João Ronaldo Del-Ducca Cunha

SoCManager: Uma Ferramenta de Apoio ao Gerenciamento de Configuração de Software

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Orientador:
Prof. Dr. Antonio Francisco do Prado

Co-orientador:
Prof. Dr. Antonio Carlos dos Santos

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO
DEPARTAMENTO DE COMPUTAÇÃO
PPG/CC - PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE SÃO CARLOS

São Carlos – SP

Junho / 2005

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

C972sf

Cunha, João Ronaldo Del-Ducca.
SoCManager: uma ferramenta de apoio ao gerenciamento de configuração de software / João Ronaldo Del-Ducca Cunha. -- São Carlos : UFSCar, 2005.
111 p.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2005.

1. Engenharia de software. 2. Qualidade de software. 3. Melhoria de processo. I. Título.

CDD: 005.1 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"SoCManager: Uma Ferramenta de Apoio ao Gerenciamento de Configuração de Software"

JOÃO RONALDO DEL-DUCCA CUNHA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Antonio Francisco do Prado
(Orientador – DC/UFSCar)

Prof. Dr. Antonio Carlos dos Santos
(Co-Orientador – DC/UFSCar)

Profa. Dra. Selma Shin Shimizu Melnikoff
(POLI/USP)

São Carlos
Março/2005

Dedico esta dissertação sobretudo a Deus e a Nossa Senhora,
e a meus pais e irmãos, cujo exemplo de honestidade e
trabalho tem sido um norteador para a minha vida,
me apoiando principalmente nos momentos difíceis
e me ensinando a simplicidade de ter esperança.

Agradecimentos

Muitos foram os que contribuíram para a realização deste projeto. Destaco no entanto alguns, não necessariamente os mais importante, mas os que foram essenciais, em momentos precisos, para a consolidação desta conquista.

Primeiramente e sobretudo agradeço a Deus por ter me dado a oportunidade e capacidade de aprender, e a Nossa Senhora que, por meio de sua interseção, me deu forças e coragem para enfrentar os desafios e buscar a realização de meus sonhos, nunca deixando que eu me abatesse pelas quedas e tristezas, me conservando na humildade.

Aos professores Antonio Francisco do Prado e Antônio Carlos dos Santos, pela orientação e incentivo, pelos ensinamentos, amizade e principalmente pela confiança depositada em mim.

Aos professores Ethan V. Munson, Alexandre Vasconcelos e Rosely Sanches, por terem contribuído de forma essencial na concepção e realização do projeto.

Ao meu pai, que com a sua experiência e dedicação profissional, me ajudou a enxergar a solução para muitos problemas enfrentados.

À equipe do Laboratório de Engenharia de Software do DC da UFSCar, em especial ao amigo Daniel Lucrédio por estar sempre disposto a ajudar e discutir novas idéias e projetos, pelo companheirismo e pelos momentos de descontração; ao Vinicius Garcia, por partilhar diversas opiniões e ajuda em diversos momentos; e à Val Fontanette pelas inúmeras traduções e auxílio no inglês.

Aos colegas de mestrado pelo essencial apoio em momentos difíceis e pelos momentos de descontração, especialmente, André Constantino, Raphael Souza, Genta, Mairum, Gustavo, Pegoraro, Vinicius Garibaldi, Ivan, e outros.

Aos meus amigos e colegas da BCP Telecomunicações que, de alguma forma, me apoiaram na decisão de deixar a empresa e encarar o desafio do mestrado, especialmente, Marcelo Toda, João Paulo, Guilherme, Rodrigo, Glauco, Beatriz, Joércio e Joel.

Aos amigos e colegas da finda república “Doze Torto”, em especial aos meus amigos Júlio Maia, Luiz Henrique, Rodrigo “Dracena” e Damaso, exemplos de honestidade, seriedade e temor a Deus, pelos vários momentos de descontração e pelos valiosos conselhos.

À Vanesa Carboni Barbosa, minha noiva, por ter tido a paciência em aceitar o meu nervosismo e anciedade durante vários momentos, e por entender a loucura que é a vida de um estudante de mestrado. Sem a sua compreensão, incentivo, carinho e apoio, a conclusão e o sucesso deste trabalho não seria possível.

Aos meus pais e irmãos pelo incentivo, amor, carinho, e cujo suporte foi essencial para a minha formação e para a realização deste projeto. É por vocês que eu procuro sempre mais. Com certeza este trabalho deve-se muito a todo esse amor. Amo todos vocês.

Finalmente, à todos aqueles que colaboraram direta ou indiretamente na realização deste projeto.

Meus sinceros agradecimentos.

“É muito melhor arriscar coisas grandiosas,
alcançar triunfos e glória, mesmo expondo-se a derrota,
do que formar fila com os pobres de espírito
que nem gozam muito nem sofrem muito,
porque vivem nessa penumbra cinzenta,
que não conhece vitória nem derrota ”

T. Roosevelt

Resumo

Os processos de desenvolvimento de software buscam promover o sucesso dos projetos por meio da execução de atividades que contribuem com o cumprimento de prazos e o atendimento dos requisitos estabelecidos pelo cliente. O Processo de Gerenciamento de Configuração de Software (GCS), por exemplo, auxilia no desenvolvimento do projeto definindo atividades para Identificação, Controle, Administração de Estado e Auditagem da Configuração, que contribuem para a elevação da qualidade do projeto através do controle dos artefatos produzidos e do auxílio aos demais processos, com destaque, o de manutenção. No entanto, muitas empresas de software ainda não implantaram o processo de GCS, devido às dificuldades de tornar a execução deste processo em algo realmente operacional. Estas dificuldades vão desde a falta de abordagens para a implantação do processo de GCS até a escassez de ferramentas que dêem apoio a todas as atividades de um processo de GCS. Neste contexto e buscando atender as necessidades das empresas de software em melhorar a qualidade de seus projetos, este projeto de pesquisa foi desenvolvido com o objetivo de auxiliar a implantação do processo de GCS, em empresas de software. Para tanto esta pesquisa foi dividida em três partes: proposição de uma abordagem para implantação do processo de GCS em empresas de software; proposição de uma abordagem para o processo de GCS, que define os agentes e atividades dentro do processo; e construção de uma ferramenta de apoio à execução do processo de GCS, implementada com base na abordagem para o processo de GCS proposta nesta pesquisa.

Abstract

The processes of Software development contribute to the success of software projects by means of the execution of activities that contribute to the fulfillment of the deadlines and the requirements agreed with the customer. The Software Configuration Management (SCM) process, for instance, defines activities for Identification, Control, Status Management and Configuration Audit, that contribute for the rise and leverage of the quality of the project through the control of the produced artifacts and the assistance to the other processes especially maintenance. However, there are still many software companies which yet have not implemented the SCM process, due to the difficulties in turning the execution of this this process in something actually operational. These difficulties include the lack of a policy to the implementation of SCM process and the lack of tools that support SCM activities. In this context and searching to take care of the needs of software companies in improving the quality of its projects, this research project was developed with the objective of assisting the implementation of the SCM process, in software companies. So this research it was divided in three parts: a) an approach for the implementation of a SCM process in software companies; b) a proposal of an approach for the SCM process, which defines the agents and activities inside the process; and c) the implementation of a tool to support the SCM process approach proposed in this research.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos, Siglas e Abreviaturas

ABNT - Associação Brasileira de Normas Técnicas

API - *Application Programming Interface*

C-CORE - *Component Construction and Reuse*

CASE - *Computer-Aided Software Engineering*

CAR - *Configuration Audit Register*

CCB - *Configuration Control Board*

CI - *Configuration Item*

CMMi - *Capability Maturity Model Integration*

CMO - *Configuration Management Officer*

CRF - *Change Request Form*

CVS - *Concurrent Version System*

DBC - Desenvolvimento Baseado em Componentes

EAD - Educação a Distância

EJB - *Enterprise Java Beans*

EUP - *Enterprise Unified Process*

GoES - Grupo de Engenharia de Software

GCS - Gerenciamento de Configuração de Software

IEC - *International Electrotechnical Commission*

IEEE - *Institute of Electrical and Electronics Engineers*

IDEAL - *Initiating-Diagnostics-Establishing-Acting-Leveraging*

IPD-CMM - *Capability Maturity Model for Integrated Product Development*

ISO - *International Standard Organization*

JMI - *Java Metadata Interface*

JSP - *Java Server Pages*

KPA - *Key Process Area*

MCT - Ministério de Ciência e Tecnologia

MDR - *MetaData Repository*

MOF - *Meta Object Facility*

MP - *Modification Plan*

MRF - *Modification Request Form*

MVC - *Model-View-Controller*

MVCASE - *Multiple View CASE*

NASA - *National Aeronautics and Space Administration*

NBR - Norma BRasileira

OMG - *Object Management Group*

PA - *Process Area*

PEB - *Process Engineer Board*

PRAXIS - PRocesso para Aplicativos eXtensíveis InterativoS

QAB - *Quality Assurance Board*

RUP - *Rational Unified Process*

SA-CMM - *Capability Maturity Model for Software Acquisition*

SCM - *Software Configuration Management*

SE-CMM - *Capability Maturity Model for System Engineering*

SEI - *Software Engineer Institute*

SoCManager - *Software Configuration Manager*

SubCM - *Subsystem Configuration Management*

SW-CMM - *Capability Maturity Model for Software*

SPICE - *Software Process Improvement and Capability dEtermination*

UFSCar - Universidade Federal de São Carlos

UML - *Unified Modeling Language*

UP - *Unified Process*

VO - *Value Object*

XMI - XML *Metadata Interchange*

XML - *eXtensible Markup Language*

Sumário

1 Introdução

É notória a dificuldade das empresas em adotar conceitos e práticas da Engenharia de Software que contribuam para a qualidade e, conseqüentemente, o sucesso dos projetos de software desenvolvidos, acarretando em prejuízos que transcendem o financeiro, como por exemplo, o sucesso profissional da equipe do projeto e a imagem organizacional frente ao mercado e aos seus clientes.

Esta dificuldade muitas vezes reside no fato de que a adoção das práticas de Engenharia de Software exige um bom investimento tanto em recursos financeiros como humanos. Recursos estes que são limitantes principalmente para pequenas e médias empresas, o que acarreta muitas vezes num impedimento à adoção dessas práticas que poderiam contribuir para o crescimento dessas empresas no mercado de informática. Outra dificuldade enfrentada tanto por grandes como médias e pequenas empresas é o aspecto cultural da organização, isto é, a resistência das pessoas em adotar as práticas da Engenharia de Software na sua rotina de trabalho.

Almejando reduzir estas dificuldades, muitos esforços já foram realizados, por meio de estudos e pesquisas que buscam contribuir para a evolução da Engenharia de Software e de suas áreas de conhecimento com a definição de processos, métodos, modelos e padrões. Dentre essas áreas de conhecimento, cita-se a de Gerenciamento de Configuração de Software (GCS) que está relacionada à Gestão da Qualidade, contribuindo para a qualidade do projeto por meio do suporte às atividades relacionadas tanto aos aspectos gerenciais quanto aos aspectos técnicos do projeto. Nesta área, os estudos resultaram na elaboração de documentos como, por exemplo, o documento de padrão da IEEE (IEEE, 1998), que define o escopo, conceitos e outros aspectos relacionados ao GCS.

Neste contexto, foram sendo definidos processos de software, relacionados a cada uma das áreas de conhecimento da Engenharia de Software, com o objetivo de auxiliar as empresas no desenvolvimento dos projetos de software. Além disso, outras pesquisas foram realizadas para definir modelos e normas de qualidade cujo objetivo é estabelecer metas a serem seguidas por organizações que desenvolvem projetos de software. Estas metas estão relacionadas a uma série de áreas de conhecimento da Engenharia de Software como, por exemplo, o GCS, e tem o objetivo de direcionar os esforços realizados pela empresa para alcançar qualidade no desenvolvimento do

software e, por conseqüência, aumentar o grau de sucesso dos projetos de software pela aferição de qualidade dos processos utilizados para desenvolvê-los. Dentre as normas e modelos de qualidade de software, citam-se o CMMi (*Capability Maturity Model Integration*) (SEI, 2002) definido pelo SEI (*Software Engineering Institute*) e o SPICE (*Software Process Improvement and Capability dEtermination*) (ISO/IEC, 1998).

Com base nesses processos e modelos de qualidade de software, foram elaboradas abordagens para atender às necessidades específicas das empresas de software, definindo as atividades a serem realizadas, os papéis e responsabilidades dentro da empresa, assim como informações que devem ser consumidas e geradas por cada uma das atividades da abordagem, levando-se em consideração a quantidade de recursos tanto humanos como financeiros disponíveis nas empresas de software que irão aplicá-las.

Portanto, a adoção de um processo de software em uma empresa está, em parte, na elaboração de abordagens para o processo visando atender às metas previstas nos modelos de qualidade. Além disso, para auxiliar a aplicação destas abordagens, muitas vezes se faz necessário o apoio de ferramentas como, por exemplo, as chamadas ferramentas CASE (PRESSMAN, 2001). Estas ferramentas auxiliam a equipe de desenvolvimento na execução das atividades definidas nas abordagens.

No caso específico do processo de GCS, à medida que aumentam o grau de complexidade e a magnitude de um projeto de software, o seu apoio por parte de uma ferramenta de GCS passa a ser cada vez mais essencial para o sucesso do projeto (IEEE, 2004). Isto ocorre, pois nestes projetos há uma quantidade muito grande de informações, exigindo uma maior atenção sobre as evoluções realizadas nos seus artefatos. Além disso, ao se construir um produto a partir dos artefatos do projeto, um único erro pode acarretar no funcionamento incorreto do software (SOMMERVILLE, 2003). Porém, segundo dados fornecidos pela mais recente pesquisa realizada sobre qualidade e produtividade no setor de software brasileiro, disponibilizados pelo Ministério da Ciência e Tecnologia (MCT) (MCT, 2002), estimou-se que, no ano de 2000, cerca de 23,4% das empresas de software adotavam práticas de GCS no desenvolvimento e manutenção de software e cerca de 20,1% utilizavam ferramentas de apoio a este processo.

Neste contexto, este pesquisa tem como maior objetivo auxiliar empresas de software na implantação do processo de GCS e conseqüentemente na melhoria da qualidade e produtividade de seus projetos. Para tanto, foi construída uma ferramenta computacional, denominada *SoCManager* (*Software Configuration Manager*) de apoio ao GCS de tal forma a auxiliar as empresas de software na operacionalização deste processo. Esta ferramenta está integrada a outras ferramentas computacionais desenvolvidas no laboratório GoES [1]: MVCASE para modelagem de projetos de

[1]GrupO de Engenharia de Software do Departamento de Computação da Universidade Federal de São Carlos (UFSCar)

software; e C-CORE para implementação e testes de software.

Parte dos requisitos da *SoCManager* foram obtidos a partir de uma abordagem para o processo de GCS elaborada neste projeto de pesquisa com base na área de processo de Gerenciamento de Configuração do modelo de capacitação CMMi.

Nesta abordagem foram definidas as atividades, os agentes e as responsabilidades destes agentes sob o contexto do processo de GCS.

Adicionalmente, foi proposta uma abordagem para auxiliar empresas de software na implantação do processo de GCS. Esta abordagem foi elaborada tendo como base o modelo IDEAL (*Initiating, Diagnostics, Establishing, Acting, Leveraging*) (MCFEELEY, 1996) para melhoria de processos de software.

Esta dissertação encontra-se organizada em capítulos. Apresentou-se no capítulo 1 uma introdução aos demais capítulos que compreendem o documento.

Apresenta-se no capítulo 2 a fundamentação teórica desta pesquisa. Inicialmente apresentam-se os conceitos referentes a processos de software e, em específico, o processo de GCS. Posteriormente apresentam-se os conceitos relacionados a modelos e normas de qualidade de processos de software. Em seguida apresentam-se conceitos sobre a melhoria de processos de software. Finalmente, apresentam-se soluções existentes para implantação de GCS que dizem respeito tanto a abordagens como também a ferramentas de GCS.

Apresentam-se no capítulo 3 os resultados desta pesquisa. Dá-se uma visão geral sobre o que foi desenvolvido e em seguida são apresentados os resultados em três partes: na primeira, parte com a abordagem para a implantação do processo de GCS; na segunda parte, com a abordagem para o processo de GCS; na terceira parte, com a ferramenta *SoCManager*.

Apresenta-se no capítulo 4 a avaliação desses resultados por meio de estudos de caso que envolveram a construção de componentes para o domínio de Educação a Distância (EAD) e Multimídia, além da construção de duas aplicações para a reutilização desses componentes.

Por fim, apresentam-se no Capítulo 5 as considerações finais e os trabalhos relacionados e futuros referentes a esta pesquisa.

2 *Fundamentação Teórica*

Apresentam-se neste capítulo conceitos utilizados como fundamentação teórica para o projeto de pesquisa descrito neste documento. Inicialmente citam-se processos de software e a sua contribuição para o desenvolvimento de projetos de software, dando destaque ao Gerenciamento de Configuração de Software (GCS) que é um processo de apoio ao desenvolvimento de projetos de software.

Posteriormente apresentam-se as normas, padrões e modelos para avaliar a qualidade de processos de software. Dá-se destaque ao modelo de capacitação CMMi (*Capability Maturity Model Integration*) que foi utilizado como base no desenvolvimento deste projeto de pesquisa. Iso se deve ao fato deste modelo substituir o modelo de capacitação SW-CMM (*Capability Maturity Model for Software*) que possui um alto grau de aceitabilidade por parte das empresas de software (conforme demostrado em pesquisa realizado pelo MCT (MCT, 2002)).

Apresentam-se também conceitos referentes a melhorias e implantação de processos de software e, mais especificamente, da implantação do processo de GCS.

Finalmente citam-se soluções de GCS referentes a abordagens para este processo e a ferramentas computacionais de apoio à execução deste processo.

2.1 Processos de Software

Nos primeiros anos da Engenharia de Software, o termo “processo” era utilizado para denotar o processo de software como um todo, ou seja, um processo macro e abstrato que levava à construção do software e que não possuía muitos métodos e disciplinas associadas, prevendo apenas o desenvolvimento do produto em si (SANT’ANNA, 2001).

Porém, à medida que foram sendo realizados novos estudos relacionados ao desenvolvimento das áreas de conhecimento de Engenharia de Software, os processos foram sendo aprimorados, permitindo que os projetos que os utilizavam pudessem ser mais bem planejados, gerenciados, executados, monitorados e controlados.

Atualmente, um processo de software pode ser definido como sendo um conjunto de passos parcialmente ordenados, constituídos por atividades, métodos e práticas usado no desenvolvimento de software (PÁDUA, 2003), cujo objetivo é realizar a tradução das necessidades do usuário para um produto de software (IEEE, 1990). Portanto, este processo envolve a tradução das necessidades do usuário para requisitos de software, transformação destes requisitos em modelos de projeto, implementação dos modelos em código, teste deste código e, em alguns casos, instalação do software no ambiente operacional.

Em Engenharia de Software, processos de software são utilizados para atividades como desenvolvimento, manutenção, aquisição e contratação de software.

De acordo com os seus objetivos no contexto do desenvolvimento do software, os processos de software podem ser categorizados como técnicos ou como gerenciais (PÁDUA, 2003). Os processos técnicos visam à construção do software. Já os gerenciais visam suportar, apoiar e controlar os processos técnicos para que o software tenha qualidade e atenda a prazos e a custos estabelecidos. Citam-se como processos técnicos os de identificação de requisitos, análise, desenho, implementação e testes. Já os gerencias estão relacionados ao Gerenciamento de Projetos e Gerenciamento de Qualidade.

Já o modelo SPICE (ISO/IEC, 1998) define as categorias dos processos de software como sendo: processos cliente-fornecedores, que estão diretamente relacionados com os clientes; processos de engenharia, que são os de especificação, implementação ou manutenção de um produto de software; processos de projeto, que estabelecem o projeto e coordenam e gerenciam os seus recursos; processos de suporte, que oferecem suporte aos outros processos no projeto; e processos organizacionais, que estabelecem os objetivos de negócio da organização e desenvolvem o processo, o produto e os recursos que irão contribuir para que a organização alcance os seus objetivos de negócio.

No contexto desta dissertação, define-se projeto como sendo uma unidade de gestão a qual realiza algo único, que possui uma data de início e término (PMI, 2000) e que representa a execução de processos de software.

2.1.1 Processo de GCS

Dentre os processos de software, tem-se o de GCS, tido como um processo gerencial de qualidade, segundo (PÁDUA, 2003) e como um processo de suporte ao desenvolvimento do projeto segundo o modelo SPICE (ISO/IEC, 1998). Este processo realiza o controle e fornece informações do projeto para suporte às atividades definidas em outros processos, como por exemplo, a

manutenção.

Atualmente, existe uma série de definições para GCS. Porém, não há uma mais correta que a outra e sim mais concisa, com palavras mais apropriadas ou aquela que é mais aceita pela comunidade. Seguem algumas definições de GCS existentes:

- GCS é uma disciplina para controlar a evolução de sistemas de software (DART, 1992);
- GCS é o processo de identificar e definir componentes em um sistema, controlando a atualização e alteração através do seu ciclo de vida, registrando e relatando o status dos componentes, das solicitações de alteração e verificando a completeza dos componentes do sistema (IEEE, 1998); e
- GCS é o processo cujo objetivo é identificar a configuração do software em pontos discretos no tempo e sistematicamente controlar as modificações à configuração identificada com o objetivo de manter a integridade e rastreabilidade do software ao longo do seu ciclo de vida (OLIVEIRA et al., 2001).

Neste projeto de pesquisa define-se GCS como sendo uma área de conhecimento da Engenharia de Software cujo objetivo é estabelecer e manter a integridade dos resultados de um projeto de software, tanto intermediários como finais, ao longo do seu ciclo de vida dando suporte e auxilio as atividades dos demais processos de software e contribuindo para o desenvolvimento do projeto e sua posterior manutenção.

A finalidade do GCS é estabelecer e manter a integridade dos produtos do projeto de software ao longo do seu ciclo de vida. O GCS envolve (PAULK et al., 1993):

- Identificar a configuração de software (artefatos de software selecionados e sua descrição) em um dado momento;
- Controlar sistematicamente as mudanças na configuração;
- Manter, por meio da gerência, a integridade e rastreabilidade da configuração ao longo do ciclo de vida de software;
- Controlar a integridade de artefatos compostos, levando em conta a versão de cada um dos componentes, ou seja, controlar a configuração do artefato composto; e
- Registrar e relatar o estado do processo de alteração.

No contexto de GCS, a palavra configuração de software representa o conjunto de características físicas e funcionais de software conforme estabelecidas em algum documento técnico ou realizadas por um produto desenvolvido em um projeto de software (IEEE, 1990).

Esta configuração é composta por artefatos gerados durante a execução das atividades de desenvolvimento do projeto, definidas pelos processos de software adotados. Além dos artefatos, a configuração é composta de itens de configuração de software *(Configuration Items - CIs)* que são artefatos que num dado momento do projeto passaram a ter a sua evolução controlada, isto é, só poderão ser alterados após uma prévia autorização por parte de alguns agentes responsáveis pelo controle de configuração. Portanto, define-se o CI como sendo cada um dos elementos de informação que são criados durante o desenvolvimento de um produto de software, ou que para este desenvolvimento sejam necessários, que são identificados de maneira única e cuja evolução é passível de rastreamento (PRESSMAN, 2001).

Tanto os artefatos ainda não controlados quanto os CIs são armazenados numa estrutura própria para o projeto chamada de biblioteca de software do projeto. Assim sendo, define-se biblioteca de software como sendo uma coleção controlada de software e documentos a ele relacionados, que auxilia o desenvolvimento, uso e manutenção do software (IEEE, 1990), sendo também um instrumento utilizado para realizar atividades de distribuição e entrega do software (IEEE, 2004). As técnicas e métodos usados pelo GCS geralmente estão centrados no controle dessas bibliotecas (ANSI/IEEE, 1987) que podem ser categorizadas de acordo com o grau de maturidade e significado dos elementos nela armazenados (IEEE, 2004). Dentre os possíveis tipos de biblioteca, existem as de desenvolvimento, onde são armazenados os artefatos que fazem parte do desenvolvimento do projeto que ainda não são controlados, e as de produção, onde são armazenados os CIs.

Os CIs compõem, em pontos de controle definidos durante o ciclo de vida do projeto, *baselines* que determinam uma versão do software desenvolvido no projeto (IEEE, 1990) naquele momento. Uma *baseline* tem como objetivo servir de referência para as próximas atividades do desenvolvimento do projeto sendo, no entanto, apenas um conceito lógico composto pelos CIs do projeto. Além disso, a *baseline* pode ser entregue ao cliente como sendo um resultado do projeto e neste caso é chamado de distribuição. Existem dois tipos de distribuições definidos para projetos (OLIVEIRA et al., 2001):

- **Distribuição Regular:** é uma distribuição planejada do projeto. Após o término do projeto, é criada uma distribuição regular do que foi desenvolvido. Posteriormente são definidas novas datas para outras entregas regulares de novas versões que venham a incorporar as modificações realizadas sobre os CIs da baseline; ou
- **Distribuição Excepcional:** é uma distribuição não planejada do projeto. Ocorre quando é

necessário corrigir um erro urgente que não pode esperar pela próxima distribuição regular.

