contingência? – Nesse passo, o gerente de sistemas repassará o fluxo para implantação no ambiente de contingência, se existir. Esse passo não está sincronizado com o ambiente produtivo como garantia, caso a nova versão apresente uma falha grave que não aconteceu inicialmente no ambiente de homologação.

Registro pelo *Librarian* da configuração para contingência – Os mesmos arquivos utilizados nos ambientes de homologação e produção devem ser promovidos no ambiente de contingência.

Implementação no ambiente de contingência – Os procedimentos executados no ambiente de homologação e produção serão executados no ambiente de contingência.

Evidências da implantação – O objetivo das evidências é o mesmo de produção e homologação, mas no caso de contingência, o primordial é permitir uma fiscalização, se necessário.

Evidências da validação – O ambiente de contingência é uma cópia do ambiente de produção e essa validação é necessária para garantir que o ambiente esteja operacional. A validação pode ser realizada pela equipe de TI e o usuário realiza a validação completa durante exercícios programados chamados de DR (Desastre e Recuperação).

O fluxo apresentado na Figura 6.2 e na Figura 6.3 é de uma implementação normal.

Necessidade do usuário e É uma alteração emergencial? possuem as mesmas características do fluxo emergencial.

Especificação – Esse passo é dedicado para que o gerente de sistemas crie a especificação ou inclua informações relevantes ao time de desenvolvedores.

Estimativas para o desenvolvimento – Baseado nas informações do usuário responsável e do gerente de sistemas o time de desenvolvedores estimará o tempo necessário para desenvolvimento e as necessidades técnicas para realizar a implementação.

