FUNDAÇÃO DE ENSINO EURÍPIDES SOARES DA ROCHA
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

MARLIANE ALDIVINA OLIVEIRA FERREIRA

MR.GC-PARFAIT: Modelo de Referência para a Atividade de Gerência de Configuração do PARFAIT

MARÍLIA
2007

MARLIANE ALDIVINA OLIVEIRA FERREIRA

MR.GC-PARFAIT: Modelo de Referência para a Atividade de Gerência de Configuração do PARFAIT

Dissertação apresentada ao Programa de Mestrado do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, para obtenção do título de Mestre em Ciência da Computação (Área de Concentração: Engenharia de *Software*).

Orientadora: Prof[a]. Dr[a]. Maria Istela Cagnin Machado

MARÍLIA
2007

Dedico esta dissertação a Deus.

AGRADECIMENTOS

Agradeço a Deus pela sabedoria e pela oportunidade desse aprendizado.

À minha querida orientadora Dra. Maria Istela Cagnin Machado pela paciência, dedicação e delicadeza nas orientações.

Aos meus professores pelo ensinamento.

Aos meus pais pelo apoio, incentivo e confiança.

À minha irmã Marlette pela ajuda e colaboração durante este trabalho.

Ao meu irmão, à minha cunhada e aos meus sobrinhos que souberam entender a minha ausência.

Ao meu namorado que conviveu e incentivou toda essa jornada.

Aos meus amigos do mestrado Ana Claudia, Fabrício, Ivair, José Ivo, Juliana, Laurindo, Marco, Silvia, Sérgio, Richard e Vânia pelo companheirismo, pelos momentos felizes e de descontração vividos.

Aos meus amigos que estiveram me apoiando e compreenderam a minha ausência.

A todas as pessoas que ajudaram direta ou indiretamente na realização deste sonho.

“As reticências são os três primeiros passos do pensamento que continua por conta própria o seu caminho”(MÁRIO QUINTANA).

“Não existe um caminho para a felicidade. A felicidade é o caminho”(GANDHI).

“Nunca ande pelo caminho traçado, pois ele conduz somente até onde os outros foram”(GRAHAM BELL).

“Digno de admiração é aquele que tendo tropeçado ao dar o primeiro passo, levanta-se e segue em frente”(CARLOS FOX).

FERREIRA, Marliane Aldivina Oliveira. **MR.GC-PARFAIT: Um Modelo de Referência para a Atividade de Gerência de Configuração do PARFAIT**. 130 f. Dissertação (Mestrado em Ciência da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007.

RESUMO

A Gerência de Configuração (GC) é de vital importância para garantir a qualidade do *software*, pois auxilia o seu desenvolvimento apoiando o controle das mudanças e das versões dos artefatos elaborados. A busca pela qualidade não é diferente no contexto de processos de reengenharia de *software*, no entanto há carência no tratamento de GC nesses processos. PARFAIT é um processo ágil de reengenharia baseado em *framework* que também não fornece tratamento da GC de forma explícita e, devido à sua característica incremental, versões do sistema alvo são liberadas a cada iteração do processo. A problemática do controle de versões quando se utiliza *framework* como apoio computacional, como é o caso do processo PARFAIT, se torna maior uma vez que é necessário controlar tanto as versões do *framework* quanto as versões do sistema alvo. Assim, este trabalho tem como objetivo estabelecer um modelo de referência para a atividade de GC do PARFAIT, tomando como base as normas e os modelos de maturidade existentes. Um estudo de caso de reengenharia de um sistema legado, utilizando o processo PARFAIT e o modelo de referência proposto, é conduzido a fim de analisar se tal modelo não afeta a agilidade do processo e se apóia efetivamente o controle de versões do *framework* e do sistema alvo, mantendo a integridade dos mesmos.

Palavras-chaves: Gerência de Configuração, Qualidade de Processo, Reengenharia Ágil e PARFAIT.

FERREIRA, Marliane Aldivina Oliveira. **MR.GC-PARFAIT: A reference model of the activity of Configuration Management of the PARFAIT.** 130 f. Dissertação (Mestrado em Ciência da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007.

ABSTRACT

Configuration Management (CM) is too important in order to guarantee software quality, because it helps its development by supporting change control and elaborated artifacts versions. Searching for quality is not different in context of software reengineering processes, however there is a lack in CM treatment in these processes. PARFAIT is an agile reengineering process based on framework which does not supply CM treatment in an explicit way either and, due to its incremental characteristic, target system versions are delivered to each process iteration. Version control problematic becomes greater when it is used framework as computational support, as it is PARFAIT process case, since it is necessary to control as the framework versions as the target system ones. Thus, this work has as aim to establish a reference model for PARFAIT CM activity, by taking as base norms and existing maturity models. A reengineering case study of a legacy system, using PARFAIT process and the reference model proposed, is lead in order to analyze if such a model does not affect the process agility and to check if it supports effectively versions control of framework and target system, by keeping their integrity.

Keywords: Configuration Management, Process Quality, Agile Reengineering and PARFAIT.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ARA	Arcabouço de Reengenharia Ágil
CELEPAR	Companhia de Informática do Paraná
CenPRA	Centro de Pesquisa Renato Archer
CESAR	Centro de Estudos e Sistemas Avançados de Recife
CMM	*Capability Maturity Model*
CMMI	*Capability Maturity Model Integration*
COPPE/UFRJ	Departamento de Computação da Universidade Federal do Rio de Janeiro
CRF	C*hance Request Form*
CVS	*Concurrent Versions System*
FaPRE/OO	Família de Padrões de Reengenharia Orientada a Objeto
GC	Gerência de Configuração
GG	*Generic Goal*
GP	*Generic Practice*
GQM	*Goal Question Metric*
HTML	*HyperText Markup Language*
IEEE	*Institute of Electrical and Electronics Engineers*
IPD-CMM	*Integrated Product Development Capability Maturity Model*
ISO	*International Standards Organization*
LPA	Linguagem de Padrões de Análise
MA-MPS	Método de Avaliação do MPS.BR
MAS	Modelo de Análise do Sistema
MASA	Modelo de Análise do Sistema Atual
MCT	Ministério da Comunicação e Tecnologia
MPS.BR	Melhoria de Processo do *Software* Brasileiro
MR-MPS	Modelo de Negócio do MPS.BR
MR.GC-PARFAIT	Modelo de Referência para a atividade de Gerência de Configuração do PARFAIT
OO	Orientação a Objetos

PARFAIT	Processo Ágil de Reengenharia baseado em *Framework* no domínio de sistemas de Informação com técnicas de VV&T
PREF	Processo de Evolução de *Framework*
PMBOK	*Project Management Body of Knowledge*
PMI	*Project Management Institute*
PMQ	*Project Management Quarterly*
PRE/OO	Processo de Reengenharia Orientada a Objeto
RCS	*Revision Control System*
RIOSOF	Sociedade Núcleo de Apoio à Produção e à Exportação de *Software* do Rio de Janeiro
RUP	*Rational Unified Process*
SECM	*Systems Engineering Capability Model*
SEPIN	Secretaria de Política de Informática do Ministério da Ciência e Tecnologia
SG	*Specific Goal*
SOFTEX	Associação para Promoção da Excelência do *Software* Brasileiro
SP	*Specific Practice*
SW-CMM	*Capability Maturity Model for Software*
TR	Teste de Regressão
UML	*Unified Modeling Language*
VV&T	Verificação, Validação e Teste

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto

O progresso da Engenharia de *Software* carrega consigo a preocupação com a incorporação da qualidade no desenvolvimento, proporcionando uma melhoria no produto final entregue aos usuários. Para garantir essa qualidade durante o ciclo de vida do *software* são empregadas diversas atividades de apoio, dentre elas destaca-se a Gerência de Configuração (GC) (SOMMERVILLE, 2003; PRESSMAN, 2006) que será tratada neste trabalho.

Constata-se que mudanças no *software* são inevitáveis durante e após o seu desenvolvimento, as quais podem ser classificadas em quatro tipos: adaptativa, perfectiva, corretiva e preventiva. Na primeira, o sistema alvo é adaptado a novas tecnologias tanto de *software* quanto *hardware*. Na segunda, novas funcionalidades relacionadas às regras de negócio ou às mudanças nas políticas governamentais são adicionadas. Na terceira, erros ocorridos durante o desenvolvimento ou nas mudanças realizadas posteriormente são corrigidos. Na quarta, melhorias no sistema com o intuito de facilitar futuras correções e aumentar a confiabilidade são providas. Assim, a GC é empregada como atividade de apoio, cuja responsabilidade está presente na identificação, na documentação, no armazenamento, no controle e no relato das mudanças ocorridas nos artefatos elaborados durante e depois do desenvolvimento do sistema (CUNHA *et al.*, 2004; PRESSMAN, 2006). Apesar da importância de GC, poucas empresas a utilizam devido ao custo, tempo e a ausência de profissionais qualificados (PRESSMAN, 2006).

A importância da qualidade também não é diferente no contexto de reengenharia ágil, que entrega uma versão do sistema alvo funcionando adequadamente o mais rápido possível. Para que isso ocorra, o usuário participa ativamente durante todo o projeto, que é realizado de forma incremental. Casos de testes funcionais são utilizados para apoiar o entendimento do sistema legado[1] e são empregados posteriormente para testar o sistema alvo (novo sistema) (CAGNIN, 2005a). A reengenharia reimplementa o sistema legado levando em consideração as funcionalidades

[1] É um sistema desatualizado que pode não ter ou ter apenas uma parte da documentação adequada para a manutenção. Esse sistema permanece em uso pela dificuldade de substituí-lo e de modificá-lo, bem como do custo em transformá-lo em um novo sistema (sistema alvo) (CAGNIN, 2005; ESPINDOLA *et al.*, 2004).

e as regras de negócio nele contidas (SOMMERVILLE, 2003; PRESSMAN, 2006), podendo incorporar novas funcionalidades de acordo com as necessidades dos usuários.

Pode-se observar a carência na literatura da atividade de GC em processos de reengenharia (CAGNIN, 2005a; LEMOS *et al.*, 2003; FONTANETTE *et al.*, 2002a), pois, como no desenvolvimento de *software*, é necessário se preocupar com o controle sistemático das modificações que devem ser realizadas durante a reengenharia, a fim de produzir um sistema alvo com qualidade.

Ressalta-se ainda a problemática no controle de versão em processos ágeis de reengenharia baseados em *framework*[2], uma vez que é necessário controlar as mudanças e as versões tanto no *framework* quanto nos sistemas gerados por ele. Salienta-se que, como a arquitetura do sistema gerado passa a englobar o próprio *framework* (ARIMOTO *et al.*, 2007), modificações sem precaução no *framework* podem prejudicar o comportamento dos sistemas gerados, fazendo com que os mesmos deixem de fornecer o comportamento desejado.

Para apoiar a atividade de GC tanto no desenvolvimento quanto na reengenharia de *software* é imprescindível a utilização de ferramentas específicas, como é o caso da *Revision Control System* (RCS) (TICHY, 1997), da *Concurrent Versions System* (CVS) (XIMBIOT, 2005), da GNU *Arch* (TOSUN, 2004) e da *Subversion* (COLLINS-SUSSMAN *et al.*, 2006).

1.2 Motivação

A GC é um dos itens principais que levam à garantia da qualidade de *software* por meio do controle das mudanças e das versões dos artefatos. O PARFAIT (**P**rocesso **Á**gil de **R**eengenharia baseado em ***Fr****A****mework*** no domínio de sistemas de **I**nformação com técnicas de verificação, validação & **T**este) (CAGNIN, 2005a) é um processo ágil de reengenharia que contém algumas diretrizes que tratam da GC, no entanto, não possui atividades específicas de GC. Essa carência foi observada durante alguns estudos de caso utilizando tal processo (CAGNIN *et al.*, 2004; CAGNIN *et al.*, 2003b). Assim, é necessário minimizar essa carência sem afetar a agilidade do processo, de acordo com o tratamento de GC provido pelas normas e modelos de qualidade existentes na

[2] Permite o desenvolvimento de sistemas poupando tempo e esforço, por meio do reúso de diretrizes durante toda a fase do desenvolvimento de *software*.

literatura. O PARFAIT utiliza *framework*, portanto, é necessário um tratamento especial das atividades de GC, levando-se em consideração que, além de gerenciar as versões dos artefatos produzidos durante a reengenharia, devem-se gerenciar as versões do *framework* e as versões do sistema gerado a partir dele.

1.3 Objetivos

Este trabalho tem como objetivo elaborar um modelo de referência para a atividade de GC do PARFAIT, sob a perspectiva dos diferentes modelos e normas de maturidade de *software* existentes, por exemplo, ISO/IEC 12207 (1998), ISO/IEC 15504 (1999), ISO/IEC 10007 (2005), ISO/IEC TR 15846 (1998), CMMI (SEI, 2001), MR-MPS (SOFTEX, 2005) e PMBOK (PMI, 2004). Para isso é observada a ocorrência das atividades de GC em cada norma/modelo, com o apoio do *Goal Question Metric* GQM (BASILI *et al.*, 1994), a fim de definir as melhores práticas para o PARFAIT, sem afetar a sua agilidade. Para avaliar o modelo de referência definido é conduzido um estudo de caso de reengenharia, a fim de observar se afeta a agilidade do processo PARFAIT e se mantém a integridade das versões do *framework* e do sistema alvo. Salienta-se que o processo PARFAIT emprega atividades de VV&T para o entendimento das funcionalidades do sistema legado e, posteriormente, as utiliza no sistema alvo a fim de validá-lo e de verificar se o mesmo possui comportamento similar ao do sistema legado.

Deve-se levar em consideração que, além destas normas e modelos estudados ainda existem algumas que tratam da GC como é o caso da ISO/IEC 15846 (1998), que apresenta um relatório técnico definido pela norma ISO/IEC 12207, da IEEE Std 828 (1998), que apresenta os artefatos que devem ser abordados pela GC e da IEEE 1042 (1986), que apresenta um guia de apoio para a aplicação da GC. No entanto, tais normas não serão tratadas neste trabalho.

1.4 Organização do Trabalho

Este trabalho está organizado em sete capítulos e três apêndices, como descritos a seguir.

O presente capítulo contextualizou a importância da gerência de configuração como atividade de apoio tanto no desenvolvimento como na reengenharia, e delimitou a problemática existente no controle de versões durante a reengenharia ágil baseada em *framework.*

No Capítulo 2 apresentam-se conceitos e a importância de processos de *software*, tanto para o desenvolvimento quanto para a reengenharia. Também se confere um enfoque especial ao processo ágil de reengenharia PARFAIT, que é de interesse deste trabalho.

No Capítulo 3 são abordados os conceitos para o entendimento de gerência de configuração, as atividades principais para executá-la e algumas ferramentas de apoio encontradas na literatura.

No Capítulo 4 apresenta-se a descrição de algumas normas e modelos de maturidade encontrados na literatura sob a perspectiva da atividade de GC.

No Capítulo 5 é observado o procedimento adotado na escolha das atividades e dos artefatos essenciais de GC para o PARFAIT com o apoio do método GQM (*Goal Question Metric*) e da "métrica grau de importância" criada nesta dissertação. A partir desta seleção compõe-se o modelo de referência denominado MR.GC-PARFAIT, proposto neste trabalho, apresentando um detalhamento das atividades de GC estabelecidas para esse modelo e a aplicabilidade das mesmas no PARFAIT.

O Capítulo 6 consiste da condução de um estudo de caso quantitativo de um sistema legado para analisar a viabilidade do emprego do MR.GC-PARFAIT, em conjunto com o processo PARFAIT, e verificar se tal modelo não afeta a agilidade desse processo e se a integridade dos artefatos produzidos é mantida. Adicionalmente, apresenta-se a melhoria de uma abordagem de reúso de teste existente, denominada ARTe, que foi obtida com a criação de casos de teste genéricos.

Finalmente, no Capítulo 7 encontram-se as conclusões do trabalho realizado, destacando-se as suas limitações e perspectivas de trabalhos futuros.

No Apêndice A são apresentadas a documentação completa do modelo MR.GC-PARFAIT, contendo a descrição das atividades de GC e da aplicabilidade das mesmas nas atividades do PARFAIT.

No Apêndice B apresenta-se a versão final do diagrama de casos de uso e do diagrama de classe criados no estudo de caso conduzido neste trabalho.

O Apêndice C reúne os casos de teste genéricos criados neste trabalho e que foram incorporados à abordagem ARTe, aumentando o seu potencial de apoio ao reúso de teste.

2 PROCESSO DE *SOFTWARE*

2.1 Considerações Iniciais

Neste capítulo encontra-se o embasamento teórico relacionado a processos de *software* para a compreensão deste trabalho. Na Seção 2.2 são apresentados os conceitos de processo de desenvolvimento e de reengenharia de *software*. Além disso, são abordados alguns processos de reengenharia, destacando a carência existente no tratamento de Gerência de Configuração por tais processos. E, finalmente, na Seção 2.3 são apresentadas as considerações finais deste capítulo.

2.2 Processo de Desenvolvimento de *Software*

O processo de desenvolvimento de *software* possui várias definições na literatura. Para Pfleeger (2004), é chamado de ciclo de vida do *software* que descreve a produção de um *software* desde o surgimento da proposta de desenvolvimento até o seu desuso. Já Sommerville (2003) define processo de desenvolvimento de *software* como um agregado de atividades e de resultados que irá proporcionar a produção do *software*. Pressman (2006) descreve como um procedimento que elabora o *software*, incluindo os métodos, as técnicas e as ferramentas de apoio. Peters; Pedrycz (2001) definem o processo de desenvolvimento de *software* como uma seqüência de etapas com *feedback* que resultam na produção e na evolução do *software*. Segundo Fuggeta (1997 apud FIORINI, 2001), trata-se de "um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos que são necessários para conceber, desenvolver, entregar e manter um produto de *software*". Assim, o ciclo de vida examina, entende, controla, aprimora e padroniza as atividades de desenvolvimento do *software*.

A vantagem de se empregar um processo padronizado é a redução de tempo referente ao treinamento e ao compartilhamento das experiências que contribuem para o aprendizado (PFLEEGER, 2004; SOMMERVILLE, 2003). Pressman (2006), Sommerville (2003) e Pfleeger (2004) descrevem as principais atividades que compõem um processo de *software*: especificação

dos requisitos; projeto de desenvolvimento; implementação das funcionalidades dos requisitos; verificação; validação e teste (VV&T), bem como manutenção do mesmo.

A melhoria no processo de desenvolvimento de *software* tem sido alvo de pesquisas dentro da Engenharia de *Software* e visa aprimorar as técnicas e os modelos utilizados. Para alcançar essa melhoria, as atividades dos processos são executadas com o emprego de padrões e de diretrizes, com base nas normas ISO/IEC 15504 (1999), ISO/IEC 12207 (1998), ISO/IEC 10007 (2005) e nos modelos CMMI (SEI, 2001), MR-MPS (SOFTEX, 2005) e PMBOK (PMI, 2004) que foram estudados neste trabalho para apoiar a definição do modelo de referência proposto e são apresentados no Capítulo 3 sob a perspectiva de GC.

Os padrões aplicados no processo indicam como um modelo de processo deve ser seguido durante todo o desenvolvimento do *software*; desta forma, oferece uma solução para resolver os problemas que são encontrados, possibilitando compartilhar as experiências dos desenvolvedores com o auxílio de uma documentação (PFLEEGER, 2004).

Os modelos de processo de desenvolvimento de *software* vieram suprir a necessidade de auxiliar a criação do *software*. Com o emprego desses modelos pode-se desenvolver um *software* com qualidade. Nesta seção é apresentado o modelo de processo incremental, que é de interesse deste trabalho por ser utilizado pelo PARFAIT.

O modelo de processo de desenvolvimento incremental foi sugerido por Mills *et al.* (1980 apud SOMMERVILLE, 2003) e apresenta uma seqüência linear das atividades a cada incremento do modelo, além disso, proporciona a participação direta do usuário na indicação de quais funcionalidades têm prioridade de desenvolvimento (SOMMERVILLE, 2003). As atividades que compõem cada incremento são: análise, projeto, implementação e teste, como ilustrado na Figura 1.

As vantagens apresentadas por este modelo, além de ser incremental e iterativo, são: oferecer uma visão antecipada dos riscos; facilitar a criação de medidas para controlá-los ou eliminá-los; facilitar as alterações futuras; bem como possuir um conjunto de regras que auxilia o controle dessas alterações. Entre as desvantagens destaca-se que o incremento deve ser relativamente pequeno (SOMMERVILLE, 2003; PRESSMAN, 2006).

No desenvolvimento e na reengenharia incremental deve haver controle de versão em cada incremento. Visando facilitar esse controle e proporcionar a integração das versões produzidas em cada incremento para compor o produto final, emprega-se a GC que é discutida no Capítulo 3.

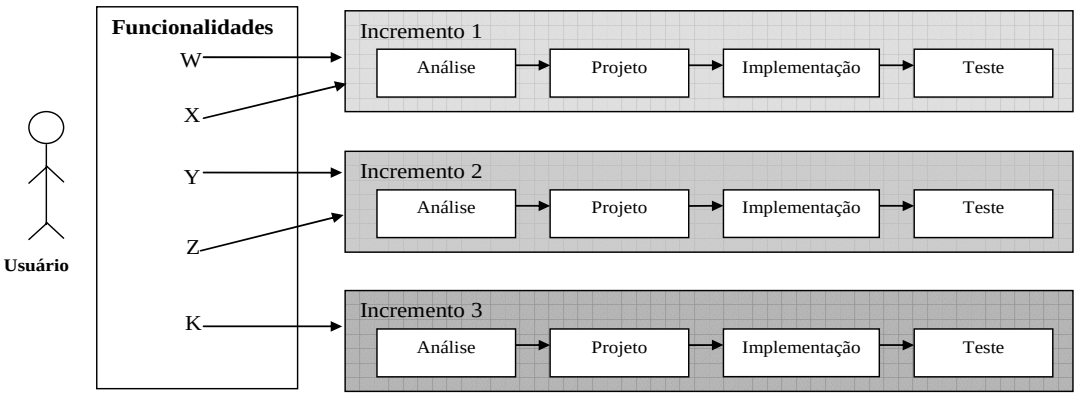

Figura 1 - Modelo de processo de desenvolvimento incremental (adaptado de PRESSMAN, 2006)

2.2.1 Processos de Reengenharia de *Software*

Salienta-se que processos de *software* também são importantes no contexto de reengenharia, pois apóiam a migração de um sistema existente, denominado sistema legado (obsoleto), para um novo sistema, denominado sistema alvo.

Em geral os sistemas legados estão em uso há muito tempo e as constantes atividades de manutenção neles realizadas podem colaborar para a degradação do código fonte e para a desatualização da documentação. Isso faz com que novas atividades de manutenção se tornem inviáveis ou até mesmo impossíveis de serem conduzidas, pois a única documentação existente é o próprio código fonte desestruturado (POSTEMA; SCHIMIDT, 1998). Portanto, o sistema legado necessita ser evoluído para adequar-se tanto às necessidades dos usuários e clientes quanto às de *hardware* e de *software*.

Existem várias definições de reengenharia, entretanto, a mais utilizada é a de Chikofsky e Cross (1990), que definiram termos relacionados: engenharia avante, engenharia reversa, reestruturação e reengenharia, descritos a seguir.

A engenharia avante consiste na transferência das abstrações, dos modelos lógicos e do projeto do sistema para uma implementação física. Enquanto que a engenharia reversa refere-se a um processo de análise do sistema legado, que identifica seus componentes, seus relacionamentos e os apresenta em um nível mais alto de abstração. A engenharia reversa pode ser classificada como redocumentação e recuperação do projeto.

A redocumentação consiste em criar ou revisar uma representação da abstração semântica do sistema, como por exemplo, o fluxo de dados e de controle, o diagrama de entidades e relacionamentos (DER) e a listagem de referência cruzada dos elementos do código fonte.

A recuperação do projeto identifica no sistema as abstrações de alto nível, isto é, o domínio do conhecimento e das informações externas e daquelas obtidas diretamente pela análise do sistema, a fim de fornecer completo entendimento sobre “o quê” o sistema faz, “como” ele faz e “por que” ele faz de uma determinada maneira.

A reestruturação consiste na transformação da representação do *software* para outra no mesmo nível de abstração, preservando o seu comportamento externo, isto é, a sua funcionalidade e semântica. A reestruturação pode ser denominada, também, de refatoração (FOWLER *et al.*, 1999).

A reengenharia é conhecida também como renovação e consiste na análise e alteração do sistema para reconstruí-lo em uma nova forma. Geralmente inclui engenharia reversa seguida por engenharia avante ou reestruturação.

Os principais objetivos da reengenharia foram apresentados por Sneed; Nyary (1995 apud CAGNIN, 2005a) e constituem-se de: melhoria da manutenibilidade do sistema; migração do sistema; uso de novas tecnologias para alcançar a confiabilidade e preparação do sistema para aumentar a sua funcionalidade.

Dentre as definições apresentadas, a de reengenharia é de interesse deste trabalho. Desta forma, alguns processos de reengenharia serão apresentados com maiores detalhes neste capítulo.

A seguir encontram-se dois processos (PRE/OO e PARFAIT) que utilizam a GC parcialmente durante a reengenharia e um outro (PRBCD) que não emprega essa atividade de apoio.

2.2.1.1 Processo de Reengenharia Orientada a Objeto (PRE/OO)

O PRE/OO é uma evolução da família de padrões de reengenharia denominada FaPRE/OO (RECCHIA *et al.*, 2002) e permite a reengenharia de sistemas legados desenvolvidos em *Delphi* procedimental para o paradigma orientado a objetos na linguagem *Object Pascal* (BORLAND, 2000) com o apoio do ambiente *Delphi*. As adaptações realizadas no FaPRE/OO para permitir essa evolução ocorrem nos seguintes padrões: reconstruir a arquitetura do sistema legado; recuperar o

Modelo de Análise do Sistema Atual (MASA)[3] e criar o Modelo de Análise do Sistema (MAS)[4]. Sendo assim, deve-se mapear o MAS para o MASA, elaborar o projeto avante e particionar o sistema. A *Unified Modeling Language* (UML) (BOOCH *et al.*, 2000) foi utilizada no PRE/OO para representar a documentação do sistema alvo (LEMOS *et al.*, 2003).

PRE/OO é composto por sete grupos de padrões de reengenharia, os quais foram divididos em padrões para a melhoria da qualidade do processo de reengenharia orientada a objetos (grupos 1 e 2) e padrões para a condução da reengenharia orientada a objetos (grupos 3 a 7). A melhoria da qualidade do processo considerada pelos grupos 1 e 2 foi baseada no nível 2 de maturidade do *Capability Maturity Model* (CMM) (SEI, 2001). Os demais grupos tomaram como base o método de engenharia reversa Fusion/RE (PENTEADO, 1996) e os padrões da FaPRE/OO.

O primeiro grupo de padrões tem como objetivo preparar e planejar o processo de reengenharia, ou seja, determinar quais artefatos serão desenvolvidos e planejar as atividades de reengenharia a serem executadas.

O segundo grupo de padrões visa à melhoria da qualidade do processo de reengenharia, mantendo e acompanhando o processo com as inspeções de garantia de qualidade e com o controle da configuração, que estabelece e mantém a integridade dos artefatos elaborados. Dentre as atividades de GC foram empregadas três neste processo: 1) identificar os itens de configuração; 2) documentar todas as mudanças dos itens de configuração com o auxílio de um cabeçalho que disponibiliza as seguintes informações: o responsável pela mudança, a data, o nome do item de configuração, a descrição da mudança, a versão e a origem (ou seja, passo do processo em que a versão do artefato foi produzida); e 3) estabelecer a criação de uma *baseline* ao final da execução de cada passo do processo (LEMOS *et al.*, 2003). Assim, foi constatada a presença destas atividades de GC, no entanto, ainda há necessidade de um aprimoramento do processo PRE/OO nesse sentido.

O terceiro grupo de padrões cria a arquitetura do sistema legado com a elaboração da documentação dos procedimentos e funcionalidades do sistema legado, obtendo o entendimento desses.

O quarto grupo modela os dados do sistema legado com a construção do Diagrama de Entidades e Relacionamentos (DER), relaciona as anomalias referentes aos dados identificados e cria uma visão Orientada a Objetos (OO) desses dados. Além disso, cria o diagrama de casos de uso do MASA contendo a descrição das funcionalidades e do comportamento do sistema legado

[3] Sistema legado.
[4] Sistema alvo.

(LEMOS *et al.*, 2003). Salienta-se que as anomalias são identificadas segundo quatro critérios: 1) procedimento ou função que observa a tabela de dados; 2) procedimento ou função que altera a tabela de dados; 3) procedimento ou função relacionado a implementação; e 4) procedimento ou função que observa e/ou altera duas ou mais tabelas de dados.

O quinto grupo apóia a elaboração de um Modelo de Análise do Sistema alvo (MAS) com a abstração do diagrama de pseudo-classes, a criação do diagrama de casos de uso do MAS baseado na visão OO do sistema legado e a descrição dos casos de uso do MAS a partir da reespecificação das descrições do sistema legado, mas com alteração na abstração (LEMOS *et al.*, 2003).

O sexto grupo de padrões visa construir um projeto avante que contém a elaboração do diagrama de classe e do diagrama de seqüência. Esse último irá documentar a lógica do negócio do sistema legado (LEMOS *et al.*, 2003).

O sétimo grupo tem como objetivo implementar as classes, resultantes da aplicação dos padrões anteriores, bem como as regras de negócio, o encapsulamento dos métodos de acesso aos dados e a interface gráfica do sistema alvo (LEMOS *et al.*, 2003).

2.2.1.2 Processo Ágil de Reengenharia baseado em *Framework* no domínio de sistemas de Informação com técnicas de VV&T (PARFAIT)

O PARFAIT (CAGNIN *et al.*, 2003a) é um processo ágil, incremental e iterativo, que visa migrar sistemas legados procedimentais para o paradigma orientado a objetos. Ele é um dos recursos do Arcabouço de Reengenharia Ágil (ARA) que apóia a reengenharia ágil baseada em linguagens de padrões (APPLETON, 1997) e *frameworks* (FAYAD *et al.*, 1999). *Framework* é considerado uma aplicação “semicompleta”, com componentes estáticos e dinâmicos que com sua instanciação geram aplicações específicas a um domínio para o usuário (FAYAD *et al.*, 1999).

A Linguagem de Padrões de Análise (LPA) é utilizada pelo PARFAIT como apoio no entendimento do domínio do sistema legado e na documentação orientada a objetos desse sistema. A LPA consiste de um conjunto de informações sobre as decisões tomadas com a ajuda das regras e das diretrizes que serão transferidas por meio destas experiências adquiridas pelo desenvolvedor na solução de problemas encontrados no recorrer da análise (APPLETON, 1997).

A justificativa de utilização de *framework* pelo ARA, consequentemente pelo PARFAIT, é o apoio que este oferece na criação de uma versão do sistema alvo o mais rápido possível. O ARA conta também com a participação dos clientes durante todo o projeto de reengenharia para validar os artefatos produzidos, e também utiliza "reengenharia guiada por teste". Nessa última prática, casos de teste são criados por meio de critérios de teste funcionais (MYERS, 2004) e são executados no sistema legado para apoiar a engenharia reversa no entendimento das funcionalidades e na identificação de regras de negócio e, posteriormente, são utilizados para validar o sistema alvo. Salienta-se que as funcionalidades presentes no sistema legado podem ser evoluídas de acordo com as necessidades dos usuários.

Por ser considerado ágil, o PARFAIT utiliza diversas práticas de métodos ágeis, como por exemplo: versões pequenas, clientes presentes, testes constantes, jogo do planejamento, programação em pares, propriedade coletiva do código, integração contínua, metáfora e 40 horas semanais.

Além das práticas ágeis, o PARFAIT também trata da Modelagem Ágil (MA) (AMBELER, 2004) que, de acordo com CAGNIN (2005a), tem como objetivo principal, no contexto do PARFAIT, documentar o sistema legado com agilidade, promovendo o entendimento e facilitando a comunicação entre os engenheiros de *software*. Para isso, o processo PARFAIT fornece gabaritos dos artefatos que devem ser produzidos durante a reengenharia para reduzir o tempo de criação da documentação; motiva a participação dos usuários na validação dos artefatos; permite que os requisitos sofram modificações quando necessário; motiva a construção de um modelo simplificado e de fácil entendimento; constrói a documentação do sistema de maneira iterativa e incremental, levando em consideração as necessidades e prioridades do usuário; e cria diversos tipos de modelos para tratar de cada particularidade do sistema legado, de acordo com as necessidades do projeto, sendo que alguns são opcionais.

A Modelagem Ágil (MA) contribui para o emprego de um conjunto de práticas que gera uma documentação e uma modelagem eficaz, entretanto não descreve os procedimentos detalhados, mas oferece indicações de como modelar de maneira eficiente, pois mistura as práticas simples de modelagem com os artefatos essenciais de modelagem de *software* (AMBLER, 2004).

No PARFAIT há reúso em vários níveis de abstração (análise, projeto, implementação, teste e manutenção), que é proporcionado indiretamente pelo arcabouço ARA. Detalhes sobre o PARFAIT são obtidos em CAGNIN (2005a).

A documentação de todas as atividades do PARFAIT foi baseada no formato da documentação do *Rational Unified Process* (RUP) (KRUCHTEN, 2000), mas no contexto de reengenharia. Este formato é composto por fases, marcos de referência, atividades, passos, papéis, artefatos de entrada e de saída, disciplinas, gabaritos, diretrizes e guia de ferramenta de apoio computacional. A modelagem dos diagramas é feita em UML.

As fases do PARFAIT se dividem em: Concepção, Elaboração, Construção e Transição (CAGNIN, 2005a), como ilustrado na Figura 2.

A **Fase de Concepção** contém atividades que visam: a) familiarizar o engenheiro de *software* com o domínio do *framework* disponível; b) identificar o domínio do sistema legado; c) comparar o domínio do sistema legado em relação ao domínio do *framework* disponível, para que este possa ser utilizado como apoio computacional; e d) elaborar o planejamento de reengenharia baseado no conhecimento adquirido em projetos anteriores com características similares, estabelecendo os riscos, o custo e o tempo que serão gastos no projeto de reengenharia. Na elaboração do cronograma, aconselha-se considerar pares de profissionais trabalhando em conjunto, no máximo, 40 horas semanais. Ainda como parte do planejamento do projeto de reengenharia, o responsável pela reengenharia deve estabelecer quais artefatos serão submetidos ao controle de versão (itens de configuração).

A **Fase de Elaboração** contém atividades que: a) criam e executam os casos de teste funcionais no sistema legado para a sua compreensão, elaborando os diagramas de casos de uso; b) criam o diagrama de classes do sistema legado no paradigma OO com o emprego de padrões da linguagem de padrões de análise; e c) documentam as regras de negócio identificadas no sistema legado.