Também é tido como distribuição o uso de CIs fora do contexto do projeto onde eles foram desenvolvidos (IEEE, 1990). Citam-se como exemplo para este caso os componentes que são construídos em um projeto de software e reutilizados em outro.

Com base nestes conceitos, o GCS define atividades cuja realização eleva a confiança e a qualidade do software, provendo (IEEE, 1998):

- Uma estrutura para identificação e controle dos CIs: documentação, código, interfaces, bancos de dados, entre outros; e
- Informações de gerenciamento do projeto e do produto preocupando-se com o controle das *baselines*, das mudanças, dos testes, das distribuições, das auditorias, entre outras atividades de desenvolvimento de um projeto de software.

Para tanto, tem-se a execução de atividades dentro do processo GCS, citadas a seguir (IEEE, 1998)(IEEE, 2004):

- **Identificação de Configuração (*Configuration Identification*):** Esta atividade está relacionada a procedimentos para identificar *baselines* do projeto, CIs a serem controlados pelo GCS e a ferramentas e técnicas a serem utilizadas para aquisição e gerenciamento dos CIs. Além disso, nesta atividade são definidos esquemas para nomear cada CI de tal forma que eles sejam unicamente identificados na configuração do projeto. A Identificação de Configuração provê a base de sustentação para as demais atividades de GCS.
- **Controle de Configuração (*Configuration Control*):** Esta atividade está relacionada com o gerenciamento de mudanças realizadas durante o desenvolvimento do projeto e envolve procedimentos para requisitar mudanças no software, determinar que mudanças realizar, implementá-las e, posteriormente, aprová-las.
- **Administração de Estado (*Status Accounting*) :** Esta atividade está relacionada ao registro e posterior divulgação de informações relacionadas ao processo de GCS, buscando apoiar as atividades gerenciais do projeto e a aferição da execução das atividades definidas para o processo de GCS. A partir da Administração de Estado é possível, por exemplo, medir as demais atividades de GCS quanto à sua eficiência dentro do contexto dos processos de software adotados pela organização, dando suporte a decisões como a de realizar uma alteração sobre estas atividades. Isto permite que haja uma evolução quanto à maturidade organizacional no que diz respeito à implantação de GCS em seus projetos de software.

- **Auditoria da Configuração (*Configuration Auditing*):** Esta atividade está relacionada às auditorias realizadas sobre CIs e *baselines*, buscando verificar se há conformidade entre características físicas e funcionais do software (IEEE, 2004) representadas pela documentação do projeto e pelo produto de software. As auditorias podem ser tanto informais como formais. Auditorias informais podem ser realizados em pontos chave do ciclo de vida do projeto. Já as auditorias formais podem ser tanto no escopo físico como funcional. As auditorias funcionais são realizadas para garantir que os artefatos do projeto estão consistentes em relação às especificações. As auditorias físicas são realizadas para garantir que a documentação do projeto está consistente com o produto de software construído.

- **Gerenciamento e Entrega de Liberações (*Release Management and Delivery*):** Esta atividade está relacionada com as liberações tanto internas como para clientes, sendo aplicada, por exemplo, em casos onde se tem diferentes versões de uma dado CI disponíveis (para diferentes plataformas ou com características diferentes) para entrega. Para este caso, tem-se muitas vezes a necessidade de recriar versões específicas e empacotar o material correto para a entrega da versão. Portanto esta atividade envolve a construção do software, onde se tem a combinação de versões específicas dos itens de configuração para compor um programa executável, e o gerenciamento das liberações, onde é feita a identificação, empacotamento, e entrega dos elementos que compõem o produto como, por exemplo, o programa executável, a documentação, relatos da liberação e dados da configuração.

Os artefatos de software são armazenados numa estrutura própria para o projeto chamada de biblioteca de software (*Software Library*). As técnicas e métodos, referentes às atividades de GCS citadas, geralmente estão centrados no controle dessas bibliotecas (ANSI/IEEE, 1987) (Figura 1).

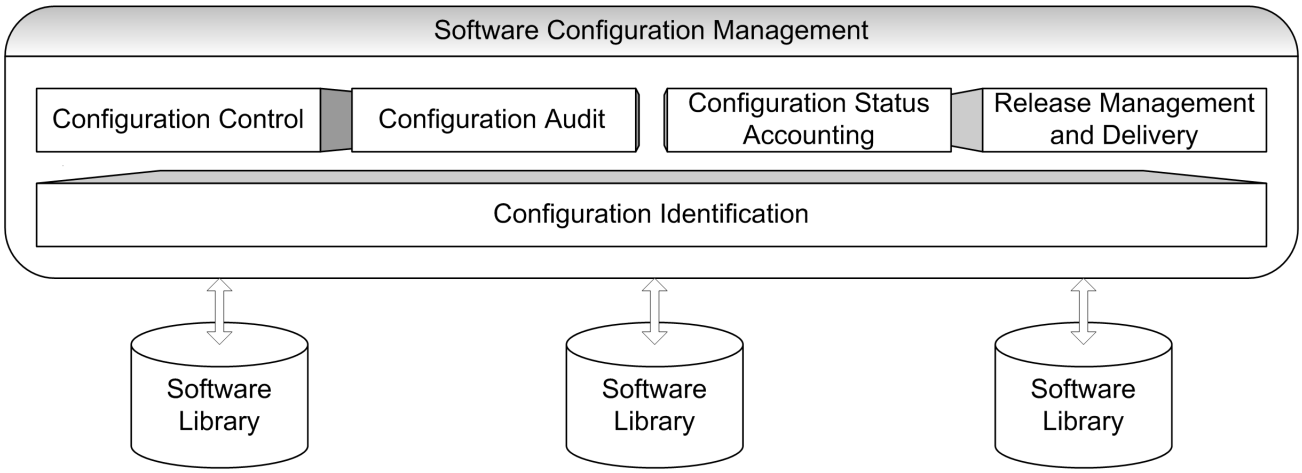

Figura 1: Atividades do Processo de GCS

2.1.2 Processos de Desenvolvimento de Software

Os processos de desenvolvimento de software englobam uma série de processos de software, como o de GCS, necessários para o desenvolvimento do projeto de software.

Mais recentemente um processo, o chamado Processo Unificado (Unified Process - UP) (JACOBSON; BOOCH; RUMBAUGH, 1999), foi proposto e obteve grande aceitação no mercado. Este processo é descendente de métodos anteriores definidos por seus autores e apresenta uma visão ortogonal das Fases (*Phases*) e Fluxos (*Flows*) do projeto, sendo que as fases representam divisões gerenciais do ciclo de vida do projeto e os fluxos representam processos de software executados no processo UP (Figura 2).

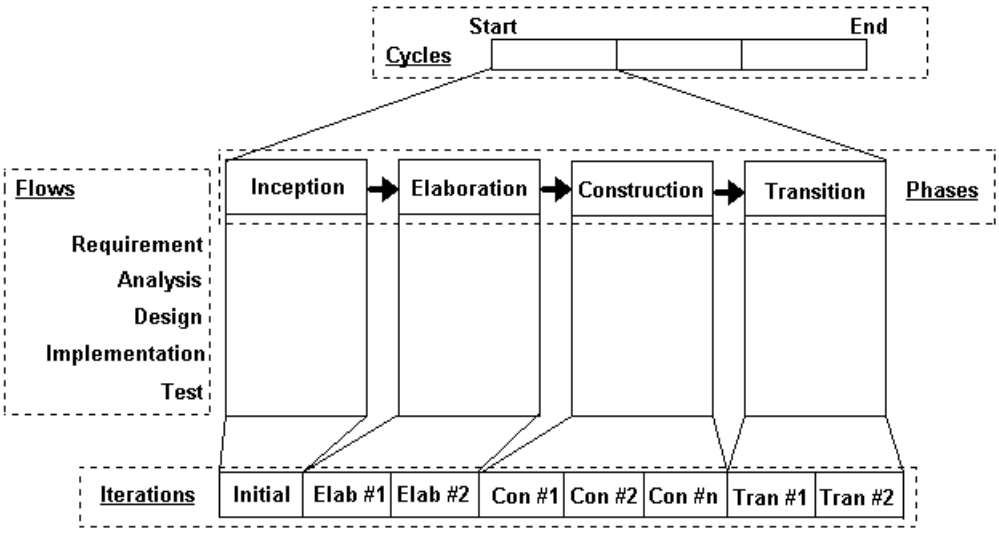

Figura 2: Processo UP

Os fluxos possuem uma série de atividades as quais são distribuídas, num nível gerencial mais genérico, dentre as fases (Tabela 1) e, mais especificamente, em iterações (*Iterations*) dentro destas fases.

Estes fluxos do processo UP, chamados fluxos técnicos (Tabela 2), apresentam uma série de atividades, sendo que cada uma dessas atividades prevê resultados que utilizam a UML como notação padrão.

A partir do processo UP foram propostos outros que detalharam os fluxos já existentes e ampliaram seu material incluindo novos fluxos e definindo novas atividades e artefatos (insumos e resultados). Dentre os processos descendentes do UP pode-se citar o RUP (*Rational Unified Process*) (KRUCHTEN, 2003), o EUP (*Enterprise Unified Process*) (AMBLER; NALBONE, 2004), e o PRAXIS (PRocesso para Aplicativos eXtensíveis InterativoS) (PÁDUA, 2003).

Tabela 1: Fases do Processo UP

Fase	**Descrição da Fase**
Concepção (*Inception*)	Fase na qual as necessidades dos usuários e os conceitos da aplicação são analisados o suficiente para justificar a especificação de um produto de software, resultando em uma proposta de especificação.
Elaboração (*Elaboration*)	Fase na qual a especificação do produto é detalhada o suficiente para modelar conceitualmente o domínio do problema, validar os requisitos em termos deste modelo conceitual e permitir um planejamento acurado da fase de construção.
Construção (*Construction*)	Fase na qual é desenvolvida (desenhada, implementada e testada) uma versão completamente operacional do produto que atende aos requisitos especificados.
Transição (*Transition*)	Fase na qual o produto é colocado à disposição de uma comunidade de usuários para testes finais, treinamento e uso inicial.

Tabela 2: Fluxos do Processo UP

Fluxo Técnico	**Objetivo do Fluxo**
Requisitos	Obter um conjunto de requisitos de um produto, acordado entre cliente e fornecedor.
Análise	Detalhar, estruturar e validar os requisitos, de forma que esses possam ser usados como base para o planejamento detalhado.
Projeto	Formular um modelo estrutural do produto que sirva de base para a implementação.
Implementação	Realizar o desenho em termos de componentes de código.
Testes	Verificar os resultados da Implementação.

O PRAXIS, por exemplo, possui, além dos cinco fluxos técnicos, três fluxos gerenciais, referentes a processos gerenciais, relacionados à Gestão de Projetos, Gestão da Qualidade e Engenharia de Sistemas, sendo que cada um deles possuem sub-fluxos como, por exemplo, o sub-fluxo de GCS do fluxo de Gestão de Qualidade o qual está relacionado ao processo de GCS. Já no RUP e EUP, o GCS é definido como um fluxo de suporte e gerenciamento, assim como os de Gerenciamento de Projeto e Ambiente.

O PRAXIS também apresenta uma série de iterações distribuídas entre as fases do processo, assim como consta na Tabela 3.

Assim como ocorre no processo UP, em cada iteração do PRAXIS são executadas atividades relacionadas aos seus fluxos. Essas atividades são obtidas a partir dos processos de software relacionados com cada fluxo do processo PRAXIS, sendo que dependendo da iteração pode-se ter um maior número de atividades de um dado fluxo e, em alguns casos, nenhuma atividade de um outro fluxo. A Tabela 4 mostra as atividades típicas, isto é, atividades normalmente executadas, da iteração de Levantamento de Requisitos do processo Práxis.

Tabela 3: Iterações do Processo PRAXIS

Fase	**Iteração**	**Descrição da Iteração**
Concepção	Levantamento de Requisitos	Levantamento das funções, interfaces e requisitos não funcionais desejados para o software.
Concepção	Análise de Requisitos	Modelagem conceitual dos elementos relevantes do domínio do problema e uso desse modelo para validação dos requisitos e planejamento detalhado da fase de Construção.
Construção	Desenho Implementável	Definições interna e externa dos componentes do software, em nível suficiente para decidir as principais questões de arquitetura e tecnologia e para permitir o planejamento detalhado das liberações.
Construção	Liberação 1	Implementação de um subconjunto de funções do produto que será avaliado pelos usuários.
Construção	Liberação N	Idem.
Construção	Testes Alfa	Realização dos testes de aceitação, no ambiente dos desenvolvedores, juntamente com elaboração da documentação de usuário e possíveis planos de Transição.
Transição	Testes Beta	Realização dos testes de aceitação no ambiente dos usuários.
Transição	Operação Piloto	Operação experimental do software em instalação piloto do cliente, com a resolução de eventuais problemas por meio do processo de manutenção.

Tabela 4: Atividades da Iteração de Levantamento de Requisitos do Processo Práxis

Fluxo	**Atividades Típicas**
Requisitos	Definição de requisitos; Detalhamento dos requisitos de interface (preliminar); Detalhamento dos requisitos funcionais (preliminar).
Análise	Identificação das classes (preliminar); Organização das classes (preliminar); Identificação dos relacionamentos (preliminar).
Projeto	Projeto Arquitetônico (preliminar).
Implementação	Prototipagem de fachada (preliminar).
Testes	-
Gestão de Projetos	Cadastramento de Requisitos (preliminar); Instanciação do processo; Estimativa de tamanho (preliminar); Estimativa do esforço (preliminar); Estimativa de recursos (preliminar); Estimativa de cronograma (preliminar); Estimativa de riscos (preliminar).
Gestão da Qualidade	Auditoria Informal

2.2 Qualidade de Processos de Software

Em paralelo à elaboração de processos de software, foram sendo definidos modelos e padrões, buscando contribuir com a melhoria da qualidade destes processos. Cita-se como exemplos os padrões ISO/IEC-12207 e ISO 15504, as normas da série ISO 9000 e os modelos SW-CMM (*Capability Maturity Model for Software*) e CMMi (*Capability Maturity Model Integration*). Contudo, o CMMi será descrito em maiores detalhes devido ao fato desta pesquisa ter se baseado neste modelo. Esta escolha se deve ao alto grau de aceitabilidade do modelo SW-CMM (modelo percursor do CMMi) em empresas de software, conforme pode ser visto em (MCT, 2002).

O padrão ISO/IEC-12207 (ABNT, 1998) provê uma arquitetura para o ciclo de vida do software e um *framework* para aquisição, fornecimento, desenvolvimento, operacionalização e manutenção do software. Este padrão agrupa os processos a serem executados durante o ciclo de vida do projeto em: Processos fundamentais (dentre eles, o de Desenvolvimento e Manutenção), Processos de apoio (dentre eles, os de Gerência de Configuração e Garantia da Qualidade), e Processos organizacionais (dentre eles, os de Gerência e Infra-Estrutura).

As normas da série ISO 9000 foram desenvolvidas para a aplicação em qualquer setor produtivo. Como exemplo, cita-se a ISO 9001, referente a sistemas de qualidade e que pode ser aplicada a qualquer empresa ou atividade. Contudo, a norma ISO 9000-3 foi elaborada para conter diretrizes para a aplicação da ISO 9001 ao projeto, desenvolvimento, fornecimento, instalação e manutenção de software.

O SW-CMM é um modelo de capacitação, criado pelo SEI (*Software Engineering Institute*), que visa a melhoria contínua dos processos de software por meio da aferição do nível de maturidade das organizações que estejam relacionadas com o desenvolvimento de projetos de software. Para tanto, o SW-CMM possui uma arquitetura em estágios com cinco níveis de maturidade (inicial, repetitivo, gerenciado, definido e otimizado), sendo que cada nível descreve a maturidade da organização em desenvolver software. Cada nível de maturidade contém áreas chave de processo (*Key Process Áreas* - KPAs) que estabelecem metas que devem ser atendidas pelas empresas de software. Uma empresa só poderá ser considerada madura segundo um dos níveis apresentados neste modelo caso ela cumpra todas as metas definidas pelas KPAs daquele nível e dos anteriores. Portanto, o SW-CMM pode ser definido como um guia que apresenta uma série de diretrizes a serem seguidas por organizações de desenvolvimento de software que desejam melhorar seu nível de maturidade por meio do controle e manutenção de seus processos de software. Atualmente este é o modelo mais difundido no mercado e foi utilizado como base para definição de atividades em vários processos como, por exemplo, o RUP, EUP e PRAXIS.

Além das normas ISO 9000, a ISO verificou a necessidade de desenvolver um padrão específico para o setor de software. Para tanto foi criado um projeto, denominado SPICE (*Software Process Improvement and Capability dEtermination*), cujo resultado foi um conjunto de documentos (Figura 3) que correspondem ao padrão ISO 15504. Este padrão foi elaborado com o intuito de harmonizar diversos modelos nos quais ela se baseia, como por exemplo, o SW-CMM.

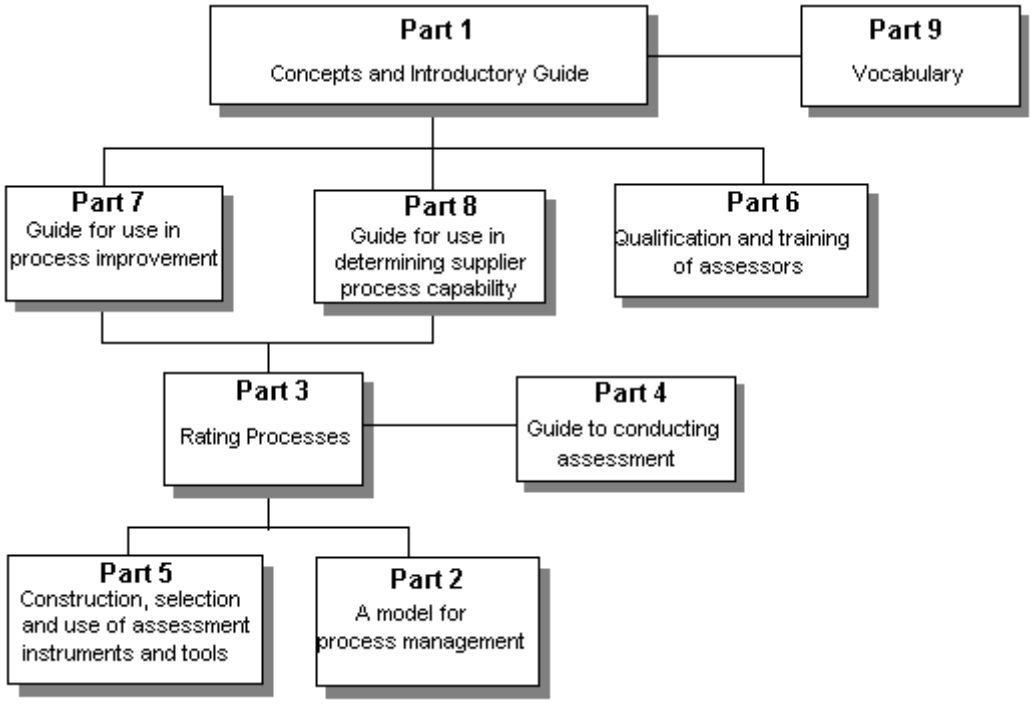

Figura 3: Norma SPICE

A ISO 15504 pode ser utilizada tanto para melhoria de processos como para a determinação das capacidades dos processos de uma organização. A parte referente a melhoria de processos foi criada com base no padrão ISO/IEC 12207. Já a parte referente a capacitação apresenta um arquitetura em estágios, assim como o SW-CMM, para verificar a maturidade dos processos. Contudo, a ISO 15504 possui algumas diferenças em relação ao SW-CMM. A primeira delas é: 1) a existência de um nível de maturidade adicional (o nível de maturidade inicial do SW-CMM corresponde a dois níveis de maturidade da ISO 15504); 2) a capacitação, aplicada para cada um dos processos de software do padrão e que possuem relação direta com os processos da ISO/IEC 12207, permitindo que se tenha uma maior flexibidade do modelo a medida que a empresa pode estar melhorando cada um dos seus processo de software isoladamente.

Por fim, o CMMi, também criado pelo SEI, é um modelo de capacitação recente que aborda, de forma integrada, todos os modelos de capacitação definidos por esta instituição. Este modelo, além de envolver processos de software (SW-CMM), envolve a engenharia de sistemas (SE-CMM), a definição de produtos (IPD-CMM) e a aquisição de software (SA-CMM). Portanto, o CMMi irá

substituir estes modelos, inclusive o SW-CMM.

De acordo com as diretrizes definidas pelo CMMi, o processo de desenvolvimento de software de uma organização pode ser avaliado segundo sua maturidade segundo dois tipos de arquitetura: contínua e em estágios. A arquitetura em estágios é similar a definida pelo SW-CMM, onde a maturidade organizacional é categorizada em níveis (Figura 4).

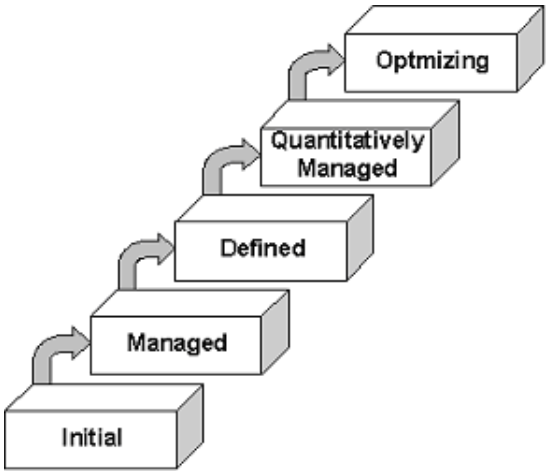

Figura 4: Níveis de Maturidade do CMMi

Seguem os níveis de maturidade definidos no modelo CMMi (SEI, 2002):

- **Nível Inicial (*Initial*):** indica que as organizações não provêem um ambiente estável para desenvolver e manter o software e normalmente enfrentam dificuldades em cumprir com os compromissos assumidos para o desenvolvimento de projetos de software, como custo e prazo. Para estas organizações, o sucesso do projeto acaba dependendo das chamadas forças caóticas (PÁDUA, 2003) dentro da organização: magia, gurus e heróis. O processo de desenvolvimento de software neste nível de maturidade é dito como caótico.

- **Nível Gerenciado (*Managed*):** indica que a organização já possui alguma maturidade no desenvolvimento de projetos de software. Existem políticas e regras definidas para o projeto, além de abordagens de implantação para cada um dos processos utilizados, sendo que o planejamento e gerenciamento de projetos se baseiam no sucesso de outros projetos já desenvolvidos. Portanto, os projetos de organizações deste nível são gerenciados e os processos são planejados, realizados e controlados.

- **Nível Definido (*Defined*):** indica que o processo de software da organização é documentado, incluindo tanto informações relacionadas à Engenharia de Software quanto ao gerenciamento de processos. Além disso, os processos são conhecidos e descritos em padrões,

procedimentos, ferramentas e métodos. Os processos de software, neste nível de maturidade, são ditos como padronizados e consistentes devido ao fato de tanto as atividades da engenharia como do gerenciamento de software serem estáveis e repetitivas.

- **Nível Gerenciado Quantitativamente (*Quantitatively Managed*):** indica que a organização já alcança objetivos de qualidade quantitativos tanto para produtos como para processos de software. A produtividade e qualidade, portanto, são medidas e, para tal, um banco de dados organizacional com informações dos processos é utilizado. Portanto, os processos de software, neste nível de maturidade, são ditos como quantitativos e previsíveis devido ao fato de serem medidos.

- **Nível Otimizado (*Optimized*):** indica que a organização está focada na contínua evolução dos processos de software com base nas informações quantitativas. Para tal, a organização possui meios de identificar as falhas e possíveis melhorias nos processos de forma pró-ativa com o objetivo de eliminar erros e reduzir os riscos no desenvolvimento de projetos de software. Portanto, o processo de desenvolvimento de software neste nível de maturidade é dito como em melhoria contínua devido ao fato de sempre haver esforços para realizar a melhoria na forma como desenvolver os projetos de software na organização.

O CMMi é um modelo de capacitação recente e portanto ainda existem poucas empresas capacitadas com base em seus níveis de maturidade. Já o SW-CMM, que é o seu predecessor, está bem difundido no mercado sendo utilizado como ponto de referência por muitas organizações preocupadas com a qualidade no desenvolvimento de projetos de software.

Conforme visto em (MCT, 2002), grande parte das empresas de software ainda se encontra no nível inicial e, portanto seu processo pode ser caracterizado pela anarquia (já discutida por Yourdon em 1995 (YOURDON, 1995)): Existem padrões, entretanto, estes são normalmente ignorados. As ferramentas disponíveis são utilizadas esporadicamente. Além disso, os processos oficiais de software são praticados “informalmente”, ou seja, de acordo com as conveniências de cada Engenheiro de Software.

Para que uma organização alcance maiores níveis de maturidade, o CMMi estabelece que esta deve atender a áreas de processo (*Process Areas* - PAs) que definem objetivos a serem atingidos.