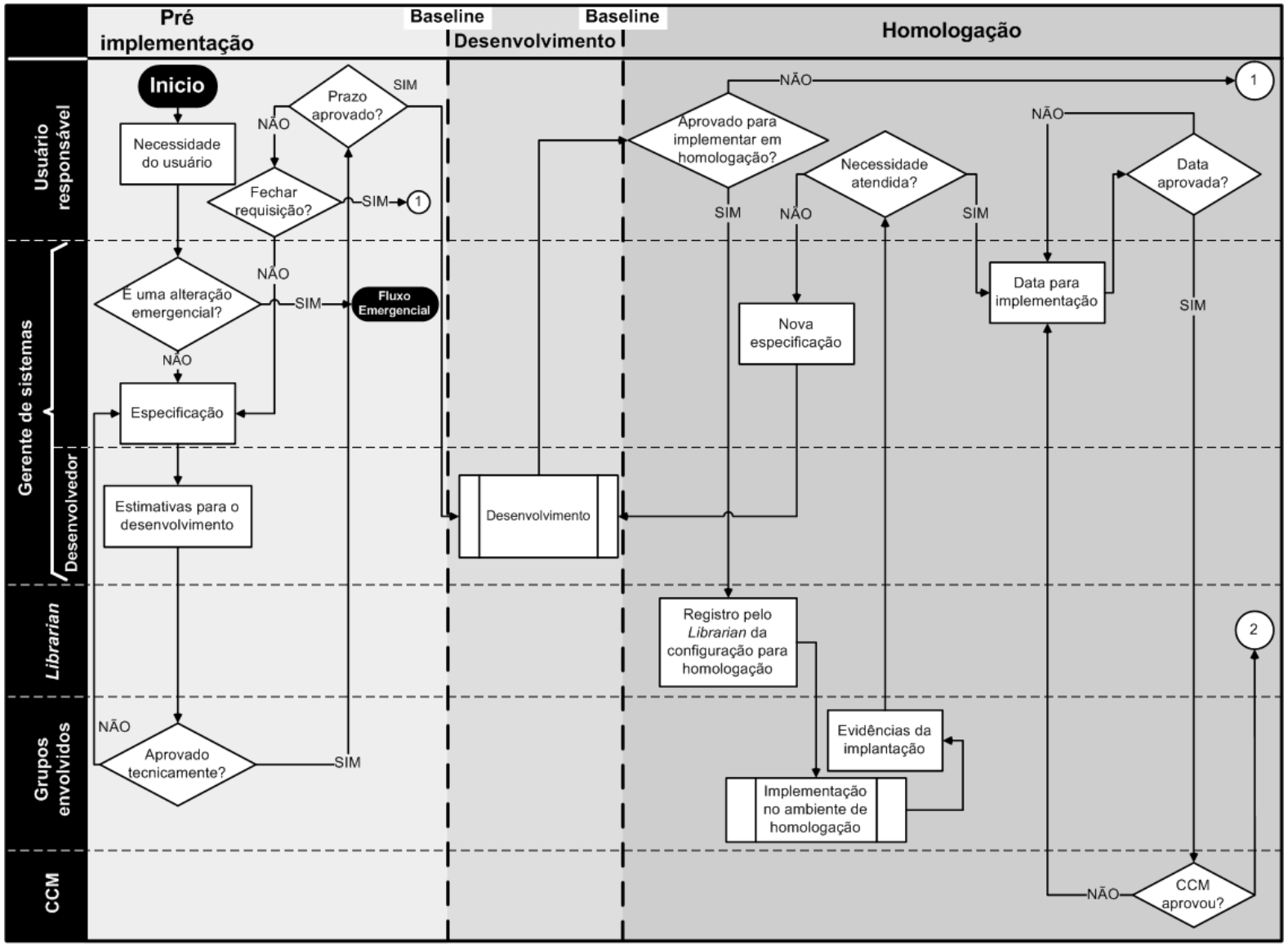

Figura 6.2 - Fluxo de GCS para implementações normais.