Ainda no contexto das atividades da Fase de Elaboração, ressalta-se que toda classe e relacionamento representados no diagrama de classes, mas não reutilizados dos padrões de análise da linguagem de padrões, são documentados em um artefato específico. Esse artefato é utilizado na próxima fase, durante a execução da atividade que adapta o sistema alvo em relação ao sistema legado, pois o sistema gerado a partir da instanciação do *framework* fornece apenas as funcionalidades cobertas pelo domínio da linguagem de padrões de análise. A atividade de adaptar o sistema alvo também se baseia na documentação das regras de negócio do sistema legado que, em geral, não são fornecidas pelo *framework*.

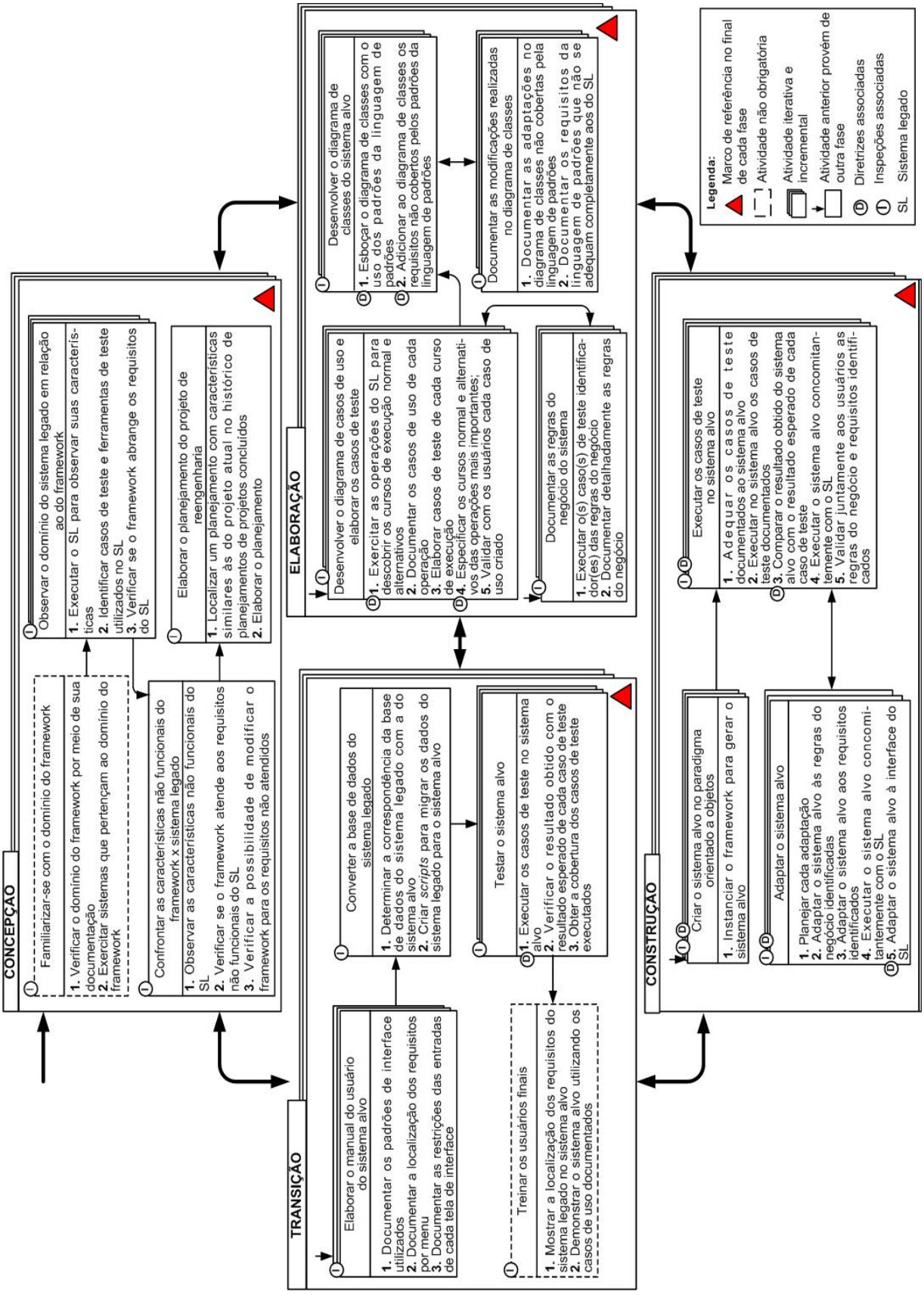

Figura 2 - Visão geral do PARFAIT (CAGNIN, 2005a)

A **Fase de Construção** possui atividades que têm por objetivo: a) instanciar o *framework*, promovendo o desenvolvimento de uma versão do sistema alvo o mais rápido possível; b) executar os casos de teste funcionais no sistema alvo, criados na Fase de Elaboração, para verificar a compatibilidade funcional do sistema alvo em relação ao sistema legado, permitindo identificar os requisitos e as regras de negócio do sistema legado que não foram cobertos pelo *framework*; e c) adaptar o sistema alvo para torná-lo funcionalmente semelhante ao legado.

Na **Fase de Transição** as atividades têm por finalidade: a) gerar o manual do usuário do sistema; b) substituir a base de dados do sistema legado para a do sistema alvo; c) treinar os usuários, caso seja necessário; e d) avaliar a maturidade do sistema alvo, executando todos os casos de teste documentados e observando a sua cobertura, caso seja de interesse.

Deve-se ressaltar que em cada fase apresentada anteriormente há um marco de referência que visa observar o cumprimento do projeto estabelecido na Fase de Concepção e pode estabelecer a continuidade ou não do projeto (CAGNIN, 2003).

O PARFAIT trata apenas de alguns itens da GC de forma manual, assim, observou-se a necessidade de aperfeiçoá-lo sob essa perspectiva. Dentre os itens de GC tratados pelo PARFAIT tem-se o controle de adaptações implementadas manualmente no código fonte das versões do sistema alvo. Essas adaptações são realizadas em cada iteração do processo, a fim de tornar o sistema alvo funcionalmente equivalente ao sistema legado. Sem esse controle, essas adaptações são perdidas em subseqüentes instanciações do *framework*. Quando o *framework* GREN é utilizado como apoio computacional do PARFAIT, a ferramenta *GREN–WizardVersionControl* (CAGNIN *et al.*, 2004a) é empregada para permitir esse controle.

Apesar de o PARFAIT preocupar-se com a identificação dos itens de configuração logo no início da aplicação do processo e ressaltar a importância de controlar as versões de todos os artefatos produzidos durante a reengenharia, não mostra como aplicar a GC de maneira sistemática.

Durante o uso do PARFAIT evoluções podem ser necessárias no *framework* para que o mesmo possa apoiar mais efetivamente a reengenharia. Para isso, sugere-se o uso do Processo de Evolução de *Framework* (PREF) que pode ser executado em paralelo ao PARFAIT e utiliza um histórico de requisitos funcionais e não funcionais de sistemas que não foram cobertos pelo domínio do *framework* para decidir se deve ou não conduzir a evolução do *framework* (CAGNIN *et al.*, 2004a). Entre as atividades apresentadas pelo PREF destaca-se a oitava atividade, que foi utilizada como base para apoiar o desenvolvimento deste trabalho. Esta atividade é denominada “Tratar do Gerenciamento de Controle de Configuração”, que deve ser executada com certo cuidado, pois

evoluções em *framework* podem mudar o seu projeto e, conseqüentemente, o dos sistemas gerados a partir da instanciação deles levando tais sistemas a deixarem de fornecer o comportamento desejado. É necessário estabelecer quando será feita uma determinada alteração, definir em quais versões do *framework* esta manutenção será introduzida, especificar se os sistemas gerados pelo *framework* devem ou não ser modificados para incorporar a evolução do *framework*. CAGNIN (2005a) ressalta que a GC no contexto de *framework* é similar a outro *software*, mas pode se tornar mais complexa devido a inúmeras versões do *framework* e ferramentas de apoio a ele associadas. Além disso, a GC deve ser aplicada tanto no *framework* quanto nas versões dos sistemas criados a partir dele.

Ressalta-se, ainda, o emprego da Abordagem de Reúso de Teste (ARTe) proposta por Cagnin *et al.* (2004). Esta abordagem tem como objetivo associar requisitos de teste a linguagens de padrões (LP) a fim de permitir o reúso não somente dos padrões, mas também dos requisitos de teste. A ARTe é um dos recursos disponíveis no ARA, e vem suprir a necessidade de otimizar as atividades de VV&T do PARFAIT. Tal abordagem é composta de duas etapas: na primeira são definidos os requisitos de teste funcional para cada padrão da LP com o apoio dos critérios de teste Particionamento de Equivalência e Análise de Valor Limite (MYERS, 2004); na segunda etapa, têm-se diretrizes para possibilitar o reúso tanto dos requisitos de teste definidos quanto das classes de equivalência criadas quando do uso dos padrões. Estudos de caso conduzidos (CAGNIN, 2005a) com e sem a abordagem ARTe mostraram uma redução de mais de 50% do tempo da reengenharia quando os requisitos de teste são reutilizados.

2.2.1.3 Processo de Reengenharia Baseado em Componente Distribuído (PRBCD)

Fontanette *et al.* (2002a, 2002b, 2002c) têm empregado a reengenharia com o apoio do método *Catalysis* (2006), que é baseado em Componentes Distribuídos, dos diagramas da UML, da ferramenta MVCASE (PRADO; LUCRÉCIO, 2001) e do Sistema Transformacional *Draco* (GARCIA *et al.*, 2002).

Os princípios adotados por Fontanette *et al.* (2002a) foram: abstração, precisão e componentes "*plug-in*". A abstração auxilia na busca das características indispensáveis ao sistema; a precisão tem como finalidade encontrar erros e problemas na modelagem; e os componentes

"*plug-in*" possibilitam a reutilização dos mesmos durante a reengenharia (FONTANETTE *et al.*, 2002a, 2002b, 2002c). Este processo utiliza o modelo de desenvolvimento espiral cuja principal característica consiste de uma seqüência de atividades que possui um retorno, em *loop*, para a próxima atividade (PFLEEGER, 2004; SOMMERVILLE, 2003).

Catalysis trata-se de um método de desenvolvimento de *software* baseado em componentes distribuídos. É composto pelos seguintes níveis lógicos: domínio do problema, especificação dos componentes e projeto interno dos componentes. Estes níveis correspondem às seguintes atividades que fazem parte deste processo: identificação dos requisitos, especificação, projeto e implementação (FONTANETTE *et al.*, 2002a, 2002b, 2002c).

O nível "domínio do problema" encontra os requisitos, determina soluções para os problemas e identifica tipos de objetos e ações, possibilitando diferentes combinações e visões por área de negócio. O nível "especificação dos componentes" identifica, determina o comportamento e a responsabilidade dos componentes. Este nível contém o aprimoramento dos modelos de negócio e do domínio do problema. Já o nível "projeto interno dos componentes" se restringe ao meio físico, isto é, aos requisitos não-funcionais e às distribuições físicas (FONTANETTE *et al.*, 2002a).

O Sistema Transformacional *Draco* (STD) é empregado em conjunto com o *Catalysis* com a finalidade de construir um *software* por transformação orientada a domínio, segundo o paradigma *Draco* que é divido em três partes: linguagem (apresenta as especificações do domínio e gera automaticamente a representação interna dos programas com as definições da gramática da linguagem, denominada *Draco Systax Abstract Tree* (DAST)); *prettyprinter* (transforma a DAST em texto na linguagem de domínio); e transformadores (mapeiam as estruturas sintáticas de uma determinada linguagem para outras estruturas e são responsáveis pela automação do processo de reconstrução do *software*) (FONTANETTE *et al.*, 2002a, 2002b).

Para realizar a reengenharia de *software* com o apoio das transformações empregam-se quatro passos: organizar código do sistema legado, recuperar projeto, reprojetar e reimplementar (PENTEADO, 1996).

No passo "organizar código do sistema legado" utiliza-se o *Draco* e as técnicas de Engenharia Reversa, que organizam as declarações e os comandos do código fonte do sistema legado sem prejudicar sua semântica e lógica.

O passo "recuperar projeto" parte do código do sistema legado, organizado anteriormente, e com o auxílio do *Draco*, gera as especificações em UML do projeto do sistema legado e as

armazena em descrições em *Modeling Domain Language* (MDL). Além disso, criam-se protótipos das respectivas classes na linguagem Java (FONTANETTE *et al.*, 2002a, 2002c).

O passo “reprojetar” tem o apoio da ferramenta MVCASE, desenvolvendo o reprojeto orientado a componentes distribuídos do projeto recuperado a partir do código do sistema legado. Neste passo utiliza-se o método *Catalysis* no domínio do problema, na especificação dos componentes e no projeto interno dos componentes (FONTANETTE *et al.*, 2002a, 2002b).

O passo “reimplementar” pode gerar automaticamente o código do sistema alvo de duas maneiras: a) transformando as especificações em MDL com o apoio do *Draco*; ou b) transformando os modelos de objetos e componentes de reprojeto com o apoio da ferramenta MVCASE (FONTANETTE *et al.*, 2002a).

Deve-se destacar que este processo não emprega atividades de GC e, portanto, nota-se a necessidade de aperfeiçoar tais atividades para atender aos processos de reengenharia de *software*, auxiliando o controle e a integridade dos artefatos gerados.

2.3 Considerações Finais

Neste capítulo foram apresentados os conceitos básicos sobre processo de *software* no desenvolvimento e na reengenharia. Foram mostrados alguns processos de reengenharia, dando ênfase ao processo PARFAIT, que é de interesse deste trabalho. Devido a isso, apresentou-se uma breve descrição desse processo, destacando os objetivos de cada uma de suas atividades. Além disso, ressaltou-se a deficiência desse processo em tratar da Gerência de Configuração de forma sistemática. Como o PARFAIT é baseado em *framework*, é necessário aplicar a GC não somente nos artefatos produzidos durante o uso do processo como também no *framework*, utilizado como apoio computacional pelo processo.

Observou-se que existem poucos processos de reengenharia presentes na literatura que aplicam a GC. Dentre os processos de reengenharia estudados, dois aplicam a GC (PRE/OO e PARFAIT), mas nenhum deles a considera de maneira sistematizada. Sob essa perspectiva, notou-se a necessidade de incorporar a GC de forma sistemática em processos de reengenharia, como é o caso do PARFAIT, pois a GC é um dos elementos importantes para a garantia da qualidade do *software*, não somente resultante do desenvolvimento como também da reengenharia.

No próximo capítulo aborda-se detalhadamente a GC, dada a importância neste trabalho do conhecimento sobre as atividades de GC, bem como dos artefatos que devem ser produzidos. Adicionalmente, discute-se a problemática da GC na reengenharia baseada em *framework* e citam-se algumas ferramentas existentes de controle de versão.

3 QUALIDADE DE *SOFTWARE* DO PONTO DE VISTA DE GERÊNCIA DE CONFIGURAÇÃO

3.1 Considerações Iniciais

Este capítulo apresenta os principais aspectos relacionados à Qualidade de *Software*, com ênfase na Gerência de Configuração e algumas ferramentas *open-source* de apoio, ao controle de versões existente na literatura. Na Seção 3.2 apresenta-se o conceito de Qualidade de *Software* com os padrões e diretrizes que são seguidos para garantir a qualidade no desenvolvimento. Na Seção 3.3 abordam-se os conceitos de Gerência de Configuração, os passos que são seguidos para gerenciar as mudanças nos artefatos criados e para controlar as versões e os mesmos. Na Seção 3.4 descrevem-se algumas ferramentas existentes de apoio à Gerência de Configuração, mais especificamente o controle de versões, e faz-se um estudo comparativo dessas ferramentas.

3.2 Qualidade de *Software*

Pressman (2006) descreve qualidade como “um procedimento para garantir a satisfação de padrões de desenvolvimento de *software*”, aplicando métodos, ferramentas, revisões, testes e controle de mudanças de seus artefatos. Já Tsukumo *et al.* (1997) apresentam a qualidade como a satisfação das necessidades dos requisitos explícitos e implícitos, ou seja, os requisitos informados e os necessários.

Natali; Falbo (2004) citam a grande dificuldade em alcançar a qualidade no desenvolvimento de *software*, ou seja, seguir o uso de padrões de desenvolvimento, localizar a causa das imperfeições e modificá-las. Para isto, é necessário um planejamento para avaliar a qualidade durante todo o desenvolvimento do *software*, estabelecendo como o projeto deve ser executado, controlado e avaliado pelo gerente do projeto. Ele deve ter em mente qual a prioridade do uso de ferramentas para o apoio no gerenciamento do desenvolvimento que indique o que avaliar, quando deve ser avaliado, quem será responsável pela avaliação. Finalmente, quando todas

essas questões são respondidas têm-se as diretrizes no desenvolvimento de *software* com a eficiência e com o conhecimento adquirido no decorrer do tempo, que pode diminuir os erros posteriores, criando um *software* com qualidade.

A qualidade no processo de desenvolvimento é obtida com o emprego de diretrizes em todo o ciclo de vida do *software*, de estruturas das organizações que desenvolvem o *software* e de atividades de suporte, que fornecem apoio durante todo o desenvolvimento. Constata-se que a qualidade do produto depende diretamente da qualidade do processo de desenvolvimento, verificando se o produto atende aos requisitos definidos com o emprego de uma avaliação sistemática que oferece apoio para determinar as etapas a serem seguidas para essa avaliação de qualidade, como pode ser o caso da ISO/IEC 14598-5[5] (TSUKUMO *et al.*, 1997).

O engenheiro de *software* tem um papel importante na garantia da qualidade, pois deve garantir o controle do processo de desenvolvimento com o emprego de revisões e testes. O grupo de Garantia da Qualidade do *Software* (SQA) auxilia a produção de um *software* por meio do planejamento das atividades, do emprego de padrões de desenvolvimento, da satisfação dos usuários, da documentação e da manutenção (PRESSMAN, 2006).

Jomori *et al.* (2004) afirmam que as empresas buscam a qualidade no desenvolvimento do *software* mediante o planejamento das atividades de desenvolvimento, assim, minimizam o número de defeitos e de falhas. Já Sant'Anna *et al.* (2002) destacam que a melhoria da qualidade e da confiabilidade do *software* pode reduzir custo e alcançar o aumento da produtividade nas organizações modernas.

Outro ponto marcante pelo emprego da qualidade é na reengenharia, pois a reengenharia transforma o sistema legado em um novo sistema evoluído tecnologicamente e pronto para ser utilizado pelo cliente. A padronização na criação da documentação e a possível reutilização do código no desenvolvimento do novo sistema possibilitam a diminuição do tempo, do custo e o aumento da qualidade. Deve-se acrescentar a Validação, a Verificação e o Teste (VV&T) nos artefatos produzidos durante a reengenharia para também garantir a qualidade no sistema alvo (CAGNIN, 2005a).

Na reengenharia baseada em *framework*, devido à problemática anteriormente mencionada, as atividades de GC devem ser mais criteriosas, pois o *framework* também sofre mudanças que

[5] ISO/IEC 14598-5, *International Standard. Information Techonology - Software product evaluation – Part 5: Process for evaluators*, 1996.

podem comprometer o comportamento dos sistemas gerados anteriormente com o apoio do *framework*.

Para garantir a qualidade do *software* tanto no desenvolvimento quanto na reengenharia apresentam-se algumas atividades de apoio que devem ser aplicadas durante todo o ciclo de vida do *software*, como é o caso da Gerência de Requisitos (HAZAN; LEITE, 2003; ESPINDOLA *et al.*, 2005; PRESSMAN, 2006), da Gerência de Projeto (SOMMERVILLE, 2003; PRESSMAN, 2006) e da Gerência de Configuração.

A Gerência de Configuração é de interesse deste trabalho por ser responsável pelo controle das mudanças ocorridas nos artefatos de *software*, e pelo controle das suas versões, como exposto na seção a seguir (PRESSMAN, 2006; SOMMERVILLE, 2003).

3.3 Gerência de Configuração de *Software*

A Gerência de Configuração (GC) identifica e controla a integridade, o armazenamento de todos os artefatos criados no desenvolvimento do *software* e expõe as modificações sofridas em cada versão do *software* e em seus respectivos artefatos. Além disso, facilita o controle das versões dos artefatos, pois várias versões podem estar em operação com diferentes configurações de *hardware* (PRESSMAN, 2006).

As principais atividades da GC, de acordo com Sommerville (2003), são: identificar os artefatos; criar um repositório de configuração; gerenciar as mudanças; gerenciar as versões; gerenciar as *releases*; efetuar revisões de GC; e definir a ferramenta de apoio à GC.

Sommerville (2003) e Pressman (2006) apresentam a GC como responsável por controlar as mudanças no ciclo de vida do *software*, desde o seu desenvolvimento até a sua manutenção, pois a necessidade de adaptação é inevitável. Assim, a GC é descrita como uma gerência que identifica, documenta, armazena, controla e relata as modificações ocorridas nos artefatos (IEEE, 1990 apud LOPES *et al.*, 2005).

A necessidade de manutenção é indiscutível durante a vida do sistema, seja para se adequar à plataforma e ao *hardware*, seja para corrigir os erros e mesmo para atender às necessidades do cliente. A fim de garantir a qualidade, aplica-se a padronização nos artefatos e o uso de ferramentas

para auxiliar tanto na organização quanto no desempenho da manutenção do sistema (SOMMERVILLE, 2003; PRESSMAN, 2006; ROCHA *et al.*, 2001).

Para realizar a manutenção são delegados alguns cargos (CUNHA *et al.*, 2004):

- **gerente do projeto:** decide e distribui as atividades do projeto;
- **engenheiro de *software*:** implementa e faz a manutenção em todos os artefatos;
- **comitê de controle de configuração:** define o número de pessoas, conforme o grau de complexidade, geralmente é formado pelo gerente de projeto e os responsáveis por cada módulo de desenvolvimento;
- **bibliotecário:** controla todos os artefatos armazenados;
- **responsável pelo gerenciamento de configuração:** efetua todo o controle de mudanças nos artefatos com o auxílio de ferramenta;
- **comitê de engenharia de processo:** coordena o processo de desenvolvimento;
- **auditor:** aplica os critérios de qualidade de *software* no projeto;
- **comitê de garantia de qualidade:** determina quais os critérios de aprovação serão empregados em todos os artefatos mantidos, possibilitando a criação da *baseline* (Seção 3.3.1).

Observa-se que a grande dificuldade em aplicar a GC está relacionada ao **custo** com equipamentos destinados apenas a essa atividade, com os programas específicos e com os encarregados aptos para exercer este controle de mudanças; à **cultura da empresa** na resistência em seguir regras e padrões; ao **aumento da formalidade e burocracia** com a solicitação de autorizações de mudanças com o emprego de formulários; à **falta de conhecimento** de GC; e à **ausência ou pouco comprometimento** presentes na empresa em desenvolver as atividades relacionadas à GC (OLIVEIRA *et al.*, 2001).

A principal vantagem em se aplicar a GC, apontada por Pressman (2006), é poder facilitar o aumento da produtividade e a diminuição dos erros com o gerenciamento dos artefatos, por meio da organização, a qual favorece a localização de um determinado artefato que será reutilizado ou até mesmo modificado. Assim, a GC existe para facilitar o controle das versões e de seus respectivos artefatos, durante todo o ciclo de vida do *software*.

Sommerville (2003) descreve a economia de tempo, durante e depois do desenvolvimento incremental, como a facilidade de gerar e obter versões diferentes de um mesmo produto, por meio de um histórico do artefato. O auxílio da GC no controle dos artefatos se torna rápido e preciso com o uso de ferramentas, sem as quais a aplicação de GC se tornaria impossível podendo gerar questões

de dificuldade de controlar vários artefatos com tempo e custo baixos (PRESSMAN, 2006; ROCHA *et al.,* 2001).

3.3.1 Conceitos importantes de GC

Os conceitos de GC, descritos nesta seção, visam esclarecer a citação dos termos empregados nesta dissertação.

- **artefatos ou itens de configuração:** são todas as documentações produzidas durante o desenvolvimento do *software*, por exemplo, documentos de requisitos, modelos de dados, modelos de análise, modelos de projeto, código fonte, executável, etc. (ROCHA *et al.*, 2001; CUNHA *et al.*, 2004).
- ***baseline:*** é apontada como um artefato utilizado como base para o desenvolvimento futuro, podendo ser alterado somente com o emprego do controle de mudança. Determina-se que este controle revise e aprove esta *baseline*, que é alocada em um repositório para manter a integridade dos dados (PRESSMAN, 2006; ROCHA *et al.*, 2001; CUNHA *et al.*, 2004; OLIVEIRA *et al.*, 2001) e a conformidade com os requisitos do projeto (BERSOFF, 1984). Peters; Pedrycz (2001) descrevem *baseline* como um artefato que já tenha sido revisado e aprovado. A definição de *baseline* apresentada por Leite (*et al.*, 1997 apud FIORINI; LEITE, 1998) determina um conjunto de artefatos aceitos e controlados que poderá ser utilizado posteriormente, sendo que sua evolução é obtida com as alterações, mediante uma solicitação e aprovação prévia.
- **repositório:** armazena os artefatos estabelecendo a criação da *baseline*, registra as informações e apóia a avaliação do impacto que as mudanças podem causar no sistema. O procedimento utilizado é o *check-in*/*check-out*. Isto é, quando há necessidade de alteração de alguma *baseline*, uma cópia do artefato é disponibilizada na área de trabalho do desenvolvedor (*check-out*), e após o término da alteração do artefato este é restituído no repositório (*check-in*), em seguida, uma nova *baseline* deve ser traçada. O responsável pela GC define os procedimentos para registrar e recuperar os artefatos solicitados (SOMMERVILLE, 2003).

- **versões:** são variações distintas de um mesmo *software* que podem ser determinadas por diferentes funcionalidades, desempenhos ou por uma diferente configuração de *software* ou de *hardware*. Caso sofram pequenas mudanças podem ser chamadas de variantes da versão original. Um sistema variante possui um conjunto de atributos e de alguns elementos que sofreram modificações, formando uma nova versão (SOMMERVILLE, 2003; PRESSMAN, 2006).
- ***release*:** é uma versão previamente distribuída aos clientes (SOMMERVILLE, 2003; OLIVEIRA *et al.*, 2001).

3.3.2 Atividades de Gerência de Configuração

Sommerville (2003) e Rocha *et al.* (2001) estabelecem o planejamento de GC por meio de padrões e procedimentos que podem ser aplicados em quaisquer empresas, podendo ser adaptados, se necessário.

No Quadro 1 encontram-se as atividades de GC que consistem, principalmente, em definir e identificar quais artefatos gerenciar; em criar um repositório que mantenha e garanta a integridade das *baselines*; em definir quais os procedimentos serão adotados para que ocorra a mudança nos artefatos, como garantir o controle das versões e *releases*, quais procedimentos devem ser adotados para se implantar a GC; e em definir qual ferramenta se enquadra melhor dentro das expectativas esperadas (SOMMERVILLE, 2003).

Atividades de GC	Objetivo
1 – Identificar os itens de configuração	Definir e identificar quais os artefatos que serão gerenciados.
2 – Criar um repositório de configuração	Armazenar e manter as *baselines*.
3 – Gerenciar as mudanças	Informar quais documentos e autorizações são necessários para que a mudança ocorra.
4 - Gerenciar as versões	Garantir o controle das versões.
5 – Gerenciar as *releases*	Garantir o controle das *releases*.
6 – Efetuar revisões de GC	Determinar quais procedimentos serão adotados para implantar a GC.
7 – Definir a ferramenta de apoio à GC	Definir qual ferramenta é adequada ao controle de mudanças no sistema.

Quadro 1 - Atividades de gerenciamento de configuração (SOMMERVILLE, 2003)

3.3.2.1 Identificar os itens de configuração

Retomando a definição de itens de configuração, anteriormente apresentada na Seção 3.3.1, que se refere a todo tipo de documentação que possa servir para consultas futuras, deve-se ter cautela na identificação de quais os itens de configuração são importantes para todo o ciclo de desenvolvimento do *software* (SOMMERVILLE, 2003; PRESSMAN, 2006).

Para melhor identificar tais itens de configuração, Pressman (2006), Sommerville (2003) e Rocha *et al.* (2001) sugerem a necessidade de determinar quais documentos devem ser controlados, investigar os planos de projeto, as especificações, os projetos, os programas, os conjuntos de dados de teste e analisar todos os documentos necessários para realizar a modificação do *software*. O cuidado na evolução dos artefatos deve ser levado em conta, podendo criar um histórico desta evolução.

3.3.2.2 Criar um repositório

O repositório armazena as informações sobre todos os artefatos, como já foi abordado na Seção 3.3.1, referenciando a versão do sistema, organizando e delimitando os tipos de consultas e descrevendo quais consultas podem ser feitas no repositório sobre uma determinada versão do *software*, como é mostrado no Quadro 2.

Tipos de consultas realizadas no repositório
1 – Quais clientes receberam uma determinada versão do sistema?
2 – Quais *hardware*s e sistemas operacionais são específicos para uma determinada versão?
3 – Quantas e quais as datas das versões criadas?
4 – Se um componente for modificado, quais versões podem ser afetadas?
5 – Quantas solicitações de mudanças podem ser realizadas para uma determinada versão?
6 – Quantos defeitos uma versão específica contém?

Quadro 2 - Consultas típicas no repositório de *baseline* (adaptado de SOMMERVILLE, 2003)

Para melhor entender os procedimentos ocorridos no repositório apresenta-se a Figura 3, a qual ilustra um conjunto de Tarefas de Engenharia de *Software* que desenvolve os artefatos que sofrem as revisões técnicas e formais, isto é, os artefatos são validados, verificados e testados garantindo a sua consistência e integridade. Após a aprovação de cada artefato, é feito o *check-in* e o seu armazenamento no repositório, criando uma *baseline*. Existe também a recuperação de uma *baseline* por meio do *check-out.* O histórico das *baselines* permite estabelecer a finalidade de cada uma e reutilizá-la para a criação de uma nova versão tanto do artefato quanto do *software*.

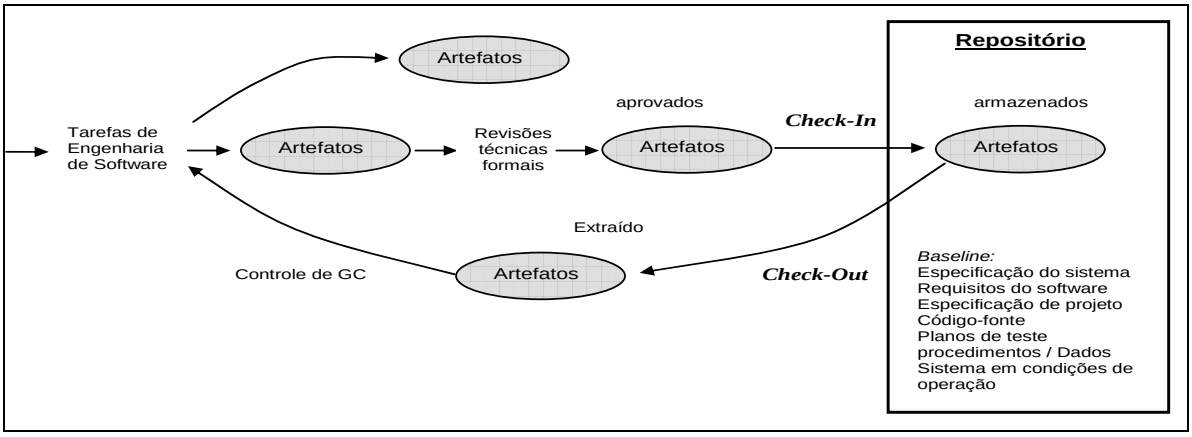

Figura 3 - Procedimentos que ocorrem no repositório (PRESSMAN, 2006)

3.3.2.3 Gerenciar mudanças

Inicialmente será necessário delegar a responsabilidade a um determinado encarregado (podendo ser o engenheiro de *software*) para o preenchimento do formulário de solicitação de mudanças (CRF - *chance request form*), apresentado no Quadro 3, que apresenta e registra a modificação requerida no sistema. Nesse quadro consta também o custo da mudança, a data da solicitação da mudança para a submissão ao comitê de controle de mudanças, a decisão tomada por esse comitê, bem como a data dessa decisão, a implementação da mudança (se for o caso), as datas de submissão efetuada para a garantia de qualidade e para a GC. Após a solicitação é ponderado e decidido pelo comitê de garantia da qualidade ou pelo gerente de projeto se tal mudança deve ser

atendida ou não, sendo a decisão determinada por uma análise de impacto[6]. Se a solicitação não for atendida, então o motivo deve ser encaminhado ao responsável que fez a requisição da modificação (SOMMERVILLE, 2003; CUNHA *et al.,* 2004).

Formulário de Solicitação de Mudanças

Projeto: Nome do projeto **Número:** Número da versão
Requisitante da Mudança: Nome do requisitante **Data:** data da solicitação
Mudança solicitada: nome do arquivo que contém o artefato que será modificado

Analista da Mudança: Nome do responsável pela mudança
Artefatos afetados: Nome dos artefatos que serão afetados pelas mudanças
Artefatos associados: Nome dos artefatos associados a mudanças, mas não modificados
Avaliação da mudança: grau de dificuldade de implementação e o que mudança requerida precisa
Prioridade de mudanças: grau de prioridade da mudança

Implementação da mudança:
Esforço estimado: tempo estimado
Data para comitê de controle de mudanças: data da submissão
Data da decisão do comitê de controle de mudanças: data do resultado da submissão
Decisão do comitê de controle de mudanças: decisão tomada

Data da submissão do comitê de garantia da qualidade: data do teste
Data de submissão a GC: data do resultado da GC
Decisão do comitê de garantia da qualidade: verifica e descreve se a modificação foi implementada corretamente

Quadro 3 - Modelo do formulário de solicitação de mudanças (adaptado de SOMMERVILLE, 2003)

O comitê de controle de configuração e o responsável pela GC devem observar a necessidade de modificar os outros artefatos presentes na mesma versão do sistema e levantar o custo real para a mudança (SOMMERVILLE, 2003; PRESSMAN, 2006; CUNHA *et al.*, 2004).

Peters; Pedrycz (2001) apresentam um modelo, mais sucinto (Quadro 4), contendo apenas as principais características tratadas pelo modelo descrito por Sommerville (2003) (Quadro 3). Entretanto, com acréscimo de algumas características, como a opção que registra a aprovação ou não do pedido de mudança, ilustrada mais claramente, o custo/economia que a mudança irá proporcionar e a prioridade da mudança.

Caso o registro das mudanças e dos problemas seja armazenado diariamente pela empresa desenvolvedora, apenas os componentes e os módulos individuais serão afetados, isto é, mantém-se um registro das mudanças sofridas por cada componente, realizando uma documentação padronizada no início do código fonte, como segue o modelo apresentado no Quadro 5. Caso as

[6] A análise de impacto oferece um entendimento das implicações que uma determinada mudança pode proporcionar no *software* (LEE, 1998).

mudanças sejam feitas em módulos, por diferentes desenvolvedores, também existe a necessidade de serem controladas (SOMMERVILLE, 2003).