Com exceção do nível inicial, todos os demais níveis apresentam uma série de PAs. O nível gerencial, por exemplo, contém as PAs de: Gerenciamento de Requisitos, Planejamento de Projeto, Monitoração e Controle de Projeto, Gerenciamento de Fornecedores, Medição e Análise, Garantia de Qualidade do Processo e Produto, e Gerenciamento de Configuração.

Assim como definido no modelo SW-CMM, para que uma organização possa ser considerada

madura segundo um dos níveis apresentados no CMMi, esta deve cumprir todas as metas definidas para as áreas chave de processo daquele nível e dos anteriores. Estas metas prevêem uma série de práticas para nortear os esforços para a sua aplicação no processo de desenvolvimento de software da organização.

Conforme citado anteriormente, uma das PAs definidas no segundo nível de maturidade do CMMi é a de Gerenciamento de Configuração, cuja finalidade é estabelecer e manter a integridade dos produtos do projeto de software ao longo do seu ciclo de vida. Esta PA, que está relacionada diretamente com o processo de GCS, define três metas específicas a constar:

- **Estabelecer *Baselines*:** *Baselines* dos artefatos identificados são estabelecidos;
- **Rastrear e Controlar Mudanças:** Mudanças nos CIs são rastreadas e controladas;
- **Estabelecer Integridade:** Integridade entre as *baselines* é estabelecida e mantida

Cada uma dessas metas prevê práticas (Tabela 6) que descrevem a infra-estrutura necessária e as atividades que deverão ser desempenhadas pela organização para atender à PA de GCS.

Tabela 5: Práticas definidas para as metas da PA de GCS do CMMi

Meta	**Práticas**
Estabelecer Baselines	Identificar CIs; Estabelecer um Sistema de Gerenciamento de Configuração; e Criar ou distribuir Baselines.
Rastrear e Controlar Mudanças	Rastrear as Requisições de Mudança; e Controlar os CIs.
Estabelecer Integridade	Estabelecer Registros do Gerenciamento de Configuração; e Realizar Auditorias sobre a Configuração.

Além das metas específicas, o CMMi define para a PA de GCS uma meta genérica que diz respeito a práticas de institucionalização dessa PA na organização. A institucionalização tem como objetivo tornar o processo, definido pela PA, em algo intrínseco à cultura organizacional, isto é, um hábito para equipes de desenvolvimento (PÁDUA, 2003) garantindo que, mesmo em momentos de crise organizacional ou de um dado projeto que esta sendo desenvolvido, o processo esteja sendo executado.

2.3 Implantação de Processos de Software

O sucesso do projeto de software depende de vários fatores. Um desses fatores é a adoção de processos de software como, por exemplo, o de GCS. Porém, a adoção desses processos é uma

tarefa complexa que muitas vezes acaba por inviabilizar as empresas em sair do diagnóstico, onde se observa a necessidade em implantar o processo, para chegar no operacional, onde se tem a execução destes processos nos projetos de software.

Isto se deve muitas vezes ao fato de se acreditar que a implantação de processos em uma empresa de software, como o caso da implantação do processo de GCS, ocorre simplesmente pela adoção de ferramentas. No entanto, a maior parte dos custos envolvidos nesta implantação relacionam-se a gastos (envolvendo novas contratações, treinamentos, entre outros) com recursos humanos e na estruturação organizacional.

O processo de GCS, especificamente, apóia o desenvolvimento do software, estando relacionado a diversas atividades desempenhadas durante este desenvolvimento. Portanto, implantá-lo implica em afetar vários setores da organização. Além disso, os processos de GCS acabam por acarretar numa maior lentidão do processo de desenvolvimento de software, incorporando práticas muitas vezes vistas como burocráticas. E por esses motivos, seu sucesso acaba dependendo mais de aspectos culturais do que dos técnicos, gerenciais e organizacionais. Os seguintes aspectos estão envolvidos na implantação de GCS (OLIVEIRA et al., 2001):

- **Aspectos Gerenciais:** Como planejar o processo, acompanhar as modificações, estabelecer prioridades, cronogramas;
- **Aspectos Organizacionais:** Infra-estrutura da organização, autoridades e responsabilidades necessárias para a execução do processo;
- **Aspectos Culturais:** Como as pessoas lidam com a qualidade de software e as práticas de engenharia de software, qual a cultura existente na organização e qual a maneira mais adequada de se fazer mudanças em tal cultura; e
- **Aspectos Técnicos:** Qual ferramenta adotar, como configurá-la, entre outros.

Devido a estes aspectos e aos custos, muitas empresas acabam por não conseguir implantar o GCS, sendo que a maioria diz respeito a empresas de pequeno porte. Como o GCS é um processo que auxilia na qualidade do projeto, então este fato acaba prejudicando a economia relacionada ao desenvolvimento de software, pois muitas destas empresas não conseguem se firmar no mercado de informática.

Uma das formas de facilitar a implantação de um dado processo de software, é por meio da adoção de uma abordagem ou modelo. O SEI definiu um modelo de programas de melhoria que pode ser utilizado como mecanismo para realizar a implantação de processos de software. Este modelo, chamado IDEAL (MCFEELEY, 1996), propõe que um ciclo de melhoria de um processo

de desenvolvimento ocorre de acordo com cinco fases principais sendo que ao fim de cada ciclo, atinge-se um estágio de capacitação. As cinco fases maiores, cujas iniciais formam o acrônimo que deu nome ao modelo, são:

- **Inicio (*Initiating*):** lançar as bases para um programa bem sucedido;
- **Diagnóstico (*Diagnosing*):** determinar onde se está e aonde se quer chegar;
- **Estabelecimento (*Estabilishing*):** planejar os detalhes de como chegar ao destino;
- **Ação (*Acting*):** realizar os planos; e
- **Lição (*Learning*):** aprender com a experiência. Esta tradução foi utilizada para que a letra inicial de casa fase, cujo nome foi traduzido para o Português, forme também o mesmo acrônimo que, em Inglês, dá nome ao modelo.

Cada uma das fases do modelo IDEAL (Figura 5) define atividades (Tabela 6) a serem executadas para realizar a melhoria desejada.

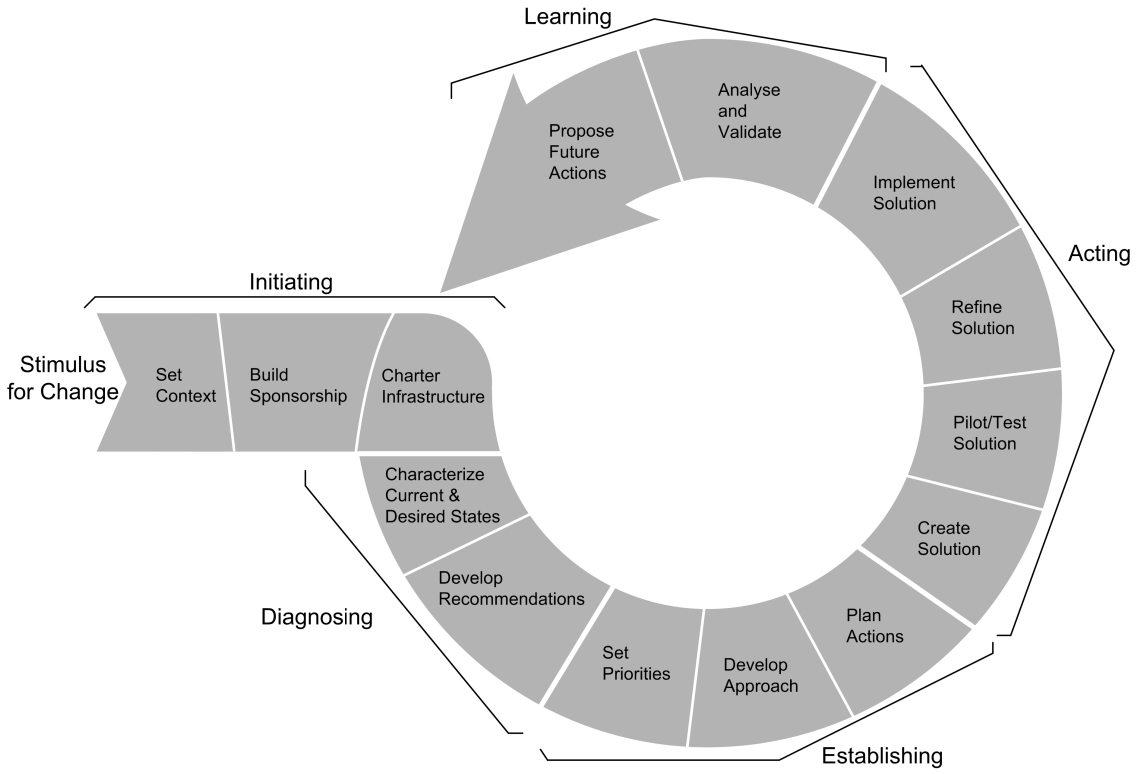

Figura 5: Modelo IDEAL (GREMBA; MYERS, 1997)

A aplicação do processo de GCS pode ser dividida em duas etapas: planejamento, onde se tem a elaboração do plano de GCS; e implementação, onde se tem a execução desse plano no projeto (ANSI/IEEE, 1987). Dessa forma, a implantação efetiva do GCS em um projeto ou organização

Tabela 6: Atividades definidas pelo Modelo IDEAL (PÁDUA, 2003)

Fase	**Nome da Atividade**	**Descrição da Atividade**
Início	Motivação	Motivar a organização para mudar os processos.
Início	Patrocínio	Obter patrocinadores para a mudança.
Início	Infra-Estrutura	Criar a infra-estrutura mínima necessária, principalmente um grupo responsável pela melhoria no processo.
Diagnóstico	Aferição	Caracterizar o estado atual e desejado dos processos.
Diagnóstico	Recomendações	Desenvolver recomendações para resolver os problemas encontrados pela aferição.
Estabelecimento	Priorização	Estabelecer prioridades para os esforços de mudança, considerando a realidade da organização.
Estabelecimento	Abordagem	Definir os passos para a realização da mudança.
Estabelecimento	Planejamento	Elaborar o plano detalhado de realização considerando prazos, custos, riscos, pontos de controle, responsabilidades e outros elementos.
Ação	Criação da Solução	Reunir os elementos da solução, como padrões, modelos, ferramentas e treinamento.
Ação	Testes-piloto	Testar a solução em um ou mais projetos-piloto.
Ação	Refinamento	Aperfeiçoar a solução para refletir o que foi aprendido com os testes-piloto.
Ação	Implantação Completa	Transferir a solução para o restante dos projetos da organização.
Lições	Análise e Validação	Coletar, analisar e documentar as lições aprendidas.
Lições	Proposição de ações futuras	Desenvolver recomendações para o próximo ciclo de melhoria de processos.

inicia-se com a elaboração de um plano de GCS (IEEE, 1998), que é uma das práticas definidas para a PA de GCS do CMMi.

Tem-se como objetivo básico, com o plano de GCS, delinear como o processo será aplicado por meio da definição de uma série de aspectos relacionados à gerência, atividades de GCS, implantação e recursos necessários para sua realização.

Segundo (PÁDUA, 2003), a elaboração do plano de GCS está diretamente relacionada ao porte e tipo do projeto em que ele será aplicado.

Em relação ao porte, Pádua afirma que para projetos de grande porte, as atividades de GCS devem ser planejadas com detalhe e estar bem definidas no plano específico para o projeto. Já para projetos de pequeno e médio porte, boa parte das informações contidas no plano de GCS não varia. Por este motivo, é comum definir, para pequenas e médias empresas, um plano de GCS para

reger todos os projetos por ela desenvolvidos. Isso ocorre, pois normalmente estes projetos são de pequeno e médio porte.

Já em relação ao tipo do projeto, a IEEE descreve (ANSI/IEEE, 1987) quatro exemplos de planos de GCS. Cada exemplo está associado a um tipo de projeto, que pode ser um projeto crítico para sistemas embarcados, um projeto pequeno; um projeto somente de manutenção ou um projeto de sistema de linha de produção.

De qualquer forma, o fato de o plano de GCS ter sido elaborado é um demonstrativo de que houve a preocupação com o controle da evolução do que foi desenvolvido no projeto e, conseqüentemente, com a qualidade. Uma vez definido, o plano deve ser implantado e seguido pela equipe de desenvolvimento podendo evoluir durante o projeto.

Para elaborar o plano de GCS, referente ao planejamento deste processo, deve-se definir qual será a sua estrutura e como preenchê-la de tal forma que se atenda aos pontos de GCS. Segue o exemplo de uma estrutura definida para o plano de GCS segundo o padrão do IEEE (IEEE, 1998):

1. Introdução
2. Gerência
   2.1. Organização
   2.2. Responsabilidades de GCS
   2.3. Políticas, Diretivas e Procedimentos Aplicáveis
3. Atividades de GCS
   3.1. Identificação de Configuração
   3.2. Controle de Configuração
   3.3. Administração de Estado
   3.4. Auditorias e Revisões de Configuração
   3.5. Controle de Interfaces
   3.6. Controle de Fornecedores e Subcontratados
4. Implantação
5. Recursos
6. Manutenção do Plano

Uma vez preenchido, o plano de GCS define todos os pontos deste processo para um dado projeto ou empresa, garantindo uma maior formalidade e maturidade organizacional referente à área de conhecimento de GCS.

Dois aspectos do processo presentes na estrutura do plano de GCS, segundo o padrão IEEE

(IEEE, 1998), são as atividades que devem ser realizadas (item “Atividades de GCS”) e as responsabilidades atribuídas a agentes para que estes executam essas atividades (item “Responsabilidades de GCS”) e que corresponde à estrutura organizacional para a execução do plano no projeto. Esses agentes podem ser tanto uma entidade dentro da organização como, por exemplo, uma pessoa ou um departamento, ou entidades externas presentes em outras organizações prestadoras de serviços.

Na NASA (NASA, 1995), dentre as responsabilidades presentes no GCS, existem as associadas ao responsável pelo gerenciamento da configuração (*Configuration Management Officer - CMO*), ao comitê de controle de configuração (*Configuration Control Board - CCB*) e ao bibliotecário (*Librarian*).

O CMO é responsável pelos aspectos operacionais para controle do processo de GCS e suas atividades, isto é, responsável por coordenar a troca de informações entre os agentes envolvidos com o GCS e realizar o registro dessas informações.

O CCB possui responsabilidades que lhes conferem a autoridade tanto para aceitar quanto para rejeitar propostas de mudanças sobre a configuração do projeto.

Já o bibliotecário é quem tem sob custódia e controle a biblioteca de software do projeto e os artefatos nela armazenados.

O levantamento completo das responsabilidades definidas no plano, no entanto, depende da abordagem para o processo de GCS definida para a organização.

2.4 Abordagens para o processo de GCS

Tanto os conceitos, definidos no documento padrão da IEEE (IEEE, 1998) que descreve o plano de GCS, quanto a PA deste processo do CMMi, definem o que deve ser feito e não como. Isto se deve ao fato de cada organização ter necessidades diferentes no que diz respeito ao GCS.

Por isso, deve-se definir uma abordagem que descreva como realizar as atividades de GCS com base nos conceitos deste processo e nas características dos projetos de software os quais elas serão aplicadas.

A seguir são apresentadas algumas das abordagens de GCS existentes. Duas delas foram elaboradas para serem aplicadas em projetos de software específicos e, portanto, definem apenas alguns aspectos do processo. As outras duas são partes integrantes de processos de software aplicados a diferentes organizações e, portanto, não definem especificamente como realizar cada uma das atividades de GCS no desenvolvimento do projeto.

2.4.1 Abordagem INPE de GCS

Esta abordagem (CUNHA, 1997) foi desenvolvida para ser aplicada em um projeto de desenvolvimento de software para controle de satélites do Instituto Nacional de Pesquisas Especiais.

Na abordagem, os conceitos para CIs são estendidos para elementos controlados. Elementos controlados são elementos produzidos durante o desenvolvimento de sistemas e que, num dado momento, passam a ser controlados segundo os procedimentos definidos para controle de configuração do GCS desta abordagem. Estes elementos são tanto produtos (unidades de software e documentos) quanto os itens internos aos produtos e os relacionamentos entre produtos, entre itens e entre produtos e itens. A Tabela 7 mostra tipos de Elementos Controlados para o Paradigma Orientado a Objetos.

Tabela 7: Exemplo de Elementos Controlados para o Paradigma Orientado a Objetos (ITAMI, 1997)

Produtos	**Itens Internos**	**Relacionamentos**
Classes	Métodos	Subclasse_de Herda_de
Componente Domínio do Problema	Dispositivos de Armazenamento	Instancia_de
Componente de Gestão de Tarefas	Tarefas	Parte-de
Componente Interface Humana	Telas	Cliente_de
Componente de Gestão de Dados		Define
Camada Assunto		Implementa
Camada Classe-Objeto		Usa
Camada Estrutura		Pertence
Camada Atributo		Contém
Camada Serviço		Testa
Doc. de Especificação de Requisitos		
Plano de Integração e Teste		
Manual do Usuário		
Manual de Instalação		
Relatório de Testes		

As responsabilidades de GCS estão relacionadas a um conjunto de ambientes que compõem o ambiente de desenvolvimento definido para o projeto de software. Cada um destes ambientes possui responsabilidades para com o ambiente de Controle de Configuração definido na arquitetura e são compostas por pessoas que integram a equipe de desenvolvimento do projeto. Os ambientes de desenvolvimento são (ITAMI, 1997): Ambiente de Sistema; Ambiente de Controle de Configuração; Ambiente de Controle de Qualidade; Ambiente de Subsistemas; Ambiente de Integração de Subsistema; Ambiente de Integração de Sistema e Ambiente de Prototipação.

Em relação às atividades, esta abordagem define aquelas para Encaminhamento e Alteração de Elementos Controlados:

- Identificação de Configuração: são definidas duas atividades principais:
  - Definição dos Elementos Controlados: É definida a tabela (Tabela 7, por exemplo) de elementos que serão controlados no projeto.
  - Encaminhamento de Elementos para Controle: Uma vez concluído o desenvolvimento de um dado elemento, este é entregue para controle para que possa ser analisado quanto à qualidade e padronização e, se aprovado, possa ser tido como elemento controlado. Este procedimento é dividido em: Entrega de Elementos; Recebimento do FEP (Formulário de Entrega de Produtos); Análise dos Produtos; e Manipulação dos Produtos.
- Controle de Configuração: é definida uma atividade para o controle de configuração:
  - Alteração de Elementos Controlados: Uma vez definido um elemento controlado, este só poderá ser alterado com aprovação prévia do ambiente de controle de configuração. Este procedimento é dividido em: Comunicação de Problema/Modificação; Avaliação do Formulário de Comunicação de Problema; Gerenciamento das Autorizações de Modificação de Produtos; Alteração de Elementos; e Controle de Elementos Controlados.

2.4.2 Abordagem ITI de GCS

Esta abordagem foi desenvolvida pelo ITI (Instituto nacional de Tecnologia da Informação) (OLIVEIRA et al., 2001) para ser aplicada numa pequena empresa para que esta pudesse realizar o GCS de um produto.

As responsabilidades aqui definidas são associadas a agentes e não ambientes como no caso da abordagem do INPE. As responsabilidades são:

- **Responsável pelo GCS:** encarregado pela operação geral do processo;
- **Conselho de Controle de Modificações:** responsável pela aprovação ou não de mudanças propostas;
- **Equipe técnica:** responsável pela implementação das modificações aprovadas pelo Conselho de Controle de Modificações;
- **Responsável pela Verificação:** encarregado de verificar se as modificações foram implementadas apropriadamente por meio da inspeção de documentos e testes funcionais.

Em relação às atividades de GCS, esta abordagem define um processo para Controle de Configuração que estabelece os passos que devem ser executados para realizar a modificação nos CIs.

Este processo é constituído basicamente por dois procedimentos: Requisição de Modificação de Software; e Autorização de Modificação de Software. Cada um destes procedimentos compõe o processo definindo um conjunto de passos que englobam: a) Proposta de Modificação; b) Solicitações de Modificação, com base em uma ou mais propostas; c) Descrição do Projeto de Modificação, que descreve como será realizada a modificação; e d) Descrição da Modificação, que descreve a modificação realizada para cada módulo alterado do projeto.

2.4.3 Abordagem PRAXIS de GCS

O processo PRAXIS (PÁDUA, 2003) define o GCS como sendo um sub-fluxo do fluxo gerencial de Gestão da Qualidade. Além disso, este sub-fluxo está relacionado à KPA de GCS do SW-CMM e, portanto, busca atender às metas definidas nesta KPA. Os agentes definidos nesta abordagem são:

- **Grupo de Gestão de Configuração:** centraliza o funcionamento dos mecanismos dessa disciplina, isto é, coordena a execução das atividades de GCS;
- **Comissão de Controle de Configuração:** toma decisões importantes relativas ao GCS; e
- **Gerente de Projeto:** responsável por cobrar dos desenvolvedores a disciplina de GCS no dia-a-dia do projeto.

Devido ao fato de este ser um processo de software, a abordagem apenas descreve como definir as atividades de GCS com base nos agentes citados. Assim sendo, os agentes executam três atividades principais do fluxo. O gerente do projeto descreve as atividades de GCS por meio da atividade de Planejamento, gerando o plano de GCS que, no caso deste processo, é uma seção relacionada ao Plano da Qualidade de Software.

Portanto, as atividades relacionadas com a Identificação, Controle e Auditoria de configuração, e também com a Administração de Estado, são definidas na atividade “Planejamento”, da abordagem, que é executada pelo Gerente do Projeto.

De acordo com o plano de GCS, a equipe de desenvolvimento envia os artefatos que irão compor uma determinada *baseline* do projeto ao Grupo de Gestão de Configuração que a insere em uma biblioteca oficial de configurações conforme definido pela atividade de **Gestão de *Baselines***. Atividade esta que abrange outros procedimentos de manipulação das *baselines*, como, por exemplo, a verificação, a alteração e a extração de software. As bibliotecas, como um todo, são também administradas pelo Grupo de Gestão de Configuração por meio da atividade de **Gestão de Bibli-**

otecas de Configurações, que envolve a verificação, *backup*, análise e auditorias dos artefatos armazenados na biblioteca de software.

2.4.4 Abordagem RUP de GCS

O RUP (KRUCHTEN, 2003) define o GCS como sendo um dos fluxos executados no decorrer do desenvolvimento do projeto, assim como ocorre no caso dos fluxos do processo UP. Este fluxo, no entanto, recebe o nome de Gerenciamento de Configuração e Mudança e envolve uma série de processos:

- Planejar Configuração do Projeto e Controle de Mudança;
- Criar ambientes para o GCS do projeto;
- Alterar e liberar CIs;
- Gerenciar *Baselines* e Releases;
- Monitorar e Relatar Status de Configuração; e
- Gerenciar Solicitações de Mudança.

Cada um destes processos possui atividades que são desempenhadas por agentes, definindo assim as responsabilidades de cada um deles. Estes agentes são:

- **Gerente do Projeto:** é aquele responsável pelas atividades de Gerenciamento de Projetos, como por exemplo, alocação de recursos e definição de prioridades;
- **Gerente de Configuração:** disponibiliza o ambiente e a infra-estrutura geral de GCS para a equipe de desenvolvimento.
- **Gerente do Controle de Mudança:** É quem supervisiona as atividades relacionadas ao Controle de Configuração.
- **Integrador:** É quem realiza a integração dos artefatos desenvolvidos pela equipe do projeto.

Em relação às atividades, muitas delas não são definidas pelo RUP e sim por membros do projeto por meio do plano de GCS. Seguem as considerações sobre as atividades de GCS definidas no fluxo:

- **Identificação de Configuração:** A definição dos CIs, baselines do projeto, e estrutura de armazenamento são definidos pelo Gerente de Configuração no momento em que é escrito o plano de GCS. Deste ponto em diante, caso os artefatos desenvolvidos sejam incorporados a uma dada baseline então estes passam a ser referenciados como CIs.

- **Controle de Configuração:** O gerente de controle de modificação define qual será o processo de controle de modificação a ser incorporado ao plano de GCS. Além disso, o processo define a Solicitação de Mudança como meio de requisitar alguma alteração sobre a configuração do projeto. Esta solicitação é avaliada pelo Gerente de Controle de Mudança e implementado por qualquer membro da equipe do projeto. Uma vez concluída a mudança, esta é testada e, se aprovada, é atualizada nas *baselines* do projeto.

- **Administração de Estado:** O Gerente de Configuração é responsável por relatar o estado da configuração e, portanto, gerar relatórios solicitados pelo Gerente do Projeto.

- **Auditoria de Configuração:** O Gerente de Configuração é responsável por realizar a auditoria de configuração que envolve: Determinar se uma *baseline* contém todos os artefatos necessários; e determinar se uma *baseline* atende aos requisitos. A auditoria é realizada em três passos que consistem em realizar a auditoria de configuração física e funcional, além de reportar as descobertas por meio de um Registro de Auditoria de Configuração.

2.5 Ferramentas de GCS

As ferramentas de GCS são softwares de apoio a este processo e são utilizadas com o objetivo de se reduzir o grau de complexidade na execução das atividades de GCS definidas no processo.