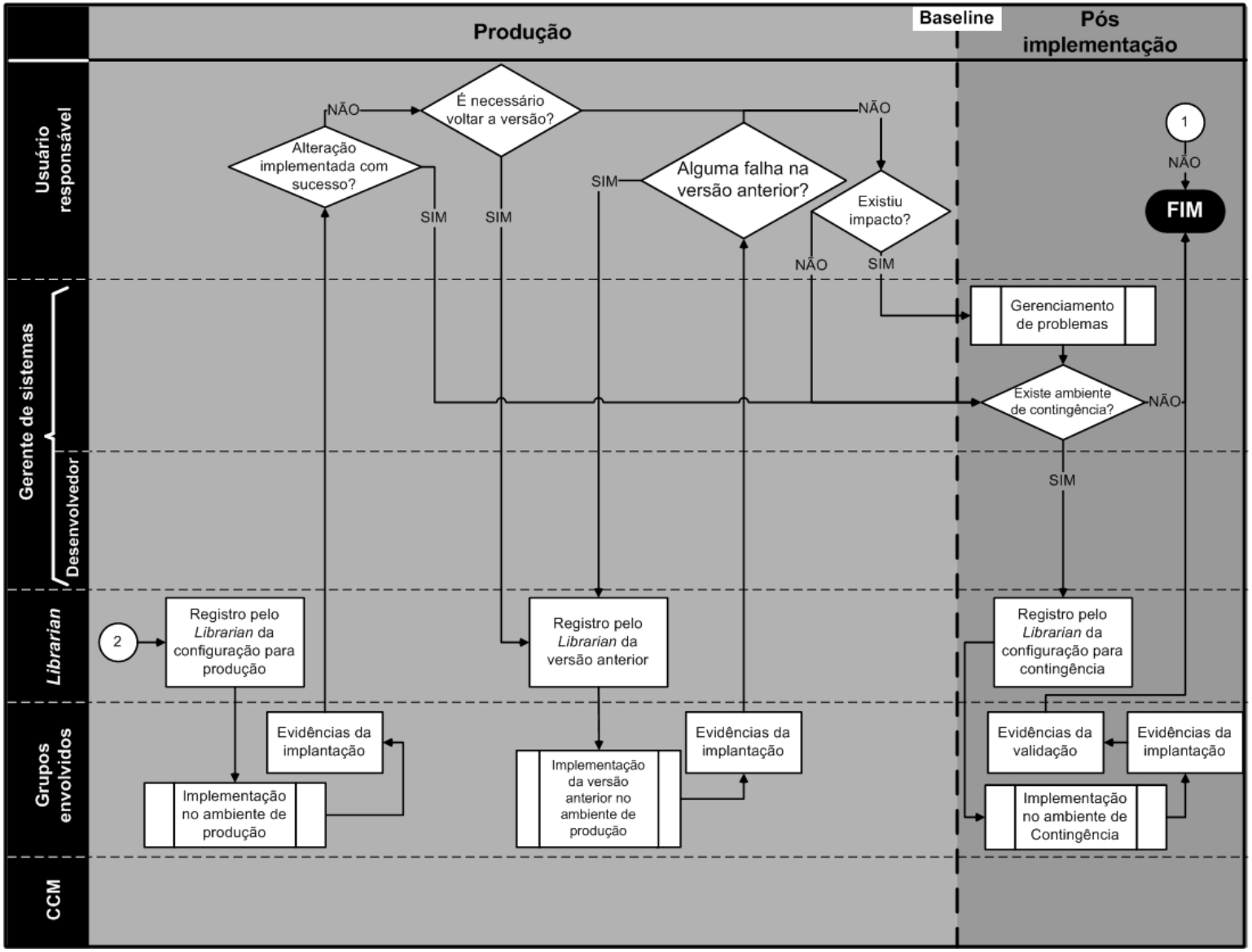

Figura 6.3 - Fluxo de GCS para implementações normais. (cont.)

Aprovado tecnicamente? – Antes de iniciar o desenvolvimento, é necessário que os grupos envolvidos aprovem tecnicamente a implementação. O gerente de sistemas é o responsável por envolver todos os grupos necessários. O objetivo desse passo é de garantir que as alterações do *software* sejam suportadas no ambiente produtivo, ou que alterações necessárias sejam planejadas com antecedência. Por exemplo: O time de desenvolvedores pretende incluir funções no *software* que necessita de uma versão mais nova do banco de dados. Se as alterações técnicas não forem aprovadas, o fluxo voltará para especificação e as alterações no *software* devem ser revistas para se adequarem às possibilidades do ambiente produtivo. Caso as alterações técnicas sejam aprovadas, o fluxo seguirá para o próximo passo.

Prazo aprovado? – Nesse passo, o usuário responsável analisará a estimativa de prazo e se for aprovada, será iniciado o desenvolvimento. Caso a implementação não seja aprovada, o fluxo seguirá para o passo Fechar Requisição.

Fechar requisição? – É possível que o usuário responsável, baseado nas estimativas de tecnologia, não queira continuar com a implementação. Se necessário, a requisição será fechada. Caso contrário, a requisição seguirá para a especificação novamente, para que sejam feitos novos ajustes nas estimativas.

Desenvolvimento - O processo de desenvolvimento não será controlado pelo fluxo proposto, mas é representado nesse passo. A interação do time de desenvolvimento com a unidade de análise é feita através do gerente de sistemas. A saída desse passo é o *software* e a documentação necessária para realizar a implementação.

Aprovado para implementar em homologação? – Nesse passo, o usuário responsável aprovará o inicio da implementação no ambiente de homologação.

Registro pelo *Librarian* da configuração para homologação – Após a aprovação, o *software* e a documentação devem ser registrados na ferramenta de versionamento. A ferramenta de versionamento garantirá, em caso de necessidade, a volta da versão anterior. Outra vantagem do versionamento é a de que, no momento de reinstalar o *software* em um ambiente novo, com as versões guardadas, é possível instalar o *software* novamente desde o início, sem a necessidade de pacotes especiais.

Os **passos Implementação no ambiente de homologação** e **Evidências da implantação** possuem as mesmas definições do fluxo emergencial.

Necessidade atendida? – O usuário responsável informará se a implementação atendeu às necessidades ou não. Se a necessidade foi atendida, o fluxo seguirá para o passo Data para implementação. Caso a especificação não tenha atendido as necessidades dos usuários, o fluxo seguirá para o passo Nova especificação.

Nova especificação – Esse é o momento em que as falhas encontradas serão analisadas para que a especificação seja corrigida. Após este passo, o fluxo seguirá para a equipe de desenvolvimento que alterará o *software*.

Data para implementação – Nesse momento, o gerente de sistemas definirá a data para implementação no ambiente produtivo.

Data aprovada? – Aqui, o usuário responsável analisará a data de implantação em produção. Se o usuário não aprovar a implementação nesta data, o fluxo volta para o passo Data para implementação, caso contrário, segue o fluxo para aprovação do CCM.

CCM aprovou? – Nesse passo, o CCM coordenará as implementações programadas, procurando conflito entre elas. O objetivo é que os grupos envolvidos estejam cientes das implementações e que sejam capazes de realizar as atividades sem dificuldades externas.

Registro pelo *Librarian* da configuração para produção – Os mesmos arquivos utilizados no ambiente de homologação devem ser promovidos no ambiente de produção. O objetivo é evitar que durante a manipulação dos arquivos do *software* algo seja modificado.