Proposta de alteração	**Data da submissão __/__/__**	**Número do formulário**
Nome e endereço da organização que a originou	**Prioridade**	**Tipo**
***Baseline* afetada pela alteração:**	**Projetos afetados pela alteração:**	
Descrição da alteração:		
Necessidade de alteração:		
Programação estimada de entrega: __/__/__	**Custo/ economia estimada pela alteração:**	
☐ **Aprovada** ☐ **Desaprovada**	**Assinatura:**	**Data da decisão: __/__/__**

Quadro 4 - Modelo de formulário de solicitação de mudança (adaptado de PETERS; PEDRYCZ, 2001)

// **Nome do Projeto**
//
// local que está o artefato
//**Objeto**: nome do artefato
//**Autor:** nome do desenvolvedor
//**Data da criação:** data da criação do artefato
//**Histórico de Alterações**
//

//**Versão**	**Modificador**	**Data**	**Mudança**	**Razão**
// Nº versão	responsável	data	o que mudou	razão da mudança

Quadro 5 – Documentação no código fonte das mudanças ocorridas (adaptado de SOMMERVILLE, 2003)

Além de planejar e controlar as mudanças, é necessário testar o artefato modificado. Assim, o teste de *software* se tornou de grande importância na identificação e eliminação de erros, especialmente na fase de manutenção do *software* (ROCHA *et al.*, 2001). Dessa forma, o Teste de Regressão (TR) foi criado para ser empregado exclusivamente nesta fase, com a reutilização do conjunto de casos de teste já aplicado na versão anterior para observar se novos erros não foram introduzidos na versão atual do *software*, diminuindo custo e tempo associados. Caso esta reutilização seja insuficiente, aconselha-se empregar novos casos de teste para verificar as modificações aplicadas na versão atual do sistema (MARTIMIANO, 1999).

3.3.2.4 Gerenciar as versões

Como já foi apresentado na Seção 3.3.1, as versões são variações de um mesmo *software*. As mudanças ocorridas na vida do *software* podem gerar problemas, caso não sejam gerenciadas. Deste modo, a atividade de gerenciamento de versões identifica e segue todo o processo de desenvolvimento de diferentes variantes de um determinado sistema. A liberação da nova versão deve ser realizada pela equipe de GC, a qual deve manter a qualidade do repositório de dados (SOMMERVILLE, 2003; PRESSMAN, 2006).

Para ajudar neste gerenciamento é necessário identificar os artefatos gerados para cada versão, facilitando a recuperação de uma determinada versão. Para auxiliar nessa identificação são empregadas três técnicas, as quais se encontram descritas no Quadro 6.

Técnica	Descrição
1 – Numeração das versões	distribuir um número específico e exclusivo de versão aos artefatos.
2 – Identificação baseada em atributos	determinar nome diferente aos artefatos em cada versão e um conjunto de atributos distintos para cada versão do artefato.
3 – Identificação orientada a mudanças	também idealizada na identificação baseada em atributos, mas associa ao nome do artefato as mudanças efetuadas.

Quadro 6 - Técnicas para a identificação dos artefatos (SOMMERVILLE, 2003)

Na numeração de versão (técnica 1) deve-se constar o nome do componente ou do sistema acrescido de um número referente à versão. A derivação de uma versão para outra pode fazer parte de uma versão anterior. Na Figura 4 ilustra-se uma estrutura com o controle das versões e suas respectivas evoluções (SOMMERVILLE, 2003). Pode-se constatar nesta figura que a versão V1.0 irá derivar a V1.1 que, por sua vez, poderá dar origem às versões V1.1b e V1.2, e a versão V1.1a derivará por sua vez a versão V2.0, e assim por diante.

A identificação baseada em atributos (técnica 2) localiza as versões que possuem o mesmo conjunto de atributos, como por exemplo, versões de cada componente, versão criada em um intervalo de data e versão para uma determinada plataforma. A identificação orientada à mudança (técnica 3) possui a facilidade de recuperação de uma determinada versão, pois utiliza a numeração simples no sistema, requer conhecimento de quais os atributos contidos e aplicação separadamente do gerenciamento de mudanças para descobrir as versões e as respectivas mudanças ocorridas (OLIVEIRA *et al.*, 2001; SOMMERVILLE, 2003).

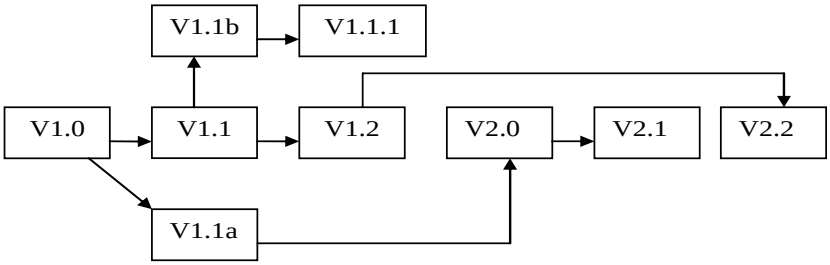

Figura 4 - Estrutura de derivação de versões (SOMMERVILLE, 2003)

3.3.2.5 Gerenciamento de *releases*

Como definido anteriormente na Seção 3.3.1, as *releases* são versões do sistema distribuídas aos clientes. Pode conter um arquivo descrevendo a configuração para a sua instalação, arquivos de dados utilizados para o bom funcionamento do sistema, documentação eletrônica apresentando o sistema e a embalagem e a publicidade relacionada à *release*.

A distribuição é feita pelos gerentes de *release*, levando em consideração que uma nova *release* não deve depender da versão atual, pois o usuário pode migrar ou não para a nova *release* (OLIVEIRA *et al.*, 2001; SOMMERVILLE, 2003). O Quadro 7 aponta alguns fatores que podem influenciar a criação de uma nova *release*.

A documentação produzida no desenvolvimento da *release* é extremamente eficaz, pois pode ser empregada no código-fonte, no sistema operacional, em todos os dados, nos arquivos de configuração, nas bibliotecas, nos compiladores, nas ferramentas e nas plataformas devem possuir as mudanças ocorridas na *release* (SOMMERVILLE, 2003).

Fator	Descrição
Qualidade técnica do sistema	Em caso de defeitos muito sérios será indispensável a utilização de uma *release* de correção, mas se os defeitos forem pequenos um *patch* pode ser distribuído.
Quinta lei de Lehman	Com o aumento de funcionalidade em uma *release*, uma *release* pode ser seguida de uma outra *release* de correção.
Competição	Se o concorrente disponibilizar uma outra versão com alguma funcionalidade, pode ser inevitável a criação de uma nova *release*.
Requisitos de mercado	A data de lançamento de determinada *release* nova será definida pelo pessoal do *marketing*.
Propostas de mudanças do cliente	Caso o cliente tenha solicitado algumas mudanças, deve-se esperar a *release* com as modificações requisitadas.

Quadro 7 - Fatores que determinam uma estratégia de *release* (SOMMERVILLE, 2003)

3.3.2.6 Implantação de Gerência de Configuração

Para melhor administrar o gerenciamento dos artefatos são emitidos relatórios durante todo o processo de desenvolvimento que poderão versar sobre as *baselines* ou sobre qualquer tipo de artefato. Entre os relatórios apresentados destacam-se os citados no Quadro 8, de acordo com Cunha *et al.*(2004).

As revisões são fundamentais nesta atividade, pois apresentam o desempenho dos artefatos referente ao seu objetivo proposto. Assim, devem ser empregadas as revisões técnicas e as auditorias. A primeira propõe a melhoria do artefato que sofreu modificação, por meio de avaliações de efeitos colaterais e omissões. Já nas auditorias, a revisão formal é efetuada com o emprego da avaliação de algumas características – se a modificação foi feita em conformidade com o que foi proposto, se necessitará de adaptação dos outros artefatos, se o processo e os padrões foram atendidos e se as especificações para o registro das modificações foram adequadamente preenchidas e atualizadas (PRESSMAN, 2006).

Relatórios	Descrição
Relatório de artefatos	Descreve a respectiva versão e as especificações do artefato
Relatório de mudanças	Estabelece as solicitações e a idealização das mudanças sugeridas
Relatório de *baseline*	Descreve todos os artefatos que compõem a *baseline*
Relatório de projeto	Especifica qual *baseline* constitui uma determinada versão

Quadro 8 - Relatórios de gerência de configuração (CUNHA *et al.*, 2004)

3.3.3 Problemática de GC na reengenharia baseada em *framework*

A GC se torna essencial tanto no desenvolvimento quanto na reengenharia de *software* incremental, principalmente quando essa última emprega o uso de *framework* como apoio computacional para a geração do sistema alvo. Ressalta-se que, se um determinado sistema alvo é desenvolvido com o emprego de um *framework*, o mesmo se torna dependente dessa tecnologia de reúso, pois a arquitetura do sistema passa a englobar o próprio *framework* (ARIMOTO *et al.*, 2007). Assim, as mudanças no *framework* podem prejudicar o comportamento do sistema alvo na próxima instanciação do *framework*, caso alguma funcionalidade já existente no *framework* tenha sido modificada (CAGNIN, 2005a).

Dentro desse contexto, como podem haver evoluções e modificações tanto no *framework* quanto no sistema alvo, gerando diferentes versões desses *software*s, deve-se ter um controle de versões que gerencie a versão do *framework* que está relacionada com cada versão do sistema. Devido a isso, é preciso tomar alguns cuidados na evolução do *framework* com o emprego da análise de impacto, verificando quais artefatos serão afetados por tal mudança, bem como o grau de prioridade da solicitação de mudança e o tempo que será gasto.

Na Figura 5 encontra-se um exemplo de várias versões de um determinado *framework* e as versões dos sistemas gerados a partir dele. Com isso, é possível observar em quais versões dos sistemas as mudanças em determinada versão do *framework* podem causar impacto. A primeira versão V 1.1 do *framework* gerou as versões V 4.0 e V 4.1 do sistema. A próxima versão do *framework* (V 1.2) gerou a versão V 4.2 do sistema, assim, pode-se observar que, apesar do sistema ter sido gerado novamente em outra versão do *framework,* o seu comportamento não foi prejudicado, pois as classes do *framework* utilizadas apresentavam o mesmo comportamento que a versão anterior do *framework*.

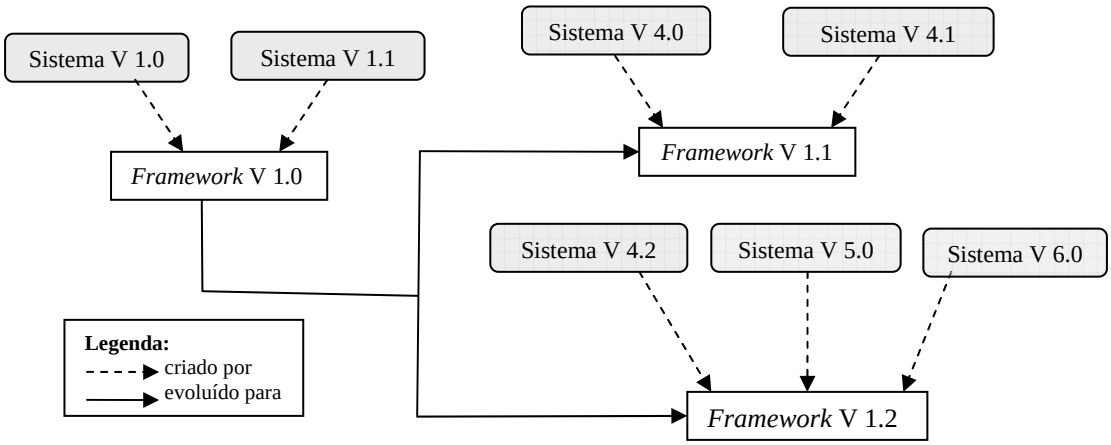

Figura 5 – Grafo de controle de versão de *framework* e dos sistemas gerados (adaptado de CAGNIN, 2005a)

3.3.4 Trabalho correlato

Dentre os trabalhos apresentados na literatura que se preocupam com a GC, destaca-se o de Murta e Werner (2007) uma vez que trata de GC no contexto de componentes, que como

frameworks, também é uma técnica que possibilita o reúso de software e necessita de precauções durante o controle de versões.

A **abordagem de GC no desenvolvimento baseado em componentes** (Murta e Werner, 2007) possui o diferencial em delegar a responsabilidade de implementação das solicitações de mudanças. A preocupação dessa abordagem está em evitar a introdução de defeitos durante mudanças dos componentes que podem afetar todos os sistemas que reutilizam os componentes modificados (MURTA; WERNER, 2007). Para permitir isso, a abordagem conta com uma ferramenta específica denominada **Odyssey-CCS**, que controla as modificações e coleta as informações dos colaboradores (pessoas que participam do controle de modificação), proporcionando uma comunicação e coordenação do trabalho e mantendo informações sobre as modificações realizadas nos componentes (LOPES *et al*., 2005).

Por meio da ferramenta **Odyssey-CCS** são realizadas as seguintes atividades: a) modelagem de um processo de controle de mudanças para cada *software* produzido na empresa, efetuada pelo gerente de configuração; b) definição de quais informações necessitam ser coletadas durante o controle de mudanças, efetuado pelo gerente de configuração, para que os colaboradores realizem as atividades e tomem as decisões necessárias; c) atribuição de responsabilidades, efetuada pelo gerente de configuração, designando os papéis modelados no processo para os colaboradores, identificação do momento de realizar a coleta da informação, por meio de uma associação de formulários a produtos modelados no processo; d) verificação, efetuada pelos colaboradores, de quais atividades e decisões encontram-se pendentes para execução; e) execução de uma determinada atividade ou decisão, que estava pendente, pelo colaborador, a fim de que outras pessoas que desempenhem a mesma função tenham conhecimento deste fato; f) após execução das atividades e decisões, os colaboradores devem fornecer informações pertinentes para que aja compreensão sobre tal procedimento futuramente (LOPES *et al*., 2005).

3.4 Ferramentas de apoio à GC

As ferramentas devem apoiar as atividades de GC com a finalidade de armazenar, gerenciar, controlar e manter a integridade dos artefatos. O apoio de ferramentas se tornou essencial pela rapidez em executar o controle das versões, pela redução do custo e o aumento do desempenho

das equipes de desenvolvimento. Além disso, facilita o acesso às informações necessárias sobre as versões e garante a integridade dos artefatos por meio do controle de solicitações de mudanças.

Deve-se ressaltar a variedade de ferramentas *open-source* que podem atender totalmente ou parcialmente às necessidades de GC, dentre elas destacam-se: a RCS (TICHY, 1997), a CVS (XIMBIOT, 2005), a *Arch* (KRAUSE, 2002) e a *Subversion* (COLLINS-SUSSMAN *et al.*, 2006), que serão apresentadas a seguir.

3.4.1 *Revision Control System* (RCS)

A ferramenta RCS foi umas das pioneiras no controle de versões. O seu desenvolvimento foi realizado por meio de comandos Unix e muitas de suas funcionalidades vêm sendo empregadas para apoiar o desenvolvimento de outras ferramentas de controle de versão, como por exemplo, a CVS e a *Subversion* (TICHY, 1997; ASKLUND; MAGNUSSON, 2001 apud QUADROS, 2005).

Esta ferramenta está diretamente relacionada às tarefas de controle de versão, as quais são feitas por *branches* (ramos). Define-se um ramo para cada versão modificada, por exemplo, caso ocorra uma modificação de determinado artefato por dois desenvolvedores, são criadas duas versões distintas com a mesma raiz ou arquivo inicial. O *merge* agrupa as versões respeitando a estrutura, apresentando na raiz a versão original e, em seguida, as derivadas desta (TICHY, 1997; SILVA, 2002). O armazenamento das versões dos documentos com extensão ".txt" é realizado em formato de árvore, sendo que o primeiro arquivo se encontra na raiz e os demais que se apresentarem na seqüência são considerados as modificações ou evoluções deste arquivo inicial (TICHY, 1997).

A RCS estabelece a integridade dos artefatos, tanto no armazenamento quanto na permissão da modificação. No armazenamento é feita uma cópia da versão que será modificada na área de trabalho do desenvolvedor com o uso do comando *check-out*. Depois deste procedimento, a versão do artefato é bloqueada no repositório para que não ocorram várias mudanças no artefato ao mesmo tempo. Após a modificação e aprovação do artefato, é executado o comando *check-in*. Neste instante, a ferramenta identifica o arquivo modificado e define uma nova versão que pode ser disponibilizada aos demais desenvolvedores, mantendo apenas as linhas que possuem caracteres diferentes da versão anterior deste artefato. Com isto, emprega-se o conceito de delta (TICHY, 1997; SILVA, 2002).

As revisões são realizadas e armazenadas em árvore no repositório, como ocorre com as versões. Os documentos criados nas revisões são: a revisão atual e as revisões das versões anteriores (TICHY, 1997).

O histórico das mudanças é gerado automaticamente com apoio dos critérios, os quais podem estar relacionados à data de criação, ao nome do desenvolvedor e ao nome de uma determinada versão ou até mesmo de uma determinada característica da versão (TICHY, 1997; SILVA, 2002).

As permissões são tratadas de acordo com o *login* de acesso. Os usuários anônimos têm acesso sem restrição. Os usuários com restrições devem informar a senha de acesso aos artefatos contidos no repositório.

3.4.2 *Concurrent Versions System* (CVS)

A ferramenta CVS se aplica em diversas plataformas e tem como objetivo controlar as versões adequadamente (XIMBIOT, 2005). Segundo Silva (2002), a ferramenta CVS pode ser considerada uma interface entre a ferramenta RCS e o usuário, pois contém as mesmas características apresentadas por essa ferramenta (ramos e *merges*), mas com algumas funcionalidades a mais, como a programação paralela simultânea de diversos programadores em uma mesma versão do artefato e o tratamento das permissões com restrições de *login* igualmente apresentado na RCS.

A disposição dos artefatos encontra-se hierarquicamente dentro de um único repositório, com o emprego do comando *import*. Pode-se criar um ramo no tronco principal, em que é armazenado o artefato original, assim qualquer alteração feita neste artefato irá prover uma nova versão. Nesse caso é realizado um *merge* (agrupamento dos artefatos) e, em seguida, é adicionado um novo lugar na árvore, partindo da versão original. A criação de lugares na árvore deve dispor de um nome simbólico que está associado ao diretório e aos arquivos dentro do repositório (SILVA, 2002; XIMBIOT, 2005).

A CVS cumpre as principais funções de controle de versões, seja no armazenamento, seja nas modificações dos artefatos contidos no seu repositório. A identificação é feita por um determinado número chamado revisão, que contém a modificação, o seu autor, a data, o tamanho,

entre outros dados, de acordo com o critério da empresa (BRUCKER *et al.*, 2002; CAETANO, 2004). O comando *check-out* apresenta as mesmas características da ferramenta RCS, isto é, cria uma cópia do artefato na área de trabalho do desenvolvedor. Em seguida, emprega-se o comando *commit*, correspondente ao comando *check-in* do RCS. O comando *commit* verifica e armazena o artefato no repositório, registrando sua evolução (SILVA, 2002).

A integridade dos artefatos é garantida com o controle dos acessos e das modificações, que contém os mesmos procedimentos apresentados pela RCS (CAETANO, 2004). Deve-se ressaltar que o armazenamento das versões apresenta economia de espaço, pois armazena apenas as diferenças apresentas em cada versão (SILVA, 2002), como ocorre na RCS.

3.4.3 GNU *Arch*

A ferramenta *Arch* tem a finalidade de controlar as versões que ainda estão em desenvolvimento (KRAUSE, 2002). Esta ferramenta possui similaridade com a ferramenta CVS, levando em consideração a organização e o controle das versões, encontra-se disposta em árvore conforme apresentado nas ferramentas comentadas anteriormente, com algumas vantagens, como: a) as permissões são tratadas tanto com restrições quanto com perfis (que não eram tratadas nas demais ferramentas); b) o registro das mudanças nos artefatos, que podem ser do tipo de reorganização e renomeação de um determinado artefato; c) a associação de um conjunto de mudanças com uma descrição exata do que foi alterado; e d) a distribuição dos repositórios favorece o *merge*, como ocorre com a ferramenta RCS. Um ponto fraco encontrado em *Arch* é que seu emprego não se estende para todas as plataformas, apenas *Linux* (TOSUN, 2004).

3.4.4 *Subversion*

A ferramenta *Subversion* objetiva o controle de versão dos sistemas. Como foi constatado na maioria das ferramentas supracitadas, esta também emprega as funcionalidades dos comandos e de armazenamento (*commit* e *check-out*). Essas similaridades facilitam a migração para esta

ferramenta com pouco esforço. A *Subversion* pode se adaptar a todas as plataformas e as permissões podem ser com e sem restrições, sendo que no primeiro caso utiliza-se perfil (COLLINS-SUSSMAN *et al.*, 2006).

A diferença encontrada nesta ferramenta está no seu armazenamento, pois não considera apenas o artefato, mas também os diretórios, cópias e renomeações das versões (COLLINS-SUSSMAN *et al.*, 2006). Como ocorre com as demais ferramentas, a *Subversion* armazena apenas as diferenças entre as versões.

3.4.5 Comparação das ferramentas de GC

Para analisar qual ferramenta se adequa melhor às necessidades previstas neste trabalho, que visa ao controle das versões dos artefatos produzidos durante a reengenharia baseada em *framework*, são apresentadas no Quadro 9 algumas das principais características encontradas nessas ferramentas.

Ferramentas / **Características**	**RCS**	**CVS**	***Arch***	***Subversion***
Mudança de localização nos artefatos e nos diretórios	Não suporta essa mudança gerando quebra de histórico	Ocorre a quebra do histórico em duas partes	Ocorre a renomeação das versões	Ocorre a renomeação das versões
Migração para outras ferramentas	Não dá suporte	Não dá suporte	Aceita a importação dos arquivos gerados pela ferramenta CVS	Aceita a importação dos arquivos gerados pela ferramenta CVS
Revisões	Armazena apenas a última na íntegra e as antigas apenas as diferenças	Armazena apenas a última na íntegra e as antigas apenas as diferenças	Realiza-se com restrições aplicadas em perfis	Realiza-se com restrições aplicadas em perfis
Documentação	Pouca documentação, restringindo a tutoriais existentes	Várias documentações dispostas em livros e tutorias em várias línguas	Pouca documentação, restringindo a tutoriais existentes	Pouca documentação, restringindo a tutoriais existentes
Características em comum	Possui código-fonte e licença livre Comandos similares aos de outras ferramentas	Possui código-fonte e licença livre Utiliza-se de alguns comandos aplicados na ferramenta RCS e a criação do *commit*	Possuem código-fonte e licença livre Utiliza-se dos comandos existentes na CVS	Possui código-fonte e licença livre Utiliza-se dos comandos existentes na CVS
Plataforma	Unix	Windows, Unix E Linux	Linux	Windows, Unix E Linux

Quadro 9 - Comparação das ferramentas de GC

Devido as características encontradas nas ferramentas de controle de versão estudadas neste trabalho, foi selecionada a *Subversion* principalmente devido ao tratamento dado por ela com relação ao repositório, isto é, possibilita a renomeação e a importação de arquivos de outras ferramentas de controle de versão. Essa característica ainda não foi implantada na CVS.

3.5 Considerações Finais

Neste capítulo foram apresentadas as atividades adotadas pela GC a fim de garantir a qualidade no desenvolvimento do *software*. Assim, foram descritas resumidamente a definição de qualidade de *software*, dando maior ênfase à Gerência de Configuração, descrevendo os passos para administrar e controlar os artefatos, as versões e as *releases*.

A Gerência de Configuração deve ser considerada, principalmente no desenvolvimento e na reengenharia incremental, uma vez que versões do sistema (novo ou alvo, respectivamente) são produzidas a cada iteração do processo. No contexto de reengenharia baseada em *framework* essa gerência deve ser aplicada com cautela, pois é necessário controlar tanto as versões geradas do sistema alvo quanto as versões do *framework*, bem como a dependência entre ambas uma vez que mudanças efetuadas no *framework* podem alterar o comportamento dos sistemas gerados, fazendo com que os mesmos forneçam comportamento indesejável.

Para apoiar mais efetivamente o controle de versão foram apresentadas algumas ferramentas *open-source*, bem como uma comparação das mesmas a fim de auxiliar na escolha daquela que melhor atenda às necessidades do projeto, que foi a *Subversion*.

No próximo capítulo são apresentadas as normas e os modelos de maturidade estudados, do ponto de vista de GC, para apoiar a definição do modelo de referência proposto neste trabalho.

4 NORMAS E MODELOS DE QUALIDADE DE *SOFTWARE*

4.1 Considerações Iniciais

São abordados neste capítulo as normas e os modelos de qualidade, pois proporcionam um alicerce para a identificação das atividades de GC do modelo de referência de GC proposto para o PARFAIT. Na Seção 4.2 apresenta-se, especificamente, como a atividade de GC é tratada no CMMI; na Seção 4.3 aborda-se a GC na ISO/IEC 12207; na Seção 4.4 ilustra-se a GC segundo a ISO/IEC 15504; na Seção 4.5 expõe-se uma descrição da norma ISO/IEC 10007, que tem em seu foco a atividade de GC; na Seção 4.6 aborda-se como o modelo MR-MPS trata a atividade de GC; e na Seção 4.7 apresenta-se o tratamento de GC pelo modelo PMBOK. Na Seção 4.8 faz-se uma comparação entre as normas e os modelos estudados neste capítulo e, na Seção 4.9, encontram-se as considerações finais deste capítulo.

4.2 CMMI sob a perspectiva da Gerência de Configuração

O *Capability Maturity Model Integration* (CMMI) é um modelo que focaliza o desenvolvimento integrado do produto e do processo. Esse modelo se originou do aperfeiçoamento do *Capability Maturity Model* (CMM) (PAULK, 1993; HUMPHREY *et al.,* 1991) que é o modelo de avaliação da maturidade dos processos de *software*, desenvolvido por pesquisadores do *Software Engineering Institute* (*SEI*)[7] da Universidade Carnegie Mellon.

A GC é apresentada no CMMI, especificamente no nível 2 de maturidade, como prática genérica. Os controles gerenciais e técnicos da GC são apresentados rigorosamente na seleção dos itens de configuração, que contém a *baseline* e sua integridade, no controle das mudanças dos artefatos, no estabelecimento das diretrizes para o desenvolvimento do produto com gerenciamento,

[7] http://www.sei.cmu.edu/

na confiabilidade e nas atividades de configuração para o desenvolvedor e para o usuário (SEI, 2001).

No CMMI as práticas específicas aplicadas na GC são: identificar os itens de configuração e transformá-los em *baselines*, controlar as modificações dos itens de configuração e estabelecer a integridade das *baselines* (SEI 2001). As metas genéricas (GG- *Generic Goal*) e específicas (SG - *Specific Goal*), bem como as práticas genéricas (GP - *Generic Practire*) e específicas (SP - *Specific Practire*) encontram-se descritas no Quadro 10.

Metas	**Práticas**
SG 1 estabelecer *baseline*s	SP 1.1 identificar itens de configurações SP 1.2 estabelecer um sistema de gerenciamento de configurações SP 1.3 criar ou liberar *baseline*s
SG 2 rastrear e controlar alterações	SP 2.1 rastrear solicitações de mudanças SP 2.2 controlar itens de configurações
SG 3 estabelecer as integridades	SP 3.1 estabelecer os registros de gerenciamento de configurações SP 3.2 executar auditorias de configurações
GG 1 executar os objetivos específicos	GP 1.1 executar as práticas básicas
GG 2 instituir um processo controlado	GP 2.1 estabelecer uma política organizacional GP 2.2 planejar o processo GP 2.3 fornecer recursos GP 2.4 atribuir responsabilidades GP 2.5 treinar as pessoas GP 2.6 gerenciar configurações GP 2.7 identificar e envolver os *stakeholders* relevantes GP 2.8 monitorar e controlar o processo GP 2.9 avaliar objetivamente a aderência GP 2.10 revisar o status com o nível mais alto de gerência
GG 3 instituir um processo definido	GP 3.1 estabelecer um processo definido GP 3.2 coletar informações de melhorias
GG 4 instituir um processo quantitativamente controlado	GG 4.1 controlar com técnicas estatísticas
GG 5 instituir um processo otimizado	GG 5.1 empregar a melhoria contínua

Quadro 10 - Metas e práticas da área de processo GC do CMMI (SEI, 2001)

Para melhor entendimento do Quadro 10, segue uma breve explicação das metas apresentadas. Ao estabelecer as Metas Específicas (SG) define-se o *status* de aprovação da mudança, mantêm-se os artefatos e as versões corretas com o auxílio de auditorias. Em SG 1 (Meta Específica 1) pode-se estabelecer, criar e disponibilizar as *baselines* somente após a identificação dos itens de configuração, isto é, todos os artefatos necessários para a criação e para a especificação do *software* que compõem a *baseline* devem ser identificados primeiramente. Dentre os itens de configuração destacam-se: o projeto, a descrição do processo de GC, os planos, os procedimentos de teste, as ferramentas de apoio, o código fonte especificado tanto pela linguagem quanto pela extensão do arquivo. Assim, é necessário gerenciar o armazenamento, a solicitação, a recuperação e a alteração dos artefatos, sem esquecer de fornecer os relatórios desse gerenciamento, o *backup* e, quando houver necessidade, as revisões de GC .

A SG 2 (Meta Específica 2) rastreia as solicitações e o controle das alterações nos itens de configuração, possibilitando o registro tanto dos novos quanto dos velhos artefatos, e analisa o impacto que pode ser gerado com a mudança. Como pode ser o caso de uma alteração ocorrida em um determinado artefato, que pode afetar outros artefatos, resultando em custo monetário e investimento além do esperado.

A SG 3 (Meta Específica 3) especifica as diferenças de cada *baseline* alterada, sua respectiva versão e apresenta um histórico de revisões: dos itens de configuração, de solicitações de mudanças, do *status* e das diferenças entre as versões e das *baselines.*

Na GG 1 (Meta Genérica 1) são produzidos os artefatos e os serviços previstos durante a aplicação do processo.

Na GG 2 (Meta Genérica 2) é estabelecida e mantida a política organizacional que planeja, executa e controla a GC. No controle das alterações das *baselines* e da sua integridade executa-se o plano do processo de GC, fornecem-se os recursos necessários para desenvolver o produto e prover os serviços, como por exemplo, as ferramentas para apoiar a administração das mudanças ocorridas nos artefatos, e delegam-se a responsabilidade e a autoridade para executar o processo de GC. Ainda é preciso: treinar as pessoas tanto na execução quanto no suporte do processo de GC em cada mudança ocorrida, nas responsabilidades, nos padrões, nos procedimentos, nos métodos e no repositório das *baselines*; gerenciar as configurações com as listas de acesso, com os relatórios de *status* de mudanças e com as *baselines* arquivadas no repositório; e identificar e comprometer os interessados em analisar o impacto das mudanças e em estabelecer as *baselines* ao rever os resultados das auditorias de GC.

A GG2 também realiza as seguintes atividades: controla a quantidade de mudanças e de auditorias dos itens de configuração, e se não contêm erros; avalia o objetivo referente às ligações do processo de GC com padrões e procedimentos que estabelecem e mantêm os *baselines*; rastreia e controla as mudanças; revisa as atividades, os *status* e os resultados apresentados no processo de GC.

A GG 3 (Meta Genérica 3) estabelece oficialmente um processo definido de GC e coleta as informações sobre os resultados, as medidas, o planejamento e a execução do processo para melhorias futuras.

Em GG 4 (Meta Genérica 4) o processo é controlado com o emprego de técnicas estatísticas ou quantitativas, o artefato e o desempenho do processo são mensurados e controlados em todo o projeto. Já em GG 5 (Meta Genérica 5) emprega-se a melhoria contínua do desempenho do processo com iniciativas incrementais e inovadoras.

4.3 NBR ISO/IEC 12207 sob a perspectiva da Gerência de Configuração

A norma ISO/IEC 12207 (*Information technology - Software Life Cycle Process*) foi desenvolvida em 1995, pela *International Organization for Standardization* (*ISO*) e pelo *International Electrotechnical Commission* (*IEC*), sofreu alteração em 1998 e se encontra atualmente no Brasil, sob o cuidado da Associação Brasileira de Normas Técnicas (ABNT), com a denominação NBR ISO/IEC 12207 (Tecnologia da Informação - Processos de Ciclo de Vida do *Software*).

A norma NBR ISO/IEC 12207 (1998) tem como principal ponto positivo uma estrutura comum de processos de ciclo de vida, que auxiliam as empresas nos negócios relacionados aos produtos de *software*, tornando eficaz o desenvolvimento e a manutenção do *software*. Esta norma se adapta às necessidades da empresa desenvolvedora e se destaca por ser a pioneira em apresentar os processos e as tarefas que podem ser empregadas durante o ciclo de vida do *software* para garantir a melhoria do mesmo.

A GC é vista pela NBR ISO/IEC 12207 e contém detalhes com a descrição das atividades que apóiam a empresa ao desenvolver essa prática.

As atividades exclusivas de GC estão relacionadas à implementação do processo, à identificação da configuração, ao controle da configuração, ao relato da situação da configuração, à avaliação da configuração e à gerência de liberação e distribuição, como se apresenta a seguir (NBR ISO/IEC 12207, 1998).

A atividade “implementação do processo” cria um plano de GC que descreve as atividades de GC, os procedimentos, o cronograma e o responsável por executar estas atividades.

A atividade “identificação da configuração” consiste em identificar e controlar os artefatos e as suas versões. Todos os itens de configuração devem apresentar a documentação que estabelece a *baseline*, as referências de versão e quaisquer detalhes que ajudem na sua identificação.

A atividade “controle da configuração” executa as tarefas de: identificar e registrar os pedidos de alteração; avaliar as alterações; aprovar ou não o pedido; e implementar, verificar e liberar os itens de *software* que foram alterados. Sem se esquecer dos registros de auditoria que controlam os acessos aos artefatos gerando confiança, segurança e proteção aos mesmos.

Com o emprego da atividade “relato da situação da configuração” são descritos os relatórios e os registros da situação do histórico de todos os artefatos controlados e as *baselines*. Este relatório deve conter o número de alterações, as últimas versões, os identificadores, a quantidade de liberações e as comparações empregadas nos artefatos.

Na “avaliação da configuração” ocorre a determinação e a garantia de qualidade dos artefatos, referente às funcionalidades dos requisitos.

Na “gerência de liberação e distribuição” disponibilizam-se os produtos e a documentação que são gerenciados formalmente com a utilização de cópias de segurança do código e da documentação. A proteção dos artefatos é outro fator importante na manipulação, no armazenamento, no empacotamento e na distribuição, segundo a política da empresa.

4.4 ISO/IEC 15504 sob a perspectiva da Gerência de Configuração

A norma ISO/IEC 15504, conhecida também como *SPICE*, foi criada em 1999 com o apoio do grupo de trabalho dedicado ao campo de Tecnologia da Informação e Engenharia de

Software[8](ISO/IEC TR 15504, 1999). Esta norma é composta por nove partes cujo objetivo é elaborar um guia de orientação que examine e avalie o *software*, promovendo uma melhoria contínua dos processos.