Neste contexto, à medida que aumenta o grau de complexidade e a magnitude de um projeto de software, o seu apoio por parte de uma ferramenta de GCS passa a ser cada vez mais essencial para o sucesso do projeto (IEEE, 2004). Isto ocorre, pois, nestes projetos há uma quantidade muito grande de dados, o que exige uma maior atenção sobre as evoluções realizadas nos CIs. Além disso, ao se construir um produto a partir dos CIs do projeto, um único erro pode levar ao funcionamento incorreto do software (SOMMERVILLE, 2003).

As ferramentas de GCS podem ser categorizadas de acordo com o apoio oferecido tanto à execução do processo de GCS como também a sua integração com outras ferramentas CASE. No que diz respeito ao apoio a execução do processo de GCS, as ferramentas podem ser categorizadas como (SOMMERVILLE, 2003):

1. **Ferramentas de Apoio ao Gerenciamento de Configuração:** São ferramentas que controlam um repositório onde são armazenados os arquivos do projeto e, com base neste repositório, oferecem o controle de versão sobre os arquivos criados no projeto. A cada arquivo presente no repositório é associada uma versão. Estas ferramentas oferecem recursos como, por exemplo, identificação de versões dos arquivos do projeto, gerenciamento de armazenamento que permite o controle sobre o acesso dos arquivos armazenados, registro de histórico de mudanças sobre estes arquivos, obtenção de versões antigas, entre outros;

2. **Ferramentas de Apoio ao Gerenciamento de Mudanças:** São ferramentas que oferecem apoio à atividade de Controle de Configuração, fornecendo recursos como, por exemplo, um editor de formulários para a realizar uma solicitação de mudança, um sistema de fluxo de trabalho que permite que se tenha o controle sobre o ciclo de vida de uma dada requisição, um banco de dados de mudanças para armazenar todas as solicitações de mudança, entre outros; e

3. **Ferramentas de Apoio à Construção de Sistemas:** São ferramentas para realizar a construção de sistemas com base nos arquivos do projeto evitando erros que podem vir a ocorrer principalmente quando há uma grande quantidade de arquivos envolvidos na construção do produto de software. Estas ferramentas oferecem recursos como, por exemplo, uma linguagem de especificação de dependência para definir as dependências entre arquivos do projeto para auxiliar a compilação do sistema, seleção de ferramentas para executar a compilação sobre os arquivos, compilação distribuída para possibilitar a compilação em diferentes localidades, entre outros.

Dentre as ferramentas existentes no mercado, a sua grande maioria oferece apoio ao gerenciamento de configuração e construção de sistemas. Isto ocorre pois os serviços relacionados ao controle de versão e construção do sistema são os mais adotados em empresas que desenvolvem software. Já as ferramentas de apoio ao gerenciamento de mudanças normalmente são desenvolvidas dentro das empresas, pois dependem da abordagem de GCS adotada pela empresa. Na tabela 2.5 são mostradas algumas das ferramentas disponíveis no mercado, citando suas principais funcionalidades e categorias, de acordo com as citadas anteriormente.

Das ferramentas de GCS, o CVS[1] é o mais amplamente utilizado. Praticamente todas as comunidades de software livre, como “Sourceforge[2]”, “Java.net[3]”, “Tigris[4]”, e os projetos abertos,

[1] `http://cvshome.org`, Consultado em Março/2004
[2] `http://sourceforge.net`, Consultado em Outubro/2004
[3] `http://www.java.net/`, Consultado em Dezembro/2004
[4] `http://www.tigris.org/`, Consultado em Janeiro/2005

como “Netbeans[5]”, mantêm seus artefatos organizados em repositórios CVS. As empresas também o utilizam, principalmente por ser uma ferramenta simples, aberta e já consolidada. Também por sua popularidade, é grande a quantidade de APIs (*Application Programming Interfaces*) e outros softwares de suporte ao CVS. Na tabela 2.5 são mostradas algumas das ferramentas disponíveis no mercado. As categorias da ferramenta são relacionadas com as definidas por (SOMMERVILLE, 2003) e citadas anteriormente, sendo que os números referentes a estas categorias são: 1) Ferramentas de Apoio ao Gerenciamento de Configuração; 2) Ferramentas de Apoio ao Gerenciamento de Mudanças; e 3) Ferramentas de Apoio à Construção de Sistemas.

Tabela 8: Ferramentas de GCS

Ferramenta	**Empresa**	**Categoria**	**Funcionalidades**
ClearCase [6]	IBM	1	Realiza o controle de versões de artefatos, permitindo que se tenha a combinação e efetivação de mudanças, a checagem de diferenças entre versões, além de auxiliar no desenvolvimento paralelo e colaborativo. Pode ser integrada com a *Rational ClearQuest*, uma ferramenta de controle de mudanças e defeitos.
CVS	Projeto Open Source	1	Software livre de controle de versões de artefatos dando suporte à combinação e efetivação de mudanças, e permite a checagem de diferenças entre as versões. É amplamente utilizado e conhecido, principalmente por se tratar de um software livre.
BitKeeper Source Management [7]	BitMover, Inc	1	Comercializado pela BitMover, Inc., esta ferramenta faz controle de mudanças, dando suporte à visualização de mudanças, detecção e resolução de conflitos de mudanças, e combinação de mudanças em um arquivo.
Visual SourceSafe [8]	Microsoft Corporation	1,2	Sistema de controle de versão que trabalha junto com outro produto da *Microsoft*, o Visual Studio .NET. Controla arquivos-fonte que podem ser classificados como valiosos, controla o uso concorrente de arquivos, põe etiquetas em arquivos para controle de versão, controla mudanças em nível de linhas de código, entre outros.
AllChange [9]	Intasoft	1,2	AllChange é um sistema integrado de suporte ao Gerenciamento de Configuração. As principais funcionalidades são: Rastreabilidade de mudanças; segurança de acesso a artefatos controlados; trabalho cooperativo; e disponibilização de informação.
ChangeMan [10]	Serena	1, 2, 3	ChangeMan gerencia alterações paralelas em todas as aplicações e oferece recursos como o gerenciamento de configuração de sistemas distribuídos e construção automática da aplicação e implantação.

[5] `http://www.netbeans.org/`, Consultado em Julho/2004

Muitas vezes as ferramentas de GCS são utilizadas em conjunto para dar suporte as atividades deste processo definidas por uma dada organização. Assim sendo, ocorre o fato de se utilizar uma ferramenta de GCS como suporte para outras ferramentas que, por meio do uso dos serviços oferecidos pela primeira, permitem o apoio a outras atividades de GCS não contempladas. Por exemplo, normalmente as ferramentas de apoio ao Gerenciamento de Configuração são utilizadas por outras ferramentas que oferecem apoio tanto ao Gerenciamento de Mudanças quanto à Construção de Sistemas.

Porém, mesmo assim muitas empresas ainda não conseguiram adotar as práticas previstas pelas atividades de GCS, realçando que somente a existência de uma ferramenta de GCS não garante o sucesso na implantação desta área de conhecimento. Além disso, muitas empresas precisam realizar o desenvolvimento de uma outra ferramenta que atenda as suas necessidades não contempladas pelas ferramentas adquiridas do mercado. Este fato acaba por acarretar em gastos que muitas vezes se apresentam como decisivos para a não implantação de GCS.

Buscando minimizar estes gastos, empresas devem seguir critérios para direcionar a tomada de decisão no que diz respeito à aquisição de ferramentas de GCS. Porém, estes critérios só poderão ser utilizados depois que a empresa tenha definido uma abordagem para implantação de GCS que atenda as características organizacionais e dos projetos por ela desenvolvidos. Esta abordagem (SCM *Approach*) irá definir os serviços de GCS (SCM *Services*) que deverão ser atendidos pelo conjunto de ferramentas de apoio ao GCS (SCM *Tools*) - Figura 6.

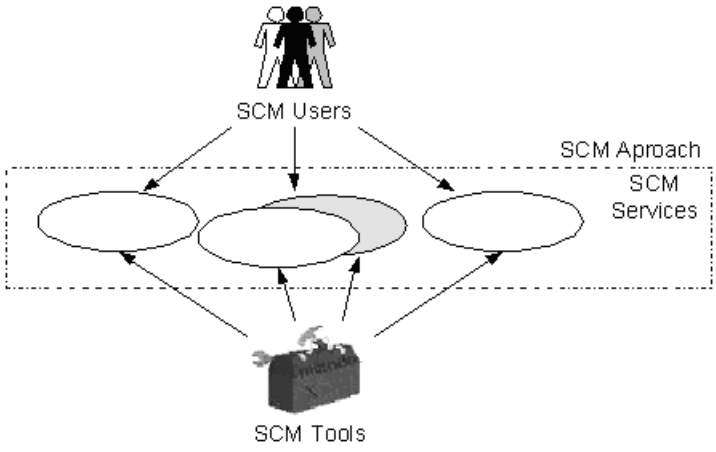

Figura 6: Ferramentas atendendo aos serviços de GCS (adaptado de (DART, 1992))

Seguem alguns critérios que podem ser adotados para se escolher ferramentas de GCS (BURROWS; GEORGE; DART, 1996):

[6] `http://www-306.ibm.com/software/awdtools/clearcase/index.html`, Consultado em Março/2004
[7] `http://www.bitkeeper.com/`, Consultado em Abril/2004
[8] `http://msdn.microsoft.com/ssafe/`, Consultado em Março/2004
[9] `http://www.intasoft.net/`, Consultado em Abril/2004
[10] `http://www.serena.com/Products/changeman/home.asp`, Consultado em Abril/2004

- **Suporte à Equipe de Desenvolvimento:** Análise das funcionalidades das ferramentas aplicadas ao desenvolvimento de projetos de uma equipe de programadores conectados a uma rede local. Deve ser observado o processo de gerenciamento das distintas áreas de trabalho de cada desenvolvedor, o processo de desenvolvimento paralelo, bem como a visualização gráfica do controle de versões aplicada a projetos que possuem várias ramificações; isto é, diferentes caminhos variantes de desenvolvimento;

- **Suporte ao Desenvolvimento Remoto:** Análise das funcionalidades aplicadas a projetos com diversos sítios de acesso remoto, realidade que se faz presente no trabalho de equipes que se distribuem em diferentes áreas geográficas, onde cada membro realiza uma atividade, e a integração do trabalho do grupo é feita continuamente;

- **Suporte a Configurações:** Observa-se a capacidade da ferramenta ligada à identificação de distintas versões de arquivos que compõem uma distribuição e recuperação da mesma. É analisado o impacto de uma dada alteração, por meio da estrutura hierárquica do projeto que o software deve permitir exibir; além dos aspectos de rastreabilidade, que tornam possível o estabelecimento de relações entre distintos CIs, e seus arquivos de histórico de modificações;

- **Suporte à Gerência de Mudanças:** A habilidade de prover o rastreamento de modificações, gerando relatórios de acompanhamento, de forma a permitir uma visão precisa, em tempo real, do estado das modificações;

- **Suporte à Construção e de Distribuição de Produto:** Análise das funcionalidades ligadas à geração de uma determinada versão de um produto e à recuperação de uma distribuição anterior e modificação da mesma;

- **Suporte de Processo:** Análise da capacidade de manter um controle efetivo do desenvolvimento, dando suporte à aplicação de políticas, implementando o conceito de ciclo de vida, estabelecendo autoridades de acesso a serviços de acordo com os papéis dos diversos integrantes do projeto e registrando os históricos e os estados dos diversos artefatos envolvidos no desenvolvimento sob controle da ferramenta;

- **Usabilidade:** Deve ser observado se a documentação se mostra acessível e prática, se há um sistema de auxílio on-line, qual é o nível de integração da ferramenta ao sistema operacional em uso e o grau de facilidade de acesso às informações de GCS controladas;

- **Facilidade de "set-up":** Análise da facilidade de a partir da sua aquisição, colocar a ferramenta em operação, englobando: a instalação, a parametrização da ferramenta, a capacitação de pessoal e a integração no processo de GCS previamente definido; e

- **Personalização:** Avaliação da capacidade de personalização da ferramenta, de modo a atender as necessidades específicas de um usuário no que diz respeito a aspectos de modelos de formulários e relatórios, políticas específicas de desenvolvimento, dentre outros.

Outros critérios ainda podem ser utilizados para selecionar a ferramenta de GCS mais adequada para o desenvolvimento do projeto. Estes podem ser baseados nas funcionalidades (DART, 1992) que podem ser oferecidas por uma ferramenta de GCS. São elas:

- **Componentes:** Identificar, classificar armazenar e acessar os componentes que compõem o produto desenvolvido no projeto;
- **Estrutura:** Possibilitar a representação da arquitetura do produto;
- **Construção:** Dar suporte à construção dos produtos e seus artefatos;
- **Auditoria:** Possibilitar a realização de auditorias sobre o produto e processo de GCS;
- **Informativos:** Disponibilizar estatísticas sobre o produto e o processo;
- **Controle:** Controlar como e quando mudanças são realizadas;
- **Processo:** Dar suporte ao gerenciamento de como o produto evolui; e
- **Equipe:** Possibilitar a definição de equipes de desenvolvimento para o projeto.

Estes critérios devem ser utilizados em conjunto com uma metodologia como a apresentada por Cassidy (CASSIDY, 1998). Segundo esta metodologia, deve-se ter a definição de uma lista de vendedores dessas ferramentas (três ou quatro vendedores) que irão realizar demonstrações de suas ferramentas. Adicionalmente, deve-se ter uma equipe de profissionais da empresa que será responsável pela seleção das ferramentas de GCS. A partir disso, a empresa interessada pela seleção das ferramentas deve:

- Elaborar um *script* das informações de seu interesse e como elas devem ser exibidas pelos vendedores;
- Definir o pacote de dados com informações que deverão ser utilizadas pelas ferramentas na demonstração;
- Elaborar uma lista de requisitos chave desejados e que poderão ser utilizados para avaliar as ferramentas demonstradas pelos vendedores. Para tanto, pode-se utilizar como base os critérios de seleção já citados;

- Decidir a metodologia de classificação (*ranking*) a ser utilizada pelas pessoas da empresa que irão participar da demonstração das ferramentas, podendo ser aplicada para cada item do *script* ou apenas para os requisitos chave. Pode-se, por exemplo, atribuir pesos para cada item de acordo com o participante que irá fazer a sua avaliação, sendo que o peso maior deve ser dado para aquele participante que possuir maior conhecimento sobre o item;
- Elaborar um guia de demonstração a ser utilizado pelo vendedor;
- Elaborar um guia de demonstração para a equipe responsável pela seleção das ferramentas, preparando os participantes de modo a obter o maior número de informações durante a demonstração das ferramentas; e
- Apresentar-se para cada vendedor, a fim de que este conheça mais sobre os seus processos e negócios.

A partir disto, tem-se as apresentações das ferramentas e, por fim, a seleção daquela que melhor atende as necessidades da empresa.

Na mesma direção, Hope-Ross e outros (HOPE-ROSS; DISBROW; WOOD, Agosto, 2003) apresentam 5 recomendações para auxiliar empresas na escolha de soluções computacionais e que podem ser aplicadas para a seleção de ferramentas de GCS. São elas:

1. **Seguir as melhores práticas no processo de aquisição:** O processo de aquisição pode ser aperfeiçoado por meio do uso de metodologias, gerenciamento do processo, gerenciamento do projeto, e pela clara definição dos papéis e responsabilidades deste processo. Apenas a partir destas ações preliminares, a seleção da ferramenta deve começar. Além disso, durante a seleção, a empresa deve: a) Identificar todos os *stakeholders* e construir equipes com uma composição que reflita as diversidades geográficas, funcionais e técnicas da empresa; b) Documentar o processo a ser apoiado pela ferramenta em seu estado atual e depois da implementação; c) Manter o processo de seleção autamente competitivo; d) Atribuir um ponto interno de contato para os vendedores e estabelecer guias para que estes vendedores possam buscar informações mais detalhadas sobre os requisitos a serem atendidos pela ferramenta; e e) Incluir profissionais treinados e experientes na negociação da licença da ferramenta e do contrato de integração de sistemas.
2. **Não perguntar se é possivel e sim como é feito:** Deve-se evitar perguntas do tipo “O vendedor X pode suportar o conjunto de funcionalidades y”, pois mesmo a resposta sendo afirmativa e tecnicamente correta, o suporte ou implementação daquelas funcionalidades pode ser custoso, podendo exigir inclusive trabalhos manuais para completar as atividades. Portanto, sempre deve-se perguntar como um dado produto suporta uma determinada funcionalidade.

3. **Executar cenários:** Um cenário descreve um problema que a empresa deseja resolver. A descrição do problema deve ser bem detalhada e idealmente deve conter dados que o vendedor deverá utilizar para realizar a demonstração da ferramenta.

4. **Requisitar referências:** Em conjunto com a execução dos cenários por parte dos vendedores, a empresa deve requisitar referências de outros clientes destes vendedores e entrar em contato com eles. Para tanto, a equipe de seleção deve: a) escolher referências de uma lista completa de clientes; b) se atentar àqueles do mesmo ramo de negócio, com infra-estruturas similares e que usam o mesmo conjunto de produtos de software e versões de software; c) procurar mais de um ponto de contato para cada referência; d) conversar não apenas com contatos da área executiva e sim com gerentes do projeto e pessoas que utilizam a ferramenta no seu dia-a-dia; d) procurar clientes que compraram a ferramenta a mais de 6 meses, para garantir maior confiabilidade nas suas impressões em relação a ferramenta; e e)incluir a análise dos pontos positívos e negativos para entender as razões que levaram os clientes a adquirir a ferramenta.

5. **Deixar tudo por escrito:** A chave para o sucesso na aquisição da ferramenta é fazer um grande contrato que evite surpresas desagradáveis. Este contrato, por exemplo, deve explicitamente definir quem será responsável pelas tarefas de configuração e customização da ferramenta.

Enfim, a seleção de uma ferramenta de GCS é uma parte fundamental na implantação do processo de GCS e deve ser realizada da melhor maneira possível de tal forma que a empresa obtenha aquela que melhor atenda as suas necessidades em relação à execução deste processo.

2.6 Considerações Finais

A partir da correta execução de processos de software, como o de GCS, tem-se uma redução dos riscos de falha no desenvolvimento de projetos de software. Além disso, é por meio dos processos que as empresas de software são avaliadas, segundo os modelos, normas e padrões, quanto à sua capacidade em desenvolver software. O processo de GCS contribui com a qualidade dos projetos de software desenvolvidos na organização provendo:

- Uma estrutura para identificação e controle dos CIs: documentação, código, interfaces, bancos de dados, entre outros; e

- Informações de gerenciamento do projeto e do produto preocupando-se com o controle das

> baselines, das mudanças, dos testes, das liberações, das auditorias, entre outras atividades de desenvolvimento de um projeto de software.

No entanto, a implantação do processo de GCS só é possível após a adoção ou elaboração de uma abordagem para este processo por meio do qual são definidos os conceitos, agentes, atividades e responsabilidades em concordância com as características dos projetos de software desenvolvidos pela empresa. É a partir desta abordagem que se tem definição o plano de GCS, a ser aplicado nos projetos de software, e seleção de ferramentas de apoio ao processo de GCS, reduzindo assim as dificuldades na execução deste processo.

3 *SoCManager: Uma Ferramenta de Apoio ao Gerenciamento de Configuração de Software*

Motivado pelos estudos realizados e pelo objetivo de se facilitar a implantação do processo de GCS por empresas de software, construiu-se nesta pesquisa uma ferramenta de apoio ao GCS, denominada *SoCManager* (*Software Configuration Manager*). Para tanto, foi elaborada uma abordagem para o processo de GCS de onde são descritos os conceitos, agentes e atividades para este processo. Adicionalmente, foi elaborada uma abordagem para implantação de um processo de GCS, onde foram definidas as atividades que devem ser realizadas para a implantação de um processo de GCS em uma empresa de software.

Dessa forma, esta pesquisa consistiu em:

- Elaborar uma abordagem que define as atividades para realizar a implantação de um processo de GCS em empresas de software. Esta abordagem foi baseada no modelo IDEAL (descrito na seção 2.3) para melhoria de processos de software e possui um conjunto de atividades que norteiam as empresas na implantação de um processo de GCS;
- Elaborar uma abordagem para o processo de GCS, que estabelece aspectos deste processo conforme as metas definidas pela PA de Gerenciamento de Configuração do modelo de capacitação CMMi (descrito na seção 2.2); e
- Construir uma ferramenta CASE de apoio ao processo de GCS com base nos aspectos levantados na abordagem para este processo elaborada nesta pesquisa.

Apresentam-se a seguir os três resultados desta pesquisa. Inicialmente apresenta-se a abordagem para implantação de um processo de GCS e suas atividades. Em seguida, apresenta-se a abordagem para o processo de GCS, seus conceitos, agentes e atividades. Por fim, apresenta-se a ferramenta *SoCManager*, com os seus principais requisitos, a sua arquitetura e a sua integração com o ambiente *Orion* (LUCRÉDIO et al., 2004) do laboratório de Engenharia de Software do

Departamento de Ciência da Computação da Universidade Federal de São Carlos.

Para facilitar a leitura, a abordagem para a implantação de processo de GCS será referenciada como abordagem da implantação e a abordagem para o processo de GCS, como abordagem do processo.

Maiores informações acerca dos resultados desta pesquisa podem ser obtidas no *site* `http://recope.dc.ufscar.br/socmanager/`.

3.1 Uma Abordagem para Implantação de um Processo de Gerenciamento de Configuração de Software

Apresenta-se nesta seção uma abordagem da implantação. Esta abordagem é uma instanciação simplificada do modelo IDEAL (MCFEELEY, 1996) que enfoca as atividades que deveriam estar sendo realizadas para realizar a implantação de um processo de GCS.

Adicionalmente, a elaboração da abordagem também recebeu contribuições de experiências (CUGOLA et al., 1997) (HEDGE, 1992) na implantação do processo de GCS que contêm informações sobre as necessidades das empresas, a abordagem para o processo de GCS, as resistências enfrentadas e as lições aprendidas.

Apesar de o objetivo principal ser auxiliar na implantação da abordagem de GCS definida nesta pesquisa, a idéia foi manter essa abordagem de implantação genérica o suficiente para auxiliar na implantação de qualquer abordagem de GCS.

3.1.1 Descrição da Abordagem

A abordagem da implantação estabelece um conjunto de atividades que devem ser realizadas a fim de implantar este processo. Estas atividades devem ser realizadas por um agente denominado "Agente de Mudança", que é um profissional da própria empresa que deve estar comprometido com a implantação e que, muitas vezes, não pode se desfazer de suas demais obrigações. A definição deste agente se deve ao fato da abordagem poder estar sendo aplicada em empresas de pequeno e médio porte que não possuem muitos recursos, tanto humanos como financeiros, e que, portanto, dificilmente poderão contar com equipes especializadas na execução das atividades de implantação de processos como o de GCS.

Podem existir casos em que os Agentes de Mudança surgem da iniciativa de profissionais que desejam melhorar as suas condições de trabalho e a qualidade dos projetos desenvolvidos na sua empresa. Mas normalmente espera-se que a iniciativa seja da própria empresa, selecionando os

profissionais que deverão atuar como Agentes de Mudança.

Uma vez definidos, os Agentes de Mudança deverão trabalhar com o intuito de convencer as demais pessoas que trabalham na empresa dos benefícios em se implantar o GCS e obter o aval em realizar as atividades da abordagem da implantação nos projetos de software da empresa. Além disso, estes agentes devem estar cientes das resistências, relacionadas principalmente ao aspecto cultural, que irão sofrer no decorrer da execução das atividades da abordagem (a descrição dos aspectos envolvidos na implantação do processo de GCS podem ser vistos na seção 2.3).

A abordagem da implantação possui 12 atividades que foram agrupadas em 3 etapas conforme mostrado na figura 7.