Os passos **Implementação no ambiente de produção** e **Evidências da implantação** possuem as mesmas definições do fluxo emergencial.

Alteração implementada com sucesso? – Essa etapa serve para garantir que usuário responsável realize as validações necessárias no ambiente de produção e verifique se o *software* está funcionando corretamente. Caso a implementação não tenha sido implementada com sucesso, será necessário seguir o fluxo para o passo É necessário voltar a versão?.

É necessário voltar a versão? – Uma vez que a implementação não tenha sido realizada com sucesso, é necessário que a situação seja avaliada. Este passo serve para que o usuário responsável avalie a necessidade de voltar ou não à versão anterior do *software*. Caso necessário, o fluxo seguirá os passos para voltar a versão anterior, caso contrário seguirá para o passo Existe ambiente de contingência?.

Registro pelo *Librarian* da versão anterior – Para que a versão anterior seja implementada com sucesso, é necessário que os componentes anteriores estejam disponíveis. A função do *Librarian* neste passo é a de garantir que a versão anterior fique disponível para que grupos envolvidos possam voltar a ela, caso seja necessário.

Implementação da versão anterior no ambiente de produção – Nesse passo, os grupos envolvidos executarão o plano da volta da versão anterior. Para garantir que a volta será executada corretamente, eles deverão seguir o plano de volta da versão anterior, criado pelo time de desenvolvimento.

O passo **Evidências da implantação** possui a mesma definição do fluxo emergencial.

Alguma falha na versão anterior? – Nesse passo, é necessário que o usuário responsável realize a validação da versão anterior para garantir que o *software* esteja funcionando corretamente. Caso seja encontrada alguma falha na versão anterior, o fluxo seguirá para uma nova implementação da versão anterior. Caso nenhuma falha seja encontrada o fluxo seguirá para o passo Existiu Impacto?.

Existiu Impacto? – Neste passo o usuário responsável determinará se houve impacto no negócio em função da falha. Caso exista alguma falha, o fluxo seguirá para o processo de Gerenciamento de problemas caso contrário irá para o passo Existe ambiente de contingência?

Gerenciamento de problemas – O processo de gerenciamento de problemas não será controlado pelo fluxo proposto, mas esse passo serve para garantir que o impacto sofrido seja devidamente registrado e conduzido pelas regras de gerenciamento de problemas da instituição.

Os passos: **Existe ambiente de contingência?, Registro pelo *Librarian* da configuração para contingência, Implementação no ambiente de Contingência, Evidências da implantação** e **Evidências da validação** possuem as mesmas definições do fluxo emergencial.

7 Conclusões e considerações finais

Este trabalho apresentou uma proposta de um roteiro para a execução de um processo de gerenciamento de *software* em um ambiente corporativo e conforme resultados apresetados no capitulo *Proposta do roteiro* este objetivo foi alcançado.

Há dois focos de contribuição: teórico e prático. As contribuições teóricas envolvem um melhor conhecimento dos aspectos mais relevantes relacionados com a implantação de GCS em ambiente corporativo. Do ponto de vista prático, o trabalho contribui para que profissionais que pretendem implementar o processo GCS tenham um roteiro para obtenção (ou complementação) de conhecimento e sua aplicação. Os resultados obtidos, também trazem contribuição aos profissionais já envolvidos com o processo da GCS, mas que carecem de aprimorar ou aperfeiçoar algumas de suas funções.

Conclusões

O objetivo geral - propor um roteiro para Gerência de Configuração de *Software* - foi desenvolvido em etapas, resultando no roteiro apresentado no capítulo 6. No capítulo 2, foi apresentada uma introdução sobre a GCS com base em pesquisa bibliográfica, que, em conjunto com as informações obtidas no capítulo 5, por meio de estudo de caso, foram usados para o desenvolvimento do roteiro.

Toda empresa, que tenha um parque de informática, implementa novas versões dos seus sistemas. A utilização do roteiro proposto proporcionará maior controle nas atividades necessárias até o ambiente produtivo.