A GC encontra-se na quinta parte da norma, com a denominação de suporte. Essa parte contém processos dentre os quais se destacam aqueles que tratam de GC:

- Criar uma estratégia de GC para desenvolver todas as atividades,
- Estabelecer o emprego de padrões, de procedimentos e de ferramentas,
- Identificar os artefatos que foram armazenados em repositórios,
- Controlar as modificações com o formulário de solicitação de mudanças e de autorização,
- Gerenciar o histórico do artefato com um nível de detalhes aceitável, caso seja necessário recuperar e garantir a integridade do artefato,
- Administrar a liberação de uma determinada versão somente após revisão e autorização.

4.5 ISO/IEC 10007 - Sistemas de Gestão da Qualidade – Diretrizes para a Gestão de Configuração

A norma ISO/IEC 10007 (Sistemas de Gestão da Qualidade – Diretrizes para a Gestão de Configuração) foi elaborada pelo Comitê Brasileiro da Qualidade (ABNT/CB-25) e pela Comissão de Estudo de Sistema de Gestão de Qualidade, e se encontra na segunda edição. Esta norma tem como objetivo fornecer um guia com diretrizes para a atividade de GC, cujo enfoque está voltado principalmente para o ciclo de vida do produto, atendendo aos requisitos de identificação e de rastreabilidade dos artefatos (ISO/IEC 10007, 2005).

Assim, a ISO/IEC 10007 é empregada para estimular o uso de GC no planejamento, na identificação, no controle das alterações, na situação de configuração e nas auditorias (ISO/IEC

[8] WG10 - composto por 13 países: Austrália, Brasil, Canadá, França, Alemanha, Israel, Itália, Japão, África do Sul, Espanha, Suécia, Inglaterra e Estados Unidos.

10007, 2005). Salienta-se que esta norma utiliza como apoio a ISO/IEC 9001 (2000) e a ISO/IEC 9004 (2000).

As responsabilidades apresentadas nesta norma consistem em delegar: **as responsabilidades e as autoridades** tanto na implementação quanto na verificação do processo de GC; a **autoridade pela disposição**, ou seja, verificar se as alterações propostas são necessárias e aceitáveis, se essas possuem uma documentação adequada e se as mesmas são planejadas satisfatoriamente.

O processo de GC, por sua vez, pode ser dividido em seis partes, iniciando-se pelas generalidades e finalizando em auditoria de configuração, como se apresenta a seguir (ISO/IEC 10007, 2005):

1ª Parte: denominada de **generalidade** que enfoca os requisitos do cliente do produto final e todo contexto de execução, detalhando o plano de GC por completo com a descrição dos procedimentos específicos e a extensão de sua aplicação durante todo o ciclo de vida;

2ª Parte: se encontra o **planejamento** que coordena todas as atividades de gestão de configuração, com uma documentação devidamente aprovada, que descreve os procedimentos relevantes, as responsabilidades e os profissionais incumbidos na execução de GC;

3ª Parte: apresenta a **identificação de configuração** que seleciona os artefatos necessários com o emprego dos critérios de seleção relacionados aos requisitos, à segurança e ao risco, à tecnologia do projeto, à interface com os artefatos e ao suporte;

4ª Parte: engloba o **controle de alterações** que possibilita a solicitação das alterações com a verificação do seu impacto, a análise de sua aprovação e, em seguida, o seu gerenciamento. A documentação deve ter um nível de detalhe adequado, isto é, com numerações, com uma breve descrição da justificativa e registro da alteração e com revisão, facilitando a rastreabilidade dos artefatos;

5ª Parte: emprega a **contabilização da situação da configuração** que impulsiona o registro das informações importantes do artefato, por exemplo, o número de identificação da versão, a data efetiva da liberação da versão, a situação da revisão, o histórico do artefato, enfim melhora o rastreamento, mantendo a integridade do artefato;

6ª Parte: contém a **auditoria de configuração** que deve ser aplicada nas revisões para favorecer a verificação dos requisitos, se estes foram atingidos nos artefatos**.**

4.6 MR-MPS sob a perspectiva da Gerência de Configuração

O projeto Melhoria do Processo de *Software* Brasileiro (MPS.BR) se encontra em desenvolvimento desde 2003, em conjunto com a Associação para Promoção da Excelência do *Software* Brasileiro (SOFTEX), a Sociedade Núcleo de Apoio à Produção e à Exportação de *Software* do Rio de Janeiro (RIOSOFT), o Programa de Engenharia de Sistemas e o Departamento de Computação da Universidade Federal do Rio de Janeiro (COPPE/UFRJ), o Centro de Estudos e Sistemas Avançados de Recife (CESAR), o Centro de Pesquisa Renato Archer (CenPRA) e a Companhia de Informática do Paraná (CELEPAR) (SOFTEX, 2005), e tem por objetivo possibilitar que empresas de *software* de pequeno e médio porte obtenham certificação baseada em normas internacionais a um custo acessível.

O MPS.BR é um projeto que criou o modelo de referência MR-MPS que visa a "maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de produtos de *software* e serviços correlatos", adequando-o à realidade brasileira tanto relacionada ao custo quanto ao desenvolvimento, com o auxílio de padrões internacionais, sendo inspirado no modelo CMMI (SEI, 2001) e nas normas ISO/IEC 12207 (1998) e ISO/IEC 15504 (1999).

Os 14 objetivos de GC apresentados no modelo MR-MPS (SOFTEX, 2005) são os seguintes:

1. Selecionar os artefatos com o emprego de critérios documentados.
2. Delegar responsabilidades para o desenvolvimento de cada artefato e *baseline.*
3. Identificar, armazenar, testar, revisar, alterar, definir, manter os artefatos e submetê-los à *baseline*.
4. Instituir uma política de GC que controle os diferentes níveis, que possua um armazenamento dos artefatos para posteriormente serem recuperados, bem como uma solicitação de mudanças.
5. Preservar todos os artefatos de GC.
6. Autorizar a criação ou a liberação das *baselines* pela GC.

7. Controlar as modificações e as liberações dos artefatos.
8. Disponibilizar as modificações e as liberações dos artefatos.
9. Registrar, relatar e analisar o impacto das solicitações de mudanças que os artefatos podem sofrer e sua situação.
10. Assegurar com as revisões, que as mudanças ocorridas nas *baselines* não sofram efeitos inesperados.
11. Adicionar os artefatos no repositório, já aprovados e devidamente controlados;
12. Assegurar a consistência e a completeza dos artefatos.
13. Controlar o armazenamento, o manuseio e a liberação dos artefatos.
14. Estabelecer e manter a integridade das *baselines*, com os registros e a auditoria de GC.

4.7 PMBOK sob a perspectiva da Gerência de Configuração

O *Project Management Body of Knowledge* (PMBOK) é um conjunto de conhecimento em Gerenciamento de Projetos, criado inicialmente em 1969 pelo *Project Management Institute* (PMI), no Estado da Pensilvânia, EUA. A primeira publicação do boletim *Project Management Quarterly* (PMQ) ocorreu nos anos 70, contudo deve-se ressaltar que no final desta década o PMBOK já possuía 2000 afiliados em todo o mundo (PMI, 2006). O principal objetivo do PMBOK é fornecer uma visão geral dos conhecimentos de Gerência de Projeto.

A GC se encontra na área de conhecimento de “Gerenciamento de Integração de Projeto”, que possui processos imprescindíveis para a coordenação de todos os elementos que constituem o projeto e é composta por: gerenciamento das mudanças, revisões e aprovações das mudanças, perfil do mantenedor para a autorização e método para validar as mudanças (LIMA, 2004; PMI, 2004).

O controle das atividades de mudanças desempenhadas em GC é documentado formalmente com supervisão tanto técnica quanto administrativa. Essas atividades podem ser divididas em controlar, verificar, registrar e relatar as mudanças ocorridas na elaboração do artefato e de suas características. Ressalta-se que o controle de mudanças possui uma preocupação na aprovação das mudanças para manter o artefato íntegro (PMI, 2004).

O processo de controle integrado de mudança é essencial pela dificuldade de executar na íntegra o plano do projeto. Suas atividades são: indicar se uma determinada mudança ocorreu ou precisa ocorrer; controlar as aprovações das mudanças que serão implementadas e identificar todos os fatores que dificultaram esse controle; revisar e aprovar as mudanças com o histórico das mudanças; gerenciar as *baselines*, mantendo a integridade e a documentação quando solicitada; revisar e aprovar as ações corretivas e preventivas; gerenciar as atualizações de custo, do cronograma, do orçamento e dos riscos das mudanças; documentar o impacto das mudanças; validar as mudanças; e garantir a qualidade do projeto com base nos relatórios de qualidade.

Assim, o PMBOK oferece um processo eficaz, eficiente e padronizado para gerenciar as mudanças no projeto, que facilita a identificação dos artefatos. Entre seus objetivos destacam-se o processo para identificação e solicitação das mudanças nas *baselines*, a melhoria contínua e a validação das mudanças do projeto, conforme o seu impacto, e o registro das mudanças sofridas por meio de históricos.

4.8 Comparação das Normas e Modelos de Maturidade Estudados

Pode ser constatado no estudo das normas e modelos de maturidade, efetuado neste trabalho, que a maioria deles considera como base a norma ISO/IEC 12207 ou o modelo de maturidade CMM (SOFTEX, 2005; SEI, 2001).

No Quadro 11 encontra-se uma breve comparação entre normas/modelos que aplicam GC, apresentando o objetivo, a abordagem utilizada, se possuem flexibilidade em algum aspecto, se foram baseados em outras normas, quais características gerais e específicas de GC possuem e a locação de GC. Relata-se que os dois últimos itens apresentados são de relevância para este trabalho.

Destacam-se no Quadro 11 as **características de GC**, isto é, como a GC é tratada e também a sua devida **localização da GC** nas normas e nos modelos de maturidade abordados no estudo realizado.

No Quadro 12 trata da identificação de cada **atividade de GC** que produz algum artefato relevante nas normas e nos modelos de maturidade estudados, destaca-se neste quadro a atividade

de GC “Gerenciar as mudanças nas versões e *releases*”, pois é de suma importância à GC e deve ser realizada com cautela.

	ISO/IEC 12207	**ISO/IEC 15504**	**ISO/IEC 10007**	**CMMI**	**MR-MPS**	**PMBOK**
Objetivo	Apoiar todo o ciclo de vida do artefato, apresentando diretrizes para prover a melhoria.	Avaliar os processos e determinar a capacitação, promovendo a melhoria contínua.	Oferecer diretriz para todas as atividades de GC.	Empregar o desenvolvimento integrado ao produto e ao processo.	Avaliar e melhorar a qualidade na produtividade dos serviços empregados no *software*.	Fornecer uma visão geral dos conhecimentos de gerência de projeto.
Abordagem	Oferece apoio ao desenvolvimento e à manutenção do *software*.	Dispõe de níveis de capacitação para avaliar os processos.	Oferece apoio na execução das atividades de GC.	Oferece níveis de maturidade para a avaliação dos processos.	Dispõe de níveis de maturidade para avaliar os processos e as capacidades correspondentes a cada nível.	Oferece nove áreas de conhecimento, descrevendo o emprego das atividades para aplicar a GC.
Flexibilidade	Pode-se adaptar conforme a necessidade da empresa.	Podem-se definir quais processos e práticas são empregados conforme a necessidade da empresa.	Esta norma é específica para atividade de GC.	Os níveis são a base do modelo, porém não podem ser alterados e sua representação é feita por estágio e contínua.	Dispõe de níveis voltados à realidade das empresas brasileiras.	Pode apoiar-se em outras normas e adaptar-se as necessidades da empresa.
Características gerais	Norma internacional, possui detalhes de como aplicá-la.	Norma internacional, estabelece diretriz para melhoria contínua.	Norma brasileira, descreve todos os passos das atividades de GC.	Norma internacional, possui uma aplicação flexível, por se apresentar em níveis.	Modelo de melhoria de processo do *software* brasileiro.	Modelo de melhoria de gerenciamento de projeto internacional.
Características de GC	Não apresenta em detalhes como executar a GC, apenas cita as atividades.	Apresenta dificuldade em se aplicar a GC, pois apenas cita as atividades.	Não apresenta modelos dos formulários e documentação necessários à GC.	Apresenta apenas o que deve ser feito e não os detalhes de como alcançar a GC.	Apresenta apenas o que deve ser feito e não os detalhes de como alcançar a GC.	Não trata com detalhes a GC e é voltada à gerência de projeto.
Possuem participação ou indicação de outra norma	Pioneira	ISO/IEC 12207	ISO/IEC 9001 E ISO/IEC 9004	CMM	ISO/IEC 12207, ISO/IEC 15504 E CMMI	SECM, SW-CMM, IPD-CMM
Localização da GC	Processo de apoio	5º Parte	-	Nível 2	Nível F	Área de gerencia-mento de integração de projeto

Quadro 11 - Comparação normas/modelos estudados (adaptado de TSUKUMO *et al.*, 1997)

Normas e modelos de Maturi-dade / Atividades de GC	ISO/IEC 12207	ISO/IEC 15504	ISO/IEC 10007	CMMI	MR-MPS	PMBOK
Identificar os itens de configuração	Atividade de identificação da configuração	Processo de identificar os artefatos	Parte 3 - identificação de configuração	Meta Específica 1	Objetivo 1, 2 e 3	Atividade de identificar e controlar
Gerenciar as mudanças nas versões e *releases*	Atividade de controle de mudança e Atividade de relato da situação da configuração.	Processo de controle de mudança	Parte 4 - controle de alterações Parte 5 - contabilização da situação da configuração	Meta Específica 2 Meta Genérica 2	Objetivo 6, 7, 8, 9,10 e 11	Atividade de revisar, aprovar e manter as *baseline*s; Atividade de gerenciamento, controle e documentação
Implantação de GC	Atividade de gerência de liberação e distribuição	Processo de liberação de versão	Parte 6 - auditoria de configuração	Meta Genérica 2	Objetivo 12, 13 e 14	Atividade de revisar, aprovar, e liberar

Quadro 12 – Identificação das atividades de GC atendidas em cada norma/modelo

Já no Quadro 13 informa **os artefatos** que são elaborados durante a execução das principais atividades de GC atendidas pelas normas e pelos modelos de maturidade estudados. Salienta-se que apenas as atividades em que é elaborado algum tipo de documentação relacionada à GC são consideradas nesse quadro.

Normas e modelos de maturi-dades / Atividades de GC	ISO/IEC 12207	ISO/IEC 15504	ISO/IEC 10007	CMMI	MR-MPS	PMBOK
Identificar os itens de configu-ração	Documentação da *baseline*, referência a versão e detalhes que podem ajudar na sua identificação	Documentação da *baseline*	Documentação da *baseline*	Relatório de gerenciamento dos artefatos e revisões	Documentação da *baseline*	Documentação da *baseline*
Gerenciar as mudanças, as versões e as *releases*	Formulário de solicitação de mudança, documentação das auditorias e histórico dos artefatos	Formulário de solicitação de mudança, histórico dos artefatos	Formulário de solicitação de mudança, documentação das auditorias e histórico dos artefatos	Relatório de status de mudança, lista de acesso em as *baselines*, documentação de auditorias e histórico dos artefatos	Formulário de solicitação de mudança; histórico dos artefatos	Formulário de solicitação de mudança; histórico dos artefatos
Implanta-ção de GC	Documentação formal e cópias de segurança	Documento de revisões das atividades de GC	Documento de revisões das atividades de GC	Documento de revisões das atividades de GC	Documento de revisões das atividades de GC	Documento de revisões das atividades de GC

Quadro 13 – Artefatos de GC elaborados pelas principais atividades de GC encontradas nas normas/modelos

4.9 Considerações Finais

Este capítulo abordou as diretrizes de algumas normas e modelos de maturidade que apóiam a execução da GC. Entre essas, se destacou a norma NBR ISO/IEC 10007, que por ser uma norma específica para o tratamento da GC, apresenta mais detalhes do que as demais normas e modelos de maturidade estudados. No entanto, não fornece gabaritos de formulários que devem ser elaborados durante a aplicação da GC.

Deve-se levar em consideração neste trabalho, que por se tratar de um modelo de referência de GC de um processo ágil baseado em *framework* devem-se tomar certas precauções, pois o processo não pode perder a sua característica ágil e também é necessário controlar as versões e as mudanças tanto do *framework* quanto do sistema alvo.

Para definir o modelo de referência proposto, considerando a problemática já exposta, no próximo capítulo são analisadas as normas e os modelos de maturidade, com o apoio do método *Goal Question Metric* (GQM) (BASILI *et al.*, 1994), para selecionar as atividades e os artefatos essenciais de GC para compor tal modelo.

5 MR.GC-PARFAIT: MODELO DE REFERÊNCIA DE GC DO PARFAIT

5.1 Considerações Iniciais

Neste capítulo apresentam-se a concepção e a documentação parcial do modelo de referência proposto. Na Seção 5.2 apresentam-se as etapas seguidas para definir as atividades e os artefatos de GC do modelo de referência, bem como a aplicabilidade das atividades desse modelo nas atividades do PARFAIT. Mais especificamente, na Seção 5.2.1 mostra-se a primeira etapa que seleciona as atividades essenciais de GC para que não haja perda da agilidade do PARFAIT. A Seção 5.2.2, indica a segunda etapa que seleciona os artefatos essenciais de GC para garantir a integridade dos artefatos produzidos pelo PARFAIT, com a mesma preocupação da etapa anterior, ou seja, sem afetar as práticas de modelagem ágil utilizadas pelo processo. A Seção 5.2.3 mostra a terceira etapa com a identificação da aplicabilidade das atividades e dos artefatos essenciais de GC nas atividades do PARFAIT. Na Seção 5.3 apresenta-se a quarta etapa que mostra parte da documentação do modelo de referência proposto, baseada na estrutura da documentação do RUP (*Rational Unified Process*), contendo a indicação da aplicação de cada atividade de GC do modelo no PARFAIT. Na Seção 5.4 encontram-se as considerações finais deste capítulo.

5.2 Etapas para definição do MR.GC-PARFAIT

Para criar o modelo de referência da atividade de GC do processo PARFAIT, denominado MR.GC-PARFAIT (FERREIRA; CAGNIN, 2007), foram selecionadas inicialmente apenas as atividades e os artefatos essenciais de GC presentes nas normas e nos modelos de maturidade estudados, sem afetar a aplicabilidade das práticas de modelagem ágil definidas na proposição desse processo. Para isso, foram seguidas duas etapas distintas, conforme revela a Figura 6: a primeira etapa trata da seleção das atividades de GC e a segunda trata da seleção dos artefatos. Para a seleção dos artefatos, foram considerados aqueles produzidos nas atividades selecionadas. Posteriormente, foi realizada uma terceira etapa, que identificou em que momento da execução do PARFAIT as

atividades de GC selecionadas deveriam ser aplicadas. E, por fim, na quarta etapa a documentação do modelo de referência foi elaborada. Todas as etapas foram realizadas de maneira incremental, a fim de refinar o modelo de referência, conforme ilustrado na Figura 6.

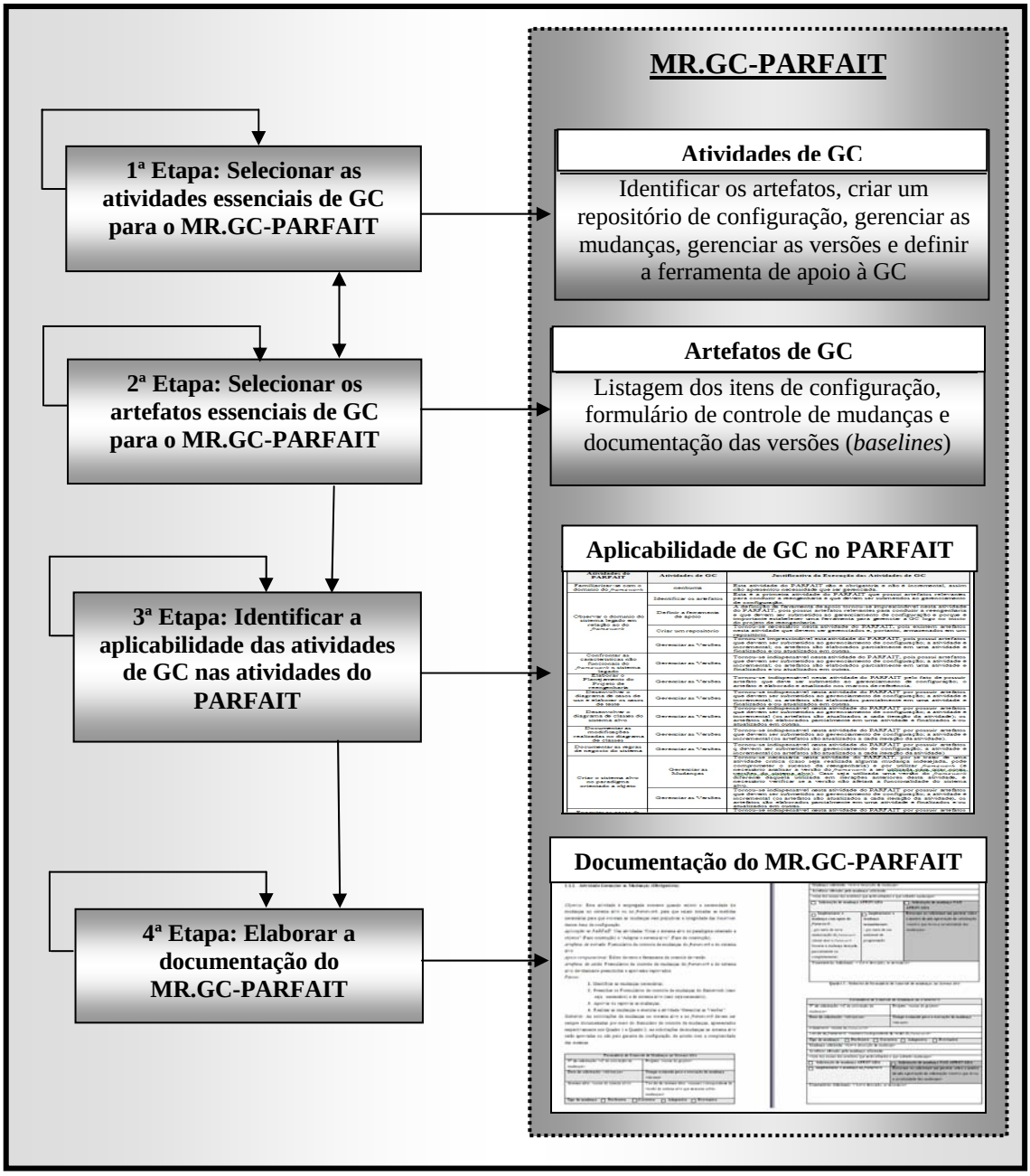

Figura 6 - Visão geral da definição do MR.GC-PARFAIT

5.2.1 1ª Etapa: Selecionar as Atividades Essenciais de GC

Nesta etapa realizou-se a seleção das atividades essenciais de GC para o MR.GC-PARFAIT, levando-se em consideração a problemática do controle de versões na reengenharia baseada em *framework*, discutida na Seção 3.3.3, bem como a permanência da agilidade do processo PARFAIT.

Para apoiar a análise quantitativa e objetiva das normas e dos modelos de maturidade a fim de selecionar as atividades essenciais de GC empregou-se o método de mensuração *Goal Question Metric* (GQM) (BASILI *et al.*, 1994). Esse método foi inicialmente desenvolvido para avaliar defeitos em projetos desenvolvidos pela NASA, na década de 80 e, posteriormente, foi estendido para um contexto mais amplo. Assim, esse método é orientado a objetivo/meta cuja aplicação é feita por meio da coleta de dados, com um objetivo específico predeterminado. Os passos adotados por este método são estabelecidos "de cima para baixo" pelos objetivos/metas, questões e métricas, sendo o resultado interpretado "de baixo para cima", a fim de alcançar a meta proposta por este modelo (SOLINGEN; BERGHOUT, 1999), como ilustrado na Figura 7.

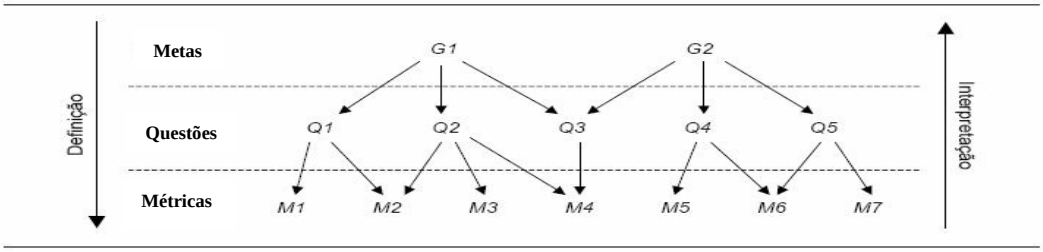

Figura 7 – Método GQM (SOLINGEN; BERGHOUT, 1999)

Segundo Gresse (1996), os objetivos/metas que devem ser alcançados com a aplicação do GQM apresentam-se em cinco maneiras distintas: objetivo do estudo (conjunto de metas relacionadas ao processo ou ao produto); propósito (que determina, caracteriza, melhora e controla questões de relevância fundamental); foco da qualidade (meta relacionada à qualidade); ponto de vista (meta relacionada à satisfação a quem interessa os resultados almejados); e ambiente (contexto relacionado à interpretação dos resultados).

As questões no GQM estão relacionadas diretamente ao objetivo proposto. Já as métricas alcançam as questões por meio dos resultados coletados de maneira quantitativa, visando obter as informações necessárias para atender ao objetivo proposto.

Para apoiar a aplicação do GQM neste trabalho, foi estabelecida uma métrica, denominada Grau de Importância (FERREIRA; CAGNIN, 2006). O objetivo dessa métrica é quantificar a importância dos elementos em um determinado contexto. No caso deste trabalho observou-se o percentual de ocorrência das atividades e dos artefatos de GC nas normas e modelos de maturidade estudados para compor o modelo de referência MR.GC-PARFAIT.

Para estabelecer o grau de importância, é necessário utilizar a seguinte escala: **grau 5** (extremamente importante): percentual de ocorrência de 100% a 80%; **grau 4** (importante): de 79% a 60%; **grau 3** (importância moderada): de 59% a 40%; **grau 2** (pouco importante): de 39% 20%; **grau 1** (sem importância): de 19% a 0%.

Dando início à aplicação do GQM para apoiar a seleção das atividades para compor o modelo de referência, foram observados os seguintes dados: a ocorrência das atividades de GC, definidas por Sommerville (2003), nas normas e nos modelos de maturidade estudados (CMMI, ISO/IEC 12207, ISO/IEC 15504, ISO/IEC 10007, PMBOK, MR-MPS); o percentual das práticas de modelagem ágil consideradas pelo PARFAIT (documente de maneira ágil, organize uma participação ativa dos clientes, atualize apenas quando necessário, crie conteúdo simples, modele incrementalmente, crie diversos modelos em paralelo), que não são prejudicadas por cada atividade de GC; e o percentual de apoio dessas atividades de GC no controle de versões tanto do *framework* quanto do sistema alvo, conforme apresentado no Quadro 14.

Considerando a questão 1 elaborada com a aplicação do GQM (Quadro 14), observou-se a ocorrência de cada atividade de GC, definida por Sommerville (2003), em cada norma e modelo de maturidade, conforme apresentado nas colunas dois a sete do Quadro 15. A partir disso, aplicou-se a métrica 1, obtendo-se o percentual de ocorrência. Exemplificando, a atividade “identificar os artefatos” (1) está presente em todas as normas e modelos de maturidade. Assim, o valor da métrica 1 para essa atividade é de 100%. Posteriormente, estabeleceu-se o grau de importância (métrica 1.1) a partir do resultado obtido na métrica 1. Nesse caso, o resultado da métrica 1.1 para a atividade de GC “identificar os artefatos” é de 5 (extremamente importante).

Objetivo: Analisar as atividades de GC dos modelos de maturidade CMMI, ISO/IEC 12207, ISO/IEC 15504, ISO/IEC 10007, PMBOK e MR-MPS **com o propósito de** selecionar as atividades essenciais de GC para o PARFAIT sem afetar sua agilidade e apoiar o controle das versões tanto do *framework* quanto das geradas a partir dele, **do ponto de vista de** GC de *Software*, **no contexto de** processos de reengenharia.
Questão 1: A atividade X[9] está presente na maioria das normas e modelos de maturidade?
Métrica 1: Percentual de ocorrência da atividade X nas normas e modelos de maturidade.
Métrica 1.1: Valor do grau de importância.
Questão 2: A agilidade do PARFAIT não é prejudicada pela atividade X?
Métrica 2: Percentual das práticas ágeis, atendidas pelo PARFAIT, que não são prejudicadas pela atividade X.
Métrica 2.1: Valor do grau de importância.
Questão 3: A atividade X apóia o controle das versões do *framework* e das aplicações geradas a partir dele?
Métrica 3: Percentual de apoio do controle de versões presente na atividade X.
Métrica 3.1: Valor do grau de importância.
Métrica 3.2: Média ponderada do grau de importância obtido nas métricas 1.1, 2.1 e 3.1 ...

Quadro 14 – Plano de mensuração GQM das atividades de GC

Atividades de GC - Questão 1								
Atividades de GC	**CMMI**	**ISO/IEC 12207**	**ISO/IEC 15504**	**ISO/IEC 10007**	**PMBOK**	**MR-MPS**	**Métrica 1**	**Métrica 1.1**
1 - Identificar os itens de configuração	√	√	√	√	√	√	100%	5
2 - Criar um repositório de configuração	√	√	√	√	√	√	100%	5
3 - Gerenciar as mudanças	√	√	√	√	√	√	100%	5
4 - Gerenciar as versões	√	√	√	√	√	√	100%	5
5 - Gerenciar as *releases*		-	√	√	√		50%	3
6 - Efetuar as revisões de GC	√	-	-	-	√	√	50%	3
7 - Definir a ferramenta de apoio à GC	√	√	√	√	√	√	100%	5

Quadro 15 - Resultado da questão 1

Com relação à questão 2 foi observado se cada prática de modelagem ágil atendida pelo PARFAIT não é prejudicada de alguma maneira pelas atividades de GC de Sommerville (2003), conforme apresentado nas colunas dois a sete do Quadro 16. Com isso, aplicou-se a métrica 2,

[9] Cada atividade de GC considerada por Sommerville (2003): 1) identificar os artefatos, 2) criar um repositório de configuração, 3) gerenciar as mudanças, 4) gerenciar as versões, 5) gerenciar as *releases*, 6) efetuar as revisões de GC e 7) definir a ferramenta de apoio à GC.

obtendo-se o percentual das práticas de modelagem ágil que não são prejudicadas. Exemplificando, as atividades de GC “gerenciar as *releases*” (5) e “efetuar as revisões de GC” (6) prejudicam metade das práticas de modelagem ágil atendidas pelo PARFAIT, ou seja, as práticas de modelagem ágil “crie apenas documentos extremamente necessários”, “atualize apenas quando necessário” e “crie conteúdo simples”. Tais práticas são prejudicadas uma vez que tais atividades produzem documentação importante, mas não essencial para o controle de versões exigido pelo PARFAIT. Em seguida, obteve-se o grau de importância (métrica 2.1) com valor 3 (importância moderada) a partir do resultado obtido na métrica 2.

Atividades de GC - Questão 2								
Atividade de GC	**Crie apenas documentos extremamente necessários**	**Organize uma participação ativa dos clientes**	**Atualize apenas quando necessário**	**Crie conteúdo simples**	**Modele incremental-mente**	**Crie diversos modelos em paralelo**	**Métrica 2**	**Métrica 2.1**
1 - Identificar os itens de configuração	√	√	√	√	√	√	100%	5
2 - Criar um repositório de configuração	√	√	√	√	√	√	100%	5
3 - Gerenciar as mudanças	√	√	√	√	√	√	100%	5
4 - Gerenciar as versões	√	√	√	√	√	√	100%	5
5 - Gerenciar as *releases*	-	√	-	-	√	√	50%	3
6 – Efetuar as revisões de GC	-	√	-	-	√	√	50%	3
7 – Definir a ferramenta de apoio à GC	√	√	√	√	√	√	100%	5

Quadro 16 – Resultado da questão 2

No Quadro 18 verificou-se se cada atividade de GC interfere de alguma maneira, nas práticas de modelagem ágil presentes no PARFAIT. Considerando a atividade de GC “identificar os itens de configuração” observou-se que a mesma não interfere na prática ágil “crie apenas documentos extremamente necessários”, pois é essencial identificar logo no começo quais artefatos serão elaborados no projeto de reengenharia. Já, com relação a prática “organize uma participação ativa dos clientes”, o cliente pode validar os artefatos que serão submetidos a GC podendo acrescentar ou retirar algum, caso seja necessário. Na prática “atualize apenas quando necessário” caso ocorram mudanças será necessário atualizar esta listagem de itens de configuração. Na prática “crie conteúdo simples”, o gabarito desta atividade pode na forma de *check-list*, assim fica a critério do engenheiro de software decidir para cada projeto empregar ou não GC em cada artefato. Na prática “modele incrementalmente”, a identificação dos itens de configuração não irá prejudicar em

tal prática. Já na prática “crie diversos modelos em paralelo”, como esta identificação é uma sugestão, não interferirá na criação de outra lista ou no acréscimo ou retirada de alguma das opções sugeridas.

Já na questão 3 (Quadro 14), verificou-se se cada atividade de GC apóia ou não o controle de versão tanto do *framework* quanto do sistema alvo, conforme apresentado nas colunas dois e três do Quadro 18. Adicionalmente, analisou-se se cada atividade de GC apóia o controle de dependência de versões entre o *framework* e o sistema alvo, conforme ilustrado na coluna quatro do Quadro 18. Exemplificando, a atividade “criar um repositório de configuração” (2) apóia o controle de versões do sistema alvo, o controle de versões do *framework* e a dependência entre as versões dos mesmos. Ressalta-se que o apóio da dependência entre as versões tanto do *framework* quanto do sistema alvo é possível por meio de uma seqüência de pastas dentro do repositório da ferramenta de controle de versão *Subversion* (Seção 6.3.3). As demais atividades não apóiam a dependência entre *framework* e o sistema alvo. Assim, o valor da métrica 3 para a atividade “Criar um repositório de configuração” é de 100%. Posteriormente, estabeleceu-se o grau de importância com o emprego da métrica 3.1, a partir do resultado obtido na métrica 3. Para a atividade “gerenciar as versões” (4) o valor obtido da métrica 3.1 é de 4.