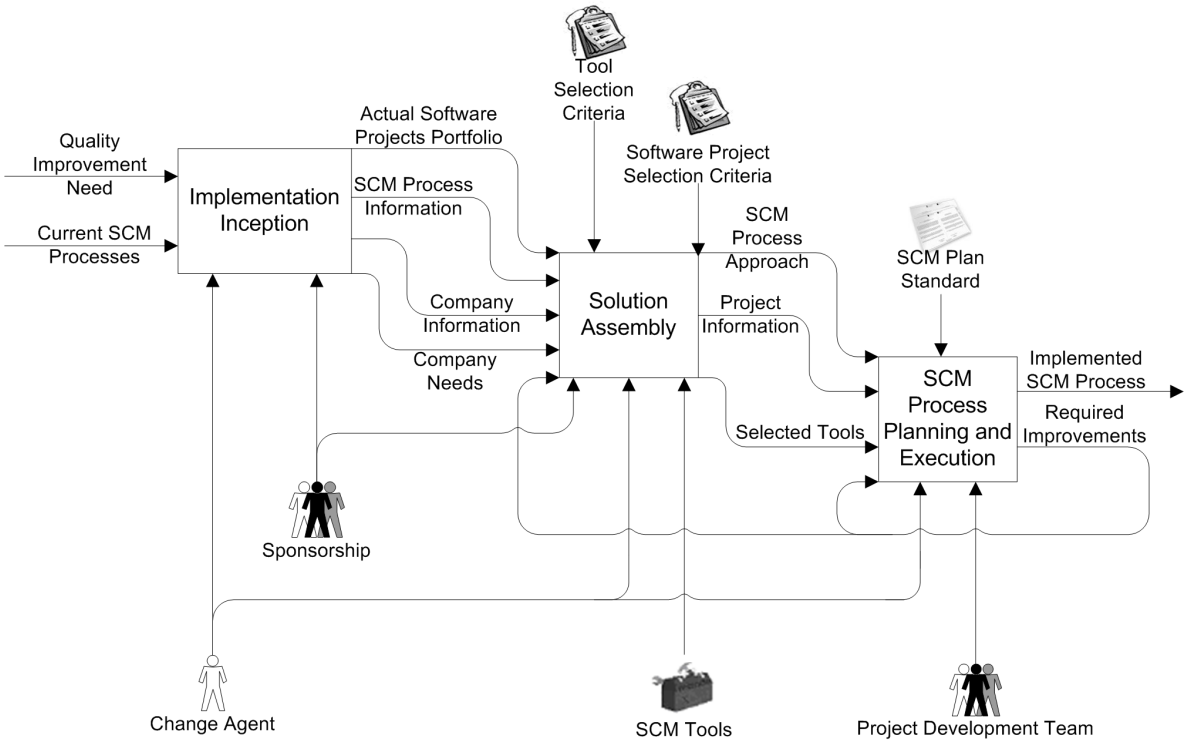

Figura 7: Etapas da Abordagem para Implantação do Processo de GCS Elaborada nesta Pesquisa (Notação SADT (ROSS, 1977))

Na etapa “Inicio da Implementação” (*Implementation Inception*), tem-se atividades para obtenção de apoio dentro da empresa para a execução da abordagem, além da coleta de informações tanto de processos de GCS como da empresa. Na figura 8 são mostradas as atividades referentes a esta etapa da implantação descritos em seguida.

1. **Obter Patrocínio (*Get Sponsorship*):** Esta atividade envolve a busca, por parte do Agente de Mudança (*Change Agent*), do apoio de pessoas dentro da empresa que tenham maior poder de decisão (*Champions*) e que irão atuar na implantação como patrocinadores (*Sponsorship*). Esta atividade é particularmente importante no caso da iniciativa em implantar GCS não ter

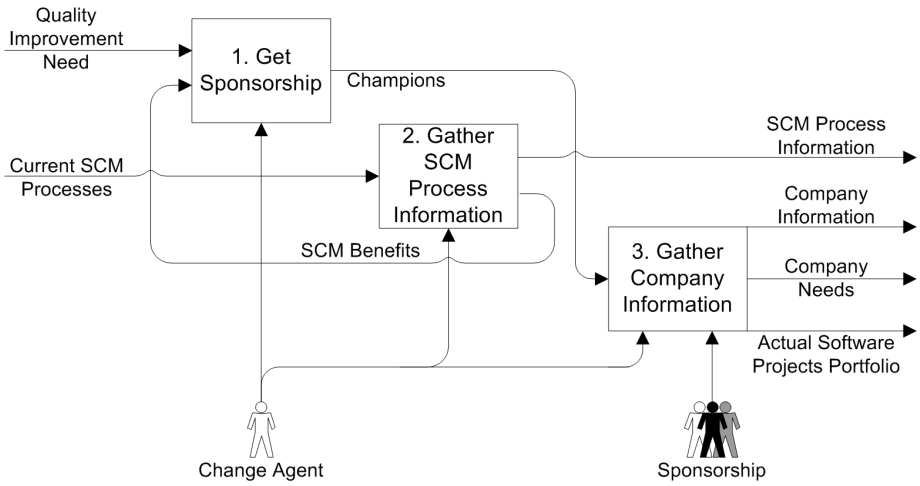

Figura 8: Atividades da Etapa "Início da Implementação" da Abordagem para Implantação do Processo de GCS (Notação SADT (ROSS, 1977))

partido da empresa e sim de pessoas dentro da organização. A partir deste patrocínio é que o Agente de Mudança poderá justificar o uso de seu tempo de trabalho para a implantação. Além disso, por meio do patrocínio é que se tornará possível a utilização de um projeto de software da organização como projeto-piloto.

No entanto, para obter o patrocínio, o Agente de Mudança deve estar ciente da necessidade de melhoria da qualidade (*Quality Improvement Need*) e dos benefícios que o processo de GCS (*SCM Benefits*) poderá trazer para a empresa. Com base nessas informações é que ele terá maiores chances de conseguir o patrocínio necessário. A partir dessas informações, o Agente de Mudança deverá marcar reuniões com as pessoas que têm maior poder de decisão dentro da empresa para fazer uma breve apresentação, explicitando as melhorias da qualidade e benefícios obtidos a partir da correta execução do processo de GCS.

2. **Levantar Informações sobre o Processo de GCS (*Gather SCM Process Information*):** Esta atividade envolve coletar informações sobre processos de GCS atuais (*Current SCM Processes*), suas atividades, relatos de experiência sobre seu uso e implantação, modelos e padrões de qualidade relacionados, entre outros. Com base nestas informações (*SCM Process Information*) é que o Agente de Mudança poderá explorar mais profundamente os aspectos do processo de GCS e daí então executar as demais atividades da abordagem da implantação.

   Uma forma de se obter estas informações é por meio do uso da técnica que diz respeito ao estudo de viabilidade, onde o Agente de Mudança procura obter o contato de pessoas, de fora da empresa, experientes na área de GCS por meio de correio eletrônico, por exemplo. Com este contato, o Agente de Mudança poderá obter informações a respeito do processo de GCS e as outras informações citadas.

3. **Levantar Informações sobre a Empresa (*Gather Company Information*):** Esta atividade envolve coletar informações sobre os projetos de software desenvolvidos naquele momento pela empresa (*Actual Software Projects Portfolio*) e sobre a empresa (*Company Information*) como, por exemplo, os processos de software utilizados e a estrutura organizacional. Além disso, tem-se nesta atividade a identificação das necessidades específicas da empresa (*Company Needs*) que poderão estar sendo atendidas por um processo de GCS.

   Para tanto, o Agente de Mudança, com o apoio dos patrocinadores, poderá utilizar-se de técnicas como a entrevista (no caso dos gerentes, coordenadores e líderes do projeto), questionários (para as equipes dos projetos como um todo), análise de documentos dos processos de software e projetos, entre outros. Caso a empresa não possua um processo ou método de desenvolvimento bem definido, então o Agente de Mudança deverá coletar informações sobre o processo informal executado nos projetos.

Já na etapa "Montagem da Solução" (*Solution Assembly*), tem-se atividades de estabelecimento de objetivos e metas na implantação do processo de GCS, de definição ou adoção de uma abordagem para este processo, de seleção de ferramentas de apoio a esta abordagem, e de escolha de projetos de software onde será implantado o processo de GCS. Na figura 9 são mostradas as atividades referentes a esta etapa da implantação descritos em seguida.

4. **Estabelecer Objetivos e Metas (*Establish Goals and Targets*):** Esta atividade envolve definir os objetivos e metas (*Goals and Targets*) da empresa quanto aos aspectos de desenvolvimento de software relacionados com o processo de GCS. Os objetivos descrevem o contexto sobre o qual a implantação estará sendo realizada. Já as metas definem a realização da implantação propriamente dita.

   Com base nas necessidades da empresa (*Company Needs*), o Agente de Mudança e os Patrocinadores poderão definir, por exemplo, que o objetivo da organização é obter uma determinada certificação de qualidade. Portanto, o processo de GCS deverá atender aos critérios exigidos por esta certificação. Além disso, por meio das necessidades, é possível priorizar a implantação das atividades do processo de GCS. Pode-se ter, por exemplo, a necessidade de possuir o Controle de Versões sobre os projetos da organização. Neste caso, a meta inicial deveria ser a implantação de atividades relacionadas ao Controle de Versões.

5. **Definir uma Abordagem para o Processo de GCS (*Define a SCM Process Approach*):** Esta atividade envolve tanto elaborar ou adotar uma abordagem para o processo de GCS (*SCM Process Approach*) (por exemplo, as citadas na seção 2.4), como também melhorar a abordagem já existente com base nas melhorias requeridas (*Required Improvements*) após

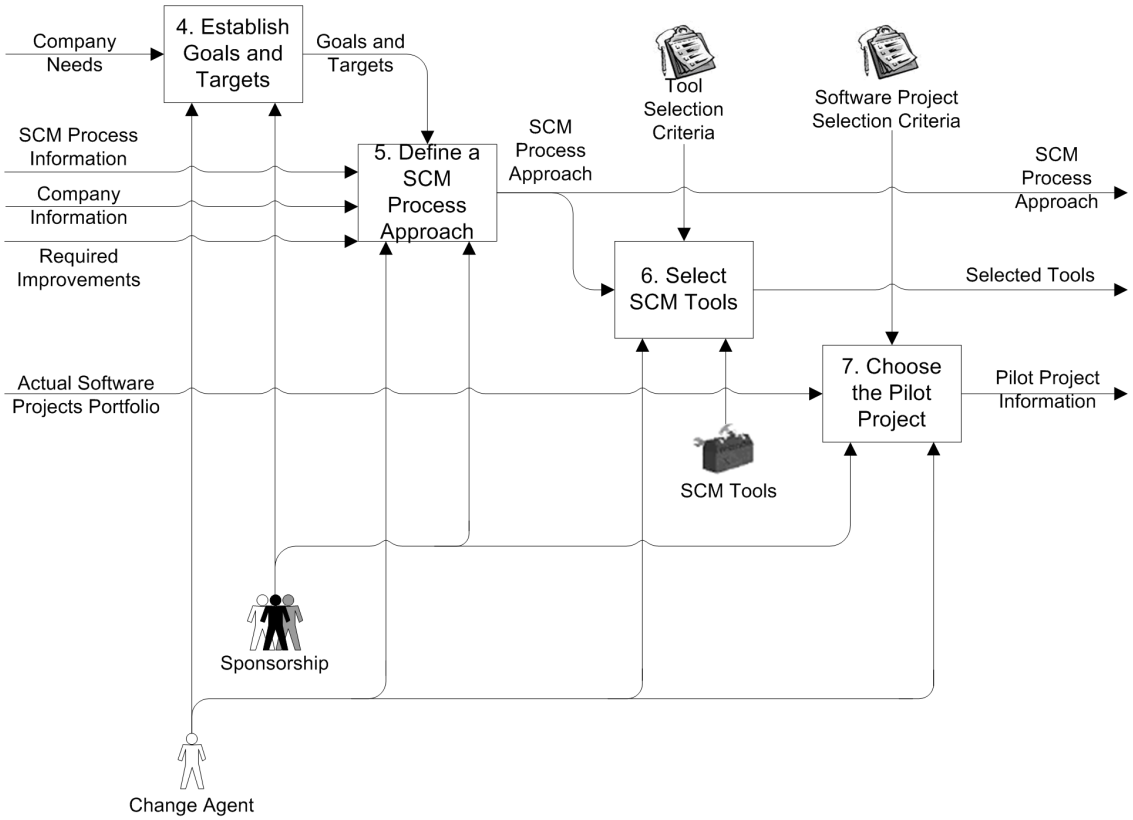

Figura 9: Atividades da Etapa “Montagem da Solução” da Abordagem para Implantação do Processo de GCS (Notação SADT (ROSS, 1977))

a implantação do processo de GCS nos projetos de software da empresa. Uma abordagem para o processo de GCS deve descrever os conceitos, agentes, atividades e os insumos e resultados dessas atividades.

No caso, o Agente de Mudança, por meio do apoio dos Patrocinadores, deverá definir a abordagem a ser utilizada com base nas informações da empresa (*Company Information*) e nas informações sobre o processo de GCS (*SCM Process Information*). No entanto, esta abordagem deve ser definida visando atender aos objetivos e metas já definidos (*Goals and Targets*).

Para a elaboração de uma nova abordagem, o Agente de Mudança deve buscar definir o escopo do processo por meio da definição dos elementos envolvidos e os conceitos relacionados a estes elementos. Posteriormente, devem ser definidos os agentes de GCS que deverão executar as atividades deste processo. Depois devem ser definidas as atividades com base, por exemplo, nas que compõem o processo de GCS (seção 2.1.1), e os insumos e resultados da abordagem.

6. **Selecionar Ferramentas de GCS (*Select SCM Tools*):** Esta atividade envolve definir as ferramentas de GCS (*Selected Tools*) que irão apoiar a execução do processo de GCS descrito

na abordagem (*SCM Process Approach*). Portanto, depois de definida a abordagem para o processo de GCS, deve-se buscar utilizar ferramentas que atendam às suas atividades.

Cada ferramenta possui características específicas e atende a determinados aspectos de GCS. Escolher qual ferramenta adotar para automatizar a abordagem é uma tarefa difícil que deve ser bem planejada e realizada. Além disso, muitas empresas precisam realizar o desenvolvimento de uma outra ferramenta que atenda às suas necessidades não contempladas por aquelas que foram adquiridas do mercado. Buscando minimizar estes gastos, empresas devem seguir critérios (*Tool Selection Criteria*) como os apresentados na seção 2.5 e analisar as ferramentas disponíveis no mercado (*SCM Tools*).

Porém, o uso desses critérios exige muitas vezes um melhor planejamento, devendo estar contextualizado sob um processo de escolha que pode envolver várias pessoas na empresa. Dentre os casos de seleção de ferramentas, cita-se (DART, 1992) e (BURROWS; GEORGE; DART, 1996).

7. **Escolher o Projeto-Piloto (*Choose the Pilot Project*):** Esta atividade envolve escolher um projeto piloto para implantar o GCS. Esta escolha é fundamental para se conseguir reduzir as dificuldades nessa implantação, principalmente as relacionadas com os aspectos culturais. Além disso, há outros motivos que levam a escolher um projeto piloto. São eles:

   - Menor complexidade tendo em vista o fato de estar sendo aplicado em um projeto específico com apenas algumas pessoas envolvidas;
   - Melhor gerenciamento da transformação;
   - No caso da implantação ter sido bem sucedida, o projeto servirá de caso de sucesso e os benefícios obtidos serão evidenciados para permitir a implantação nos demais projetos de software da empresa; e
   - No caso da implantação não ter sido bem sucedida, os impactos para a empresa serão menores e mais controlados.

   Para tanto, o Agente de Mudança e Patrocinadores deverão selecionar um dos projetos de software atualmente desenvolvidos na empresa (*Actual Software Projects Portfolio*). Para esta pode-se adotar critérios (*Software Project Selection Criteria*) como, por exemplo, selecionar projetos de software com baixo risco e que estejam num estágio inicial de desenvolvimento. Além disso, deve-se entrar em comum acordo com os responsáveis pelo projeto como, por exemplo, o gerente, o coordenador e os líderes, para que possam ser realizadas as demais atividades da abordagem da implantação naquele projeto.

Por fim, na etapa “Planejamento e Execução do Processo de GCS” (*SCM Process Planning and Execution*), tem-se atividades para planejar, implementar e acompanhar a execução do plano de GCS, além da atividade para extensão da implementação para os demais projetos de software. Na figura 10 são mostradas as atividades referentes a esta etapa da implantação descritos em seguida.

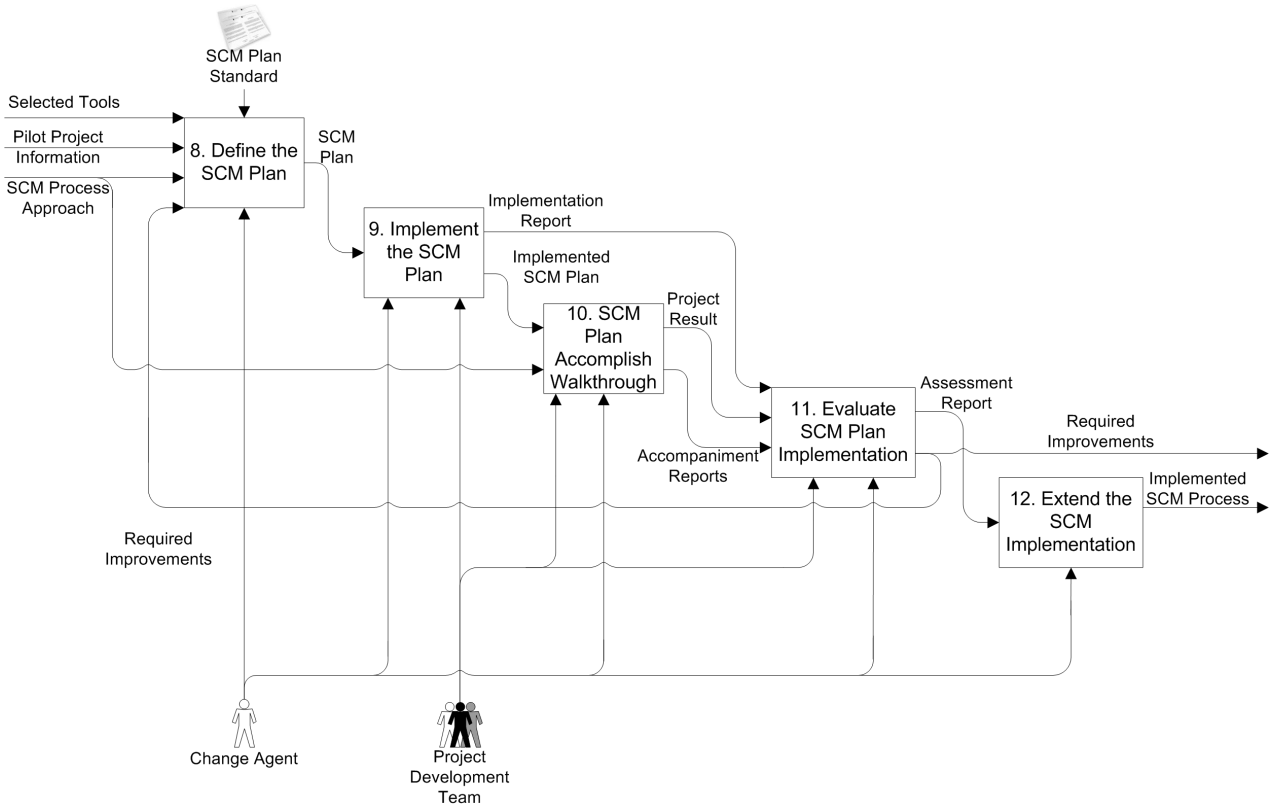

Figura 10: Atividades da Etapa “Planejamaneto e Execução do Processo de GCS” da Abordagem para Implantação do Processo de GCS (Notação SADT (ROSS, 1977))

8. **Definir o Plano de GCS (*Define the SCM Plan*):** Esta atividade envolve definir a estrutura organizacional para a execução da abordagem de GCS por meio do plano de GCS (*SCM Plan*) que deverá ser posteriormente aplicado ao projeto. Para tanto, o Agente de Mudança deverá considerar as informações sobre o projeto-piloto (*Pilot Project Information*), a abordagem de GCS definida e as informações acerca das ferramentas de GCS que serão utilizadas (*Selected Tools*). Além disso, pode-se ter também as melhorias requeridas obtidas de uma anterior execução desta abordagem da implantação na empresa.

   Para a definição do plano GCS, o Agente de Mudança poderá se utilizar de padrões (*SCM Plan Standard*) como o estabelecido pela IEEE (IEEE, 1998).

   Uma vez preenchido, esse plano garante uma maior formalidade e maturidade organizacional referente à área de conhecimento de GCS.

9. **Implementar o Plano de GCS (*Implement the SCM Plan*):** A implementação do plano de GCS envolve apresentar aos membros da equipe do projeto-piloto (*Project Development Team*) o conteúdo deste plano, designando para cada membro os papéis com base nos agentes definidos na abordagem para o processo de GCS adotado. Para tanto, esta atividade envolve realizar uma série de reuniões e apresentações com esses membros.

   Durante a implementação, deve-se buscar conciliar a execução das atividades de GCS com as demais atividades definidas por outros processos de software já utilizados no projeto-piloto. Dois pontos importantes a serem considerados pelo Agente de Mudança durante as reuniões e apresentações com a equipe são:

   - Esclarecer os benefícios em executar as atividades de GCS para que a equipe esteja motivada em realizar as atividades; e
   - Definir claramente qual será a nova rotina de trabalho das pessoas envolvidas no projeto.

   Pode haver uma certa resistência por parte dos membros da equipe do projeto-piloto neste momento. Portanto, o Agente de Mudança deve ter o apoio dos patrocinadores e dos responsáveis pelo projeto-piloto durante as reuniões e apresentações.

   Ao término desta atividade, o Agente de Mudança deverá compor um relatório sobre a implementação do plano de GCS (*Implementation Report*) descrevendo as dificuldades encontradas e suas soluções, perguntas freqüentes, entre outros.

10. **Acompanhamento da Execução do Plano de GCS (*SCM Plan Accomplish Walkthrough*):** Esta atividade envolve evoluir o plano e abordagem de acordo com o *feedback* obtido pela execução das atividades de GCS. Para tanto, o Agente de Mudança deve acompanhar a execução dessas atividades, garantir a sua correta execução e obter, por parte dos membros da equipe do projeto-piloto, sugestões e críticas. Durante esse acompanhamento, ele deve estar preparado para possíveis mudanças no plano e, conseqüentemente, na abordagem de GCS. Cada uma dessas informações devem ser registradas por meio de relatórios de acompanhamento (*Accompaniment Reports*).

    Adicionalmente, por meio desta atividade o Agente de Mudança pode detectar melhorias requeridas para a abordagem e o plano de GCS e que deverão estar sendo realizadas para a implantação do processo nos demais projetos de software da empresa.

11. **Avaliar a Implementação do Plano de GCS (*Evaluate SCM Plan Implementation*):** Esta atividade envolve avaliar, após a conclusão do projeto-piloto, os relatórios de acompanhamento e os resultados do próprio projeto (*Project Result*) para avaliar se os benefícios, des-

critos por meio das metas, foram realmente alcançados. Com base nessa avaliação e no relatório de implementação do plano de GCS é que o Agente de Mudança poderá definir as melhorias requeridas, por meio dos novos passos a serem realizados para os próximos projetos onde a implantação será realizada, além de melhorar o que foi definido na abordagem para o processo de GCS adotado pela empresa.

Para auxiliar nesta avaliação, o Agente de Mudança pode solicitar reuniões com os membros da equipe do projeto-piloto para adquirir o seu *feedback* em relação à implantação do GCS. Dentre as informações que podem ser requeridas, constam: os benefícios que a equipe esperava alcançar; as dificuldades encontradas; as formas como as dificuldades foram superadas; e as críticas e sugestões.

Pode-se ter, por exemplo, mudanças na forma como executar as atividades de GCS, de acordo com as dificuldades encontradas pelos membros da equipe do projeto-piloto, e as soluções para estas dificuldades.

12. **Estender a Implementação para Outros Projetos de Software (*Extend the Implementation to Other Software Projects*):** Esta atividade envolve estender a implementação do processo de GCS nos demais projetos de software a serem desenvolvidos na empresa com base no relatório da avaliação da implementação do plano de GCS (*Assessment Report*).

    Para tanto, o Agente de Mudança deve tornar bem claros os resultados avaliados na atividade anterior, explicitando os benefícios obtidos pela implantação. O público alvo nesta atividade são tanto os patrocinadores como as outras pessoas da empresa com poder de decisão que, no primeiro momento, não apoiaram a implantação.

    Além disso, o Agente de Mudança deverá divulgar o processo de GCS e toda a sua documentação (*Implemented SCM Process*), definir a estrutura organizacional e efetuar os treinamentos necessários para garantir a correta execução do processo nos projetos de software da empresa mesmo que não haja sua direta intervenção.

    As atividades da estratégia necessárias para adotar o processo de GCS nos demais projetos de software são:

    - Definição do Plano de GCS. Este plano pode ter como base as informações do plano aplicado no projeto-piloto;
    - Implementação do Plano de GCS;
    - Acompanhamento da Execução do Plano de GCS; e
    - Avaliação da Implementação do Plano de GCS.

É importante que se tenha o compromisso, por parte dos responsáveis pela execução do processo de GCS, de buscar aperfeiçoar a abordagem para o processo de GCS adotado na empresa, para que se tenha um esforço para a melhoria contínua deste processo e, conseqüentemente, da qualidade dos projetos de software da empresa.

3.1.2 Relação da Abordagem da Implantação com o modelo IDEAL

O modelo IDEAL (seção 2.3) é um modelo que estabelece um ciclo de melhoria de processo. Já abordagem da implantação, é uma simplificação deste modelo voltada para a implantação do processo de GCS.