Para uma empresa que não utiliza nenhum método para GCS, o roteiro exige: adoção de novos procedimentos; pessoal qualificado, para utilizar as ferramentas corretamente; comprometimento da gerência, coordenação com os usuários para garantir que o fluxo será seguido. Utilizando o roteiro corretamente os benefícios potenciais são:

- Redução dos erros humanos envolvidos no processo de implementação: o roteiro proposto apresenta um fluxo definido e suportado por ferramentas, automatizando as atividades e contribuindo para a redução de erros durante a manipulação dos arquivos;
- Redução dos riscos: o roteiro apresenta fluxos de trabalho, com aprovação dos envolvidos, aprovação nos testes e evidência da conclusão da implementação, que podem reduzir o risco de uma implementação;
- Obtenção de um histórico, e conseqüentemente a criação de um banco de dados sobre as implementações executadas ou a serem executadas, com os acontecimentos obtidos

durante todo o processo: com a utilização do repositório proposto é possível viabilizar o histórico das implementações.

Percebeu-se durante o estudo de caso que a participação do usuário no fluxo de GCS é essencial. O usuário será afetado caso o *software* não funcione, portanto algumas decisões no fluxo devem ser do usuário responsável.

Outra observação durante o estudo de caso é a utilização de quatro ambientes: desenvolvimento, homologação, produção e contingência. Na unidade de análise, o ambiente de desenvolvimento é criado quando necessário, mas os ambientes de homologação, produção e contingência são criados para todos os *softwares*. A adoção de um ambiente de pré-produção, chamado de homologação na unidade de análise, com as mesmas regras e controles aplicados em ambiente produção, se mostrou eficaz no combate à falhas. O ambiente de homologação irá proporcionar maior realidade para os usuários e durante a implementação em homologação os grupos envolvidos têm a oportunidade de validar os procedimentos, prevendo pontos de falhas e o tempo necessário para realizar as atividades no ambiente produtivo.

Para conduzir o processo de GCS de forma consistente, é necessário que se tenha um fluxo definido. A ferramenta de GCS deve permitir adaptações no fluxo, pois as adaptações irão contemplar novos passos alinhando com os objetivos empresariais e as estratégias de negócio da empresa e evitar que atividades sejam realizadas fora do fluxo controlado.

Sugestão de pesquisas futuras

O roteiro proposto tem como foco um processo de Gerência de Configuração de *Software* aplicado em ambiente produtivo de meio corporativo, abstraindo o processo de desenvolvimento. Uma pesquisa complementar, seria a de implementar os passos necessários para realizar a gerência de configuração integrada desde o desenvolvimento, integrada com o fluxo de GCS proposto.

Tanto o processo de GCS proposto quanto a sua ampliação (englobando o processo de desenvolvimento) merecem mais estudos de campo, voltados para avaliá-lo em ambientes diferenciados de negócios. Esses estudos podem ser realizados tomando o modelo proposto como referência e aplicando-os nos vários ambientes.

Outra proposta para pesquisa futura é a de análise de ferramentas de GCS, livres e proprietárias, comparando suas funcionalidades e avaliando em que medida contempla cada fluxo definido no roteiro proposto.

8 Referências

BABICH, Wayne A. ***Software configuration management coordination for team productivity***. - [S.l.]: Addison-Wesley, 1986.

BECK, Kent. **Programação extrema (XP) explicada.** Tradução de Adriana P.S. Machado; Natália N.P. Lopes. Porto Alegre, BRA. Bookman, 2004.

BRYAN, William; CHADBOURNE, Christopher; SIEGEL, Stan. ***Software configuration management.*** 1980. Califórnia, EUA: IEEE *Computer Society Press*, 1980.

CMPE - Change Management Process Education, 2007. Hortolândia, SP: IBM, 2007.

ESTUBLIER, Jacky et al. ***Impact of the Research Community for the Field of Software Configuration Management.*** [S.l.]: IEEE *Computer Society Press*, 2002.

ESTUBLIER, Jacky et al. ***Impact of software engineering research on the practice of software configuration management.*** [S.l.]: ACM Press (TOSEM), Volume 14 Issue 4, 2005.

GEIA. ***National Consensus Standard for Configuration Management.*** Arlington, EUA: GEIA - *Government Electronics and Information Technology Association*, 2004 (Relatório Técnico: ANSI/EIA 649-A).

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa.** 4.ed. São Paulo, BRA: Editora Atlas, 2002.

IEEE Std 1028-1997. ***IEEE Standard for Software Reviews***. [S.l.]: IEEE *Computer Society Press*, 1997.

IEEE Std 610.12-1990, ***IEEE Standard Glossary of Software Engineering Terminology***. [S.l.]: IEEE *Computer Society Press*, 1990.