Atividades de GC - Questão 3					
Atividades de GC	**Apóia o controle de versão do *framework***	**Apóia o controle de versão do sistema alvo**	**Apóia a dependência entre *framework* e sistema alvo**	**Métrica 3**	**Métrica 3.1**
1 - Identificar os itens de configuração	√	√	-	67%	4
2 - Criar um repositório de configuração	√	√	√	100%	5
3 - Gerenciar as mudanças	√	√	-	67%	4
4 - Gerenciar as versões	√	√	-	67%	4
5 - Gerenciar as *releases*	√	√	-	67%	4
6 - Efetuar as revisões de GC	√	√	-	67%	4
7 - Definir a ferramenta de apoio à GC	√	√	-	67%	4

Quadro 17 – Resultado da questão 3

Atividade de GC	Crie apenas documentos extremamente necessários	Organize uma participação ativa dos clientes	Atualize apenas quando necessário	Crie conteúdo simples	Modele incrementalmente	Crie diversos modelos em paralelo
1 – Identificar os itens de configu-ração	A elaboração da listagem dos itens de configuração se tornou essencial logo no começo do projeto de reengenharia, pois os artefatos serão considerados e transformados em *baselines*, assim **não** prejudica tal prática ágil	A participação dos clientes pode validar, acrescentar ou retirar algum dos artefatos que serão submetidos a GC, assim **não** prejudica tal prática ágil	Caso as mudanças ocorram, será necessário atualizar o gabarito dos itens de configuração, assim esta atividade **não** prejudica tal prática ágil	Esta atividade sugere um gabarito simples dos possíveis artefatos que serão submetidos a GC, ficaria a critério de cada projeto empregar ou não os artefatos sugeridos, assim **não** prejudica tal prática ágil	Tal prática **não** será prejudicada por tal atividade, pois é realizada apenas uma vez	Devido o gabarito ser uma sugestão não interferirá a criação de outra ou até mesmo acrescentar ou retirar alguma das opções sugeridas, assim **não** prejudica tal prática ágil
2 - Criar um repositó-rio de configu-ração	A criação do repositório se tornou de vital importância quando se trata de gerenciar com qualidade os artefatos produzidos durante o projeto de reengenharia, assim **não** prejudica tal prática ágil	As participações dos clientes podem de uma maneira ou de outra influenciar a criação do repositório que terá particularidades próprias de cada projeto de reengenharia, assim **não** prejudica tal prática ágil	Devido ao uso da ferramenta *Subversion*, caso seja necessário existe a possibilidade de mudar o nome, o local do repositório, acrescentar ou retirar algum item de configuração, assim **não** prejudica tal prática ágil	O repositório apresenta-se de maneira simples e satisfatória para condução do projeto de reengenharia, assim **não** prejudica tal prática ágil	Tal prática **não** será prejudicada por tal atividade, devido ser realizada apenas uma vez	O repositório será criado apenas uma vez, não será necessário ter vários modelos e **não** prejudica tal prática ágil
3 - Gerenciar as mudanças	Tal atividade é extremamente necessária por documentar por meio de gabaritos as mudanças ocorridas nos itens de configuração elaborados durante o projeto de reengenharia e **não** prejudica tal prática ágil	Por meio da solicitação de mudanças necessárias ou solicitadas pelo cliente é que se gerencia tal mudança que **não** prejudica tal prática ágil	O gerenciamento da mudança só será realizado caso seja necessário, assim **não** prejudica tal prática ágil	O gerenciamento será realizado por meio dos gabaritos sugeridos por tal atividade de GC, que se apresenta de maneira simples e de rápido preenchimento, assim **não** prejudica tal prática ágil	O desenvolvimento do projeto será incrementalmente implementado de acordo com a necessidade imposta por tal projeto, assim tal atividade **não** prejudica esta prática ágil	Caso seja necessário poderá criar versões diferentes devido a uma solicitação de um cliente, assim **não** prejudica tal prática ágil
4 - Gerenciar as versões	A documentação criada pelo gerenciamento das versões é extremamente necessária para controlar melhor as versões de cada artefato, assim **não** prejudica tal prática ágil	Cada versão do sistema pode possuir alguma particularidade diferente ou exclusiva para um determinado cliente. Por meio deste gerenciamento será de fácil localização e acesso a tal versão, assim **não** prejudica tal prática ágil	A documentação do gerenciamento de versão só será atualizada quando necessário, assim **não** prejudica essa prática ágil	O gabarito desta atividade se apresenta simples e de fácil preenchimento, assim **não** prejudica tal prática ágil	Os artefatos são criados incrementalmente, sendo que as versões dos mesmos podem ser atualizadas a qualquer momento do projeto de reengenharia, assim **não** prejudica tal prática ágil	Devido às diversas particulares que cada versão do sistema possa ter, assim tal atividade **não** prejudica tal prática ágil devido principalmente ao gerenciamento proporcionado por esta atividade
5 - Gerenciar as *releases*	A documentação criada nesta atividade é repetida, devido ser a mesma apresentada na atividade “Gerenciar as versões”, assim se tornou inviável armazená-la no repositório e assim prejudica tal prática ágil	Se cada versão possui alguma particularidade especial para um determinado cliente isso depende principalmente da participação do cliente, assim **não** prejudica tal prática ágil	A atualização da documentação criada nesta atividade será a mesma apresentada pela atividade “Gerenciar as versões”, assim prejudica tal prática ágil por apresentar uma documentação duplicada	O conteúdo desta atividade não será complexo, pois seria a mesma da atividade “Gerenciar as versões” e isso **não** prejudicaria tal prática	Tal atividade será realizada apenas quando o sistema será entregue ao cliente, assim **não** possuirá nenhuma modelagem incremental	Tal atividade criará várias *releases* e **não** prejudicará tal prática ágil

6 – Efetuar as revisões de GC	Esta atividade apresentará várias documentações que não serão extremamente necessárias, assim tal prática será prejudicada por tal atividade	A validação pelo o usuário das revisões pode auxiliar esta atividade, assim tal prática ágil **não** será prejudicada	Atualizar esta atividade pode prejudicar a agilidade por apresentar vários documentos, assim pode prejudicar tal prática ágil	Esta atividade apresenta documentação complexa, assim prejudica tal prática ágil	Esta atividade criar várias documentações e pode ser realizada incrementalmente, assim **não** prejudica tal prática ágil	Esta atividade cria várias documentações, mas caso for necessário pode criar outros modelos, assim **não** prejudica tal prática ágil
7 – Definir a ferra-menta de apoio à GC	Esta atividade não cria documentação, assim **não** prejudica tal prática ágil	A participação do cliente pode facilitar a escolha da ferramenta empregada na GC, assim **não** prejudica tal prática ágil	Esta atividade é realizada uma única vez, assim **não** prejudica tal prática ágil	Esta atividade não cria documentação, assim **não** prejudica tal prática ágil	Esta atividade não cria documentação, assim **não** prejudica tal prática ágil	Esta atividade não cria documentação, assim **não** prejudica tal prática ágil

Quadro 18 – Justificativas das práticas ágeis

Dando prosseguimento à primeira etapa, foi estabelecida a métrica 3.2, que trata da média ponderada dos resultados obtidos com a aplicação das métricas 1.1, 2.1 e 3.1, Quadro 19. A partir do resultado obtido no Quadro 19, observou-se que as atividades 1 (identificar os artefatos), 2 (criar um repositório de configuração), 3 (gerenciar as mudanças), 4 (gerenciar as versões) e 7 (definir a ferramenta de apoio à GC) são consideradas essenciais para a GC e devem estar presentes no MR.GC-PARFAIT.

Atividades de GC	Métrica 1.1	Métrica 2.1	Métrica 3.1	Métrica 3.2
1 - Identificar os itens de configuração	5	5	4	4,7 ➔ 5
2 - Criar um repositório de configuração	5	5	5	5
3 - Gerenciar as mudanças	5	5	4	4,7 ➔ 5
4 - Gerenciar as versões	5	5	4	4,7 ➔ 5
5 - Gerenciar as *releases*	3	3	4	3,4 ➔ 3
6 - Efetuar as revisões de GC	3	3	4	3,4 ➔ 3
7 - Definir a ferramenta de apoio à GC	5	5	4	4,7 ➔ 5

Quadro 19 - Resultados da média ponderada das métricas das atividades de GC

5.2.2 2ª Etapa: Selecionar os Artefatos Essenciais de GC

A segunda etapa foi iniciada com o resultado obtido na seleção das atividades essenciais de GC com grau de importância 5[10] para compor o modelo MR.GC-PARFAIT, realizada na etapa anterior. Para iniciar esta etapa, optou-se também pela utilização do método *Goal Question Metric* (GQM) para identificar quais artefatos presentes nas atividades selecionadas são relevantes para serem incorporados ao modelo de referência, conforme apresentado no Quadro 20. Para isso, foi analisado o percentual de ocorrência de cada artefato produzido pelas atividades de GC

[10] Atividades de GC selecionadas: Identificar os artefatos; Criar um repositório de configuração; Gerenciar as mudanças; Gerenciar as Versões; Definir a ferramenta de apoio à GC.

selecionadas, nas normas e modelos de maturidade. Adicionalmente, analisou-se também o percentual das práticas de modelagem ágil que não são prejudicadas pelos artefatos.

Objetivo: Analisar os artefatos das atividades essenciais de GC, presentes nos modelos de maturidade CMMI, ISO/IEC 12207, ISO/IEC 15504, ISO/IEC 10007, PMBOK e MR-MPS **com o propósito de** selecionar os artefatos essenciais de GC para o PARFAIT, sem afetar sua agilidade, e apoiar o controle das versões tanto do *framework* quanto das aplicações geradas à parte dele **do ponto de vista de** GC de *Software*, **no contexto de** processos de reengenharia.
Questão 4: O artefato Y[11] está presente na maioria das normas e modelos de maturidade?
Métrica 4: Percentual de ocorrência do artefato Y nas normas e modelos de maturidade.
Métrica 4.1: Valor do grau de importância.
Questão 5: A agilidade do PARFAIT não é prejudicada pelo artefato Y?
Métrica 5: Percentual das práticas de Modelagem Ágil atendidas pelo PARFAIT, que não são prejudiciais pelo artefato Y.
Métrica 5.1: Valor do grau de importância.
Métrica 5.2: Média ponderada do grau de importância obtido nas métricas 4.1 e 5.1 ...

Quadro 20 – Plano de mensuração GQM dos artefatos elaborados pelas atividades de GC

Considerando a questão 4, elaborada com a aplicação do GQM (Quadro 20), observou-se a ocorrência de cada artefato de GC em cada norma e modelo de maturidade estudado, conforme apresentado no Quadro 21. A partir disso, aplicou-se a métrica 4, obtendo o percentual de ocorrência. Como pode ser visto no Quadro 20, todos os artefatos elaborados nas atividades de GC selecionadas estão presentes em todas as normas e modelos de maturidade. Assim, o valor da métrica 4 é de 100%. Em seguida, estabeleceu-se o grau de importância (métrica 4.1), obtendo-se 5 (extremamente importante) como resultado.

Com relação à questão 5, foi verificado se cada prática de modelagem ágil atendida pelo PARFAIT não é prejudicada pelos artefatos de GC. Com isso, aplicou-se a métrica 5 obtendo-se o percentual das práticas ágeis que não são prejudicadas pelos artefatos, conforme apresentado nas colunas dois a sete do Quadro 22. Observa-se que todos os artefatos não prejudicam nenhuma prática de modelagem ágil atendida pelo PARFAIT, ou seja, o valor da métrica 5 é de 100% para todos os artefatos. Em seguida, obteve-se o grau de importância 5 (extremamente importante) (métrica 5.1) a partir do resultado da métrica 5.

[11] Documento da *baseline*, formulário de controle de mudança e histórico dos artefatos.

Artefatos de GC – Questão 4								
Artefato	**CMMI**	**ISO/IEC 12207**	**ISO/IEC 15504**	**ISO/IEC 10007**	**PMBOK**	**MR-MPS**	**Métrica 4**	**Métrica 4.1**
1 - Documento da *baseline*	√	√	√	√	√	√	100%	5
2 - Formulário de controle de mudança	√	√	√	√	√	√	100%	5
3 - Histórico dos artefatos	√	√	√	√	√	√	100%	5

Quadro 21 - Resultados da questão 4

Artefatos de GC – Questão 5								
Artefato	**Crie apenas documentos extremamente necessários**	**Organize uma participa-ção ativa dos clientes**	**Atualize apenas quando necessário**	**Crie conteúdo simples**	**Modele incremental-mente**	**Crie diversos modelos em paralelo**	**Métrica 5**	**Métrica 5.1**
1 - Documento da *baseline*	√	√	√	√	√	√	100%	5
2 - Formulário de controle de mudança	√	√	√	√	√	√	100%	5
3 - Histórico dos artefatos	√	√	√	√	√	√	100%	5

Quadro 22 – Resultado da questão 5

No Quadro 23 verificou-se se cada artefato criado pelas atividades selecionadas na etapa anterior interfere, de alguma maneira, nas práticas de modelagem ágil presentes no PARFAIT. Considerando o artefato de GC "documento da *baseline*" (1) observou-se que o mesmo não interfere na prática ágil "crie apenas documentos extremamente necessários", pois é essencial documentar a *baseline* logo no início para facilitar a elaboração do projeto de reengenharia. Já, com relação a prática "organize uma participação ativa dos clientes", o cliente pode validar tal documentação podendo acrescentar ou alterar alguma informação, caso seja necessário. Na prática ágil "atualize apenas quando necessário" observou-se que caso seja necessário realizar algumas mudanças, a documentação da *baseline* deverá ser atualizada e tal procedimento não irá prejudicar tal prática ágil. Na prática ágil "crie conteúdo simples" notou-se que a utilização de um gabarito da documentação da *baseline* poderá facilitar o preenchimento e simplificar tal documentação, assim não irá prejudicar tal prática ágil. Na prática ágil "modele incrementalmente", a documentação da *baseline* não irá prejudicar tal prática pois a documentação poderá ser criada ou evoluída de acordo com a evolução de um determinado artefato, já com relação a prática ágil "crie diversos modelos

em paralelo" como a documentação da *baseline* é sugerida, fica a critério do gerente do projeto de reengenharia alterá-la ou criar um modelo específico ou mais apropriado à necessidade do projeto.

Artefato	Crie apenas documentos extremamente necessários	Organize uma participação ativa dos clientes	Atualize apenas quando necessário	Crie conteúdo simples	Modele incremental-mente	Crie diversos modelos em paralelo
1 - Documento da *baseline*	O gabarito produzido é extremamente necessária para controlar melhor as versões de cada artefato, assim **não** prejudica tal prática ágil	O gabarito produzido pode ter alguma característica exclusiva para um determinado cliente, assim **não** prejudica tal prática ágil	O gabarito produzido pode ser atualizado quando necessário, assim **não** prejudica tal prática ágil	O conteúdo criado por tal gabarito será simples e **não** prejudica tal prática ágil	O gabarito pode ser produzido incrementalmente caso necessário, assim **não** prejudica tal prática ágil	O gabarito produzido pode ser criado em paralelo caso necessário, assim **não** prejudica tal prática ágil
2 - Formulário de controle de mudança	O gabarito produzido é extremamente necessária para controlar melhor as mudanças de cada artefato, assim **não** prejudica tal prática ágil	O gabarito produzido pode ter alguma característica exclusiva para um determinado cliente, assim **não** prejudica tal prática ágil	O gabarito produzido por esse artefato pode ser atualizada quando necessário, assim **não** prejudica tal prática ágil	O conteúdo criado por tal gabarito será simples e **não** prejudica tal prática ágil	O gabarito pode ser produzido incrementalmente caso necessário, assim **não** prejudica tal prática ágil	O gabarito produzido pode ser criado em paralelo caso necessário, assim **não** prejudica tal prática ágil
3 - Histórico dos artefatos	O gabarito produzido é extremamente necessária para controlar o histórico dos artefatos, assim **não** prejudica tal prática ágil	O gabarito produzido pode ter alguma característica exclusiva para um determinado cliente, assim **não** prejudica tal prática ágil	O gabarito produzido por esse artefato pode ser atualizada quando necessário, assim **não** prejudica tal prática ágil	O conteúdo criado por tal gabarito será simples e **não** prejudica tal prática ágil	O gabarito pode ser produzido incrementalmente caso necessário, assim **não** prejudica tal prática ágil	O gabarito produzido pode ser criado em paralelo caso necessário, assim **não** prejudica tal prática ágil

Quadro 23 – Justificativas das práticas ágeis

Dando continuidade à segunda etapa, foi estabelecida a métrica 5.2, que trata da média ponderada dos resultados obtidos com a aplicação das métricas 4.1 e 5.1, conforme apresentado no Quadro 24. A partir disso, observou-se que todos os artefatos de GC foram considerados como sendo essenciais e que devem compor o modelo de referência do PARFAIT.

Atividade de GC	Métrica 4.1	Métrica 5.1	Métrica 5.2
1 - Documento da *baseline*	5	5	5
2 - Formulário de controle de mudança	5	5	5
3 - Histórico dos artefatos	5	5	5

Quadro 24 - Resultados da média ponderada das métricas dos artefatos de GC

5.2.3 3ª Etapa: Identificar a Aplicabilidade das Atividades Essenciais de GC nas Atividades do PARFAIT

Ao concluir as etapas um e dois, observou-se a necessidade de identificar a aplicabilidade das atividades de GC selecionadas para o modelo de referência, nas atividades do PARFAIT. Para apoiar esta terceira etapa, foi realizada uma análise da documentação do processo PARFAIT a fim de selecionar as suas atividades que deverão ser submetidas à GC com o apoio do MR.GC-PARFAIT.

A análise realizada foi baseada em três critérios: 1) se determinada atividade do processo é incremental (conseqüentemente, os artefatos são elaborados inicialmente e posteriormente refinados); 2) se determinada atividade do processo produz artefatos relevantes para conduzir a reengenharia e que devem ser submetidos ao gerenciamento de configuração; e 3) se os artefatos produzidos em uma determinada atividade não incremental são atualizados em outras atividades ou nos marcos de referência.

O resultado da análise realizada (Quadro 25) apresenta-se uma sugestão das atividades de GC que devem ser adotadas durante a aplicação do MR.GC-PARFAIT no PARFAIT, entretanto a sua real aplicabilidade deve ser analisada de acordo com as necessidades do projeto de reengenharia. Neste quadro está estabelecida a relação entre as atividades do processo PARFAIT, as atividades de GC selecionadas para o modelo de referência deste processo e a justificativa do emprego das atividades de GC em uma determinada atividade do PARFAIT, de acordo com os critérios citados anteriormente.

Ressalta-se que a atividade de GC “gerenciar mudanças” foi necessária na atividade do PARFAIT “Criar o sistema alvo no paradigma orientado a objeto”, por ser considerada crítica (uma vez que, caso seja realizada alguma mudança indesejada no sistema alvo, pode comprometer o sucesso da reengenharia) e por utilizar *framework* (sendo necessário analisar a versão do mesmo a ser utilizada para criar novas versões do sistema alvo). Caso seja utilizada uma versão do *framework* diferente daquela utilizada em iterações anteriores desta atividade, é necessário verificar se a versão não afetará o comportamento do sistema alvo. Já a atividade de GC “gerenciar as versões” está presente em todas as atividades do PARFAIT que compõem o modelo de referência, pois em cada uma destas atividades é elaborado algum artefato relevante, que necessite deste controle.

Atividades do PARFAIT	Atividades de GC	Justificativa da Execução das Atividades de GC
Familiarizar-se com o domínio do *framework*	Nenhuma	Esta atividade do PARFAIT não é obrigatória e não é incremental, assim não apresentou necessidade de ser gerenciada.
Observar o domínio do sistema legado em relação ao do *framework*	Identificar os itens de configuração	Esta atividade do PARFAIT satisfaz os critérios 1 e 2. A ordem das atividades de GC foi determinada de acordo com os artefatos de entrada necessários para a execução de cada atividade (pré-requisito).
	Definir a ferramenta de apoio	
	Criar um repositório	
	Gerenciar as Versões	
Confrontar as características não funcionais do *framework* x sistema legado	Gerenciar as Versões	Esta atividade do PARFAIT satisfaz os critérios 1 e 2.
Elaborar o Planejamento do Projeto de reengenharia	Gerenciar as Versões	Esta atividade do PARFAIT satisfaz os critérios 1 e 3.
Desenvolver o diagrama de casos de uso e elaborar os casos de teste	Gerenciar as Mudanças	Esta atividade do PARFAIT satisfaz os critérios 1 e 2.
	Gerenciar as Versões	
Desenvolver o diagrama de classes do sistema alvo	Gerenciar as Versões	Esta atividade do PARFAIT satisfaz os critérios 1 e 2.
Documentar as modificações realizadas no diagrama de classes	Gerenciar as Versões	Esta atividade do PARFAIT satisfaz os critérios 1 e 2.
Documentar as regras de negócio do sistema	Gerenciar as Versões	Esta atividade do PARFAIT satisfaz os critérios 1 e 2.
Criar o sistema alvo no paradigma orientado a objeto	Gerenciar as Mudanças	Esta atividade do PARFAIT satisfaz os critérios 1 e 2.
	Gerenciar as Versões	Esta atividade do PARFAIT satisfaz os critérios 1 e 2.
Executar os casos de teste no sistema alvo	Gerenciar as Versões	Esta atividade do PARFAIT satisfaz os critérios 1 e 2.
Adaptar o sistema alvo	Gerenciar as Mudanças	Esta atividade do PARFAIT satisfaz os critérios 1 e 2. Além disso, no passo "Executar o sistema alvo concomitantemente com o sistema legado" desta atividade do PARFAIT é necessário analisar se as mudanças solicitadas pelos usuários são importantes e devem ser efetuadas no sistema alvo.
	Gerenciar as Versões	Esta atividade do PARFAIT satisfaz os critérios 1 e 2.
Elaborar o manual do usuário do sistema alvo	Gerenciar as Versões	Esta atividade do PARFAIT satisfaz os critérios 1 e 2.
Converter a base de dados do sistema legado	-	Não satisfaz nenhum critério, pois não possui artefato para ser gerenciado
Testar o sistema alvo	Gerenciar as Versões	Esta atividade do PARFAIT satisfaz os critérios 1 e 2.
Treinar os usuários finais	-	Não satisfaz nenhum critério, pois não possui artefato para ser gerenciado

Quadro 25 - Atividades do PARFAIT versus atividades de GC

5.3 4ª Etapa: Elaborar a documentação do MR.GC-PARFAIT

O formato da documentação do MR.GC-PARFAIT (FERREIRA; CAGNIN, 2007) definido neste estudo segue aquele utilizado no RUP: atividade, objetivo, artefatos de entrada e de

saída, apoio computacional, passos e gabaritos. Além disso, um item foi adicionado na documentação, que é “aplicação no PARFAIT”, ou seja, em que momento do processo PARFAIT (atividade) determinada atividade de GC do modelo de referência deve ser executada.

Sugere-se que o responsável pela reengenharia determine um membro da equipe como gerente de configuração, que deve ter conhecimento e habilidade de executar as atividades impostas pelo MR.GC-PARFAIT. Um breve resumo da documentação do MR.GC-PARFAIT está presente no Quadro 26, apresentando o objetivo de cada atividade de GC, os artefatos elaborados e a aplicabilidade de tal atividade no PARFAIT.

Atividade de GC	Objetivo	Artefatos Produzidos	Aplicabilidade no PARFAIT
Identificar os itens de configuração	Selecionar os artefatos do PARFAIT que deverão ser submetidos à GC	Listagem dos itens de configuração	Atividade: Observar o domínio do sistema legado em relação ao *framework*
Definir a ferramenta de apoio	Definir a ferramenta que melhor apóie a GC	-	Atividade: Observar o domínio do sistema legado em relação ao *framework*
Criar um repositório de configuração	Armazenar os itens de configuração no repositório da ferramenta de controle de versão	Repositório criado	Atividade: Observar o domínio do sistema legado em relação ao *framework*
Gerenciar as versões	Gerenciar as versões do *framework*, do sistema alvo e dos demais itens de configuração	Documentação das versões do *framework*, do sistema alvo e dos demais itens de configuração	Fase de Elaboração e Construção Atividades: Observar o domínio do sistema legado em relação ao do *framework*; Confrontar as características não funcionais do *Framework* x sistema legado; Elaborar o planejamento do projeto de reengenharia; elaborar o manual do usuário do sistema alvo; Testar o sistema alvo.
Gerenciar as mudanças	Gerenciar as mudanças no *framework* e no sistema alvo	Formulário de controle de mudanças do *framework* e do sistema alvo	Atividades: - Criar o sistema alvo no paradigma orientado a objetos e adaptar o sistema alvo.

Quadro 26 – Resumo da documentação do MR.GC-PARFAIT

Nas subseções a seguir mostra-se a documentação completa de apenas duas das atividades de GC que compõem o MR.GC-PARFAIT que são: **gerenciar as versões** e **gerenciar as mudanças**. A documentação das demais atividades desse modelo de referência encontra-se no Apêndice A.

5.3.1 Atividade: Gerenciar as Versões (Obrigatória)

Objetivo: Nesta atividade são gerenciadas as versões do *framework*, as versões do sistema alvo gerado a partir de tal *framework*, bem como as versões dos demais itens de configuração. Essa

atividade é conduzida pelos envolvidos no projeto de reengenharia. Especificamente para o caso das versões do *framework* são documentadas as particularidades apresentadas em cada versão, como por exemplo, plataforma, necessidades de *hardware* e *software*, linguagem de programação, entre outras.

Aplicação no PARFAIT: em todas as atividades das fases de Elaboração e Construção e em algumas da fase de Concepção e Transição: “Observar o domínio do sistema legado em relação ao do *framework*” (Fase de Concepção); “Confrontar as características não funcionais do *Framework* x sistema legado” (Fase de Concepção); “Elaborar o planejamento do projeto de reengenharia” (Fase de Concepção); “Elaborar o manual do usuário do sistema alvo” (Fase de Transição); “Testar o sistema alvo” (Fase de Transição).

Artefatos de entrada: Documentação das particularidades das versões do *framework* e dos sistemas gerados, bem como a dos demais itens de configuração.

Apoio computacional: Editor de texto e ferramenta de controle de versão.

Artefatos de saída: Documentação das versões do *framework*, do sistema alvo, dos demais itens de configuração (*baseline*).

Passos:

1. Documentar de forma detalhada as peculiaridades apresentadas em cada versão do *framework* e do sistema alvo gerado, a partir do mesmo e em cada versão dos demais itens de configuração;
2. Armazenar a versão do item de configuração no repositório da ferramenta de controle de versão.

Gabaritos: Nesta atividade foram desenvolvidos gabaritos para documentar as versões dos itens de configuração. Para documentar as versões do sistema alvo e do *framework* foram criados formulários específicos, apresentados no Quadro 27 e no Quadro 28, respectivamente. Para documentar as versões dos demais itens de configuração selecionados na atividade “Identificar os itens de configuração” é necessário utilizar o formulário apresentado no Quadro 29.

Salienta-se que serão documentadas apenas as versões dos itens de configuração que se tornarem baseline. Além de informações específicas sobre a versão, é necessário registrar também o número da iteração do projeto de reengenharia que o item de configuração foi criado e/ou modificado, bem como o nome da atividade em que isso ocorreu.

Documentação das Versões do Sistema Alvo	
Documentação da Versão <número da versão do sistema> **do Sistema Alvo** <nome do sistema alvo>	
Projeto	**Data de criação do formulário**
<nome do projeto>	<dd/mm/aa – hh:mm>
Versão do *framework* utilizada <número de versão do *framework* >	
Variabilidades funcionais do *framework* reutilizadas	**Variabilidades Não-Funcionais do *framework* reutilizadas**
<lista dos nomes das variabilidades funcionais>	< lista dos nomes das variabilidades não-funcionais>
Requisitos funcionais do sistema não reutilizados do *framework*	**Requisitos não-funcionais do sistema não reutilizados do *framework***
<lista dos nomes dos requisitos funcionais >	<lista dos nomes dos requisitos não-funcionais>

Quadro 27 - Gabarito da documentação das versões do sistema alvo

Documentação das Versões do *Framework*	
Documentação da Versão <número da versão do *framework* > **do *framework*** <nome do *framework* >	
Plataforma	**Data de criação do formulário**
<nome da plataforma utilizada>	<dd/mm/aa – hh:mm>
Domínio	**Linguagem de programação**
<nome do domínio coberto pelo *framework*>	<nome da linguagem de programação utilizada>
Recursos de *hardware* e *software* requeridos	
<por exemplo, ambiente de desenvolvimento, versão do compilador da linguagem de programação, sistema gerenciador de banco de dados (SGBD), versão do SGBD>	
Variabilidades funcionais	**Variabilidades não-funcionais**
<lista dos nomes das variabilidades funcionais>	<lista dos nomes das variabilidades não-funcionais>

Quadro 28 - Gabarito da documentação das versões do *framework*

Documentação das Versões dos demais Itens de Configuração				
Nome do Projeto		**Data de criação do formulário**		
<nome do projeto>		<dd/mm/aa – hh:mm>		
Data de criação ou atualização do item	**Nome dos itens de configuração**	**Versão**	**Número da iteração**	**Nome da atividade do PARFAIT**
<dd/mm/aa>	<nome do item de configuração>	<versão do item de configuração>	<número da iteração do sistema alvo>	<nome da atividade do PARFAIT que esta o item de configuração>

Quadro 29 - Gabarito da documentação das versões dos itens de configuração

5.3.2 Atividade Gerenciar as Mudanças (Obrigatória)

Objetivo: Esta atividade é empregada somente quando existir a necessidade de mudanças no sistema alvo ou no *framework*, que sejam tomadas as medidas necessárias para que ocorram as mudanças sem prejudicar a integridade das *baselines* desses itens de configuração.

Aplicação no PARFAIT: Nas atividades “Criar o sistema alvo no paradigma orientado a objetos” (Fase construção) e “Adaptar o sistema alvo” (Fase de construção).

Artefatos de entrada: Formulários de controle de mudanças do *framework* e do sistema alvo

Apoio computacional: Editor de texto e ferramenta de controle de versão.

Artefatos de saída: Formulários de controle de mudanças do *framework* e do sistema alvo devidamente preenchidos e aprovados/reprovados.

Passos:

1. Identificar as mudanças necessárias.
2. Preencher os formulários de controle de mudanças do *framework* (caso seja necessário) e do sistema alvo (caso seja necessário).
3. Aprovar ou reprovar as mudanças.
4. Realizar as mudanças e executar a atividade “Gerenciar as Versões”.

Gabarito: As solicitações de mudanças no sistema alvo, no *framework* e nos demais itens de configuração devem ser sempre documentadas por meio do formulário de controle de mudanças, apresentado respectivamente no Quadro 30, no Quadro 31 e no Quadro 32. As solicitações de mudanças no sistema alvo serão aprovadas ou não pelo gerente de configuração, de acordo com a complexidade das mesmas.

Formulário de Controle de Mudanças no Sistema Alvo	
Nº da solicitação: <nº da solicitação da mudança>	**Projeto:** <nome do projeto>
Data da solicitação: <dd/mm/aa>	**Tempo estimado para a execução da mudança** <hh:mm>
Sistema alvo: <nome do sistema alvo>	**Versão do sistema alvo:** <número correspondente da versão do sistema alvo que necessita sofrer mudanças>
Tipo de mudança: ☐ **Perfectiva** ☐ **Corretiva** ☐ **Adaptativa** ☐ **Preventiva**	
Mudança solicitada: <breve descrição da mudança>	
Artefatos afetados pela mudança solicitada:	
<lista dos nomes dos artefatos que serão afetados e que sofrerão mudanças>	
☐ **Solicitação de mudança APROVADA**	☐ **Solicitação de mudança NÃO APROVADA**
☐ **Implementar a mudança com apoio do *framework*:** - por meio de nova instanciação do *framework* (nesse caso o *framework* fornece a mudança desejada, parcialmente ou completamente)	☐ **Implementar a mudança manualmente:** - por meio de um ambiente de programação
	Retornar ao solicitante um parecer sobre o motivo da não aprovação da solicitação <motivo que levou a inviabilidade das mudanças>
Comentários Adicionais: < breve descrição, se necessário>	

Quadro 30 - Gabarito do formulário de controle de mudanças no sistema alvo

Formulário de Controle de Mudanças no *Framework*	
Nº da solicitação: <nº da solicitação da mudança>	**Projeto:** <nome do projeto>
Data da solicitação: <dd/mm/aa>	**Tempo estimado para a execução da mudança** <hh:mm>
***Framework*:** <nome do *framework*>	
Versão do *framework*: <número correspondente da versão do *framework*>	
Tipo de mudança: ☐ **Perfectiva** ☐ **Corretiva** ☐ **Adaptativa** ☐ **Preventiva**	
Mudança solicitada: <breve descrição da mudança>	
Artefatos afetados pela mudança solicitada:	
<lista dos nomes dos artefatos que serão afetados e que sofrerão mudanças>	
☐ **Solicitação de mudança APROVADA**	☐ **Solicitação de mudança NÃO APROVADA**
☐ **Implementar a mudança no *framework***	**Retornar ao solicitante um parecer sobre o motivo da não aprovação da solicitação** <motivo que levou a inviabilidade das mudanças>
Comentários Adicionais: < breve descrição, se necessário>	

Quadro 31 - Gabarito do formulário de controle de mudanças no *framework*

Formulário de Controle de Mudanças dos demais Itens de Configuração	
Nº da solicitação: <nº da solicitação da mudança>	**Projeto:** <nome do projeto>
Data da solicitação: <dd/mm/aa>	**Tempo estimado para a execução da mudança** <hh:mm>
Nome Artefato: <nome do artefato>	**Versão do Artefato:** <versão do artefato>
Tipo de mudança: ☐ **Perfectiva** ☐ **Corretiva** ☐ **Adaptativa** ☐ **Preventiva**	
Mudança solicitada: <breve descrição da mudança>	
Artefatos afetados pela mudança solicitada:	
<lista dos nomes dos artefatos que serão afetados e que sofrerão mudanças>	
☐ **Solicitação de mudança APROVADA**	☐ **Solicitação de mudança NÃO APROVADA**
Comentários Adicionais: < breve descrição, se necessário>	

Quadro 32 – Gabarito do formulário de controle de mudanças dos demais itens de configuração

Para analisar tal complexidade é preciso levar em consideração quais artefatos serão afetados pela mudança e se os mesmos necessitam de adaptações ou correções. No caso do PARFAIT com o apoio computacional do *framework* GREN, a implementação das adaptações (mudanças) realizadas manualmente no código fonte do sistema alvo é feita com o apoio da ferramenta *GREN–WizardVersionControl* a fim de que tais adaptações não sejam perdidas nas próximas instanciações do *framework.* As solicitações de mudanças no *framework* serão aprovadas ou não de acordo com uma análise, que verifica se uma mudança é pertinente ou não ao domínio do *framework.* Essa análise está fora do escopo deste trabalho e é feita com o apoio do Processo de Evolução de *Framework* (PREF) (CAGNIN *et al.*, 2004b). As mudanças no *framework* são efetuadas também com o apoio desse processo.

5.4 Considerações Finais

Neste capítulo foram selecionadas as atividades e os artefatos essenciais de GC para compor o modelo de referência definido neste trabalho, baseando-se em diversas normas e modelos de maturidade existentes com o emprego do método de mensuração GQM, a fim de selecionar quantitativamente as atividades e os artefatos, sem afetar a agilidade do processo de reengenharia

PARFAIT. Para apoiar esta seleção foi definida neste trabalho uma métrica, denominada “grau de importância”, para ser utilizada durante a aplicação do GQM.

Levando-se em consideração que o PARFAIT utiliza apoio computacional de *framework*, foi necessário se preocupar também tanto com a gerência de configuração do *framework* utilizado pelo processo quanto com a do sistema alvo.