Conforme pode ser visto na tabela 9, a abordagem da implantação focou principalmente na atividade de “Criação da Solução”, do modelo IDEAL. Existem, no entanto, atividades deste modelo (“Motivação”, “Abordagem”, “Planejamento” e “Proposição de Ações Futuras”) que não foram contempladas diretamente por atividades da abordagem da implantação. A atividade de “Motivação”, por exemplo, surge como pré-condição para a execução desta abordagem, por meio da definição do Agente de Mudança e da necessidade observada em melhorar a qualidade do desenvolvimento de software. A atividade de “Abordagem”, que engloba a definição dos passos para a realização da melhoria, já é contemplada pela própria abordagem da implantação que define esses passos por meio de suas atividades. A atividade de “Planejamento” foi suprimida, sendo realizada pelo “Agente de Mudança” durante a execução das atividades de “Implantação do Plano de GCS” e “Acompanhamento da Execução do Plano de GCS”. Por fim, a atividade de “Proposição de Ações Futuras” corresponde ao resultado da execução das atividades de “Acompanhamento da Execução do Plano de GCS” e “Avaliação da Implantação do Plano de GCS”, onde se tem as melhorias requeridas tanto para a abordagem como para o plano de GCS utilizados no projeto-piloto.

3.2 Uma Abordagem para Processo de Gerenciamento de Configuração de Software

Uma abordagem para um dado processo estabelece as atividades, os agentes e suas responsabilidades, e os insumos e resultados (em termos de artefatos) relacionados a este processo, de tal forma que ela possa ser executada no desenvolvimento de projetos de software. Apenas a partir dessa abordagem é que se torna possível a adoção/desenvolvimento de ferramentas que apóiem a sua execução.

Segundo a atividade de Seleção de Ferramentas de GCS, apresentada na seção 3.1, mesmo que haja a necessidade de se desenvolver uma nova ferramenta de GCS, a empresa deve definir

Tabela 9: Relacionamento entre as atividades do Modelo IDEAL e as atividades da Abordagem da Implantação

Atividade do Modelo IDEAL	**Atividade da Abordagem do Processo**
Motivação	-
Patrocínio	Obter Patrocínio
Infra-estrutura	-
Aferição	Levantar Informações sobre o processo de GCS Levantar Informações sobre a empresa
Recomendações	Estabelecer Objetivos e Metas
Priorização	Estabelecer Objetivos e Metas
Abordagem	-
Planejamento	-
Criação da Solução	Definir uma Abordagem para o Processo de GCS Selecionar Ferramentas de GCS Escolher o Projeto-Piloto Definir o Plano de GCS
Testes-piloto	Implementar o Plano de GCS Acompanhamento da Execução do Plano de GCS
Refinamento	Avaliar a Implementação do Plano de GCS
Implantação Completa	Extender a Implementação para Outros Projetos de Software
Análise e Validação	Implementar o Plano de GCS Acompanhamento da Execução do Plano de GCS
Proposição de ações futuras	-

previamente a abordagem que deseja implementar. Por este motivo e devido ao fato desta pesquisa envolver a construção de uma ferramenta de apoio ao processo de GCS, foi elaborada uma abordagem para o processo de GCS (CUNHA; PRADO; SANTOS, 2004), aqui denominada de abordagem do processo.

Esta abordagem teve como referência as abordagens apresentadas na seção 2.4 e os aspectos deste processo descritos em (ANSI/IEEE, 1987), (IEEE, 1998) e (IEEE, 2004).

Buscou-se nesta abordagem incorporar as diversas características definidas nessas referências de tal forma que ela atendesse aos aspectos básicos de GCS, podendo ser utilizada na implantação deste processo em empresas de software. A partir da execução dessa abordagem, acredita-se que a empresa esteja apta a aperfeiçoar o processo de GCS e atender as suas necessidades específicas por meio da melhoria continua.

A seguir apresentam-se os conceitos, agentes, e atividades da abordagem do processo. Adicionalmente faz-se uma análise sobre a aderência desta abordagem ao modelo de capacitação CMMi, citando as relações existentes entre as suas atividades e as metas definidas pela PA de Gerenciamento de Configuração do nível 2 do modelo.

3.2.1 Conceitos da Abordagem do Processo

Segundo os conceitos definidos na abordagem do processo (Figura 11), um projeto (*Project*) é uma unidade de gestão na qual se realiza algo único, que possui uma data de início e término (PMI, 2000), e que representa a execução de processos de software. Este projeto pode ser dividido em módulos (*Modules*), para facilitar o seu gerenciamento, e reutilizar outros projetos.

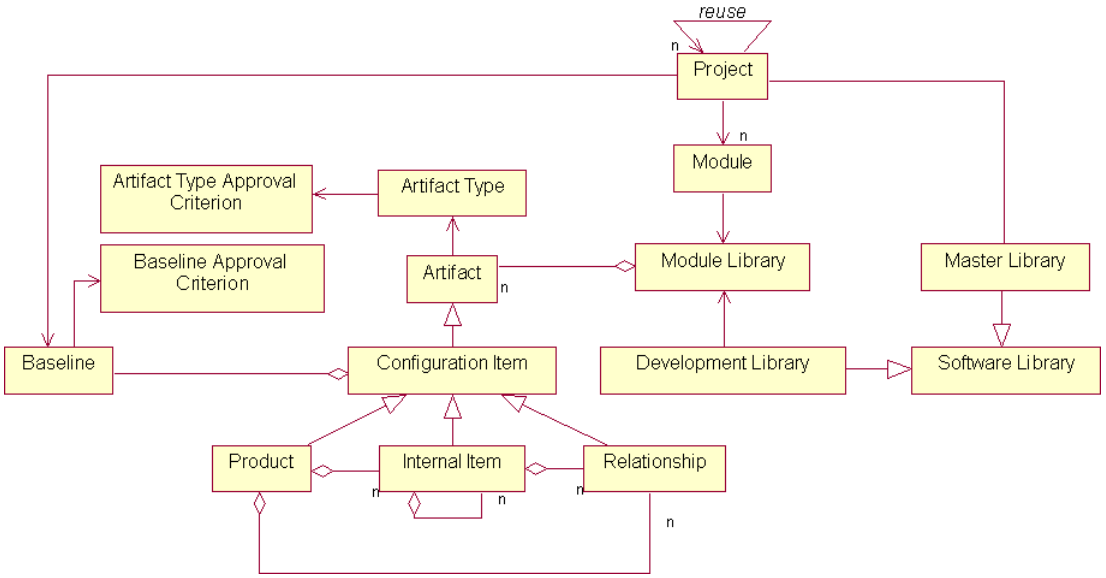

Figura 11: Conceitos da Abordagem para o processo de GCS (Notação UML (OMG, 2003))

A configuração de software do projeto é composta por artefatos (*Artifacts*) e CIs. Ambos são informações criadas durante o desenvolvimento de um produto de software, ou que sejam necessários para seu desenvolvimento (PRESSMAN, 2001). Além disso, um CI é um artefato:

- Concluído;
- Entregue e avaliado segundo atividades definidas pelo GCS para compor uma determinada versão da configuração do projeto;
- Controlado, pois só poderá ser alterado após uma aprovação prévia definida por atividades do processo de GCS; e
- Integrante de uma dada *baseline* do projeto.

Além disso, a abordagem estende o conceito de CI e artefatos. Assim, um artefato pode ser definido como sendo: um Produto (*Product*) que corresponde a um arquivo do projeto; um Item Interno (*Internal Item*) que corresponde a itens internos do artefato; ou um Relacionamento (*Relationship*) que corresponde a um relacionamento entre artefatos Produto, artefatos Item Interno

ou entre artefatos Produto e Item Interno. Este conceito permite um controle dos artefatos em diferentes níveis de abstração.

Neste contexto, para cada categoria de artefato (Produto, Item Interno ou Relacionamento) são associados tipos (*Artifact Type*) como, por exemplo, classes, documento de especificação de requisitos, entre outros. Adicionalmente, cada tipo também possui um critério de aprovação (*Artifact Type Approval Criterion*) a ser adotado nas atividades da abordagem proposta.

Tanto os artefatos quanto os CIs são armazenados em bibliotecas de software (*Software Library*) (IEEE, 2004) do projeto. Dentre os possíveis tipos de biblioteca, esta abordagem define:

- Bibliotecas de Desenvolvimento (*Development*): bibliotecas que estão diretamente relacionados com os módulos do projeto e onde são armazenados os artefatos que fazem parte do desenvolvimento do projeto e que ainda não são controlados; e
- Bibliotecas de Produção (*Master*): bibliotecas que estão relacionadas com o projeto e onde são armazenados os CIs.

Já as *baselines* do projeto contêm uma série de CIs e servem de base para as atividades de desenvolvimento. As *baselines* possuem um dado estado que pode ser: criada; aberta, ou seja, é a *baseline* que está sendo desenvolvida naquele momento (apenas uma *baseline* pode estar aberta num dado momento do projeto); e estabelecida, isto é, já foi adquirida e nenhum CI pode ser adicionado a ela. Além disso, assim como os tipos de artefatos, toda *baseline* possui um critério de aprovação (*Baseline Approval Criterion*) a ser adotado nas atividades da abordagem proposta.

3.2.2 Agentes da Abordagem de GCS

O GCS, por ser um processo fundamental de suporte dentro do desenvolvimento de software, exige a participação de vários agentes que atuam no projeto. Além dos agentes específicos do processo de GCS, como o Comitê de Controle de Configuração e o Responsável pelo Gerenciamento da Configuração, têm-se outros agentes como o gerente do projeto, o auditor e o engenheiro de software. Seguem os agentes definidos na abordagem do processo que devem ser estabelecidos na empresa para que se tenha a adoção desta abordagem:

- **Gerente do Projeto (*Project Manager*):** É aquele que definiu o projeto e que realiza as atividades relacionadas com as áreas de conhecimento de gestão de projetos como, por exemplo, a definição da equipe, o planejamento e controle de tarefas e atividades do projeto;

- **Engenheiro de Software (*Software Engineer*):** É quem realiza as implementações e manutenções dos artefatos do projeto. Pode vir a desempenhar diferentes papéis de acordo com o que é estabelecido na organização, como por exemplo, analista de sistemas, programador, integrador, entre outros;

- **Comitê de Controle de Configuração (*Configuration Control Board - CCB*):** É um comitê que dependendo da complexidade do projeto pode ser formado por uma ou várias pessoas. Normalmente é formado pelo Gerente do Projeto e os líderes de cada um dos módulos desenvolvidos no projeto;

- **Bibliotecário (*Librarian*):** É quem tem sob custódia e controle a biblioteca de software do projeto e os artefatos nela armazenados;

- **Responsável pelo Gerenciamento de Configuração (*Configuration Management Officer - CMO*):** É quem executa as atividades operacionais do GCS como, por exemplo, o controle dos formulários criados durante as atividades. Dependendo do tamanho do projeto, é imprescindível o apoio de ferramentas para auxiliar o CMO em suas responsabilidades. Inclusive as atividades sob responsabilidade deste papel podem estar sendo automatizadas parcialmente ou completamente por meio do uso de uma ferramenta de GCS;

- **Auditor (*Auditor*):** É quem está relacionado à qualidade de software e que possui responsabilidades definidas para as áreas de qualidade adotadas na organização e que, portanto, segue critérios de qualidade definidos para avaliar o que é realizado dentro do projeto;

- **Comitê de Engenharia de Processo (*Process Engineer Board - PEB*):** É um comitê que define e controla os processos de software, evoluindo-os de acordo com as necessidades organizacionais; e

- **Comitê de Garantia de Qualidade (*Quality Assurance Board - QAB*):** É um comitê que define os critérios de aprovação associados aos CIs e baselines.

Cada um destes agentes de negócio possui responsabilidades definidas com base nas atividades de GCS e, dependendo da quantidade de recursos presentes na organização, pode ocorrer de membros da equipe desempenharem o papel de mais de um agente.

3.2.3 Atividades da Abordagem de GCS

A abordagem do processo define um conjunto de atividades relacionadas com a Identificação da Configuração (*Configuration Identification*), Controle da Configuração (*Configuration Con-*

trol), Auditoria da Configuração (*Configuration Audit*) e Administração de Estado (*Status Accounting*) que são executados dentro de um projeto de software. Além disso, tem-se a atividade de Identificação de Tipos de CIs (*CI Types Identification*), que é executada fora do contexto de um projeto de software. Na Figura 12 são mostradas as atividades definidas na abordagem do processo.

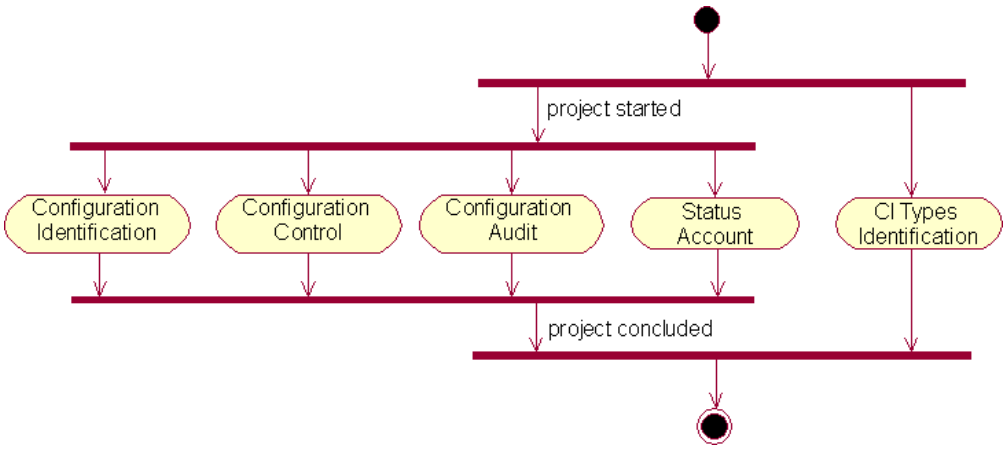

Figura 12: Atividades da Abordagem de GCS (Notação UML (OMG, 2003))

Na Figura 12 observa-se que há a execução em paralelo da atividade de Identificação de Configuração com as demais atividades executadas sob o contexto de um projeto de software. No entanto, inicia-se a execução do processo de GCS por esta atividade, pois é nela que se tem a definição das *baselines* do projeto, dos tipos CIs que serão adquiridos no desenvolvimento, dentre outros aspectos que compõem a infra-estrutura de GCS utilizada pelas demais atividades.

3.2.3.1 Identificação da Configuração

A Identificação de Configuração fornece a infra-estrutura para as demais atividades do processo de GCS. É a partir desta atividade que se tem as *baselines* que estarão sendo desenvolvidas no decorrer do projeto, os CIs (por meio da definição dos tipos de artefatos a serem controlados) e a estrutura do projeto, isto é, as bibliotecas de software onde estarão sendo armazenados os artefatos e CIs. Esta atividade também envolve a entrega de artefatos para controle e a composição das *baselines*.

Na figura 13 são mostradas as atividades de Identificação da Configuração.

- **Identificação de *Baselines* (*Baselines Identification*):** O Gerente do Projeto define, com base em informações gerenciais do projeto, quais *baselines* serão estabelecidas durante o desenvolvimento do projeto. Para cada *baseline*, o QAB define o critério de aprovação (*Baseline Approval Criterion*) que deverá ser aplicado no momento de sua aquisição.

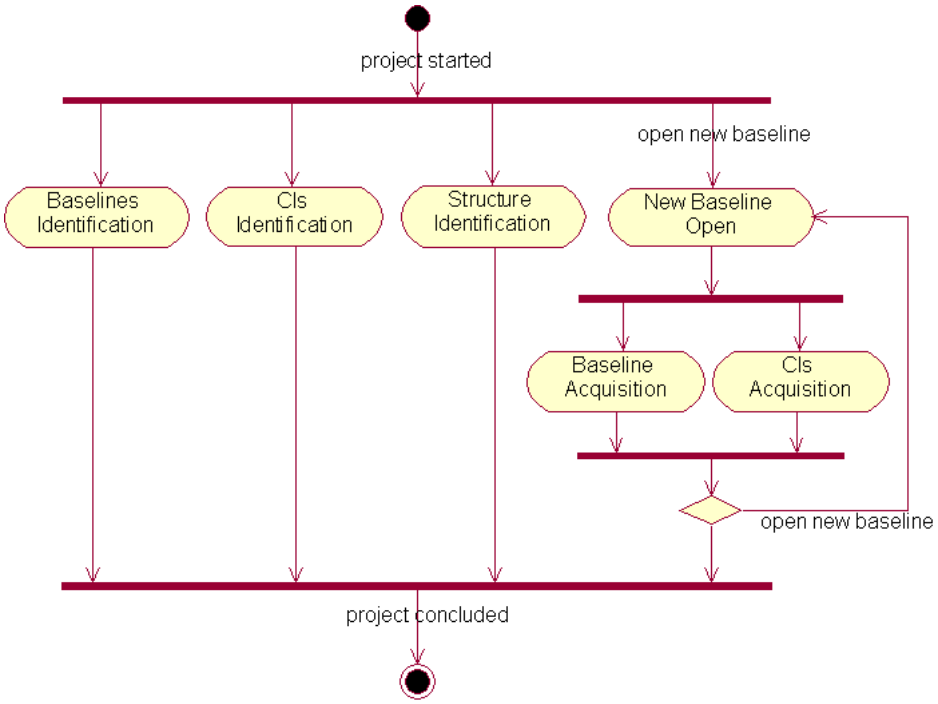

Figura 13: Atividades de Identificação da Configuração (Notação UML (OMG, 2003))

- **Identificação de CIs (*CI Identification*):** O CCB define quais tipos de informações do projeto serão controlados e irão compor as *baselines* do projeto. Estes tipos são definidos com base nos tipos de CIs definidos na atividade "Identificação de Tipos de CIs", que é executada fora do escopo do projeto.

- **Identificação da Estrutura (*Structure Identification*):** o Gerente do Projeto define os módulos do projeto, e o Bibliotecário define a estrutura da biblioteca de desenvolvimento associada a cada módulo definido.

- **Abertura de Nova *Baseline* (*New Baseline Open*):** o Gerente do Projeto abre uma nova *baseline* dentre as identificadas na atividade "Identificação de *Baselines*" para que os Engenheiros de Software possam realizar o desenvolvimento de seus artefatos. Para tanto, a *baseline* não pode já ter sido aberta e não deve haver nenhuma outra aberta no projeto, pois a abordagem estabelece que apenas uma *baseline* pode estar sendo desenvolvida num dado momento do projeto.

- **Aquisição de CIs (*CIs Acquisition*):** Os Engenheiros de Software realizam a entrega dos artefatos para que estes se tornem CIs. Estas informações são avaliadas pelo Auditor com base nos critérios de aprovação de CIs (*CI Approval Criterion*) definidos na atividade "Identificação de Tipos de CIs".

- **Aquisição da *Baseline* (*Baseline Acquisition*):** O Gerente do Projeto solicita a aquisição da

baseline que é avaliada pelo Auditor com base nos critérios de aprovação (*Baseline Approval Criterion*) definidos na atividade “Identificação de *Baselines*”.

3.2.3.2 Controle da Configuração

O Controle da Configuração permite a alteração das *baselines* e, conseqüentemente, dos CIs do projeto. Portanto, esta atividade envolve atividades para realizar desde requisições de mudança até a implementação das mudanças.

Na Figura 14 são mostradas as atividades de Controle da Configuração.

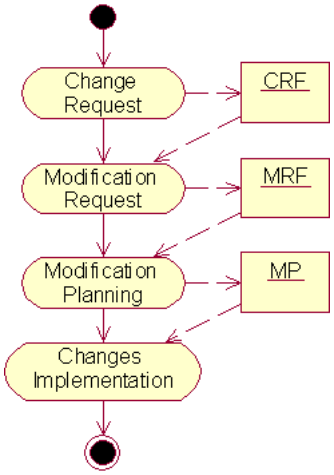

Figura 14: Atividade de Controle da Configuração (Notação UML (OMG, 2003))

- **Requisição de Mudança (*Change Request*):** O Engenheiro de Software relata um problema detectado ou uma melhoria requerida em uma dada *baseline* do projeto de software, preenchendo um formulário para requisição de mudanças (*Change Request Form - CRF*).

- **Requisição de Modificação (*Modification Request*):** O CCB avalia as requisições de mudança quanto a sua pertinência, seleciona aquelas similares para que possam compor uma requisição de modificação e preenche um formulário para realizar esta requisição (*Modification Request Form - MRF*). Além disso, o CCB define uma equipe de manutenção, formada por Engenheiros de Software, que irá ser responsável por planejar e realizar a modificação.

- **Planejamento da Modificação (*Modification Planning*):** A equipe de manutenção planeja a modificação, preenchendo um Plano de Modificação (*Modification Plan - MP*) com base na requisição de modificação e, neste planejamento, selecionam os CIs da *baseline* a serem alterados. Para tanto, o CMO realiza a análise de impacto das mudanças e o Bibliotecário disponibiliza os CIs da biblioteca de produção do projeto.

- **Implementação de Mudanças (*Changes Implementation*):** a equipe de manutenção realiza as mudanças necessárias sobre os CIs, denominados artefatos em manutenção, presentes no Plano de Modificação. No momento em que a implementação é concluída, o Auditor avalia os artefatos em manutenção segundo os critérios de aprovação associados a cada tipo dos artefatos. Uma vez aprovados, estes artefatos passam novamente a serem referenciados como CIs, compondo a *baseline* do projeto.

3.2.3.3 Auditoria da Configuração

As auditorias da configuração são muitas vezes dependentes do contrato que rege o projeto e devem ser bem planejadas e definidas, podendo requisitar a atuação de vários profissionais que irão avaliar a configuração do projeto. Elas podem ser realizadas em diferentes momentos do desenvolvimento e envolvem o trabalho do Auditor.

Na Figura 15 são mostradas as atividades de Auditoria da Configuração.

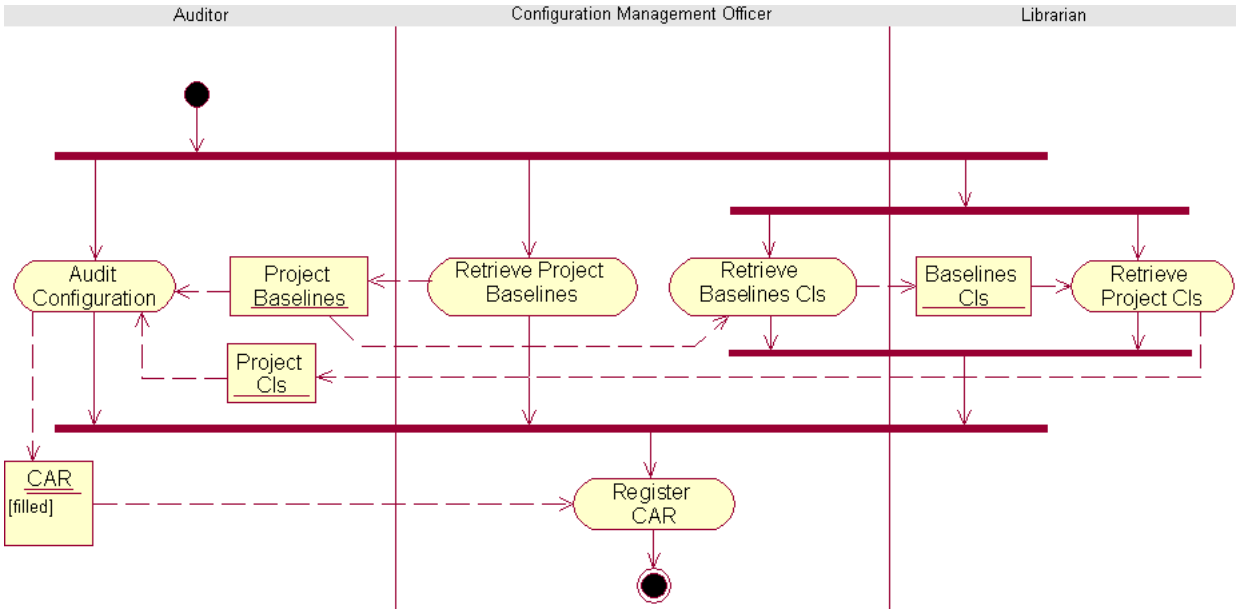

Figura 15: Atividades de Auditoria da Configuração (Notação UML (OMG, 2003))

- **Auditar Configuração (*Audit Configuration*):** O Auditor realiza a auditoria sobre as *baselines* e CIs fornecidos pelo CMO e pelo Bibliotecário. Para concluir a auditoria, o Auditor deve preencher um Registro da Auditoria da Configuração (*Configuration Audit Register - CAR*).

- **Obter as *Baselines* do Projeto (*Retrieve Project Baselines*):** O CMO obtém as *baselines* já estabelecidas e a que está em aberto, disponibilizando-as para o Auditor.

- **Obter os CIs das *Baselines* (*Retrive Baselines CIs*):** O CMO obtém os CIs de cada *baseline* obtida na atividade "Obter as *Baselines* do Project". A informação sobre estes CIs é disponibilizado ao Bibliotecário para que este possa obtê-los da biblioteca de software.

- **Obter os CIs do Projeto (*Retrive Project CIs*):** O Bibliotecário, com base nas informações sobre os CIs das *baselines*, disponibilizados pelo CMO, obtém da biblioteca de software os arquivos para que o Auditor possa realizar a auditoria.

- **Registrar CAR (*Register CAR*):** Concluída a atividade "Auditar Configuração", o CMO recebe o CAR preenchido e efetua seu registro.