IEEE Std 828-2005, ***IEEE Standard for Software Configuration Management Plans***. New York, EUA: *IEEE Computer Society*, 2005.

IEEE. ***SWEBOK Guide to the software engineering body of knowledge***. 2004 version. Califórnia, EUA: IEEE *Computer Society Press*, 2004. Disponível em: <http://www.swebok.org>. Acesso em: julho de 2006.

KEYES, Jessica. ***Software configuration management***. [S.l.]: CRC Press LLC, 2004.

LEON, Alexis. ***Software Configuration Management Handbook.*** 2.ed. Massachusetts, EUA: Artech House, 2005.

LIENTZ, Bennet P. ***Issues in software maintenance***. Califórnia, EUA: ACM Press, 1983.

MARCONI, Marina de A; LAKATOS, Eva M. **Fundamentos de metodologia científica**. 6.ed. - 3. Reimpr. São Paulo, BRA: Editora Atlas, 2006.

MEI, Hong; ZHANG, Lu; YANG, Fuqing. ***A Software Configuration Management Model for Supporting Component-Based Software Development***. Beijing, CHN: ACM Press, 2001.

OGC. ***ITIL Foundation***. 2004. GBR: *OCG - Office of Government Commerce*, 2004. Disponível em: < http://www.ogc.gov.uk>. Acesso em: Abril de 2007.

PACHECO, Renato F.; SANCHES, Rosely. **Gerenciamento de configuração de *software***. São Carlos, BRA: ICMSC-USP, 1997 (Relatório Técnico nº57).

PARIKH, Girish. ***Exploring the world of software maintenance: what is software maintenance?***. Chicago, EUA: ACM Press, ACM SIGSOFT Software Engineering Notes, Volume 11, Issue 2, 1986.

PMBOK. ***A Guide to the project management body of knowledge***. 3.ed. Pennsylvania, EUA: [s.n.], 2004.

PING, Liang; YANG, Li Jian. ***A Change-Oriented Conceptual Framework Of Software Configuration Management***. Chengdu, CHN: IEEE *Computer Society Press*, 2007.

PRESSMAN, Roger S. ***Software engineering a practitioner's approach***. 6.ed. [S.l.]: McGraw-Hill, 2005.

SEAWLHO, Pornthep; SUWANNASART, Taratip. ***A SCM Workflow Model for CMM Organizations***. Bangkok, THA: IEEE *Computer Society Press*, 2003.

SEI - Software Engineering Institute. ***CMMI® for Development***. 1.2 version. Pittsburgh, EUA: [s.n.], 2006. Disponível em: <www.sei.cmu.edu/cmmi>. Acesso em: setembro de 2006.

SOMMERVILLE, Ian. ***Software engineering***. 8.ed. St. Andrews University, Scotland: Addison-Wesley, 2007.

TONINI, Antonio C. **A contribuição do seis sigma para a melhora dos processos de *software*.** 2006. Dissertação (Mestrado em Engenharia da Produção) – Escola Politécnica, Universidade de São Paulo, São Paulo, BRA, 2006.

WESTFECHTEL, Bernhard et al. ***A Layered Architecture for Uniform Version Management***. [S.l.]: IEEE *Computer Society Press*, 2001.

YIN, Robert K., **Estudo de caso Planejamento e métodos**. Tradução de Daniel Grassi. 3 ed. Porto Alegre, BRA: Bookman, 2005.

9 Bibliografia complementar

BASILI, Victor et al. ***Understanding and predicting the process of software maintenance release.*** [S.l.]: IEEE Computer Society, 1996.

BEN-MENACHEN, Ben-Menachem; MARLISS, Garry S. ***Inventorying Information Technology Systems:Supporting the "Paradigm of Change".*** [S.l.]: IEEE *Computer Society Press*, 2004.

BERCZUK, Steve. ***Pragmatic Software Configuration Management.*** [S.l.]: IEEE *Computer Society Press*, 2003.

CERVO, Amado L.; BERVIAN, Pedro Alcino. **Metodologia científica.** 4.ed. São Paulo, BRA: Makron *Books*, 1996.

CHOU, I-Hsin; FAN, Chin-Feng. ***A Regulatory Software Maintenance Environment Using Agent-Based Software Configuration Management.*** TWN: IEEE *Computer Society Press*, 2006.