Após a seleção das atividades e artefatos de GC do modelo de referência, observou-se a necessidade de identificar a aplicabilidade destas atividades nas atividades do PARFAIT. Para isso, foi feita uma análise objetiva da documentação das atividades do PARFAIT com base em alguns critérios estabelecidos.

Posteriormente, deu-se início à elaboração da documentação do modelo de referência de GC para o processo PARFAIT, que é o principal resultado desta dissertação. Tal documentação apresentou as atividades de GC, detalhando como e quando elas devem ser executadas em conjunto com as atividades do PARFAIT para controlar tanto as versões do *framework* quanto as do sistema alvo.

O maior desafio encontrado neste trabalho foi a elaboração da documentação do MR.GC-PARFAIT. Ressalta-se que não foi encontrado nenhum modelo de referência na literatura que descrevesse com detalhes os procedimentos adotados pela GC em processos de reengenharia e nem em processos ágeis de reengenharia.

Para analisar se o MR.GC-PARFAIT apresentado neste capítulo satisfaz o que foi proposto, será conduzido no próximo capítulo um estudo de caso quantitativo de um sistema legado em conjunto com o processo PARFAIT, a fim de obter resultados dos benefícios do seu uso.

6 ESTUDO DE CASO: PARFAIT E MR.GC-PARFAIT

6.1 Considerações iniciais

Neste capítulo abordar-se-á como foi conduzido o estudo de caso quantitativo para avaliar o MR.GC-PARFAIT. Na Seção 6.2 deste capítulo apresenta-se o planejamento do estudo de caso; na Seção 6.3 descreve-se como o estudo de caso foi executado, apresentando também a análise e a interpretação dos resultados obtidos, bem como uma discussão das dificuldades enfrentadas. Na Seção 6.4 encontram-se as considerações finais deste capítulo.

6.2 Planejamento do Estudo de Caso para avaliar o MR.GC-PARFAIT

Esse planejamento, definido e documentado de acordo com o formato proposto por Wholin *et al.* (2000), visa observar o uso do modelo de referência MR.GC-PARFAIT, na reengenharia de um sistema legado, utilizando o processo PARFAIT, a fim de analisar se tal modelo não afeta a agilidade do processo e se apóia efetivamente o controle de versões do *framework*, utilizado como apoio computacional, e do sistema alvo gerado por tal *framework*, mantendo a integridade dos mesmos.

A fase do planejamento tem como objetivo identificar o que deve ser observado do MR.GC-PARFAIT, estabelecer as hipóteses para analisar o modelo de referência proposto e a sua aplicabilidade na reengenharia ágil de *software*.

6.2.1 Definição

Objeto de estudo: Modelo de referência da atividade de GC do PARFAIT (MR.GC-PARFAIT).

Objetivo: Conduzir um estudo de caso no contexto de reengenharia ágil de *software* para analisar o emprego do MR.GC-PARFAIT sem afetar a agilidade do processo PARFAIT.

Foco qualitativo: Produção da documentação de GC necessária sem afetar a agilidade do PARFAIT e controle de versão do *framework* e das versões do sistema alvo visando manter a integridade dos mesmos.

Perspectiva: A perspectiva do estudo de caso baseia-se na utilização do MR.GC-PARFAIT.

Contexto: O estudo de caso será conduzido pela própria autora do modelo de referência. O material necessário para o estudo consiste em: documentação da abordagem ARTe (CAGNIN *et al.*, 2004) que apóia na atividade de teste, da LPA GRN (BRAGA, 2002a) que auxilia no entendimento do sistema legado e na documentação na análise orientada a objeto; dos requisitos de teste dos padrões da LPA GRN (CHAN; CAGNIN, 2004) que apóia o reúso na atividade de teste e na criação de casos de teste genéricos; do *framework* GREN (BRAGA, 2003) que apóia na geração automática do sistema alvo por meio da ferramenta de instanciação *GREN-Wizard* (BRAGA, 2002b; 2002c); da ferramenta de controle de versões *GREN-WizardVersion Control* (CAGNIN, 2004b) que apóia no controle das versões do sistema alvo; manuais do SGBD MySQL(2007)[12] que apóia no entendimento do banco de dados do *framework,* da linguagem de programação *Smalltalk* que possibilita as adaptações no sistema alvo, do modelo de referência da atividade de GC que apóia a atividade de GC e da ferramenta *Subversion* que apóia o controle de versão dos itens de configuração. O estudo de caso baseia-se na reengenharia de um sistema legado de biblioteca universitária, que controla o empréstimo e a devolução de livros, foi desenvolvido em *Clipper* e possui aproximadamente 6 KLOC.

[12] http://dev.mysql.com/doc/refman/4.1/pt/

6.2.2 Planejamento

Seleção do Contexto: Para a execução do estudo de caso foi elaborado um formulário para anotar o tempo gasto na execução de cada atividade do processo de reengenharia PARFAIT e do modelo de referência. A autora do modelo de referência possui algum conhecimento na utilização dos recursos e ferramentas necessários para a condução do estudo, contudo receberá um treinamento antes de executar essa tarefa. Esse estudo será realizado no contexto específico do domínio de Engenharia de *Software*.

Formulação das hipóteses:

- **Nulas:**

1. A documentação de GC elaborada pela aplicação do MR.GC-PARFAIT não afeta a agilidade do PARFAIT.
2. A integridade do *framework* e do sistema alvo é garantida pelo controle de versões do MR.GC-PARFAIT.

- **Alternativas:**

1. A documentação de GC elaborada pela aplicação do MR.GC-PARFAIT afeta a agilidade do PARFAIT.
2. A integridade do *framework* e do sistema alvo não é garantida pelo controle de versões do MR.GC-PARFAIT.

Seleção das variáveis:

Variáveis independentes (variáveis de entrada): o conhecimento empírico adquirido pela participante referente aos recursos necessários para a realização do estudo de caso (uso do *framework* GREN, da linguagem de padrões GRN, da ferramenta de instanciação GREN-*Wizard* e da ferramenta de controle de versões GREN-

*WizardVersionContro*l, do SGBD MySQL, da linguagem de programação *Smalltalk*, dos critérios de teste funcional Particionamento de Equivalência e Análise do Valor Limite, da abordagem ARTe, do MR.GC-PARFAIT, da ferramenta *Subversion* e do domínio de *softwares* de locação) ajuda no desempenho e no uso do processo e do modelo de referência.

Variáveis dependentes (variáveis de saída ou resposta): formulário de tempo gasto no estudo de caso em cada atividade do PARFAIT e em cada atividade de GC do MR.GC-PARFAIT.

Seleção dos indivíduos: A técnica de amostragem empregada neste estudo de caso é a por conveniência efetuada pela própria autora do modelo.

Projeto do estudo de caso: O estudo de caso é definido como um objeto único, por se tratar de apenas um único participante e de um único estudo de caso.

Instrumentação: Para realizar o estudo de caso necessita-se que a participante possua conhecimento: na documentação do modelo de referência MR.GC-PARFAIT; na documentação da LPA GRN e do *framework* GREN, manual da linguagem de programação *Smalltalk*; na documentação da abordagem ARTe; na documentação dos requisitos de teste da LPA GRN; nos critérios de teste funcionais Particionamento de Equivalência e Análise do Valor Limite, da ferramenta de instanciação *GREN-Wizard*, da ferramenta para controle de versões *GREN-WizardVersionControl* e da ferramenta *Subversion*.

Avaliação da validade: A avaliação dos resultados obtidos apresenta ameaças referentes: à validade de conclusão, pois depende do conhecimento empírico adquirido pelo engenheiro de *software* nos recursos necessários para realizar o estudo de caso; à validade interna que está relacionada à experiência da própria autora do MR.GC-PARFAIT em utilizá-lo com certa facilidade, aumentando o desempenho do estudo de caso; à validade de construção, sendo necessária a condução de outros estudos de casos com grau de complexidade maior e realizados por outros profissionais visando observar a dificuldade e benefícios alcançados; e à validade externa que apresenta-se em adaptar este modelo de referência para outros processos de reengenharia baseados em *framework*.

6.3 Operação

Execução: O estudo de caso foi realizado em duas etapas. Na primeira etapa estabeleceu-se a comparação do domínio do sistema legado com o do *framework* disponível, sendo constatada a viabilidade em realizar o projeto de reengenharia.

Na segunda etapa, o projeto de reengenharia foi dividido em três iterações. Na primeira iteração os casos de teste para apoiar a execução do sistema legado foram criados a partir da documentação disponível dos requisitos de teste dos padrões da GRN. Para reduzir ainda mais o tempo de VV&T com a utilização da abordagem ARTe, a criação dos casos de teste foi efetuada de maneira genérica, conforme descrito na Seção 6.3.1. Em seguida, foram utilizados os padrões 1 e 2 da LPA GRN (padrões “Identificar o Recurso” e “Quantificar o Recurso”, respectivamente) para documentar as funcionalidades do sistema alvo, priorizadas para a primeira iteração, que tratam dos seguintes cadastros: Livro, Editora, Assunto, Área, Autor e Exemplar.

Posteriormente, dando prosseguimento à execução das atividades do PARFAIT, o *framework* GREN foi instanciado e a primeira versão do sistema alvo foi gerada e submetida aos testes funcionais, criados anteriormente, e ao teste de aceitação em conjunto com o usuário do sistema legado. Foram efetuadas adaptações na interface GUI (*Graphical User Interface*) desta primeira versão do sistema alvo, relacionadas ao dimensionamento das telas e à alteração dos nomes de alguns campos a fim de adequá-los funcionalmente ao sistema legado. A ferramenta *GREN-WizardVersion Control* foi utilizada para armazenar as adaptações realizadas nesta primeira iteração.

As atividades de GC foram executadas em todas as iterações do processo PARFAIT de acordo com o modelo de referência MR.GC-PARFAIT, sendo o controle de versão apoiado pela ferramenta *Subversion*. Ressalta-se que, durante o uso do MR.GC-PARFAIT na primeira iteração, observou-se necessidade de acrescentar o diagrama de casos de uso para ser submetido à GC, pois não havia sido considerado na concepção do modelo de referência.

O estudo de caso foi conduzido sem horários contínuos. Na Tabela 1 apresenta-se a relação da seqüência das atividades executadas em cada iteração do projeto de reengenharia, a estimativa de tempo de cada atividade foi baseada em um estudo de caso de reengenharia já conduzido com o apoio da abordagem ARTe e sem o uso do MR.GC-PARFAIT (CAGNIN, 2005a), e o tempo realmente gasto em cada iteração neste estudo de caso. Como no estudo de caso relatado em Cagnin

(2005a), a atividade do PARFAIT que trata da criação do manual do usuário também não foi efetuada.

Tabela 1 - Tempo gasto em cada iteração das atividades do PARFAIT e do MR.GC-PARFAIT

Seqüência das atividades			Atividades	Tempo gasto em horas/minutos por cada atividade					
				1ª Iteração		2ª Iteração		3ª Iteração	
1ª	2ª	3ª		T. Estimado	T. Gasto	T. Estimado	T. Gasto	T. Estimado	T. Gasto
5	-	-	**Observar o domínio do sistema legado em relação ao do *framework***	02:00	**02:10**	01:00	-	01:00	-
6	-	-	Definir a ferramenta de apoio	00:20	**00:30**	-	-	-	-
7	-	-	Criar um repositório	00:20	**00:45**	-	-	-	-
8		-	Gerenciar as Versões	00:30	**00:05**	00:30	-	00:30	-
9	-	-	**Confrontar as características não funcionais do *framework* x sistema legado**	00:20	**00:20**	00:20	-	00:20	-
10	-	-	Gerenciar as Versões	00:30	**00:05**	00:30	-	00:30	-
11	-	-	**Elaborar o Planejamento do Projeto de reengenharia**	00:15	**00:25**	00:15	-	00:15	-
12	-	-	Gerenciar as Versões	00:30	**00:05**	00:30	-	00:30	-
3	1	3	**Desenvolver o diagrama de casos de uso e elaborar os casos de teste**	30:00	**25:20**	98:00	**33:45**	126:00	**10:25**
4	2	4	Gerenciar as Versões	00:30	**00:05**	00:30	**00:05**	00:30	**00:05**
1	3	5	**Desenvolver o diagrama de classes do sistema alvo**	08:00	**04:00**	06:00	**03:10**	4:00	**02:40**
2	4	6	Gerenciar as Versões	00:30	**00:05**	00:30	**00:05**	00:30	**00:05**
13	5	7	**Documentar as modificações realizadas no diagrama de classes**	01:20	**00:30**	01:00	**00:27**	01:00	**00:10**
14	6	8	Gerenciar as Versões	00:30	**00:05**	00:30	**00:05**	00:30	**00:05**
-	-	-1	**Documentar as regras de negócio do sistema**	-	**-**	-	**-**	1:30	**00:10**
-	-	2	Gerenciar as Versões	-	**-**	-	**-**	00:30	**00:10**
15	7	9	**Criar o sistema alvo no paradigma OO**	00:20	**00:30**	00:30	**00:20**	00:45	**00:20**
-	8	10	Gerenciar as mudanças	-	**-**	00:30	**00:10**	00:30	**00:10**
16	9	11	Gerenciar as Versões	00:30	**00:10**	00:30	**00:20**	00:30	**00:05**
18	10	12	**Executar os casos de teste no sistema alvo**	03:00	**02:00**	04:00	**01:40**	1:30	**00:30**
19	11	13	Gerenciar as Versões	00:30	**00:05**	00:30	**00:10**	00:30	**-**
17	12	14	**Adaptar o sistema alvo**	03:00	**02:00**	07:00	**03:55**	08:00	**01:00**
20	13	15	Gerenciar as mudanças	00:30	**00:10**	00:30	**00:20**	00:30	**00:10**
21	14	16	Gerenciar as Versões	00:30	**00:15**	00:30	**00:30**	00:30	**00:10**
22	15	17	**Testar o sistema alvo**	01:30	**00:45**	02:00	**00:55**	01:30	**01:00**
23	16	18	Gerenciar as Versões	00:30	**00:05**	00:30	**00:05**	00:30	**00:10**
TOTAL DE TEMPO				53:55	**40:20**	127:35	**46:07**	151:50	**17:15**
TOTAL DE TEMPO GASTO COM O MR.GC-PARFAIT				**-**	**02:30**	**-**	**01:45**	**-**	**1:10**

O tempo total gasto na primeira iteração foi de 40h20, das quais 02h30 foram gastas na execução das atividades de GC; 15h55 foram gastas na documentação do sistema alvo (atividades "Observar o domínio do sistema legado em relação ao do *framework"*, "Elaborar o planejamento do projeto de reengenharia", "Desenvolver o diagrama de casos de uso e elaborar os casos de teste"[13], "Desenvolver o diagrama de classes do sistema alvo", "Documentar as modificações realizadas no diagrama de classes" e "Adaptar o sistema alvo"); e 21h05 foram gastas nas atividades de VV&T (atividades "Desenvolver o diagrama de casos de uso e elaborar os casos de teste"[14], "Executar os casos de teste no sistema alvo" e "Testar o sistema alvo").

Na segunda iteração foi utilizado o padrão 4 da LPA GRN (padrão "Alocar Recurso"), documentando assim os cadastros de Aluno e de Curso, bem como o Empréstimo de Livros. Para implementar esses requisitos uma nova instanciação do *framework* GREN foi efetuada, obtendo-se uma outra versão do sistema alvo.

Nesta segunda iteração, durante a demonstração do sistema alvo para o usuário, foi solicitada a implementação de duas regras de negócio que não estavam presentes no sistema legado. A primeira trata do período de devolução do livro (sete dias para professores e três dias para os demais leitores) e a segunda trata da cobrança de taxa de multa de R$ 1,00 para cada dia de atraso na devolução do livro.

O tempo gasto na segunda iteração foi de 46h07 (Tabela 1), das quais 01h45 foram gastas na execução das atividades de GC; 11h17 foram despendidas na documentação do sistema alvo (atividades "Desenvolver o diagrama de casos de uso e elaborar os casos de teste", "Desenvolver o diagrama de classes do sistema alvo", "Documentar as modificações realizadas no diagrama de classes" e "Adaptar o sistema alvo"); e 32h35 nas atividades de VV&T (atividades "Desenvolver o diagrama de casos de uso e elaborar os casos de teste", "Executar os casos de teste no sistema alvo" e "Testar o sistema alvo").

Na terceira iteração foram criados os relatórios pertinentes ao sistema alvo, os quais foram herdados do *framework*, e foi implementada apenas a regra de negócio referente à taxa de multa, também herdada do *framework*. Entretanto, a outra regra não foi incorporada ao sistema alvo devido à falta de tempo. Ressalta-se que essa decisão tomada não prejudicará os resultados deste estudo de caso, pois, considerando o tempo gasto para implementar tal regra no outro estudo de caso (CAGNIN, 2005a), que foi de 05h40, seria gasto neste estudo de caso apenas 5,46% a mais do

[13] O tempo para a elaboração dos casos de teste não foi considerado.
[14] O tempo para desenvolver o diagrama de caso de uso não foi considerado.

tempo total gasto na reengenharia (103h42 + 05h40), como está apresentado no anterior estudo de caso, não prejudicando dessa forma os resultados obtidos. Salienta-se que tais resultados serão utilizados para analisar, especialmente, o impacto do uso do modelo proposto neste trabalho sem afetar a agilidade do PARFAIT. O tempo total gasto na terceira iteração foi de 17h15, sendo que 01h10 foi o tempo gasto com a aplicação do MR.GC-PARFAIT.

6.3.1 Criação dos Casos de Teste Genéricos

Durante a aplicação das diretrizes da abordagem ARTe para o reúso dos requisitos de teste disponíveis, observou-se a necessidade de aprimorá-la para proporcionar um melhor desempenho na sua execução, no intuito de reduzir ainda mais o tempo de VV&T. Para isso, foram criados casos de teste genéricos a partir dos requisitos de teste disponíveis nos padrões 1, 2 e 4 da LPA GRN, utilizados no estudo de caso, conforme apresentado na Tabela 2. No caso do atributo “descrição”, que consta no padrão 1, foram criados casos de teste genéricos para diferentes tamanhos de descrições.

Adicionalmente, foram elaboradas classes de equivalência para os tipos de dados especiais do GREN (lista a partir de uma classe[15], lista discreta[16] e lista multivalorada[17]) (BRAGA, 2002c). Exemplificando as classes de equivalência válidas criadas, tem-se que um atributo do tipo de dado especial lista multivalorada pode ter mais do que um valor não duplicado, conforme apresentado na Tabela 3. Requisitos de teste foram criados a partir das classes de equivalência elaboradas e estão disponíveis para serem reutilizados (Tabela 2).

[15] O valor do atributo do tipo lista a partir de uma classe é proveniente de um objeto de outra classe.
[16] O valor do atributo do tipo lista discreta é proveniente de uma lista de valores pré-definidos.
[17] Os valores do atributo do tipo lista multivalorada são provenientes de objetos de outra classe.

Tabela 2 – Casos de testes gerais

Id. Casos teste	Requisito de teste ARTe	Dados de entrada		Saída esperada
CT13	RT30	Descrição 60	AA	Sucesso: <= 60 caracteres
CT14	RT30/RT266/RT267	Descrição 40	AA	Sucesso:< = 40 caracteres
CT15	RT30	Descrição 30	AA	Sucesso: <= 30 caracteres
CT16	RT30	Descrição 25	AA	Sucesso: <= 25 caracteres
CT17	RT30	Descrição 20	AA	Sucesso: <= 20 caracteres
CT68	RT2007	Lista a partir de uma classe	Valor A	Sucesso na verificação: permite continuar com a ação
...	...	...	...	...
CT69	RT2008	Lista a partir de uma classe	Valor B	Erro: valor não cadastrado
CT70	RT2009	Lista a partir de uma classe	“Vazio”	Erro
CT71	RT2002	Lista Multivalorada	“Vazio”	Erro: não é permitido continuar com a ação
CT72	RT2000	Lista Multivalorada	Valor A	Sucesso na verificação: permite continuar com a ação
CT73	RT2001	Lista Multivalorada	Valor A + Valor B	Sucesso na verificação: permite continuar com a ação
CT76	RT2005	Lista Discreta	Valor A	Sucesso na verificação: permite continuar com a ação
CT77	RT2006	Lista Discreta	“Vazio”	Erro

A Tabela 4 contém: o tipo de dado especial; o número da classe de equivalência base para a criação do requisito de teste (Tabela 3); a especificação do requisito de teste; a condição verificada por esse requisito de teste; e o retorno esperado. Exemplificando, o requisito de teste 2001 refere-se à inserção e à alteração de atributos do tipo lista multivalorada e tem como especificação atribuir dois ou mais valores distintos ao atributo para que a inserção ou a alteração seja efetuada com sucesso.

Posteriormente, para reduzir o tempo futuro de criação de casos de teste, foram elaborados também casos de teste genéricos a partir dos requisitos de teste criados para os tipos especiais do GREN, conforme ilustrados na Tabela 4 (a partir da linha 6).

Tabela 3 - Classes de equivalência criadas para os tipos de dados especiais do GREN

Tipo de dado especial	Classes Válidas	Classes Inválidas
Lista Multivalorada	quantidade de valores >= 1 (1166) Valores não duplicados (1168)	quantidade de valores = 0 (1167) valores duplicados (1169)
Lista Discreta	quantidade de valor = 1 (1170)	quantidade de valor > 1 (1171) quantidade de valor = 0 (1172)
Lista a partir de uma Classe	quantidade de valor = 1 (1173) idCode cadastrado (1175)	quantidade de valor = 0 (1174) idCode não cadastrado (1176)

Tabela 4 – Requisitos de teste criados para os tipos de dados especiais do GREN

Cód. do Req. Teste	Tipo de dado especial	Classes de Equivalência Utilizadas	Especificação do Requisito de Teste	Condição Verificada	Retorno Esperado
RT 2000	Lista Multivalorada	1166, 1168	determinar 1 valor	Atributo = valor A	Sucesso na verificação, permitindo continuar com a ação
RT 2001	Lista Multivalorada	1169	determinar 2 ou mais valores diferentes	Atributo = valor A e valor B	Sucesso na verificação, permitindo continuar com a ação
RT 2002	Lista Multivalorada	1167	sem valores	"Vazio"	Erro: falha na verificação, não permitindo continuar com a ação
RT 2003	Lista Discreta	1170, 1171	determinar 1 valor	valor A	Sucesso na verificação, permitindo continuar com a ação
RT 2004	Lista Discreta	1172	"Vazio"	"Vazio"	Erro: falha na verificação, não permitindo continuar com a ação
RT 2005	Lista a partir de uma classe	1173, 1175	determinar 1 IdCode já cadastrado	Valor 1	Sucesso na verificação, permitindo continuar com a ação
RT 2006	Lista a partir de uma classe	1174	determinar 1 IdCode não cadastrado	Valor 3	Sucesso na verificação, permitindo continuar com a ação

6.3.2 Análise e Interpretação dos Resultados

O estudo da LPA GRN e a documentação do *framework* GREN auxiliaram na condução do estudo de caso, pois fornecem diversos exemplos práticos que facilitaram o uso dessas técnicas no projeto de reengenharia.

O tempo gasto no estudo de caso apresentado por Cagnin (2005a) que utilizou o PARFAIT sem o modelo de referência proposto neste trabalho foi de 328h40. Já no estudo de caso de reengenharia, conduzido neste trabalho utilizando o PARFAIT e o MR.GC-PARFAIT, o tempo gasto foi de 103h42, perfazendo uma redução de 68,5%.

Essa redução de tempo da reengenharia é justificada principalmente pela criação de casos de teste genéricos, considerando um aperfeiçoamento da abordagem ARTe, uma vez que houve redução de 71,5% de tempo nas atividades do PARFAIT relacionadas à VV&T, conforme apresentado na segunda linha da Tabela 5. O tempo total gasto com as atividades do MR.GC-PARFAIT foi de 05h25, representando apenas 5,07% do tempo total da reengenharia. O valor desse percentual é considerado baixo em relação aos benefícios obtidos com o uso do modelo de referência.

Tabela 5 - Comparação do tempo gasto no estudo de caso

Tempo	Estudo de caso sem o MR.GC-PARFAIT e sem os casos de teste genéricos	Estudo de caso com o MR.GC-PARFAIT e com os casos de teste genéricos
da reengenharia	328h40	103h42
	68.5%	
das atividades VV&T	267h30	76h20
	71.5%	
das atividades de GC	-	05h25

A partir dos dados coletados durante a condução do estudo de caso, foi possível responder as hipóteses nulas identificadas durante o planejamento do estudo de caso. No Quadro 33 apresenta-se cada uma das respostas e uma breve justificativa.

Respostas das hipóteses nulas
1. A documentação de GC elaborada pela aplicação do MR.GC-PARFAIT não afeta a agilidade do PARFAIT? **Resultado:** Sim. A documentação de GC elaborada não afeta a agilidade do PARFAIT, tanto do ponto de vista das práticas de modelagem ágil tratadas pelo PARFAIT quanto do tempo gasto para elaborá-la. A documentação do MR.GC-PARFAIT se apresentou simples, clara e de fácil aplicação. **Justificativa:** O tempo gasto com o modelo MR.GC-PARFAIT correspondeu a apenas 5,07% (05h25) de todo tempo gasto no projeto de reengenharia (103h42).
2. A integridade do *framework* e do sistema alvo é garantida pelo controle de versões do MR.GC-PARFAIT? **Resultado:** Sim. A integridade tanto do *framework* quanto do sistema alvo foi garantida com o emprego do controle de versões e de mudanças do MR.GC-PARFAIT. **Justificativa:** Por meio do controle de versão e de mudanças tanto do *framework* quanto do sistema alvo manteve-se a integridade dos mesmos com o emprego dos gabaritos pertencentes ao modelo.

Quadro 33 - Dados coletados durante o estudo de caso

6.3.3 Discussão

Além da análise quantitativa dos resultados obtidos com o estudo de caso conduzido apresentada na Tabela 1 e na Tabela 5, foram coletadas também informações relacionadas à experiência e ao uso das técnicas e ferramentas utilizadas durante o estudo de caso com a autora do modelo (participante do estudo de caso). Foram relatadas algumas dificuldades no aprendizado de LPA GRN, além da elaboração da documentação durante a execução de cada atividade do PARFAIT.

Observou-se, além disso, a necessidade de adaptar a abordagem ARTe para permitir também o reúso de casos de teste, pois o tempo gasto para a criação dos testes com o apoio dessa abordagem ainda era relativamente significativo (em torno de 68% do tempo total gasto na reengenharia) e o entendimento da sua documentação para a criação dos casos de teste era de certa forma difícil.

A criação de casos de teste genéricos possibilitou uma considerável redução de tempo, conforme relatado na seção anterior, e uma maior flexibilidade nos casos de teste desenvolvidos. Entre os pontos favoráveis destacam-se o emprego do *framework* GREN em conjunto com a GRN que permitiram a criação de versões do sistema alvo o mais rápido possível, o uso do PARFAIT e da abordagem ARTe que otimizaram, de maneira considerável, o projeto de reengenharia.

Deve ser levado em consideração que os *frozen spots* impostos pelo *framework* fizeram com que diversas informações pertencentes ao domínio, mas não ao sistema legado, tivessem que ser consideradas no sistema alvo. Destaca-se, ainda, que o MR.GC-PARFAIT apresentou-se eficaz no que foi proposto, uma vez que gasto-se pouco tempo para aplicá-lo quando comparado aos benefícios providos por ele, como é o caso do controle de mudanças e de versões tanto do *framework* quanto do sistema alvo. Para observar a eficiência do MR.GC-PARFAIT é necessário efetuar a condução de um outro estudo de caso com um participante que não tenha sido influenciado pela sua definição, o que ocorreu com o estudo de caso conduzido neste trabalho.

Ressalta-se, igualmente, que apesar da ferramenta de controle de versão *Subversion* não possuir uma opção específica para controlar as versões do *framework*, foi adaptada uma seqüência de pastas dentro do repositório que possibilitou tal procedimento. Exemplificando, na Figura 8 apresenta-se a criação das pastas para o armazenamento da versão 13 (**im 13**) do *framework* GREN, bem como das versões dos sistemas gerados, como é o caso do **BibNew**, que possui a base de dados, o código fonte, os diagramas e a documentação pertencente ao projeto de reengenharia.

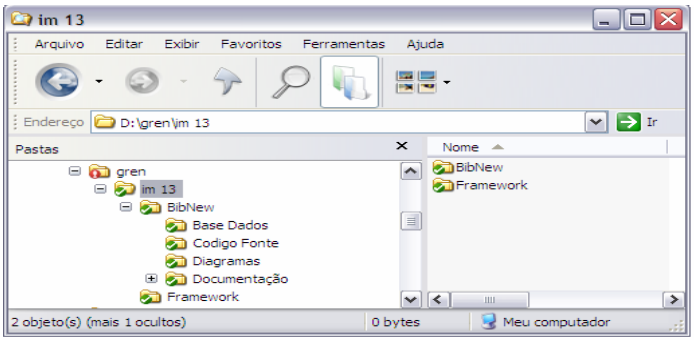

Figura 8 – Esquema do repositório

6.4 Considerações finais

Neste capítulo apresentou-se, com detalhes, a execução de um estudo de caso de reengenharia, planejado de acordo com Wholin *et al.* (2000), utilizando o processo PARFAIT e o modelo de referência MR.GC-PARFAIT. Alguns dos resultados quantitativos obtidos mostraram que o tempo de aplicabilidade do MR.GC-PARFAIT em conjunto com o PARFAIT é relativamente baixo, não interferindo significativamente no tempo total da reengenharia.

Além disso, é possível recuperar, em qualquer momento da reengenharia, uma determinada versão do sistema alvo e da versão correspondente do *framework* utilizada para apoiar na criação de tal sistema, bem como aceitar ou não solicitações de mudanças tanto no *framework* quanto no sistema alvo. Assim, é válido afirmar que o modelo proposto minimiza a problemática existente de GC durante a reengenharia baseada em *framework*.

Ressalta-se que a abordagem de teste ARTe foi aprimorada neste estudo de caso, possibilitando um melhor desempenho durante o seu uso, considerando a redução de tempo com o reúso dos casos de teste genéricos criados neste trabalho.

A conclusão, as limitações do trabalho e as sugestões de trabalhos futuros são apresentadas no próximo capítulo.

7 CONCLUSÕES

Neste trabalho foi definido o modelo de referência MR.GC-PARFAIT (Capítulo 5) com base nas diretrizes de GC de algumas normas e modelos de maturidade. Esse modelo tem como objetivo incorporar sistematicamente a GC no processo ágil de reengenharia PARFAIT, fornecendo documentação básica das atividades essenciais de GC que devem ser executadas em projetos de reengenharia com o apoio desse processo. Mais especificamente, o modelo de referência fornece: uma descrição dos passos de cada atividade; os artefatos que devem ser elaborados e o modo de elaboração (gabaritos); o apoio computacional que pode ser utilizado para apoiar cada atividade; e em quais atividades do PARFAIT cada atividade do modelo deve ser executada.

Para avaliar o MR.GC-PARFAIT, foi conduzido um estudo de caso planejado de reengenharia de um sistema legado (Capítulo 6), utilizando-se o processo PARFAIT e o *framework* GREN como apoio computacional. A partir da interpretação dos resultados obtidos nesse estudo de caso, observou-se que pouco tempo foi gasto com a aplicação do MR.GC-PARFAIT comparado aos benefícios obtidos com o mesmo, ou seja, apenas 5,07% do tempo total da reengenharia foi gasto para a execução das atividades de GC do modelo de referência.

Outra contribuição importante deste trabalho está relacionada à definição dos gabaritos que devem ser elaborados durante a execução das atividades de GC do modelo MR.GC-PARFAIT.

Adicionalmente, este trabalho contribuiu para a adaptação da abordagem de teste ARTe com relação à criação de casos de testes genéricos, baseados nos requisitos de teste disponíveis dos padrões da LPA GRN. A partir do reúso dos casos de teste genéricos durante o estudo de caso conduzido observou-se que houve redução significativa no tempo das atividades de VV&T do PARFAIT, ou seja, aproximadamente 71,5%.

Ressalta-se ainda que o MR.GC-PARFAIT trata de dez dos quatorze objetivos de GC considerados pelo modelo de referência MR-MPS. Tal modelo foi desenvolvido pela Softex cuja finalidade é adaptar normas e modelos de maturidade existentes na literatura à realidade das pequenas e médias empresas de desenvolvimento de *software* brasileiras. Os demais objetivos não são considerados pelo modelo MR.GC-PARFAIT, pois prejudicariam a agilidade do processo PARFAIT. Os objetivos do MR.MPS que são também tratados pelo MR.GC-PARFAIT encontram-se elencados a seguir: “selecionar os artefatos com o emprego de critérios documentados”; “identificar, armazenar, testar, revisar, alterar, definir, manter os artefatos e submetê-los à

baseline”; “instruir uma política de GC que controle os diferentes níveis, que possua um armazenamento dos artefatos para posteriormente serem recuperados, bem como uma solicitação de mudanças”; “preservar todos os artefatos de GC”; “autorizar a criação ou a liberação das *baselines* pela GC”; “controlar as modificações e as liberações dos artefatos”; “disponibilizar as modificações e liberações dos artefatos”; “adicionar os artefatos no repositório, já aprovados e devidamente controlados”; “assegurar a consistência e a completeza dos artefatos”; e “controlar o armazenamento, o manuseio e a liberação dos artefatos”.

7.1 Limitações

Uma das limitações deste trabalho refere-se ao fato de que o modelo MR.GC-PARFAIT é específico para o processo PARFAIT. Mas isso não impede que tal modelo seja adaptado a outros processos ou generalizado para qualquer processo de reengenharia baseado em *framework*.

Com relação às hipóteses nulas levantadas no planejamento do estudo de caso conduzido, as mesmas avaliam apenas alguns aspectos do modelo MR.GC-PARFAIT, como é o caso do controle de versão do *framework* e do sistema alvo, da permanência da agilidade do PARFAIT, bem como da integridade dos artefatos produzidos. Entretanto, é interessante observar outros aspectos do modelo de referência a fim de confirmar a sua viabilidade.

Ressalta-se que o estudo de caso foi conduzido em ambiente acadêmico, pela própria autora do modelo e utilizou um sistema legado de pequeno porte. Tudo isso pode ter influenciado nos resultados obtidos. Assim, sugere-se a condução de outros estudos de caso com outros participantes e em ambiente industrial.

7.2 Sugestões de Trabalhos Futuros

Para dar continuidade ao trabalho realizado, são sugeridos os seguintes trabalhos futuros:

- Conduzir um estudo de caso planejado de reengenharia objetivando analisar a eficiência e eficácia do MR.GC-PARFAIT em um ambiente industrial.