3.2.3.4 Administração de Estado

A atividade de Administração de Estado possibilita que os membros da equipe do projeto estejam cientes sobre o *status* das *baselines*, dos artefatos e de quaisquer outras informações referentes ao GCS. Nesta abordagem, tem-se que o CMO obtém estas informações e as fornece por meio de relatórios. Dentre estes relatórios definidos na abordagem, destacam-se:

- **Relatório sobre Tipos (*Types Report*):** Lista todos os tipos de CI da empresa. Além disso, apresenta uma lista de tipos de CIs utilizados em cada um dos projetos.

- **Relatório sobre Artefatos (*Artifacts Report*):** Lista todos os artefatos, com suas respectivas versões (indicando quais deles são CIs) e todas as aquisições de CIs realizadas sobre eles.

- **Relatório de Mudanças (*Changes Report*):** Lista todas as requisições de mudança, além das solicitações e planejamentos de modificação gerados para um dado projeto.

- **Relatório de Baselines (*Baselines Report*):** Lista todas as *baselines* de um dado projeto, assim como dos CIs que as constituem.

- **Relatório de CIs (*CIs Report*):** Apresenta informações sobre a evolução de todos os CIs, mostrando o seu histórico, das atividades de Aquisição de CIs realizadas, e dos planos de modificação relacionados a eles.

- **Relatório de Auditorias de Configuração (*Configuration Audit Report*):** Apresenta informações das auditorias realizadas na configuração do projeto, os CIs e *baselines* envolvidos, além do resultado associado a cada um deles (aprovado ou reprovado).

- **Relatório de Projetos (*Projects Report*):** apresenta informações relacionadas à configuração do projeto específico, com todas as suas *baselines*, CIs e distribuições.

3.2.3.5 Identificação de Tipos de Itens de Configuração

Esta atividade envolve definir os tipos de artefatos que podem estar sendo colocados sob controle. No caso, o PEB realiza a identificação de quais tipos de CIs podem estar sendo controlados e o QAB define, para cada tipo, os critérios de aprovação que serão utilizados pelo Auditor no projeto.

A Figura 16 ilustra a atividade de Identificação de Tipos de CIs.

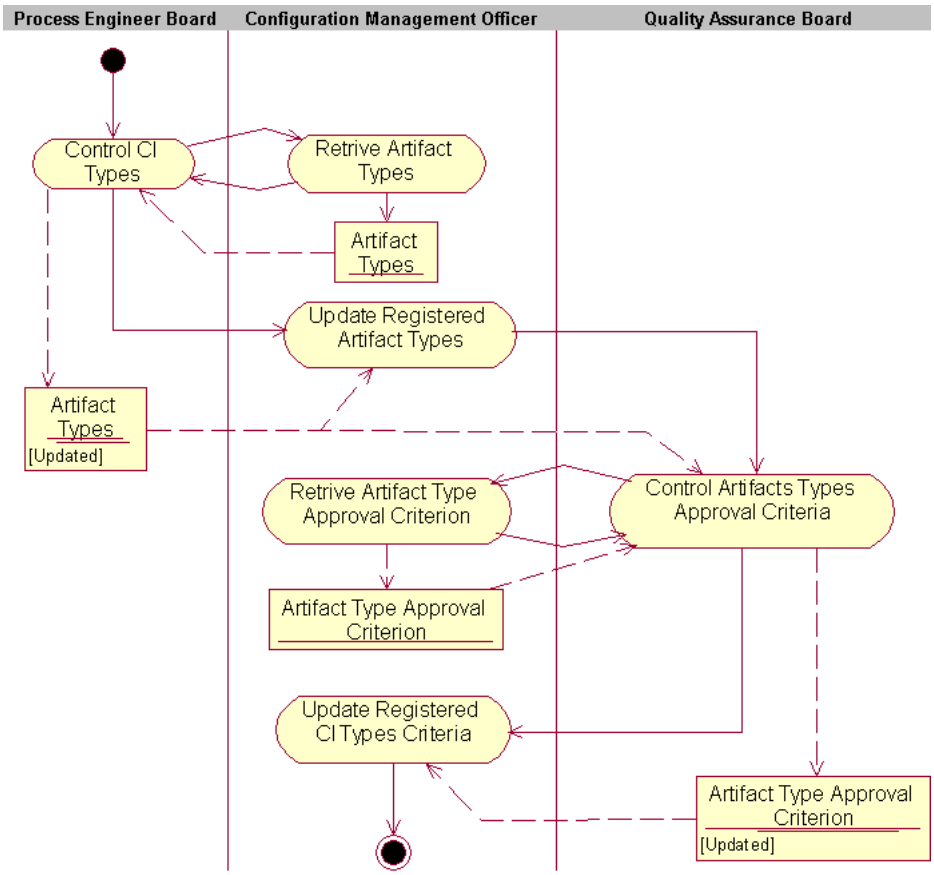

Figura 16: Atividades de Identificação de Tipos de CIs

- **Controlar Tipos de CIs (*Control CI Types*):** O PEB adiciona, altera ou remove os tipos de CIs. Para tanto, ele utiliza como base a lista de tipos de CIs existente fornecida pela atividade "Obter Tipos de CIs".
- **Obter Tipos de CIs (*Retrieve CI Types*):** O CMO obtém os tipos de CIs já cadastrados, disponibilizando-as para o PEB.
- **Atualizar os Tipos de CIs Registrados (*Update Registered CI Types*):** O CMO atualiza os tipos de CIs cadastrados, remove os excluídos e adiciona os novos.

- **Obter os Critérios de Aprovação dos Tipos de CIs (*Retrive CI Types Approval Criteria*):** O CMO obtém os critérios de aprovação, relacionados aos tipos de CIs, já cadastrados. Estes critérios são então fornecidos ao QAB.

- **Controlar Critérios de Aprovação de Tipos de CIs (*Control CI Types Approval Criteria*):** O QAB atualiza, altera ou remove os critérios de aprovação dos tipos de CIs.

- **Atualizar os Critérios de Aprovação de Tipos de CIs cadastrados (*Update Registered CI Types Criteria*):** O CMO atualiza os critérios de aprovação dos tipos de CIs cadastrados, remove os excluídos e adiciona os novos.

3.2.4 Aderência ao Modelo CMMi

A abordagem do processo, além de reunir as práticas das principais abordagens de GCS existentes, também teve como base as metas da PA de Gerenciamento de Configuração do nível 2 do modelo de capacitação CMMi, descritas em (CHRISSIS; KONRAD; SHRUM, 2003). As atividades dessa abordagem apóiam a execução das práticas destas metas conforme mostra a tabela 10:

Tabela 10: Atividades da Abordagem de GCS relacionadas com as Práticas da PA de Gerenciamento de Configuração do CMMi

Metas do CMMi	**Práticas do CMMi**	**Atividades da Abordagem**
Estabelecer *Baselines*	Identificar CIs	Identificação de CIs Aquisição de CIs
Estabelecer *Baselines*	Estabelecer um Sistema de Gerenciamento de Configuração	Identificação da Estrutura
Estabelecer *Baselines*	Criar ou Liberar *Baselines*	Identificação de *Baselines* Estabelecimento de *Baselines*
Rastrear e Controlar Mudanças	Rastrear as Requisições de Mudança	Requisição de Mudança Planejamento de Modificação
Rastrear e Controlar Mudanças	Controlar os CIs	Implementação da Mudança
Estabelecer Integridade	Estabelecer Registros do Gerenciamento de Configuração	Aquisição de CIs Identificação de *Baselines* Administração de Estado
Estabelecer Integridade	Realizar Auditorias sobre a Configuração	Auditoria da Configuração

- **Estabelecer *Baselines*:** Esta meta especifica práticas para estabelecer as *baselines* do projeto. Nela são definidas 3 práticas:

  - **Identificar CIs:** Envolve identificar os itens de configuração, componentes e artefatos que estarão sob gerenciamento de configuração. A abordagem apóia esta prática por meio das atividades de: Identificação de CIs, que define o que estará sendo controlado; e Aquisição de CIs, que permite colocar os artefatos do projeto sob controle.
  - **Estabelecer um Sistema de Gerenciamento de Configuração:** Envolve estabelecer e manter um sistema para gerenciamento de configuração e gerenciamento de mudanças para os artefatos controlados do projeto. A abordagem apóia esta prática por meio da atividade de "Identificação da Estrutura do Projeto", que permite definir como estarão sendo armazenados os artefatos. No entanto, há a necessidade de se ter o apoio de uma ferramenta computacional que permita atender tanto ao gerenciamento de configuração quanto ao gerenciamento de mudanças.
  - **Criar ou Liberar *Baselines*:** Envolve criar ou liberar *baselines* para uso interno ou para distribuir ao cliente. A abordagem apóia esta prática por meio das atividades de "Identificação de *Baselines*", que permite definir as *baselines* do projeto, e de "Aquisição de *Baselines*", que permite que estas *baselines* possam estar sendo estabelecidas e disponibilizadas para distribuição ou uso em outros projetos.

- **Rastrear e Controlar Mudanças:** Esta meta especifica práticas que permitem controlar e rastrear as mudanças nos CIs. Nela são definidas 2 práticas:

  - **Rastrear as Requisições de Mudança:** Envolve criar as requisições de mudanças sobre os CIs, rastreá-las e analisar o seu impacto sobre o produto. A abordagem apóia esta prática por meio das atividades de "Requisição de Mudança", para criar novas requisições sobre os CIs e *baselines* do projeto, de "Planejamento da Modificação", para realizar a análise de impacto de uma modificação, e de "Administração de Estado" que permite visualizar o *status* das requisições de mudança.
  - **Controlar os CIs:** Envolve rastrear a configuração de cada CI, aprovar novas configurações, se necessário, e atualizar a *baseline*. A abordagem apóia esta prática por meio da atividade de "Implementação de Mudança" que permite que os CIs possam ser alterados e que realiza a atualização da *baseline* que está sendo alterada.

- **Estabelecer Integridade:** Esta meta especifica práticas que permitem estabelecer e manter a integridade sobre as *baselines* do projeto. Nela são definidas 2 práticas:

  - **Estabelecer Registros do Gerenciamento de Configuração:** Envolve estabelecer e manter registros que descrevem os CIs. A abordagem apóia esta prática por meio das

atividades de: Aquisição de CIs, que permite descrever a configuração do CI; Implementação da Mudança, que permite realizar a alteração sobre a configuração do CI; e Administração de Estado, que permite obter relatórios sobre status de CIs e diferenças entre *baselines* do projeto.

– **Realizar Auditorias sobre a Configuração:** Realizar auditorias para manter a integridade sobre as *baselines*. A abordagem apóia esta prática por meio da atividade de Auditoria da Configuração, que permite verificar tanto a consistência dos CIs em relação à especificação do projeto (auditoria funcional) quanto para garantir que a documentação do projeto está em conformidade com o produto desenvolvido (auditoria física).

3.3 Ferramenta *SoCManager*

SoCManager (*Software Configuration Manager*) é uma ferramenta computacional de apoio ao GCS que tem como objetivo auxiliar empresas de software em implantar este processo no desenvolvimento de seus projetos de software.

Grande parte das ferramentas de GCS existentes dá suporte apenas a atividades específicas do processo de GCS como, por exemplo, controle de versões e de mudanças. Por este motivo, muitas empresas de software acabam por adotar ferramentas de GCS em conjunto para dar suporte a este processo de acordo com as suas necessidades.

No entanto, isto acaba elevando as dificuldades em operacionalizar o processo dentro da empresa, surgindo a necessidade de se ter pessoas que atuem na garantia da correta execução do processo. Conseqüentemente, acaba-se dependendo muito mais de pessoas do que de ferramentas e do processo em si, acarretando num maior custo em implantá-lo.

Neste contexto, a *SoCManager* possui como diferencial o fato de ter como objetivo apoiar todas as atividades definidas na abordagem do processo, automatizando as responsabilidades referentes ao CMO.

3.3.1 Funcionalidades da *SoCManager*

Das atividades de GCS definidas na abordagem, a *SoCManager* atualmente apóia aquelas relacionadas à Identificação e Controle de Configuração e Identificação de Tipos de CIs. Além disso, a ferramenta possui funcionalidades que permitem o gerenciamento de projetos nela cadastrados. Este gerenciamento engloba: o controle da equipe de desenvolvimento e o papel de cada membro

desta equipe no projeto; e o controle dos módulos do projeto, definindo quais dos membros que atuam como Engenheiros de Software são responsáveis pelo desenvolvimento de cada módulo.

A *SoCManager* permite o cadastro de novos membros, que podem tanto realizar o controle de seu cadastro quanto criar projetos de software que estarão tendo seu GCS apoiado pela ferramenta.

Um membro pode atuar em vários projetos e, podendo desempenhar mais de um papel em cada projeto. Estes papéis, referentes aos agentes definidos na abordagem, são: Gerente do Projeto, Engenheiro de Software, Bibliotecário, Membro do Comitê de Controle de Configuração (referente ao CCB definido na Abordagem) e Auditor. Cada um desses papéis são associados a perspectivas que dão acesso a determinados tipos de funcionalidades referentes às suas responsabilidades, definidas na abordagem do processo.

Mostra-se a seguir o diagrama de contexto dividido segundo as perspectivas definidas na ferramenta, assim como o relacionamento de cada caso de uso com as atividades da abordagem do processo, quando houver. A descrição de todos os casos de uso pode ser encontrada no *site* `http://recope.dc.ufscar.br/socmanager/`.

Na Figura 17 são mostradas as funcionalidades disponíveis aos membros cadastrados na *SoCManager*. Por se tratar de funcionalidades específicas da ferramenta, não referentes ao processo de GCS, não há nenhuma relação entre estes casos de uso e as atividades de GCS definidas na abordagem do processo.

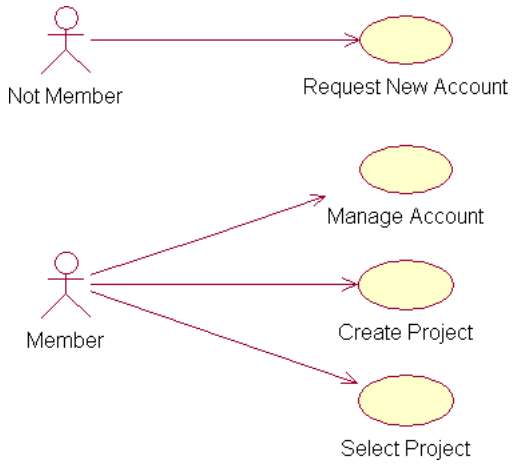

Figura 17: Diagrama de Contexto referente ao Membro da SoCManager

Na Figura 18 são mostradas as funcionalidades disponíveis aos membros que atuam num determinado projeto como Gerentes do Projeto (*Project Manager*). Esta perspectiva, denominada Perspectiva do Gerente do Projeto, é acessível quando um determinado membro acessar um projeto por ele cadastrado (por meio do caso de uso “Criar Projeto” - *Create Project*). Neste caso, existem casos de uso que estão relacionados com atividades do processo de GCS, conforme mos-

tra a Tabela 11.

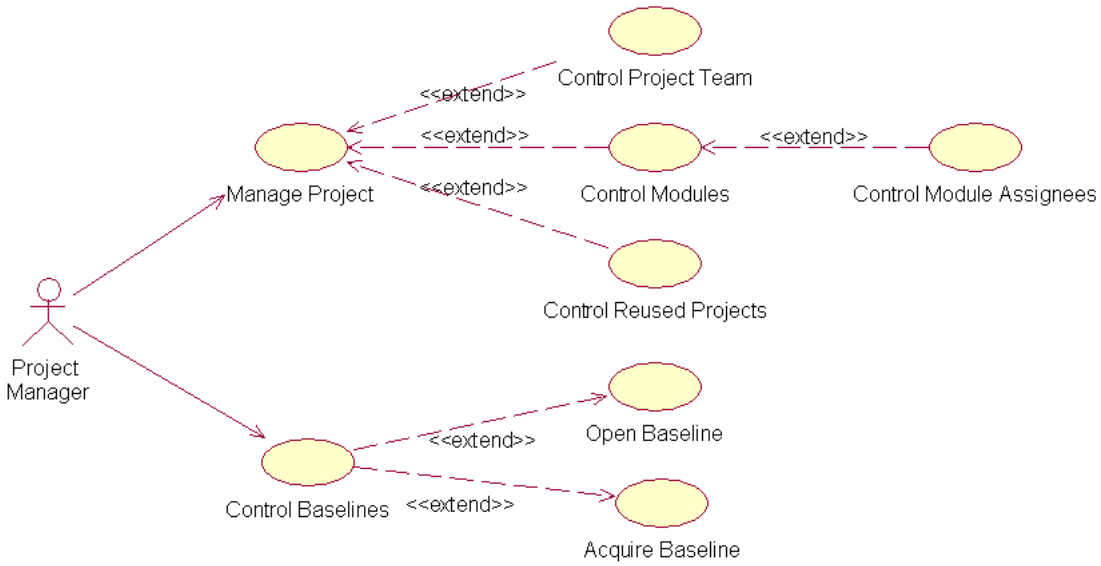

Figura 18: Diagrama de Contexto da Perspectiva do Gerente do Projeto

Tabela 11: Relação entre os Casos de Uso da Perspectiva Gerente do Projeto e as atividades da abordagem do processo

Caso de Uso	**Descrição do Caso de Uso**	**Atividade GCS**
Controlar *Baselines* (*Control Baselines*)	Adicionar, alterar e remover as *baselines* do projeto	Identificação de *Baselines*
Controlar Módulos (*Control Modules*)	Adicionar, alterar e remover os módulos do projeto	Identificação da Estrutura
Controlar Designações do Módulo (*Control Module Assignees*)	Adicionar, alterar e remover Engenheiros de Software que atuam no módulo	Identificação da Estrutura
Abrir *Baseline* (*Open Baseline*)	Abrir uma nova *baseline* para o projeto	Abertura de Nova *Baseline*
Adquirir *Baselines* (*Acquire Baseline*)	Realizar a aquisição de uma dada *baseline* em aberto do projeto	Aquisição da *Baseline*

A exemplo do papel Gerente do Projeto, a *SoCManager* estabelece, para cada um dos papéis de GCS nela definidos, a exemplo do papel Gerente do Projeto, uma perspectiva que fornece acesso a funcionalidades específicas do processo de GCS conforme estabelecido na abordagem do processo.

Dessa forma, para que um membro realize as atividades de GCS num dado projeto, ele deve selecioná-lo (caso de uso “Selecionar Projeto” - *Select Project*) dentre os projetos em que ele atua e, dependendo de seus papéis no projeto, ele terá acesso às perspectivas definidas pela *SocManager*.

Para os membros com o papel de Bibliotecário (*Librarian)*, tem-se as funcionalidades referentes ao controle de bibliotecas de software (Figura 19). A tabela 12 mostra a relação dos casos de uso do Bibliotecário com a abordagem do processo.

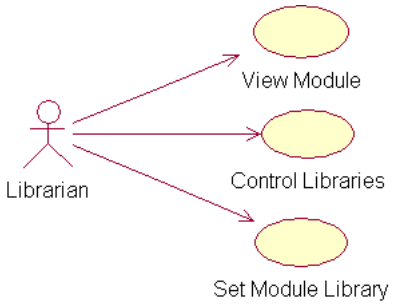

Figura 19: Diagrama de Contexto da Perspectiva do Bibliotecário

Tabela 12: Relação entre os Casos de Uso da *SoCManager* e as atividades da abordagem do processo

Caso de Uso	**Descrição do Caso de Uso**	**Atividade GCS**
Controlar Bibliotecas (*Control Libraries*)	Adicionar, alterar e remover bibliotecas de software utilizadas no projeto	Identificação da Estrutura
Definir a Biblioteca do Módulo (*Set Module Library*)	Associar um módulo do projeto a uma determinada biblioteca	Identificação da Estrutura
Visualizar Módulo (*View Module*)	Visualizar informações de um dado módulo do projeto	Identificação da Estrutura

Para os membros com o papel de Engenheiro de Software (*Software Engineer*), tem-se as funcionalidades referentes ao desenvolvimento, aquisição e manutenção de artefatos de software (Figura 20). A tabela 13 mostra a relação dos casos de uso do Engenheiro de Software com a abordagem do processo.

Tabela 13: Relação entre os casos de uso do Engenheiro de Software e as atividades da abordagem do processo

Caso de Uso	**Descrição do Caso de Uso**	**Atividade GCS**
Requisitar Aquisição de CIs (*Request CIs Acquisition*)	Requisitar Aquisição de CIs, selecionando os artefatos a serem colocados sob controle	Aquisição de CIs
Requisitar Mudança (*Request Change*)	Requisitar mudanças sobre uma dada *baseline*, em aberto ou já adquirida, do projeto.	Requisição de Mudança
Planejar Modificação (*Plan Modification*)	Planejar uma modificação que foi solicitada	Planejamento da Modificação
Analisar Impacto (*Analyze Impact*)	Verificar quais outros CIs poderão sofrer impactos caso um dado CI seja modificado.	Planejamento da Modificação
Concluir Modificação (*Conclude Modification*)	Concluir uma dada modificação	Implementação de Mudanças
Controlar Artefatos em Manutenção (*Control Maintenance Artifacts*)	Alterar artefatos de uma dada modificação	Implementação de Mudanças

Para os membros com o papel de Membro do Comitê de Controle de Configuração (*Configuration Control Board Member*), tem-se as funcionalidades referentes à definição de tipos de artefatos

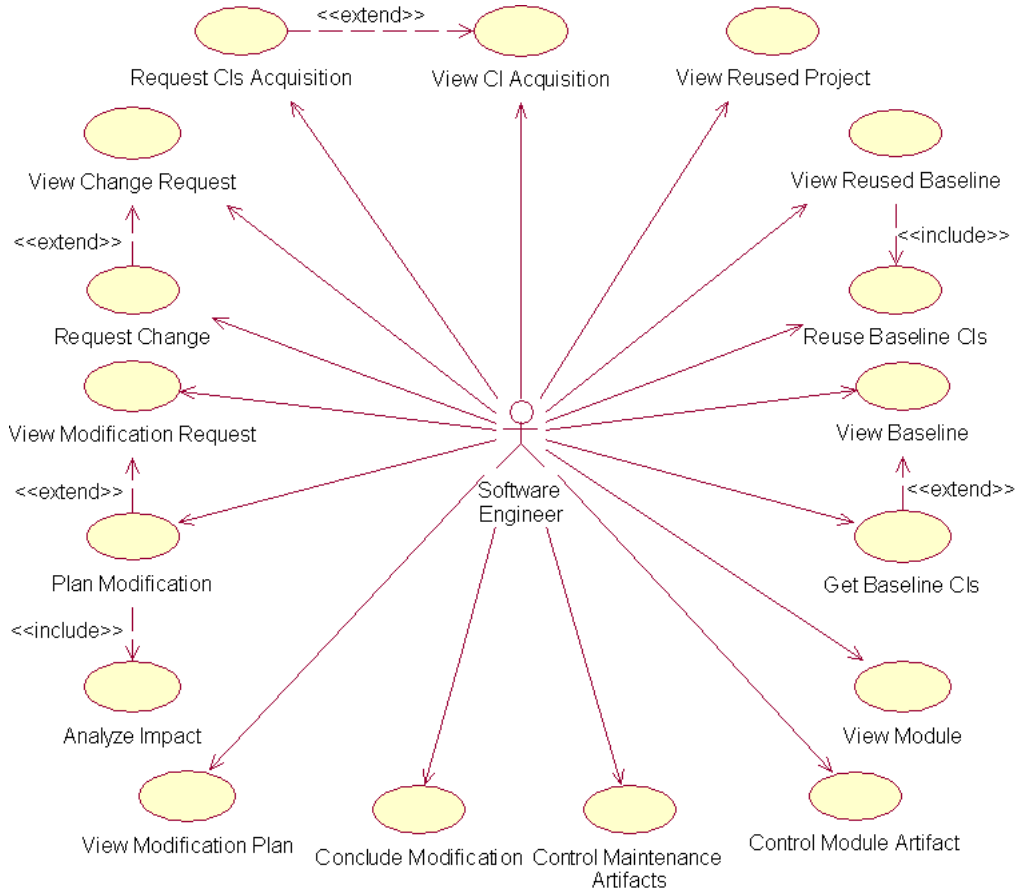

Figura 20: Diagrama de Contexto da Perspectiva Engenheiro de Software

e controle das mudanças realizadas na configuração do software (Figura 21). A tabela 14 mostra a relação dos casos de uso do Membro do Comitê de Controle de Configuração com a abordagem do processo.