ESTUBLIER, Jacky. ***Software Configuration Management: A Roadmap.*** Genebra, FRA: ACM Press, 2000.

IEEE Std 1219-1998. ***Standard Glossary of Software Engineering Terminology***. [S.l.]: IEEE *Computer Society Press*, 1998.

IPT - Instituto de Pesquisas Tecnológicas do Estado de São Paulo. **Guia para elaboração da Dissertação de Mestrado,** 2.ed, São Paulo, BRA: [s.n.], 2005.

JANERT, Philipp K. ***A Software Configuration Management Primer.*** [S.l.]: IEEE *Computer Society Press*, 2004.

JANSEN, Slinger; BRINKKEMPER, Sjaak. ***Integrated Development and Maintenance of Software Products to Support Efficient Updating of Customer Configurations: A Case Study in Mass Market ERP Software***. [S.l.]: IEEE *Computer Society Press*, 2005.

Li, Ruan; YONG, Zhong. ***A New Configuration Management Model for Software Based on Distributed Components and Layered Architecture***. Chengdu, CHN: IEEE *Computer Society Press*, 2003.

SHEN, Haifeng; SUN, Chengzheng. ***A Complete Textual Merging Algorithm for Software Configuration Management Systems***. [S.l]: IEEE *Computer Society Press*, 2004.

VISCONTI, Marcello; GUZMIN, Liliana. ***A Measurement-Based Approach for Implanting SQA & SCM Practices***. Valparaíso, CHL: IEEE *Computer Society Press*, 2000.

VOLZER, Hagen et al. ***SubCM: A Tool for Improved Visibility of Software Change in an Industrial Setting***. [S.l.]: IEEE *Computer Society Press*, 2004.

WANG, Xiuli; WANG, Yongji; ZHOU, Hui. ***QSCM: Engineering QoS in Web-based Software Configuration Management System***. [S.l.]: IEEE *Computer Society Press*, 2006.

10 Apêndices

Apêndice 1: Questionário de pesquisa.

Este documento contém as questões de pesquisa para o estudo de caso referente à dissertação de mestrado: “Gerência de configuração de *software*: um estudo de caso”.

O documento está dividido em três grupos:

- Introdução
- Dados sobre a unidade de análise;
- Dados referentes à Gerência de Configuração de *Software*.

Introdução

O pesquisador, Carlos Eduardo D. Barros, é aluno regular do IPT – Instituto de Pesquisas Tecnologias do Estado de São Paulo e participante do programa de mestrado. Como forma de compreender as dificuldades e facilidades encontradas na GCS, foi elaborado o questionário de pesquisa. O questionário de pesquisa será aplicado com um grupo de pessoas para atender aos princípios de aleatoriedade e confiabilidade dos dados, os quais serão devidamente revisados pelo pesquisador.

Todas as informações obtidas neste processo serão utilizadas única e exclusivamente neste trabalho e serão mantidas sob extremo sigilo e confidencialidade.

Dados sobre a unidade de análise

Os dados sobre a empresa têm a finalidade de descrever a unidade de análise e o ramo de negócio. Estes dados servem para descrever o cenário das questões relativas à Gerência de Configuração de *Software*.

Os dados a serem questionados são os seguintes:

- Qual a principal atividade da empresa?
- Qual a quantidade de funcionários, estagiários e colaborados da empresa?
- Qual a quantidade de clientes?
- Quais são as exigências ou características da empresa?
- Qual a dependência da tecnologia para o funcionamento da empresa?
- Quais são as políticas de segurança: regras e procedimentos?
- Qual a localização geográfica da empresa?

Dados referentes à Gerência de Configuração de Software.

Este grupo de questões diz respeito à Gerência de Configuração de Software e aos processos envolvidos para sua execução.

As questões são as seguintes:

- Os ambientes de desenvolvimento, homologação e produção estão fisicamente separados?
- O fluxo de GCS é realizado através de uma ferramenta de GCS?
- O time de desenvolvimento participa do fluxo da GCS?
- Os usuários estão envolvidos no fluxo da GCS?
- Existe controle de versão nos itens controlados na GCS?
- O fluxo da GCS é conhecido e documentado?
- Como são armazenadas as aprovações durante o fluxo de GCS?
- Quais são os relatórios sobre os dados da GCS?
- Os relatórios gerados são suficientes para a gerência e auditoria da GCS?
- Quais foram os treinamentos, cursos ou palestras sobre a GCS?