- Conduzir estudos de caso planejados de reengenharia de sistemas legados de pequeno e médio porte para verificar se os resultados observados no estudo de caso conduzido neste trabalho relacionados ao MR.GC-PARFAIT e a criação dos casos de teste genéricos são mantidos.
- Adaptar o MR.GC-PARFAIT para atender a outros processos baseados em *framework*, tanto no contexto de desenvolvimento quanto no de reengenharia.
- Definir um meta-modelo para armazenar as informações relevantes do controle de GC do modelo de referência proposto para apoiar o desenvolvimento de uma ferramenta específica de automatização do uso de tal modelo; Desenvolver uma ferramenta que apóie o controle de versão e de mudanças, tanto do *framework* quanto das versões geradas do sistema alvo.

REFERÊNCIAS

AMBLER, S.W. **Modelagem Ágil: Práticas eficazes para a Programação eXtrema e o Processo Unificado.** Porto Alegre: Bookman, 2004.

ANSI/IEEE Std-828, 1998, ***IEEE Standard for Software Configuration Management Plans***.

APPLETON, B. ***Patterns and software: Essential concepts and terminology***. 1997. Disponível em: <http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html>. Acesso em: Janeiro/2006

ARIMOTO, M. M.; CAMARGO, V.V.; CAGNIN, M. I. Uma Ferramenta de Controle de Versões de *Frameworks.* In: CLEI - LATIN-AMERICAN CONFERENCE OF INFORMATICS. **Proceedings …** São José, Costa Rica, 2007. [aceito para publicação].

ASKLUND, U.; MAGNUSSON, B. *Fine Grained Version Control of Configuration in COOP/Orm*. In: SCM - SYMPOSIUM ON CONFIGURATION MANAGEMENT. **Proceedings…** Disponível em: <http://citeseer.nj.nec.com/magnusson 96fine.html>. Acesso em: 10/02/2006.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. *Goal Question Metric paradigm.* In: ENCYCLOPEDIA of Software Engineering. John Wiley & Sons, 1994.

BERSOFF, E. H. ***Elements of Software Configuration Management***. IEEE Trans SE, V10, N1, 1984.

BRAGA, R. T. V; GERMANO, F. S. R.; MASIERO, C. M. **GRN: Uma Linguagem de Padrões para Gestão de Recursos de Negócio,** documentação da LPA GRN, ICMC - Universidade de São Paulo, 2002a.

BRAGA, R. T. V. **GREN-WIZARD User Guide,** versão 1.0, documentação do *Framework* GREN, 2002b

BRAGA, R. T. V. **Manual de Instanciação do *Framework* GREN**, versão 2.0, ferramenta de instanciação GREN-*Wizard*, 2002c

BRAGA, R. T. V.; MASIERO, P. C. ***GREN-Wizard: A Tool to Instantiate the GREN Framework.*** In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE *SOFTWARE*, XVI. Gramado-RS. Sessão de Ferramentas do SBES - Caderno de Ferramentas - **Anais do SBES**. Gramado-RS, p. 408-413, 2002d.

BRAGA, T. V. **Um Processo para Construção e Instanciação de Frameworks baseados em uma linguagem de Padrões para um Domínio Específico**. Tese (Doutorado em Computação) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, 2003.

BOOCH, G., RUMBAUGH, J., JACOBSON, I. **UML: Guia do Usuário, O mais avançado tutorial sobre *Unified Modeling Language***. Rio de Janeiro: Campus, 2000.

BORLAND, I. N. C. ***Borland Delphi for Windows 95/NT. Version 5.0: Object pascal language guide***. Scotts Valley, CA: Borland, 2000.

BRUCKER, A. D.; RITTINGER, F.; WOLFF, B. ***The* CVS-*Server Case Study***, FM-TOOLS, Augsburg, 2002. Disponível em: <http://www.brucker.ch/bibliography/download/2002/fmtools_CVS_02.pdf>. Acesso em: 20/02/2006.

CAGNIN, M. I. **PARFAIT: uma contribuição para a reengenharia de *software* baseada em linguagem de padrões e *Framework***. Tese (Doutorado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2005a.

CAGNIN, M. I.; MALDONADO, J. C.; MASIERO, P. C.; PENTEADO, R. D.; BRAGA, R. T. ***An Evolution Process for Application Framework***. In: WMSWM - WORKSHOP DE MANUTENÇÃO MODERNA DE *SOFTWARE*, I. Em conjunto com o XVIII Simpósio Brasileiro de Engenharia de *Software*. **Anais...** Brasília, DF, CD-ROM, 8 p. 2004a.

CAGNIN, M. I.; MALDONADO, J. C.; CHAN, A.; PENTEADO, R. D.; GERMANO, F. S. **Reúso na Atividade de Teste para Reduzir Custo e Esforço de VV&T no Desenvolvimento e na Reengenharia de *Software***. In: SBES - SIMPÓSIO BRASILEIRO DE ENGENHARIA DE *SOFTWARE*, XVIII. **Anais...** Brasília, DF, p. 71-85, 2004b.

CAGNIN, M. I.; MALDONADO, J. C.; GERMANO, F. S.; PENTEADO, R. D. **PARFAIT: *Towards a Framework-based agile reengineering process.*** In: ADC'2003 - AGILE DEVELOPMENT CONFERENCE. **Proceedings…** Salt Lake City, UTHA, USA: IEEE, p. 22-31, 2003a.

CAGNIN, M. I.; MALDONADO, J. C.; GERMANO, F. S.; CHAN, A.; PENTEADO, R. D. **Um Estudo de Caso de Reengenharia usando o Processo PARFAIT.** In: SDMS'2003 - SIMPÓSIO

DE DESENVOLVIMENTO E MANUTENÇÃO DE *SOFTWARE* DA MARINHA. **Anais...** Rio de Janeiro, RF, CD-ROM, 2003b.

CAGNIN, M. I.; MALDONADO, J. C.; GERMANO, F. S.; PENTEADO, R. D.; BRAGA, R. T. **GREN-WizardVersionControl: Uma Ferramenta de Apoio ao Controle de Versão das Aplicações Criadas pelo *Framework* GREN.** In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE *SOFTWARE*, XVIII. Sessão de Ferramentas'2004. **Anais...** Brasília, DF, CD-ROM, p. 74-78, 2004c.

CAGNIN, M. I.; MALDONADO, J. C. **Documentação do PARFAIT: Um Processo Ágil de Reengenharia baseado em *Framework***. Documentação do PARFAIT do Instituto de Ciências Matemáticas e de Computação – ICMC/USP, São Carlos, 2005b.

CHAN, A.; CAGNIN, M. I. **Documentação dos Requisitos de Teste dos Padrões da Linguagem de Padrão de Análise GRN**. Documento de trabalho, ICMC-USP - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2004.

CHIKOFSKY, E. J.; CROSS II, J. H. ***Reverse engineering and design recovery: A taxonomy.*** IEEE Software, 7(1):13–17, 1990.

COLLINS-SUSSMAN, B.; FITZPATRICK, B. W.; PILATO, C. M. ***Version Control with Subversion, para Subversion 1.2.*** 2006. Disponível em: <http://svnbook.red-bean.com/nightly/pt_BR/svn-book.html>. Acesso em: 15/02/2006.

CUNHA, J. R. D.; PRADO, A. F.; SANTOS, A. C.; SOUZA NETO, R. M. Uma abordagem para o Processo de Gerenciamento de Configuração de *Software*. **RESI – Revista eletrônica de Sistema de Informação,** Universidade Federal de São Carlos, São Carlos-SP, 2004. Disponível em: <http://www.presidentekennedy.br/resi/edicao04/artigo05.pdf>. Acesso em: 20/12/2005.

ESPINDOLA, R. S.; LOPES, L.; PRIKLADNICKI, R.; AUDY, J. L. N. **Uma Abordagem baseada em Gestão do Conhecimento para Gerência de Requisitos em Desenvolvimento Distribuído de *Software.*** In: WER05 - WORKSHOP EM ENGENHARIA DE REQUISITOS. **Anais…** Portugal, p. 87-99, 2005.

FAYAD, M. E..; SCHMIDT, D.; JOHNSON, R. **Building Application *Framework*: Object-Oriented Foundations of *Framework* Design.** First ed. John Wiley & Sons, September 1999.

FERREIRA, M. A. O.; CAGNIN, M. I. **Proposta de um Modelo de Referência de Gerência de Configuração para um Processo de Reengenharia baseado em *Framework***. In: SBSI – SIMPÓSIO BRASILEIRO DE SISTEMA DE INFORMAÇÃO. **Anais...** Curitiba, CD-ROM, 2006.

FERREIRA, M. A. O.; CAGNIN, M. I. **Modelo de Referência de Gerência de Configuração para um Processo Ágil de Reengenharia baseado em *Framework*.** In: SBQS - SIMPÓSIO BRASILEIRO DE QUALIDADE DE *SOFTWARE*. **Anais...** Porto de Galinhas, p. 157-169, 2007.

FIORINI, S. T. **Arquitetura para a Reutilização de Processo de *Software***. Tese (Doutorado em Informática) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2001. Disponível em: <htttp://www-di.inf.puc-rio.br/~julio/tese-soeli.pdf>. Acesso em: 10/02/2006.

FIORINI, S. T.; LEITE, J. C. S. P. **Organizando Processos de Requisitos.** In: WER98 – WORKSHOP EM ENGENHARIA DE REQUISITOS. **Anais...** Maringá, p. 1-8, 1998.

FONTANETTE, V; GARCIA, V. C.; BOSSONARO, A. A.; PEREZ, A. B.; PRADO, A. F. **Reengenharia de *Software* usando Transformações (RST).** In: SECOND IBERO-AMERICAN SYMPOSIUM ON SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING. **Anais…** Salvador, 2002a.

FONTANETTE, V; GARCIA, V. C.; BOSSONARO, A. A.; PEREZ, A. B.; PRADO, A. F. **Reengenharia de Sistemas Legados Baseada em Componentes usando Transformações**, In: WORKSHOP CHILENO DE INGENIERÍA DE SOFTWARE, II. **Anais...** Chile, 2002b.

FONTANETTE, V; GARCIA, V.C.; BOSSONARO, A. A.; PEREZ, A.B.; PRADO, A. F. **Reprojeto de Sistemas Legados Baseada em Componentes de *Software***, In: CLEI - LATIN-AMERICAN CONFERENCE OF INFORMATICS. **Proceedings...** Uruguai, 2002c.

GARCIA, V. C.; FONTANETTE, V.; BOSSONARO, A.; PEREZ, A. B.; PRADO, A. F. **DDE – Draco Domain Editor.** In: SBES - SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, XVI. **Anais...** Gramado, p. 378-383, 2002.

HAZAN, C.; LEITE, J. C. S. P**. Indicadores para a Gerência de Requisitos,** In: WER03 – WORKSHOP EM ENGENHARIA DE REQUISITOS. **Anais...** Piracicaba, p. 285-301, 2003.

HUMPHREY, W. S.; KITSON, D. H.; GALE, J. *A **comparison of U.S. and Japanese software process maturity.*** In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 13. **Proceedings…** Austin, Texas, United States, 1991.

IEEE, Std 610.12 - ***IEEE Standard Glossary of Software Engineering Terminology***. Institute of Electrical and Electronics Engineers, 1990.

ISO/IEC 10007 – Sistemas de Gestão da Qualidade – Diretrizes para a Gestão de Configuração. ABNT - Associação Brasileira de Normas Técnicas. Rio de Janeiro: ABNT, 2005.
ISO/IEC 12207 - Tecnologia de Informação - Processos de ciclo de vida de *software*. ABNT - Associação Brasileira de Normas Técnicas. Rio de Janeiro: ABNT, 1998.

ISO/IEC TR 15504 – 5: Information Technology – *Software* process assessment – Part 5: An assessment model and indicator guidance. ISO/IEC - *International Standard Organization and International Electritechnical Commission*, 1999.

ISO/IEC 9001 – Sistema de Gestão da Qualidade – Requisitos. ABNT - Associação Brasileira de Normas Técnicas. Rio de Janeiro: ABNT, 2000.

ISO/IEC 9004 – Sistema de Gestão da Qualidade – Diretrizes para a melhoria de desempenho, ABNT - Associação Brasileira de Normas Técnicas. Rio de Janeiro: ABNT, 2000.

JOMORI, S. M.; VOLPE, R. L. D.; ZABEU, A. C. P. **Qualidade de *software*.** Revista TECHOJE, (IETEC - Instituto de Educação Tecnológica), ano XIII. Disponível em: <http://www.ietec.com.br/ietec/techoje/techoje/tecnologiadainformacao/2004/07/01/2004_07_01_0001.2xt/-template_interna>, 2004. Acesso em: 10/12/2005.

KRAUSE, R. *An Introduction to the arch Version Control System*. **Linux Journal**, 2002. Disponível em: <http://www.linuxjournal.com/article/5928>. Acesso em: 10/02/2006.

KRUCHTEN, P. ***The Rational Unified Process and Introduction Second Edition***. NJ: Addison Wesley Longman, 2000.

LEITE, J. C. S. P.; ROSSI, G; BALAGUER, F.; MAIORANA, V.; KAPLAN, G., HADAD, G.; OLIVEROS, A. ***Enhancing a Requirements Baseline with Scenarios, Requirements Engineering***. London: Springer-Verlag, 1997. Disponível em: < www-di.inf.puc-rio.br/~julio/Slctpub/baseline.pdf >. Acesso em: 20/12/2005.

LEMOS, G. S.; RECCHIA, E. L.; PENTEADO, R. D.; BRAGA, R. T. V**. Padrões de Reengenharia Auxiliados por Diretrizes de Qualidade de *Software*. I**n: SUGARLOAF PLOP - THE SECOND LATIN AMERICAN CONFERENCE ON PATTERN LANGUAGES OF PROGRAMMING. **Anais…** Pernambuco, 25 p., 2003.

LIMA, P. S. B.; **Proposta de um modelo simplificado de aquisição de *software* para pequenas empresas.** Dissertação (Mestrado em Engenharia de Eletricidade) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2004. Disponível em: <http://www.pcs.usp.br/~lucia/teses/PauloLima.pdf>. Acesso em: 20/12/2005.

LOPES, L. G.; MURTA, L.; WERNER, C. **Controle de mudanças em *Software* no Desenvolvimento Baseado em Componentes.** In: WMSWM 2005 - WORKSHOP DE MANUTENÇÃO DE SOFTWARE MODERNA. **Anais...** Manaus, 16 p., 2005.

MARTIMIANO, L. A. F. **Estudo de Técnicas de Teste de Regressão Baseado em Mutação Seletiva.** Dissertação (Mestrado em Ciência de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, São Carlos, 1999.

MILLS, H.D., O'NEILL, D. et al. **The Management of Software Engineering**. IBM Sys. J., 24(2), p. 414–477, 1980.

MYERS, G. J. ***The art of software testing***. Second Edition. Wiley, 2004.

MySQL. **Manual de Referência do MySQL 4.1**. 2007. Disponível em: <http://dev.mysql.com/doc/refman/4.1/pt/>. Acesso em: 10/02/2007.

NATALI, A. C. C.; FALBO, R. de A. Gerência de Conhecimento de Qualidade de *Software***. Resi – Revista Eletrônica de Sistema de Informação**, Universidade Federal do Espírito Santo, 2004. Disponível em: <http://www.inf.ufes.br/~falbo/download/pub/ Jiisic2002paper33.pdf>. Acesso em: 10/12/2005.

OLIVEIRA, A. A. A. C. P.; PRIMO, F. F., CRUZ, J. L.; MARTINO, W. R. **Gerência de Configuração de *Software* – Evolução de *Software* sob Controle.** Instituto Nacional de Tecnologia de Informação – ITI, órgão do Ministério da Ciência e Tecnologia, 2001.

PAULK, M. C. ***Capability maturity model for software – version 11 Technical Report 93-TR-24***, CMU/SEI, 1993.

PENTEADO, R. A. D. **Um Método para a Engenharia Reversa Orienta a Objetos.** Tese (Doutorado em Física Computacional) - Instituto de Física, Universidade de São Paulo, São Carlos, 1996.

PETERS, J. F.; PEDRYCZ, W. **Engenharia de *Software***. 3. reimp. Rio de Janeiro: Elsevier, 2001.

PMI - *Project Management Institute*. **Sobre PMI – História.** Disponível: <http://www.pmisc.org.br/portugues/introducao.htm>. Acesso em: em 10/03/2006.

PMI - *Project Management Institute*. **Um Guia do Conjunto de Conhecimento em Gerenciamento de Projetos (Guia PMBOK®).** 3. ed. Four Campus Boulevard, Newtown Square, PA 19073-3299, EUA, 2004.

PRESSMAN, R. S. **Engenharia de *Software*.** 5. ed. Rio de Janeiro: McGraw-Hill, 2006.

RECCHIA, E. L.; PENTEADO, R. **FaPRE/OO: Uma Família de Padrões para Reengenharia Orientada a Objetos de Sistemas Legados Procedimentais**. In: SUGARLOAF PLOP - THE SECOND LATIN AMERICAN CONFERENCE ON PATTERN LANGUAGES OF PROGRAMMING. **Anais…** Rio de Janeiro, 2002.

ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K. C. **Qualidade de *Software***. São Paulo: Prentice Hall, 2001.

SANT´ANNA, N.; CEREJA JUNIOR, M. G. BORREGO FILHO, L. F.; LUQUE, L.; CASILLO, B. H. **e-WebProject – Um ambiente integrado para apoio ao desenvolvimento e gestão de projetos de *Software*.** In: CONGRESSO INTERNACIONAL DE TECNOLOGIA DE SOFTWARE, XII. **Anais...** Curitiba, p. 1-11, 2002.

SEI - *Software* Engineering Institute, ***Capability Maturity Model® Integration* (CMMI^{SM}),** Version 1.1, 2001. Disponível em: <http://www.sei.cmu.edu/CMMI/ models/model-components-word.html>. Acesso em: 10/02/2006.

SILVA, F. A. **RCS e CVS: Ferramentas para Versionamento de Arquivos.** Instituto de Informática da Universidade Federal do Rio Grande do Sul, RS, 2002. Disponível em: <http://www.inf.ufrgs.br/~clesio/cmp151/cmp15120021/artigo_fabricio.pdf>. Acesso em: 20/02/06.

SNEED, H. M., NYARY, E. ***Extracting object-oriented specification from procedurally oriented programs*.** In: WCRE 1995, WORKING CONFERENCE ON REVERSE ENGINEERING, 2. **Proceedings...** Toronto-Canadá, 1995.

SOFTEX, **MPS.BR – Melhoria de Processo do *Software* Brasileiro (Guia Geral – Versão 1.0)**, 2005. Disponível em: <http://www.inf.ufes.br/~falbo/download/aulas/es-m/2005-2/Texto14.pdf>. Acesso em: 10/02/2006.

SOLINGEN, R. V., BERGHOUT, E. ***The Goal Question Metric Method – A practical Guide for Quality Improvement of Software Development***. Londres: McGraw-Hill International, 1999.

SOMMERVILLE, I. **Engenharia de *Software.*** 6. ed. São Paulo: Addison Wesley, 2003.

TICHY, W. F. **RCS - *A System for Version Control***. The Revision Control System, Edition 1.1 RCS Version 5.7, 1997. Disponível em: <http://cs1.bradley.edu/~alp/ teaching/RCS.pdf>. Acesso em: 10/02/2006.

TOSUN, F. ***Evaluating Software Configuration Management tools for Opticon Sensors Europe B.V.*** Masters Thesis (Software Engineering from the UVA) - Universiteit Van Amsterdam, Amasterdam, 2004.

TSUKUMO, A.; RÊGO, C. M.; SALVIANO, C. F.; AZEVEDO, G.F.; MENEGHETTI, L. K.; COSTA, M. C. C.; CARVALHO, M. B.; COLOMBO, R. M. T. **Qualidade de *Software*: Visões de Produto e Processo de *Software***. In: SBC - ESCOLA DE INFORMÁTICA DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO REGIONAL DE SÃO PAULO, II. **Anais...** Piracicaba, p. 173-189, 1997.

WHOLIN, C.; RUNESON, P.; HOST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. ***Experimentation in Software Engineering: an Introduction*.** Kluwer Academic Publishers, 2000.

XIMBIOT, *Version Management with* CVS, 2005. Disponível em: <http://ximbiot.com/CVS/manual/>. Acesso em: 27/02/2006.

APÊNDICE A

Neste apêndice será apresentado o propósito da elaboração deste documento, o púbico alvo que irá utilizá-lo, o escopo, a padronização e a documentação completa do modelo de referência MR.GC-PARFAIT.

Propósito

Este documento fornece uma visão dos procedimentos necessários para implantar a GC durante o uso do processo ágil de reengenharia PARFAIT, visando controlar as mudanças que ocorrerão tanto no *framework* quanto nas aplicações derivadas do mesmo.

Público Alvo

Os interessados diretamente neste documento são todas as pessoas envolvidas no projeto de reengenharia utilizando o processo PARFAIT.

Escopo

Neste documento está detalhado todo o procedimento adotado para estabelecer as atividades de GC no PARFAIT.

Padronização

Todo artefato elaborado durante o projeto de reengenharia deve ser nomeado da seguinte maneira:

<NOME_PROJETO>.<nome_artefato>.<data_da_criação>.[<descrição>].

Sendo:

<NOME_PROJETO>: descrição do nome do projeto em letras maiúsculas;

<nome_artefato>: descrição do nome do artefato;

<data_da_criação>: registro da criação do artefato no formato DDMMAA, HH:MM;

<descrição>: breve descrição opcional do artefato, devendo conter caracteres maiúsculos e não acentuados (A-Z), números (0-9), hífen (-) e sublinhado (_).

Notas: a quantidade de caracteres utilizada não deve ultrapassar 50 unidades.

Documentação completa do MR.GC-PARFAIT

O formato da documentação do MR.GC-PARFAIT (FERREIRA; CAGNIN, 2007) definido neste trabalho segue aquele utilizado no RUP: atividade, objetivo, artefatos de entrada e de saída, apoio computacional, passos e gabaritos. Além disso, um item foi adicionado na documentação, que é "aplicação no PARFAIT", ou seja, em que momento do processo PARFAIT (atividade) determinada atividade de GC do modelo de referência deve ser executada.

Sugere-se que o responsável pela reengenharia determine um membro da equipe como gerente de configuração, que deve ter conhecimento e habilidade de executar as atividades impostas pelo MR.GC-PARFAIT. Nas subseções a seguir são apresentadas as atividades do modelo MR.GC-PARFAIT. A execução das atividades "Identificar os itens de configuração" e "Definir a ferramenta de apoio" não são obrigatórias. Pois, para a atividade "Identificar os itens de configuração", MR.GC-PARFAIT sugere uma listagem dos itens de configuração do PARFAIT que devem ser submetidos à GC. A não obrigatoriedade da execução da atividade "Definir a ferramenta de apoio" está relacionada à familiaridade do responsável com determinada ferramenta de controle de versão.

Nas subseções a seguir apresenta-se a documentação completa das atividades do modelo de referência proposto.

Atividade: Identificar os itens de configuração (Não Obrigatória)

Objetivo: Nesta atividade é necessário selecionar apenas os artefatos do PARFAIT que devem ser submetidos ao gerenciamento de configuração. Para facilitar o uso do modelo de referência criado e a não execução desta atividade, é sugerida uma listagem dos artefatos que devem ser gerenciados.
Aplicação no PARFAIT: Na atividade "Observar o domínio do sistema legado em relação ao do *framework*" (Fase de Concepção).
Artefatos de entrada: Documentação do PARFAIT.
Apoio computacional: Editor de texto.
Artefatos de saída: Listagem dos itens de configuração do PARFAIT.

Passos:

1. Listar todos os artefatos do PARFAIT e elencá-los de acordo com o seu grau de importância (aplicar métrica "Grau de Importância") para documentar e conduzir a reengenharia.
2. Analisar a lista de artefatos priorizados, de acordo com o grau de importância.
3. Listar os artefatos que devem ser submetidos ao gerenciamento de configuração.

Gabarito: Para selecionar quais artefatos produzidos pelo PARFAIT devem ser considerados no gerenciamento de configuração com o apoio do modelo de referência proposto utilizou-se o GQM. A abrangência da métrica "Grau de Importância" foi reduzida para simplificar a seleção: **grau 3** – extremamente importante (100% a 67%), **grau 2** – importância moderada (66% a 34%) e **grau 1** – pouco importante (33% a 0%).

O estabelecimento do grau de importância para cada artefato do PARFAIT é resultante da resposta da seguinte questão formulada com a aplicação do GQM: a agilidade do PARFAIT não é prejudicada pelo gerenciamento do artefato X[18] Para apoiar a definição do percentual, tem-se a métrica intermediária: percentual de necessidade da utilização do artefato X para criar o sistema alvo a partir da instanciação do *framework*. O resultado obtido está apresentado no Quadro 34 A1, que mostra a listagem dos artefatos elaborados durante o uso do PARFAIT com grau de importância igual a 3, ou seja, artefatos extremamente importantes e que devem ser gerenciados pelo modelo proposto.

Artefatos desenvolvidos
Documento de aceitação do *framework*
Documento de requisitos
Planejamento do projeto de reengenharia
Documentação dos casos de teste
Diagrama de Classes do sistema
Digrama de Caso de Uso
Quadro das adaptações no diagrama de classes não cobertas pela linguagem de padrões
Quadro dos requisitos da linguagem de padrões que não se adaptam completamente aos do sistema legado
Documentação de regra do negócio
Sistema alvo no paradigma orientado a objeto
Documentação de casos de teste adequados ao sistema alvo
Formulário de planejamento das adaptações
Manual do usuário
Sistema alvo no paradigma orientado a objetos liberado para uso

Quadro 34 - Listagem dos itens de configuração do PARFAIT

[18] Cada artefato produzido pelo PARFAIT.

Atividade: Definir a Ferramenta de Apoio (Não obrigatória)

Objetivo: Nesta atividade define-se a ferramenta que melhor apóia a GC, devendo ser adequada às necessidades impostas tanto para controlar as versões do *framework* quanto para controlar as versões dos sistemas gerados.

Aplicação no PARFAIT: Na atividade "Observar o domínio do sistema legado em relação ao do *framework*" (Fase de Concepção).

Artefatos de entrada: Documentação das características de ferramentas de GC.

Apoio computacional: Editor de texto.

Artefatos de saída: Ferramenta que melhor se adequou às necessidades impostas.

Passos:

1. Observar e analisar, dentre as ferramentas encontradas, os pontos positivos e negativos de cada uma, levando em consideração o controle de versão tanto do *framework* quanto do sistema alvo.
2. Escolher a ferramenta de GC mais apropriada.

Atividade: Criar um repositório de Configuração (Obrigatória)

Objetivo: Após a definição dos artefatos na atividade "Identificar os itens de configuração", estes serão armazenados e gerenciados em um repositório de configuração com o apoio da ferramenta de controle de versão selecionada.

Aplicação no PARFAIT: Na atividade: "Observar o domínio do sistema legado em relação ao do *framework*" (Fase de concepção).

Artefatos de entrada: Artefatos selecionados na atividade "Identificar os itens de configuração".

Apoio computacional: Ferramenta de controle de versão.

Artefatos de saída: Repositório criado.

Passo:

1. Criar o repositório na ferramenta de controle de versão selecionada.

Atividade: Gerenciar as Versões (Obrigatória)

Objetivo: Nesta atividade são gerenciadas as versões do *framework*, as versões do sistema alvo gerado a partir de tal *framework*, bem como as versões dos demais itens de configuração. Essa atividade é conduzida pelos envolvidos no projeto de reengenharia. Especificamente para o caso das versões do *framework* são documentadas as particularidades apresentadas em cada versão, como por exemplo, plataforma, necessidades de *hardware* e *software*, linguagem de programação, entre outras.

Aplicação no PARFAIT: em todas as atividades das fases de Elaboração e Construção e em algumas atividades da fase de Concepção e Transição: “Observar o domínio do sistema legado em relação ao do *framework*” (Fase de Concepção); “Confrontar as características não funcionais do *Framework* x sistema legado” (Fase de Concepção); “Elaborar o planejamento do projeto de reengenharia” (Fase de Concepção); “Elaborar o manual do usuário do sistema alvo” (Fase de Transição); “Testar o sistema alvo” (Fase de Transição).

Artefatos de entrada: Documentação das particularidades das versões do *framework* e dos sistemas gerados, bem como a dos demais itens de configuração.

Apoio computacional: Editor de texto e ferramenta de controle de versão.

Artefatos de saída: Documentação das versões do *framework*, do sistema alvo, dos demais itens de configuração (*baseline*).

Passos:

1. Documentar de forma detalhada as peculiaridades apresentadas em cada versão do *framework*, em cada versão do sistema alvo gerado a partir do mesmo e em cada versão dos demais itens de configuração.
2. Armazenar a versão do item de configuração no repositório da ferramenta de controle de versão.

Gabaritos: Nesta atividade foram desenvolvidos gabaritos para documentar as versões dos itens de configuração. Para documentar as versões do sistema alvo e do *framework* foram criados formulários específicos, apresentados nos Quadro 35 e Quadro 36, respectivamente. Para documentar as versões dos demais itens de configuração selecionados na atividade “Identificar os itens de configuração” é necessário utilizar o formulário apresentado no Quadro 37 . Salienta-se que serão documentadas apenas as versões dos itens de configuração que se tornarem *baseline*. Além de

informações específicas sobre a versão, é necessário registrar também o número da iteração do projeto de reengenharia que o item de configuração foi criado e/ou modificado, bem como o nome da atividade em que isso ocorreu.

Documentação das Versões do Sistema Alvo	
Documentação da Versão <número da versão do sistema> **do Sistema Alvo** <nome do sistema alvo>	
Projeto	**Data de criação do formulário**
<nome do projeto>	<dd/mm/aa – hh:mm>
Versão do *framework* utilizada <número de versão do *framework* >	
Variabilidades funcionais do *framework* reutilizadas	**Variabilidades Não-Funcionais do *framework* reutilizadas**
<lista dos nomes das variabilidades funcionais>	< lista dos nomes das variabilidades não-funcionais>
Requisitos funcionais do sistema não reutilizados do *framework*	**Requisitos não-funcionais do sistema não reutilizados do *framework***
<lista dos nomes dos requisitos funcionais >	<lista dos nomes dos requisitos não-funcionais>

Quadro 35 - Gabarito da documentação das versões do sistema alvo

Documentação das Versões do *Framework*	
Documentação da Versão <número da versão do *framework* > **do *framework*** <nome do *framework* >	
Plataforma	**Data de criação do formulário**
<nome da plataforma utilizada>	<dd/mm/aa – hh:mm>
Domínio	**Linguagem de programação**
<nome do domínio coberto pelo *framework*>	<nome da linguagem de programação utilizada>
Recursos de *hardware* e *software* requeridos	
<por exemplo, ambiente de desenvolvimento, versão do compilador da linguagem de programação, sistema gerenciador de banco de dados (SGBD), versão do SGBD>	
Variabilidades funcionais	**Variabilidades não-funcionais**
<lista dos nomes das variabilidades funcionais>	<lista dos nomes das variabilidades não-funcionais>

Quadro 36 - Gabarito da documentação das versões do *framework*

Documentação das Versões dos Demais Itens de Configuração				
Nome do Projeto		**Data de criação do formulário**		
<nome do projeto>		<dd/mm/aa – hh:mm>		
Data de criação ou atualização do item	**Nome dos itens de configuração**	**Versão**	**Número da iteração**	**Nome da atividade do PARFAIT**
...	**...**	**...**	**...**	**...**

Quadro 37 - Gabarito da documentação das versões dos itens de configuração

Atividade Gerenciar as Mudanças (Obrigatória)

Objetivo: Esta atividade é empregada somente quando existir a necessidade de mudanças no sistema alvo ou no *framework*, que sejam tomadas as medidas necessárias para que ocorram as mudanças sem prejudicar a integridade das *baselines* desses itens de configuração.

Aplicação no PARFAIT: Nas atividades "Criar o sistema alvo no paradigma orientado a objetos" (Fase construção) e "Adaptar o sistema alvo" (Fase de construção).

Artefatos de entrada: Formulários de controle de mudanças do *framework* e do sistema alvo

Apoio computacional: Editor de texto e ferramenta de controle de versão.

Artefatos de saída: Formulários de controle de mudanças do *framework* e do sistema alvo devidamente preenchidos e aprovados/reprovados.

Passos:

5. Identificar as mudanças necessárias.
6. Preencher os formulários de controle de mudanças do *framework* (caso seja necessário) e do sistema alvo (caso seja necessário).
7. Aprovar ou reprovar as mudanças.
8. Realizar as mudanças e executar a atividade "Gerenciar as Versões".

Gabarito: As solicitações de mudanças no sistema alvo, no *framework* e nos demais itens de configuração devem ser sempre documentadas por meio do formulário de controle de mudanças, apresentado respectivamente no Quadro 30, no Quadro 31 e no Quadro 32. As solicitações de mudanças no sistema alvo serão aprovadas ou não pelo gerente de configuração, de acordo com a complexidade das mesmas.

<table>
<tr><th colspan="3">Formulário de Controle de Mudanças no Sistema Alvo</th></tr>
<tr><td colspan="2">Nº da solicitação: <nº da solicitação da mudança></td><td>Projeto: <nome do projeto></td></tr>
<tr><td colspan="2">Data da solicitação: <dd/mm/aa></td><td>Tempo estimado para a execução da mudança <hh:mm></td></tr>
<tr><td colspan="2">Sistema alvo: <nome do sistema alvo></td><td>Versão do sistema alvo: <número correspondente da versão do sistema alvo que necessita sofrer mudanças></td></tr>
<tr><td colspan="3">Tipo de mudança: ☐ Perfectiva ☐ Corretiva ☐ Adaptativa ☐ Preventiva</td></tr>
<tr><td colspan="3">Mudança solicitada: <breve descrição da mudança></td></tr>
<tr><td colspan="3">Artefatos afetados pela mudança solicitada:
<lista dos nomes dos artefatos que serão afetados e que sofrerão mudanças></td></tr>
<tr><td colspan="2">☐ Solicitação de mudança APROVADA</td><td>☐ Solicitação de mudança NÃO APROVADA</td></tr>
<tr><td>☐ Implementar a mudança com apoio do framework:
- por meio de nova instanciação do framework (nesse caso o framework fornece a mudança desejada, parcialmente ou completamente)</td><td>☐ Implementar a mudança manualmente:
- por meio de um ambiente de programação</td><td>Retornar ao solicitante um parecer sobre o motivo da não aprovação da solicitação
<motivo que levou a inviabilidade das mudanças></td></tr>
<tr><td colspan="3">Comentários Adicionais: < breve descrição, se necessário></td></tr>
</table>

Quadro 38 - Gabarito do formulário de controle de mudanças no sistema alvo

<table>
<tr><th colspan="2">Formulário de Controle de Mudanças no Framework</th></tr>
<tr><td>Nº da solicitação: <nº da solicitação da mudança></td><td>Projeto: <nome do projeto></td></tr>
<tr><td>Data da solicitação: <dd/mm/aa></td><td>Tempo estimado para a execução da mudança <hh:mm></td></tr>
<tr><td colspan="2"><i>Framework</i>: <nome do <i>framework</i>></td></tr>
<tr><td colspan="2">Versão do <i>framework</i>: <número correspondente da versão do <i>framework</i>></td></tr>
<tr><td colspan="2">Tipo de mudança: ☐ Perfectiva ☐ Corretiva ☐ Adaptativa ☐ Preventiva</td></tr>
<tr><td colspan="2">Mudança solicitada: <breve descrição da mudança></td></tr>
<tr><td colspan="2">Artefatos afetados pela mudança solicitada:
<lista dos nomes dos artefatos que serão afetados e que sofrerão mudanças></td></tr>
<tr><td>☐ Solicitação de mudança APROVADA</td><td>☐ Solicitação de mudança NÃO APROVADA</td></tr>
<tr><td>☐ Implementar a mudança no framework</td><td>Retornar ao solicitante um parecer sobre o motivo da não aprovação da solicitação <motivo que levou a inviabilidade das mudanças></td></tr>
<tr><td colspan="2">Comentários Adicionais: < breve descrição, se necessário></td></tr>
</table>

Quadro 39 - Gabarito do formulário de controle de mudanças no *framework*

Formulário de Controle de Mudanças dos demais Itens de Configuração	
Nº da solicitação: <nº da solicitação da mudança>	**Projeto:** <nome do projeto>
Data da solicitação: <dd/mm/aa>	**Tempo estimado para a execução da mudança** <hh:mm>
Nome Artefato: <nome do artefato>	**Versão do Artefato:** <versão do artefato>
Tipo de mudança: ☐ **Perfectiva** ☐ **Corretiva** ☐ **Adaptativa** ☐ **Preventiva**	
Mudança solicitada: <breve descrição da mudança>	
Artefatos afetados pela mudança solicitada:	
<lista dos nomes dos artefatos que serão afetados e que sofrerão mudanças>	
☐ **Solicitação de mudança APROVADA**	☐ **Solicitação de mudança NÃO APROVADA**
Comentários Adicionais: < breve descrição, se necessário>	

Quadro 40 – Gabarito do formulário de controle de mudanças dos demais itens de configuração

Para analisar tal complexidade é preciso levar em consideração quais artefatos serão afetados pela mudança e se os mesmos necessitam de adaptações ou correções. No caso do PARFAIT com o apoio computacional do *framework* GREN, a implementação das adaptações (mudanças) realizadas manualmente no código fonte do sistema alvo é feita com o apoio da ferramenta *GREN–WizardVersionControl* a fim de que tais adaptações não sejam perdidas nas próximas instanciações do *framework.* As solicitações de mudanças no *framework* serão aprovadas ou não de acordo com uma análise, que verifica se uma mudança é pertinente ou não ao domínio do *framework.* Essa análise está fora do escopo deste trabalho e é feita com o apoio do Processo de Evolução de *Framework* (PREF) (CAGNIN *et al.*, 2004b). As mudanças no *framework* são efetuadas também com o apoio desse processo.

APÊNDICE B

Neste apêndice apresentam-se os diagramas elaborados durante o estudo de caso (Capítulo 6) realizado neste trabalho.

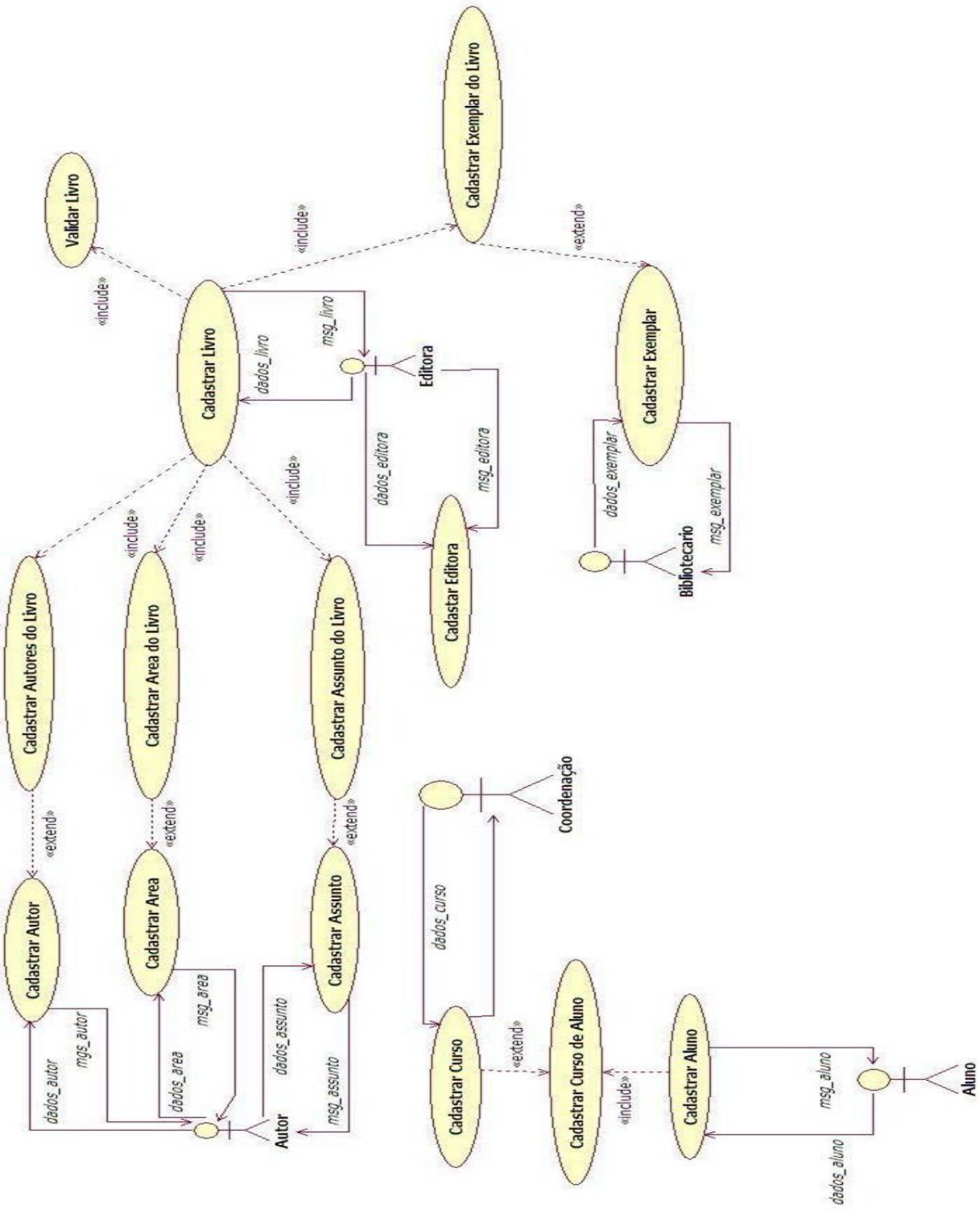

Figura 9– Diagrama de Caso de Uso

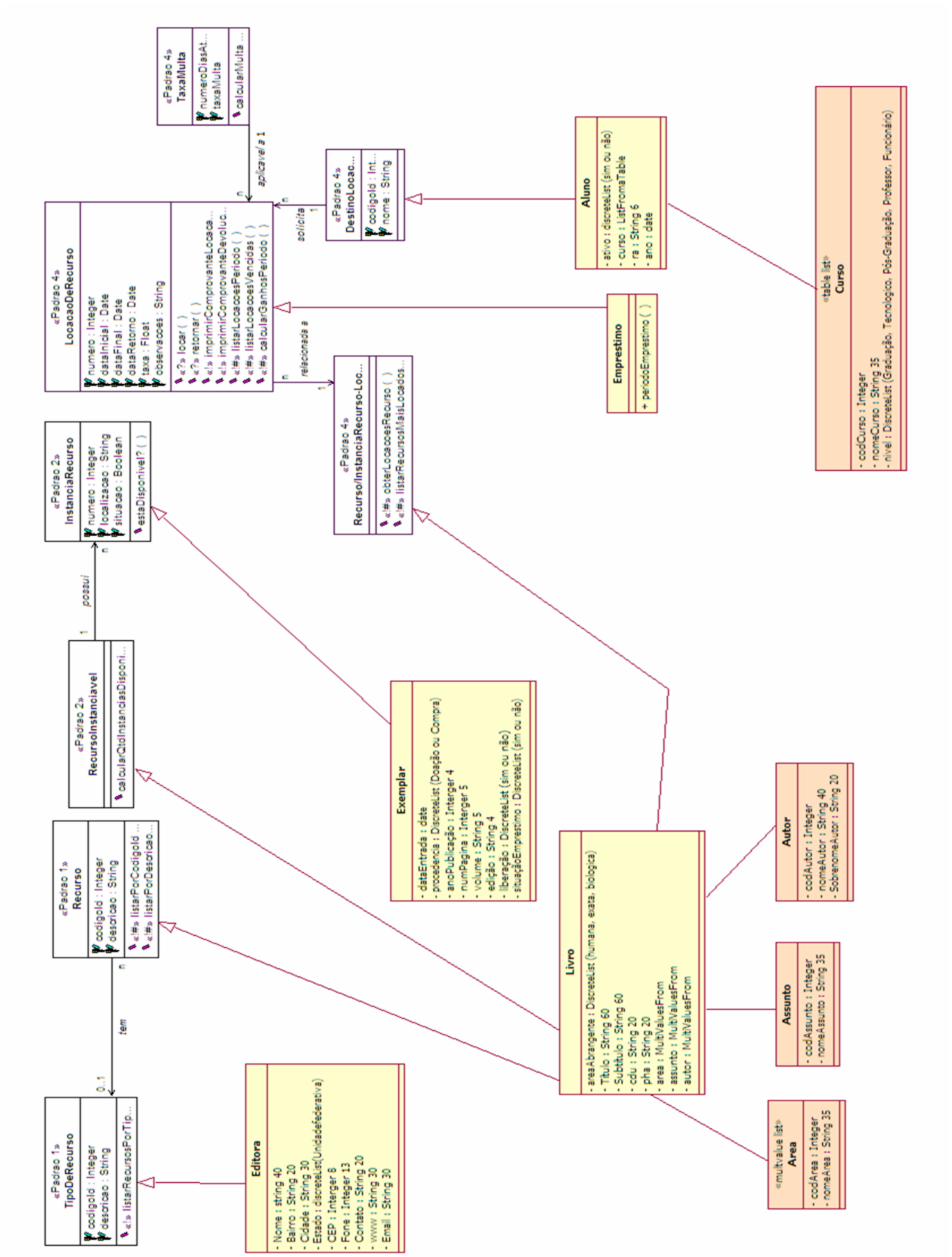

Figura 10 – Diagrama de Classes

APÊNDICE C

Neste apêndice apresentam-se os casos de teste genéricos dos padrões 1, 2 e 4 da LPA criados para aprimorar a abordagem de teste ARTe.

Id. casos de teste	Ação	Req teste ARTe	Dados de Entrada		Saída Esperada
CT1	Alteração	RT14 / RT259	Código	aa	Erro: código deve ser numérico
CT2	Alteração	RT15 / RT260	Código	0	Erro
CT3	Alteração	RT16 / RT261	Código	8	Sucesso
CT4	Alteração	RT17	Código	8	Código duplicado
CT5	Alteração	RT18 / RT263	Código	7 x 1	Erro: a quantidade de caracteres excede a expectativa
CT6	Alteração	RT19 / RT264	Código	-2	Erro: código tem de ser >= 0
CT7	Remoção	RT23	Código	x	Erro: caracteres inválidos
CT8	Remoção	RT24	Código	0	Erro: numero não existente
CT9	Remoção	RT25	Código	1	Sucesso
CT10	Remoção	RT26	Código	6	Sucesso
CT11	Remoção	RT27 / RT262	Código	7 x 9	Erro: código inexistente
CT12	Remoção	RT28	Código	-3	Erro: código tem de ser >= 0
CT13	Inserção	RT30	Descrição 60	AA	Sucesso: <= 60 caracteres
CT14	Inserção	RT30 / RT266 / RT267	Descrição 40	AA	Sucesso:< = 40 caracteres
CT15	Inserção	RT30	Descrição 30	AA	Sucesso: <= 30 caracteres
CT16	Inserção	RT30	Descrição 25	AA	Sucesso: <= 25 caracteres
CT17	Inserção	RT30	Descrição 20	AA	Sucesso: <= 20 caracteres
CT18	Inserção	RT31	Descrição 60	61 x A	Erro: a cima de 60 caracteres
CT19	Inserção	RT31	Descrição 40	41 x A	Erro: a cima de 40 caracteres
CT20	Inserção	RT31	Descrição 30	31 x A	Erro: a cima de 30 caracteres
CT21	Inserção	RT31	Descrição 25	26 x A	Erro: a cima de 25 caracteres
CT22	Inserção	RT31	Descrição 20	21 x A	Erro: a cima de 20 caracteres
CT23	Inserção	RT32	Descrição 60	“Vazio”	Não continua o cadastro
CT24	Inserção	RT32 / RT273 / RT268	Descrição 40	“Vazio”	Não continua o cadastro
CT25	Inserção	RT32	Descrição 30	“Vazio”	Não continua o cadastro
CT26	Inserção	RT32	Descrição 25	“Vazio”	Não continua o cadastro
CT27	Inserção	RT32	Descrição 20	“Vazio”	Não continua o cadastro
CT28	Inserção	RT33	Descrição 60	AA	Sucesso: esta Descrição já foi incluída deseja continuar? S/N
CT29	Inserção	RT33 / RT269 /RT 272 / RT 271	Descrição 40	AA	Sucesso: esta Descrição já foi incluída deseja continuar? S/N
CT30	Inserção	RT33	Descrição 30	AA	Sucesso: esta Descrição já foi incluída deseja continuar? S/N
CT31	Inserção	RT33	Descrição 25	AA	Sucesso: esta Descrição já foi incluída deseja continuar? S/N
CT32	Inserção	RT33	Descrição 20	AA	Sucesso: esta Descrição já foi incluída deseja continuar? S/N
CT33	Inserção	RT34	Descrição 60	1111	Sucesso
CT34	Inserção	RT34 / RT276 / RT270	Descrição 40	1111	Sucesso
CT35	Inserção	RT34	Descrição 30	1111	Sucesso

CT36	Inserção	RT34	Descrição 25	1111	Sucesso
CT37	Inserção	RT34	Descrição 20	1111	Sucesso
CT38	Alteração	RT35	Descrição 60	5 x b	Sucesso
CT39	Alteração	RT35 / RT 275	Descrição 40	5 x b	Sucesso
CT40	Alteração	RT34	Descrição 30	5 x b	Sucesso
CT41	Alteração	RT34	Descrição 25	5 x b	Sucesso
CT42	Alteração	RT34	Descrição 20	5 x b	Sucesso
CT43	Alteração	RT36	Descrição 60	61 x b	Erro: capacidade de ate 60 caracteres
CT44	Alteração	RT36	Descrição 40	41 x b	Erro: capacidade de ate 40 caracteres
CT45	Alteração	RT36	Descrição 30	31 x b	Erro: capacidade de ate 30 caracteres
CT46	Alteração	RT36	Descrição 25	26 x b	Erro: capacidade de ate 25 caracteres
CT47	Alteração	RT36	Descrição 20	21 x b	Erro: capacidade de ate 20 caracteres
CT48	Alteração	RT37	Descrição 60	“Vazio”	Não continua o cadastro
CT49	Alteração	RT37	Descrição 40	“Vazio”	Não continua o cadastro
CT50	Alteração	RT37	Descrição 30	“Vazio”	Não continua o cadastro
CT51	Alteração	RT37	Descrição 25	“Vazio”	Não continua o cadastro
CT52	Alteração	RT37	Descrição 20	“Vazio”	Não continua o cadastro
CT53	Alteração	RT38	Descrição 60	5 x b = 10 x a	Sucesso: este item já foi incluído deseja continuar? S/N
CT54	Alteração	RT38	Descrição 40	5 x b = 10 x a	Sucesso: este item já foi incluído deseja continuar? S/N
CT55	Alteração	RT38	Descrição 30	5 x b = 10 x a	Sucesso: este item já foi incluído deseja continuar? S/N
CT56	Alteração	RT38	Descrição 25	5 x b = 10 x a	Sucesso: este item já foi incluído deseja continuar? S/N
CT57	Alteração	RT38	Descrição 20	5 x b = 10 x a	Sucesso: este item já foi incluído deseja continuar? S/N
CT58	Alteração	RT39	Descrição 60	5 x b = 5 x b	Sucesso
CT59	Alteração	RT39 / RT 274	Descrição 40	5 x b = 5 x b	Sucesso
CT60	Alteração	RT39	Descrição 30	5 x b = 5 x b	Sucesso
CT61	Alteração	RT39	Descrição 25	5 x b = 5 x b	Sucesso
CT62	Alteração	RT39	Descrição 20	5 x b = 5 x b	Sucesso
CT63	Alteração	RT40	Descrição 60	423	Sucesso
CT64	Alteração	RT40	Descrição 40	423	Sucesso
CT65	Alteração	RT40	Descrição 30	423	Sucesso
CT66	Alteração	RT40	Descrição 25	423	Sucesso
CT67	Alteração	RT40	Descrição 20	423	Sucesso
CT68	Inserção/Alteração	RT2007	Lista a partir de uma classe	Valor A	Sucesso na verificação: permite continuar com a ação
CT69	Inserção/Alteração	RT2008	Lista a partir de uma classe	Valor B	Erro: Valor não cadastrado
CT70	Inserção/Alteração	RT2009	Lista a partir de uma classe	“Vazio”	Erro
CT71	Inserção/Alteração	RT2002	Lista Multivalorada	“Vazio”	Erro: não é permitido continuar com a ação
CT72	Inserção/Alteração	RT2000	Lista Multivalorada	Valor A	Sucesso na verificação: permite continuar com a ação
CT73	Inserção/Alteração	RT2001	Lista Multivalorada	Valor A + Valor B	Sucesso na verificação: permite continuar com a ação
CT74	Inserção/Alteração	RT2005	Lista Discreta	Valor A	Sucesso na verificação: permite continuar com a ação
CT75	Inserção/Alteração	RT2006	Lista Discreta	"Vazio"	Erro

Quadro 41 – Casos de teste genéricos do Padrão 1

Id. casos de teste	Ação	Req teste ARTe	Dados de Entrada		Saída Esperada
CT76	Inserção	RT330/RT360	Código	aa	Erro: código deve ser numérico
CT77	Alteração	RT331	Código	0	Erro
CT78	Alteração	RT332/RT333	Código	8	Sucesso
CT79	Alteração	RT334/ RT358	Código	8	Código duplicado
CT80	Alteração	RT335/ RT355	Código	7 x 1	Erro: a quantidade de caracteres excede a expectativa
CT81	Alteração	RT336	Código	-2	Erro: código tem de ser >= 0
CT80	Alteração	RT356	Código	"Vazio"	Erro
CT81	Alteração	RT357	Código	9 ==9	Sucesso
CT83	Remoção	RT340	Código	X	Erro: caracteres inválidos
CT84	Remoção	RT341	Código	0	Erro: numero não existente
CT85	Remoção	RT342	Código	1	Sucesso
CT86	Remoção	RT343	Código	6 x1	Sucesso
CT87	Remoção	RT344	Código	7 x 9	Erro: código inexistente
CT88	Remoção	RT346	Código	-3	Erro: código tem de ser >= 0
CT89	Inserção	RT361	Localização	AA	Sucesso: <= 35 caracteres
CT90	Inserção	RT362	Localização	36 x A	Erro: a cima de 35 caracteres
CT91	Inserção	RT363	Localização	“Vazio”	Não continua o cadastro
CT92	Inserção	RT364	Localização	AA	Sucesso: <= 35 caracteres, mesmo que duplicado
CT93	Inserção	RT365	Localização	1111	Sucesso
CT94	Alteração	RT367	Localização	36 x b	Erro: capacidade de ate 35 caracteres
CT95	Alteração	RT368	Localização	“Vazio”	Não continua o cadastro
CT96	Alteração	RT366	Localização	5 x b = 10 x a	Aviso: este item já foi incluído deseja continuar? S/N
CT97	Alteração	RT369	Localização	5 x b = 5 x b	Sucesso
CT98	Alteração	RT370	Localização	AA	Sucesso: localização já cadastrada
CT99	Alteração	RT371	Localização	111	Sucesso
CT100	Inserção	RT372	Status	1	Sucesso: valor A ou B
CT101	Inserção	RT373	Status	3	Erro: caracteres inválidos
CT102	Inserção	RT374	Status	2	Sucesso
CT103	Inserção	RT375	Status	"Vazio"	Erro
CT104	Inserção	RT376	Status	A	Erro
CT105	Alteração	RT377	Status	3	Erro: caracteres inválidos
CT106	Alteração	RT378	Status	1	Sucesso: valor A ou B
CT107	Alteração	RT379	Status	2	Sucesso: valor A ou B
CT108	Alteração	RT380	Status	"Vazio"	Erro
CT109	Alteração	RT381	Status	A	Erro
CT110	Alteração	RT382	Status	2 para 2	Sucesso
CT111	Inserção/Alteração	RT2005	Lista Discreta	Valor A	Sucesso na verificação: permite continuar com a ação
CT112	Inserção/Alteração	RT2006	Lista Discreta	"Vazio"	Erro

Quadro 42 - Casos de teste genéricos do Padrão 2

Id. casos de teste	Ação	Req teste ARTe	Dados de Entrada		Saída Esperada
CT113	Alteração	RT887	Código	aa	Erro: Código deve ser numérico
CT114	Alteração	RT888	Código	0	Erro
CT115	Alteração	RT889	Código	1	Sucesso
CT116	Alteração	RT890	Código	6 x 1	Sucesso
CT117	Alteração	RT892	Código	7 x 1	Erro: a quantidade de caracteres excede a expectativa
CT118	Remoção	RT896	Código	x	Erro: caracteres inválidos
CT119	Remoção	RT897	Código	0	Erro: numero não existente
CT120	Remoção	RT898	Código	1	Sucesso
CT121	Remoção	RT899	Código	6 x 1	Sucesso
CT122	Remoção	RT900	Código	7 x 9	Erro: Código inexistente
CT123	Remoção	RT901	Código	-3	Erro: Código tem de ser >= 0
CT124	Remoção	RT902	Código	2	Erro: código com relacionamento pendente
CT125	Inserção	RT903	Data Entrada	10/5/2007	Sucesso: Data Entrada < Data Entrada atual
CT126	Inserção	RT904	Data Entrada	10/10/2007	Erro: Data Entrada > Data Entrada atual
CT127	Inserção	RT905	Data Entrada	18/7/2007	Sucesso: Data Entrada = Data Entrada atual
CT128	Inserção	RT906	Data Entrada	x/07/07	Erro: dia não numérico
CT129	Inserção	RT907	Data Entrada	01/x/07	Erro: mês não numérico
CT130	Inserção	RT908	Data Entrada	01/07/xx	Erro: ano não numérico
CT131	Inserção	RT909	Data Entrada	00/00/00	Erro
CT132	Inserção	RT910	Data Entrada	00/01/2007	Erro: dia vazio
CT133	Inserção	RT911	Data Entrada	01/00/2007	Erro: mês vazio
CT134	Inserção	RT912	Data Entrada	1/12/0000	Erro: ano vazio
CT135	Inserção	RT913	Data Entrada	32/12/2007	Erro: dia != do esperado
CT136	Inserção	RT915	Data Entrada	10/13/2007	Erro: mês != do esperado
CT137	Inserção	RT917	Data Entrada	10/08/0090	Erro
CT138	Inserção	RT918	Data Entrada	10/10/3000	Erro
CT139	Inserção	RT920	Data Entrada	29/2/2004	Sucesso: ano bissexto
CT140	Inserção	RT921	Data Entrada	29/02/2007	Erro: ano não bissexto
CT141	Alteração	RT922	Data Entrada	10/5/2007	Sucesso: Data Entrada < Data Entrada atual
CT142	Alteração	RT923	Data Entrada	10/10/2007	Erro: Data Entrada > Data Entrada atual
CT143	Alteração	RT924/ RT941	Data Entrada	18/7/2007	Sucesso: Data Entrada = Data Entrada atual
CT144	Alteração	RT925	Data Entrada	x/07/07	Erro: dia não numérico
CT145	Alteração	RT926	Data Entrada	01/x/07	Erro: mês não numérico
CT146	Alteração	RT927	Data Entrada	01/07/xx	Erro: ano não numérico
CT147	Alteração	RT928	Data Entrada	00/00/00	Erro
CT148	Alteração	RT929	Data Entrada	00/01/2007	Erro: dia vazio
CT149	Alteração	RT930	Data Entrada	01/00/2007	Erro: mês vazio

CT150	Alteração	RT931	Data Entrada	1/12/0000	Erro: ano vazio
CT151	Alteração	RT932	Data Entrada	32/12/2007	Erro: dia != do esperado
CT152	Alteração	RT934	Data Entrada	10/13/2007	Erro: mês != do esperado
CT153	Alteração	RT936	Data Entrada	10/08/0090	Erro
CT154	Alteração	RT937	Data Entrada	10/10/3000	Erro
CT155	Alteração	RT939	Data Entrada	29/2/2004	Sucesso: ano bissexto
CT156	Alteração	RT940	Data Entrada	29/02/2007	Erro: ano não bissexto
CT157	Inserção	RT942	Observação	AA	Sucesso
CT158	Inserção	RT943	Observação	61 x A	Erro: tamanho excede o permitido
CT159	Inserção	RT944	Observação	"Vazio"	Sucesso: apesar de estar vazio
CT160	Inserção	RT945	Observação	AA	Sucesso: este item já foi incluído deseja continuar? S/N
CT161	Inserção	RT946	Observação	111	Sucesso: apenas de constar apenas números
CT162	Alteração	RT947	Observação	BB	Sucesso
CT163	Alteração	RT948	Observação	61 x B	Erro: tamanho excede o permitido
CT164	Alteração	RT949	Observação	"Vazio"	Sucesso: apesar de estar vazio
CT165	Alteração	RT951	Observação	AA	Sucesso: este item já foi incluído deseja continuar? S/N
CT166	Alteração	RT952	Observação	111	Sucesso: apenas de constar apenas números
CT167	Inserção	RT953	Status	1	Sucesso: valor A ou B
CT168	Inserção	RT954	Status	3	Erro: caracteres inválidos
CT169	Inserção	RT955	Status	2	Sucesso
CT170	Inserção	RT956	Status	"Vazio"	Erro
CT171	Inserção	RT957	Status	A	Erro
CT172	Alteração	RT958	Status	1 mudar para 2	Sucesso: valor != da atual
CT173	Alteração	RT959	Status	3	Erro: caracteres inválidos
CT174	Alteração	RT960	Status	22	Erro
CT175	Alteração	RT961	Status	"Vazio"	Erro
CT176	Alteração	RT962	Status	A	Erro
CT177	Alteração	RT963	Status	2 para 2	Sucesso
CT178	Inserção	RT1118/ RT1161	Data Chegada/ Data Retorno	10/5/2007	Erro: Data Chegada < Data atual
CT179	Inserção	RT1119/ RT1162	Data Chegada/ Data Retorno	25/7/2007	Sucesso: Data Chegada > Data atual
CT180	Inserção	RT1120/ RT1163	Data Chegada/ Data Retorno	18/7/2007	Sucesso: Data Chegada = Data atual
CT181	Inserção	RT1121/ RT1164	Data Chegada/ Data Retorno	19/7/2007	Sucesso: Data Chegada = Data atual
CT182	Inserção	RT1123/ RT1166	Data Chegada/ Data Retorno	x/07/07	Erro: dia não numérico
CT183	Inserção	RT1124/ RT1167	Data Chegada/ Data Retorno	01/x/07	Erro: mês não numérico
CT184	Inserção	RT1125/ RT1168	Data Chegada/ Data Retorno	01/07/xx	Erro: ano não numérico
CT185	Inserção	RT1126/ RT1169	Data Chegada/ Data Retorno	00/00/00	Erro
CT186	Inserção	RT1127/ RT1170	Data Chegada/ Data Retorno	00/01/2007	Erro: dia vazio

CT187	Inserção	RT1128/ RT1171	Data Chegada/ Data Retorno	01/00/2007	Erro: mês vazio
CT188	Inserção	RT1129/ RT1172	Data Chegada/ Data Retorno	1/12/0000	Erro: ano vazio
CT189	Inserção	RT1130/RT 1173	Data Chegada/ Data Retorno	32/12/2007	Erro: dia != do esperado
CT190	Inserção	RT1132/ RT1175	Data Chegada/ Data Retorno	10/13/2007	Erro: mês != do esperado
CT191	Inserção	RT1134/ RT1177	Data Chegada/ Data Retorno	10/08/0090	Erro
CT192	Inserção	RT1135/ RT1178	Data Chegada/ Data Retorno	10/10/3000	Erro
CT193	Inserção	RT1137/ RT1180	Data Chegada/ Data Retorno	29/2/2004	Sucesso: ano bissexto
CT194	Inserção	RT1138/ RT1181	Data Chegada/ Data Retorno	29/02/2007	Erro: ano não bissexto
CT195	Alteração	RT1139/ RT1183	Data Chegada/ Data Retorno	25/07/2007 para 28/072007	Sucesso: Data Chegada > Data atual
CT196	Alteração	RT1140/ RT1184	Data Chegada/ Data Retorno	18/07/07 para 17/07/2007	Erro: Data Chegada < Data atual
CT197	Alteração	RT1141/ RT1185	Data Chegada/ Data Retorno	18/7/2007	Sucesso: Data Chegada = Data atual
CT198	Alteração	RT1142/ RT1186	Data Chegada/ Data Retorno	22/7/2007	Sucesso
CT199	Alteração	RT1144/ RT1188	Data Chegada/ Data Retorno	x/07/07	Erro: dia não numérico
CT200	Alteração	RT1145/ RT1189	Data Chegada/ Data Retorno	01/x/07	Erro: mês não numérico
CT201	Alteração	RT1146/ RT1190	Data Chegada/ Data Retorno	01/07/xx	Erro: ano não numérico
CT202	Alteração	RT1147/ RT1191	Data Chegada/ Data Retorno	00/00/00	Erro
CT203	Alteração	RT1148/ RT1192	Data Chegada/ Data Retorno	00/01/2007	Erro: dia vazio
CT204	Alteração	RT1149/ RT1193	Data Chegada/ Data Retorno	01/00/2007	Erro: mês vazio
CT205	Alteração	RT1150/ RT1194	Data Chegada/ Data Retorno	1/12/0000	Erro: ano vazio
CT206	Alteração	RT1151/ RT1195	Data Chegada/ Data Retorno	32/12/2007	Erro: dia != do esperado
CT207	Alteração	RT1153/ RT1197	Data Chegada/ Data Retorno	10/13/2007	Erro: mês != do esperado
CT208	Alteração	RT1155/ RT1199	Data Chegada/ Data Retorno	10/08/0090	Erro
CT209	Alteração	RT1156/ RT1200	Data Chegada/ Data Retorno	10/10/3000	Erro
CT210	Alteração	RT1158/ RT1202	Data Chegada/ Data Retorno	29/2/2004	Sucesso: ano bissexto
CT211	Alteração	RT1159/ RT1203	Data Chegada/ Data Retorno	29/02/2007	Erro: ano não bissexto
CT212	Alteração	RT1160/ RT1204	Data Chegada/ Data Retorno	28/07/07 para 18/07/07	Sucesso
CT213	Inserção	RT1182/ RT1204	Data Retorno	1/1/1901	Sucesso
CT214	Inserção	RT1003	Destino	A	Erro: Destino do tipo numérico
CT215	Inserção	RT1004	Destino	0	Erro
CT216	Inserção	RT1005	Destino	1111	Sucesso
CT217	Inserção	RT1006	Destino	6 x 1	Sucesso
CT218	Inserção	RT1007	Destino	5	Sucesso: Destino = ao ultimo cadastrado
CT219	Inserção	RT1008	Destino	7 x 1	Erro: excede a quantidade permitida
CT220	Inserção	RT1009	Destino	-1	Erro: Destino >=1
CT221	Alteração	RT1010	Destino	1 para 1	Sucesso: mudar para o mesmo valor

CT222	Inserção	RT1434	Código do Recurso Instanciavel	7	Sucesso código do exemplar não cadastrado na tabela livro e cadastrado no exemplar e status disponível
CT223	Inserção	RT1435	Código do Recurso Instanciavel	3	Erro: código do exemplar não cadastrado na tabela livro e cadastrado no exemplar e status retirado
CT224	Inserção	RT1436	Código do Recurso Instanciavel	2	Sucesso código do exemplar cadastrado na tabela livro e cadastrado no exemplar e status disponível
CT225	Inserção	RT1437	Código do Recurso Instanciavel	1	Erro: código do exemplar cadastrado na tabela livro e cadastrado no exemplar e status retirado
CT226	Inserção	RT1438	Código do Recurso Instanciavel	8	Erro: código não cadastrado
CT227	Alteração	RT1439	Código do Recurso Instanciavel	7	Erro código do exemplar não cadastrado na tabela livro e cadastrado no exemplar e status disponível
CT228	Alteração	RT1440	Código do Recurso Instanciavel	3	Erro: código do exemplar não cadastrado na tabela livro e cadastrado no exemplar e status retirado
CT229	Alteração	RT1441	Código do Recurso Instanciavel	2	Sucesso código do exemplar cadastrado na tabela livro e cadastrado no exemplar e status livro já retira e devolvido
CT230	Alteração	RT1442	Código do Recurso Instanciavel	1	Sucesso: código do exemplar cadastrado na tabela livro e cadastrado no exemplar e status retirado
CT231	Alteração	RT1443	Código do Recurso Instanciavel	8	Erro: código não cadastrado
CT232	Inserção	RT1444	Código do Recurso Instanciavel	1	Sucesso: código do exemplar cadastrado na tabela livro e cadastrado no exemplar e status retirado
CT233	Inserção	RT1445	Código do Recurso Instanciavel	2	Erro: código do exemplar cadastrado na tabela livro e cadastrado no exemplar e status livro já retira e devolvido
CT234	Inserção	RT1446	Código do Recurso Instanciavel	3	Erro: código do exemplar não cadastrado na tabela livro e cadastrado no exemplar e status retirado
CT235	Inserção	RT1447	Código do Recurso Instanciavel	7	Erro código do exemplar não cadastrado na tabela livro e cadastrado no exemplar e status disponível
CT236	Inserção	RT1448	Código do Recurso Instanciavel	8	Erro: código não cadastrado
CT237	Alteração	RT1449	Código do Recurso Instanciavel	1	Sucesso: código do exemplar cadastrado na tabela livro e cadastrado no exemplar e status retirado
CT238	Alteração	RT1450	Código do Recurso Instanciavel	2	Erro: código do exemplar cadastrado na tabela livro e cadastrado no exemplar e status livro já retira e devolvido
CT239	Alteração	RT1451	Código do Recurso Instanciavel	3	Erro: código do exemplar não cadastrado na tabela livro e cadastrado no exemplar e status retirado
CT240	Alteração	RT1452	Código do Recurso Instanciavel	7	Erro código do exemplar não cadastrado na tabela livro e cadastrado no exemplar e status disponível
CT241	Alteração	RT1453	Código do Recurso Instanciavel	8	Erro: código não cadastrado

Quadro 43 - Casos de teste genéricos do Padrão 4