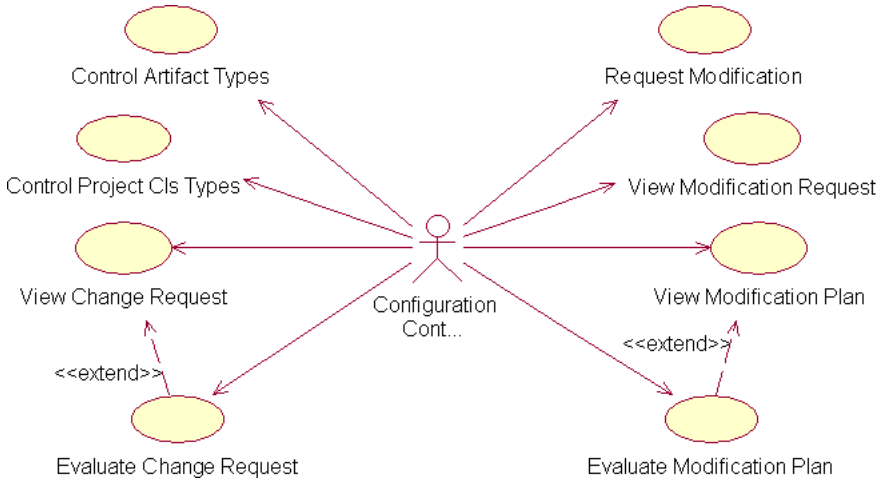

Figura 21: Diagrama de Contexto da Perspectiva Membro do Comitê de Controle de Configuração

Contudo, diferentemente do que foi definido na abordagem do processo, onde a atividade de "Identificação de Tipos de CIs" é executada pelo Comitê de Engenharia de Processo, na *SoCMana-*

Tabela 14: Relação entre os Casos de Uso da *SoCManager* do Membro do Comitê de Controle de Configuração e as atividades da abordagem do processo

Caso de Uso	**Descrição do Caso de Uso**	**Atividade GCS**
Controlar os Tipos de CIs do Projeto (*Control Project CIs Types*)	Adicionar, alterar e remover os Tipos de CIs controlados no projeto	Identificação de CIs
Avaliar Requisição de Mudança (*Evaluate Change Request*)	Aprovar ou reprovar uma dada requisição de mudança	Requisição de Mudança
Requisitar Modificação (*Request Modification*)	Requisitar a modificação sobre a *baseline* a partir de requisições de mudança	Requisição de Modificação
Avaliar Planejamento da Modificação (*Evaluate Modification Plan*)	Aprovar ou reprovar um planejamento da modificação	Planejamento da Modificação
Controlar os Tipos de CIs (*Control CIs Types*)	Adicionar, alterar e remover os tipos de CIs	Identificação de Tipos de CIs

ger esta atividade, que está relacionada com o caso de uso "Controlar os Tipos de CIs", está sendo desempenhada pelo Membro do Comitê de Controle de Configuração. Isso se deve ao fato de o apoio às atividades do Comitê de Engenharia de Processos exigir a implementação de um módulo administrativo, previsto para uma próxima etapa no desenvolvimento da *SoCManager*, citada no capítulo onde são descritos os trabalhos futuros (Capítulo 5. Para mesmo assim dar suporte à atividade de "Identificação de Tipos de CIs", importante para se ter o apoio à abordagem do processo, decidiu-se então colocá-la sob responsabilidade do Comitê de Controle de Configuração.

Já para os membros com o papel de Auditor (*Auditor*), tem-se as funcionalidades referentes à auditoria de artefatos cujo controle foi solicitado e de artefatos controlados que foram modificados (Figura 22). A tabela 15 mostra a relação dos casos de uso do Auditor com a abordagem do processo.

Tabela 15: Relação entre os casos de uso do Auditor e as atividades da abordagem do processo

Caso de Uso	**Descrição do Caso de Uso**	**Atividade GCS**
Auditar Artefatos da Aquisição de CIs (*Audit CI Acquisition Artifacts*)	Verificar conformidade dos CIs de uma dada requisição de aquisição de CIs com a configuração do software	Aquisição de CIs
Auditar Artefatos em Manutenção (*Audit Maintenance Artifacts*)	Auditar os artefatos referentes a uma modificação, aprovando ou não a modificação	Implementação de Mudanças

Com base nestas funcionalidades oferecidas pela *SoCManager*, ela pode ser categorizada como uma ferramenta de apoio ao Gerenciamento de Configuração e de Mudanças, conforme as categorias definidas por Sommerville (SOMMERVILLE, 2003) citadas na seção 2.5.

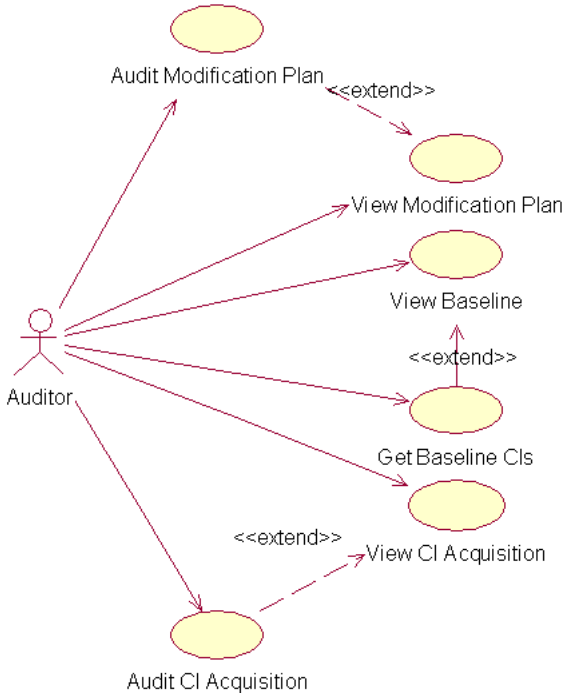

Figura 22: Diagrama de Contexto da Perspectiva do Auditor

3.3.2 Arquitetura da *SoCManager*

A *SoCManager* foi construída para ser utilizada por equipes de desenvolvimento geograficamente distribuídos e compartilhando, entre eles, as informações dos projetos de software cadastrados.

Para construir a ferramenta, foram utilizadas tecnologias referentes à linguagem de programação Java (MICROSYSTEMS, 2004b) para WEB, como JSP (*Java Server Pages*) e *Servlets*, que dão suporte à construção de aplicações distribuídas. Com isso, tem-se o uso de uma linguagem gratuita e multi-plataforma, com um conjunto grande de bibliotecas de apoio ao desenvolvimento, além de se aproveitar a existência de um grande número de comunidades que oferecem auxílio ao desenvolvimento de aplicações utilizando-se esta linguagem.

Na Figura 23 é mostrada a arquitetura da *SoCManager*, que utiliza a Intranet ou Internet como meio de comunicação entre o usuário e o servidor.

O módulo *Socmanager Server* processa as requisições do usuário e efetua o gerenciamento sobre as informações dos projetos de software cadastrados na *SoCManager*. Além disso, tem-se neste módulo parte das interfaces disponibilizadas ao usuário. O módulo *Value Object* é utilizado para transferir as informações da ferramenta entre os demais módulos e portanto está presente tanto no lado usuário como no lado servidor. O módulo *Repository* efetua o acesso aos repositórios utilizados pelos projetos para armazenar os artefatos de software, sendo utilizado também tanto no

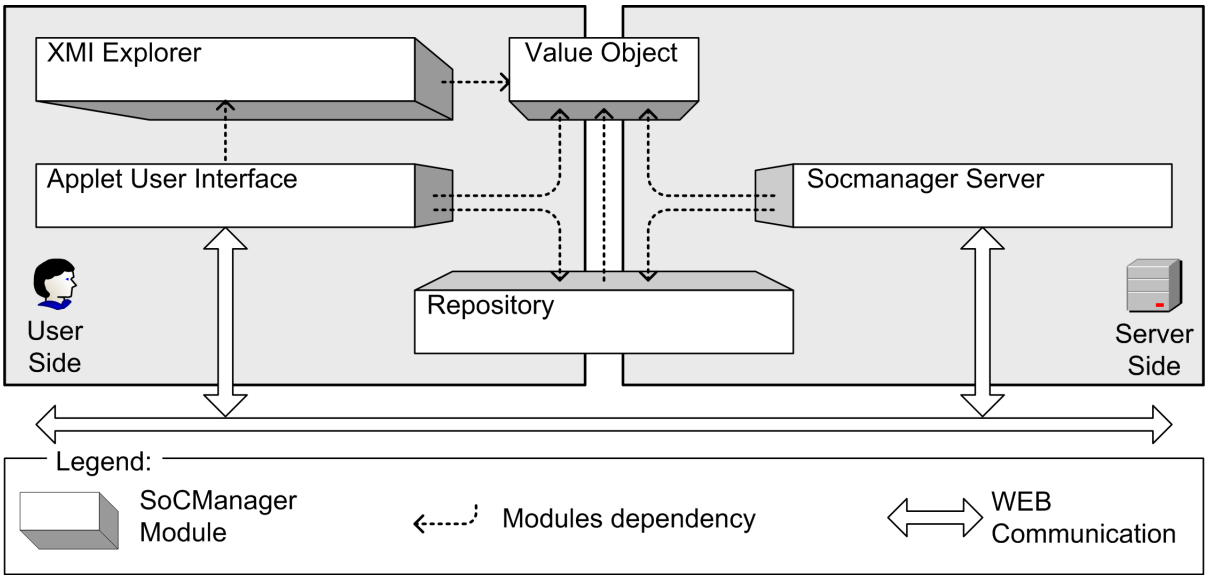

Figura 23: Arquitetura da *SoCManager*

lado usuário como no lado servidor. Já no lado usuário, tem-se o módulo *Applet User Interface*, que contem algumas das interfaces disponibilizadas ao usuário e que realiza operações nos arquivos presentes na máquina do usuário, e o módulo *XMI Explorer*, que realiza a introspecção sobre artefatos de software descritos segundo o padrão XMI (OMG, 2002b). Segue a descrição de cada módulo da *SoCManager*.

3.3.2.1 *Value Object*

Neste módulo tem-se a representação de toda informação da *SoCManager*, que é utilizada pelos demais módulos da ferramenta. Esta informação se baseia nos conceitos definidos pela abordagem para o processo de GCS (descritos na seção 3.2) que foram mostrados na figura 11. Este módulo está presente tanto no lado usuário como no lado servidor.

Para implementar este módulo, foi utilizado o padrão conhecido como *Value Object* (VO) (MICROSYSTEMS, 2004d). Decidiu-se utilizar o padrão VO, pois além dele facilitar a troca de informações entre os diferentes módulos da ferramenta, o seu uso implica numa minimização de duplicação de código.

Seguem os VOs que compõem este módulo, divididos em quatro partes:

- **Projeto:** VOs que representam informações de projeto, membros, papéis dos membros, módulos, entre outros;
- **GCS:** VOs que representam informações de artefatos, *baselines*, aquisições de CIs, requisições de mudança, auditagens, entre outros;

- **Repositório:** VOs que representam informações das bibliotecas de software do projeto.
- **Comuns:** VOs que representam informações comuns da ferramenta e que não se enquadram em Projeto, GCS e Repositório.

Na Figura 24 são mostrados os VOs referentes ao projeto.

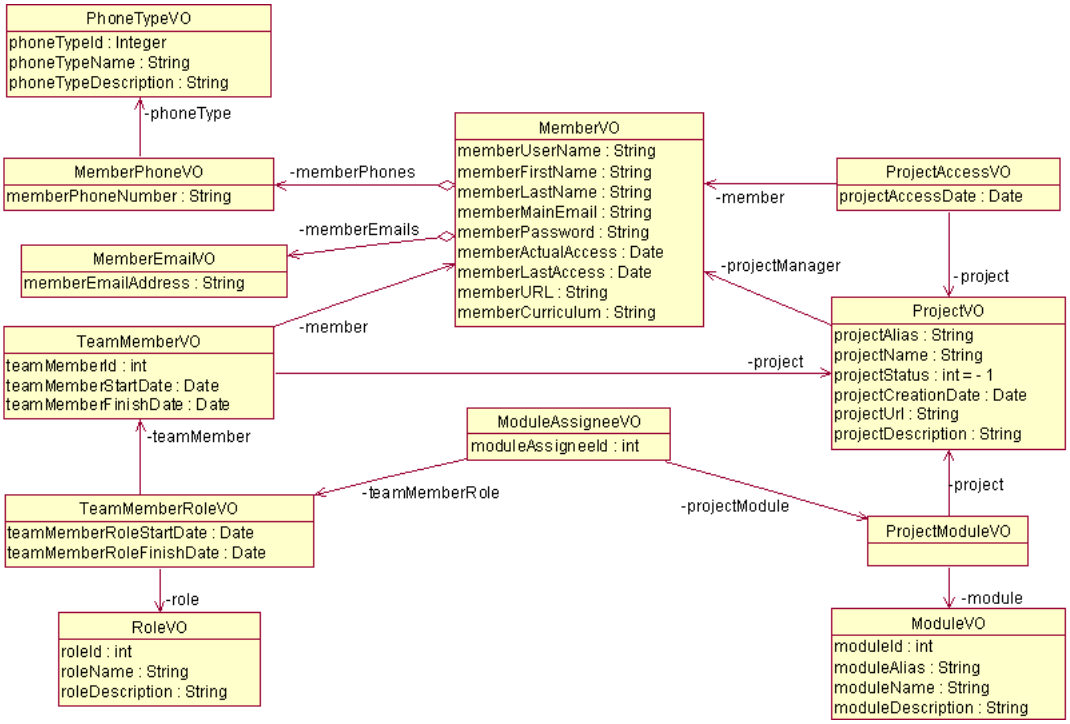

Figura 24: *Value Objects* referentes à parte “Projeto” do módulo “*Value Object*” da *SoCManager*

3.3.2.2 *Repository*

Neste módulo tem-se o acesso aos repositórios onde são armazenados os artefatos de software e onde é realizado o controle de versão sobre estes artefatos. Este módulo está presente tanto no lado usuário como no lado servidor.

Atualmente, este módulo realiza o acesso aos repositórios do CVS, ferramenta citada brevemente na seção 2.5.

Decidiu-se implementar o acesso primeiramente aos repositórios do CVS devido a sua popularidade e também porquê os trabalhos que utilizam o CVS como base possuem maior chance de serem disseminados na comunidade de desenvolvimento. Outro motivo é a existência de diversos recursos, para a linguagem de programação Java e para outros softwares que oferecem suporte ao CVS, facilitando o acesso às suas funcionalidades. Além disso, as outras ferramentas de GCS que propõem melhorias em relação ao CVS possuem basicamente os mesmos comandos. Sendo assim, uma possível adaptação para essas ferramentas não seria muito trabalhosa.

Para efetuar o acesso aos repositórios do CVS, a *SoCManager* teria de implementar o lado usuário,conforme o protocolo de comunicação do CVS, ou utilizar um software "cliente" já existente. No caso decidiu-se utilizar um "cliente" do CVS já existente: o javacvs (NETBEANS, 2004a) desenvolvido para a linguagem de programação Java e que é parte do projeto *Netbeans* (NETBEANS, 2004b).

Os principais comandos do CVS utilizados pelo módulo *Repository* são:

- ***import*:** Utilizado para inserir artefatos em um dado repositório do CVS pela primeira vez. Este comando cria um módulo de armazenamento contendo todos os artefatos inseridos;
- ***add*:** Utilizado para inserir um novo artefato em um dado repositório do CVS, quando já existir um módulo criado por um comando import;
- ***remove*:** Utilizado para remover um artefato de um dado repositório do CVS.
- ***update*:** Recupera os artefatos do repositório e os atualiza no cliente. Além disso, este comando permite realizar a união de diferentes versões de um dado artefato armazenado no repositório do CVS;
- ***commit* ou *check-in*:** Envia os artefatos que estão no cliente para o repositório. Caso haja alguma diferença entre o artefato sendo enviado e o que está armazenado no repositório, é gerada uma nova versão no repositório sendo que a versão antiga permanece armazenada;
- ***check-out*:** Recupera os artefatos do repositório para o cliente. A diferença entre o check-out e o update é que o primeiro considera que os artefatos não existem no cliente;
- ***tag*:** Colocação de etiquetas sobre os artefatos presentes no repositório. Estas etiquetas são postas em versões específicas de cada artefato e podem tanto representar uma informação referente àquela versão do artefato ou a criação de uma nova ramificação (*branch*) no grafo de versões[1] do artefato. Este último é utilizado principalmente para permitir o desenvolvimento paralelo.

Esse conjunto de comandos permitem que diferentes versões de artefatos sejam gerenciadas no repositório, sendo que a maneira e a seqüência com que são utilizados é importante.

[1]Forma de representação de espaços de versões utilizada por várias ferramentas de GCS e que consiste de nós e arcos que correspondem a um conjunto de versões e seus relacionamentos, respectivamente (CONRADI; WESTFECHTEL, 1998).

3.3.2.3 *SoCManager Server*

Neste módulo tem-se a geração das interfaces, o processamento das requisições do usuário, a realização das regras de negócio e a persistência das informações em um banco de dados. Este módulo está presente apenas no lado servidor.

Ele foi desenvolvido com base na arquitetura conhecida como MVC (*Model-View-Controller* ou Modelo-Visão-Controle) (BURBECK, 2002) a qual divide um sistema em três partes distintas (Figura 25):

- **Visão (*View*):** realiza o gerenciamento das informações que são disponibilizadas ao usuário por meio de uma interface (*Presentation*);
- **Controle (*Controller*):** obtém as entradas fornecidas pelo usuário e, baseado nestas, delega ações a serem executadas tanto pela Visão como pelo Modelo; e
- **Modelo (*Model*):** realiza o gerenciamento do comportamento (*Business*) e informações (*DataBase*) do domínio da aplicação, respondendo a requisições sobre o estado das informações, normalmente advindas da Visão, e a ações para mudança de estado, normalmente advindas do controle.

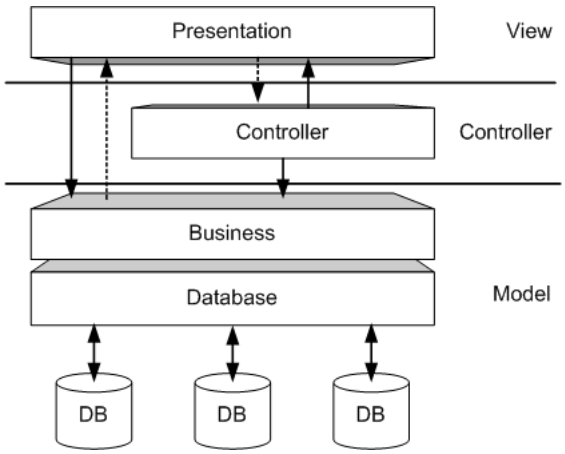

Figura 25: Arquitetura MVC (Adaptado de (MICROSYSTEMS, 2004c))

Escolheu-se a arquitetura MVC considerando que seus conceitos provêem uma divisão clara da aplicação em camadas de tal forma que estas possam ser trabalhadas e testadas isoladamente. Além disso, o uso desta arquitetura é naturalmente adaptada ao objetivo de se disponibilizar as funcionalidades da ferramenta por meio de uma rede Intranet ou da Internet, possibilitando que equipes separadas geograficamente estejam atuando num mesmo projeto.

Internamente ao módulo "*SoCManager Server*", tem-se sua implementação dividida em 3 diferentes partes:

- **Projeto:** Relacionado com a realização dos casos de uso referentes ao Projeto e aos Membros como, por exemplo, gerenciamento do projeto, controle da equipe de desenvolvimento, dos módulos, cadastro de membros e projetos, entre outros;
- **Gerenciamento de Configuração:** Relacionado com a realização dos casos de uso referentes ao GCS como, por exemplo, Controle de *Baselines*, Aquisição de *Baselines*, Aquisição de CIs, Requisições de Mudança, Planejamento da Modificação, entre outros; e
- **Repositório:** Referente a realização dos casos de uso referentes às Bibliotecas de Software como, por exemplo, Controle de Bibliotecas, Definição das Bibliotecas do módulo, entre outros.

Para auxiliar na implementação da arquitetura MVC no módulo, foi utilizado o *framework Struts* (HUSTED, 2002). Este *framework* oferece um conjunto de funcionalidades que auxiliam no desenvolvimento de aplicações, segundo o MVC, referentes ao tratamento das requisições do usuário e geração da interface.

Já para a persistência dos dados, foi utilizado o *framework Hibernate* (HIBERNATE, 2004) que realiza o mapeamento das relações existentes em um banco de dados relacional em objetos da linguagem de programação Java, auxiliando no gerenciamento das informações da ferramenta *SoCManager*. Como banco de dados relacional, foi adotado o *PostgreSQL*, por ser gratuito, trabalhar em diferentes sistemas operacionais e possuir uma grande capacidade de armazenamento de dados.

Por fim, o módulo "*SoCManager Server*" depende dos módulos "*Value Object*" e "*Repository*". Em relação a este último, o "*SoCManager Server*" utiliza a funcionalidade de conexão, permitindo que o Bibliotecário, ao definir um dado repositório e relacioná-lo a um determinado módulo de um projeto de software cadastrado na *SoCManager*, possa verificar se o mesmo está acessível.

3.3.2.4 *XMI Explorer*

Neste módulo tem-se funcionalidades que permitem a instrospecção de artefatos descritos segundo o padrão XMI (*XML Metadata Interchange*). Este módulo está presente apenas no lado usuário.

O XMI foi proposto pelo OMG (OMG, 2002b) visando facilitar o intercâmbio de metadados. Ele consiste no uso da linguagem XML (*eXtensible Markup Language*) (W3C, 2004) para representar metadados que estejam de acordo com o padrão MOF (*Meta Object Facility*) (OMG, 2002a) como, por exemplo, os modelos UML (*Unified Modeling Language*).

A *SoCManager* oferece apoio à aquisição deste artefato pois atualmente há um grande número de ferramentas que utilizam esse padrão como, por exemplo, a ArgoUML (ARGOUML, 2004), o EclipseUML (OMONDO, 2004), o *Together* (BORLAND, 2004) e a MVCASE (LUCRÉDIO, 2005), sendo que há uma tendência de que esse padrão se estabeleça, acompanhando a necessidade de meios mais flexíveis para o armazenamento e intercâmbio de informações.

Para realizar a introspecção sobre o artefato XMI, o módulo "*XMI Explorer*" utiliza o *Metada Repository* (MDR) (NETBEANS, 2004b), que é uma implementação do padrão JMI (*Java Metadata Interface*) (DIRCKZE, 2002), permitindo que se tenha um mapeamento das informações do metadados do XMI para objetos da linguagem de programação Java. Dessa forma, o módulo "*XMI Explorer*" recebe um arquivo XMI e retorna um conjunto de itens internos obtidos deste arquivo. Adicionalmente, o módulo contém um filtro (interface que deve ser estendida) que define quais informações obtidas do XMI serão transformadas em Itens Internos. A figura 26 ilustra este módulo, o JMI, o MDR e o filtro.

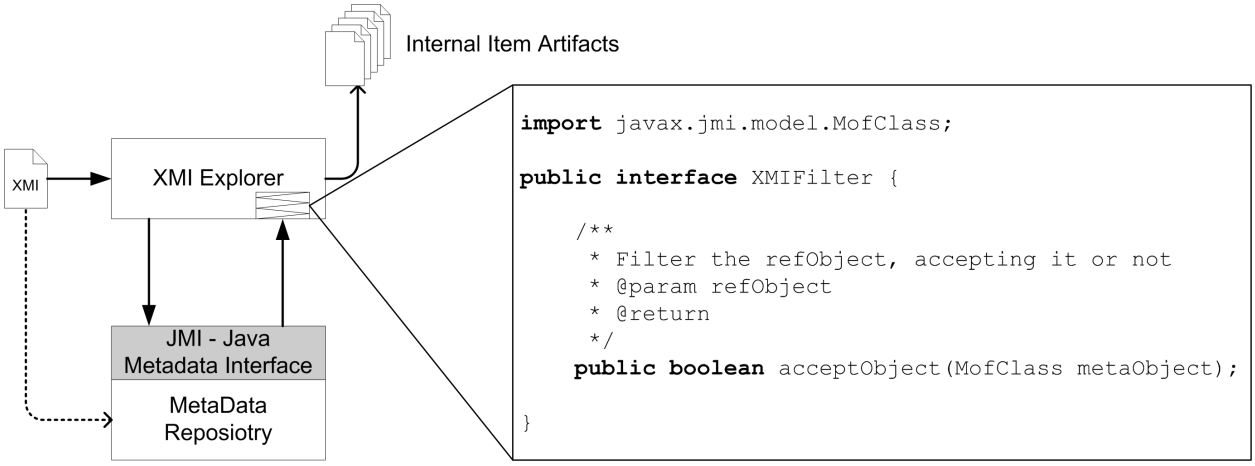

Figura 26: Módulo XMI Explorer e a instrospecção de arquivos XMI

A utilização desse filtro se faz necessária, pois o padrão XMI pode ser utilizado para qualquer tipo de metadados. Sendo assim, uma introspecção pura em um XMI genérico retornaria um número muito grande de itens internos irrelevantes. Portanto, foi utilizado um filtro genérico, que pode ser estendido para qualquer metamodelo.

Um exemplo de extensão do filtro é o definido para metadados da UML, que aceita apenas as informações referentes a pacotes, modelos, casos de uso, atores, componentes, operações, atributos, interfaces e classes (Figura 27), que são os elementos mais relevantes da UML, cujo gerenciamento seria interessante em um processo de GCS.

Pelo fato do módulo "*XMI Explorer*" realizar a instrospecção de arquivos XMI e retornar artefatos do tipos itens internos, há uma depêndencia deste módulo com o "*Value Object*".

```
import javax.jmi.model.MofClass;

public class UMLFilter implements XMIFilter {

    public boolean acceptObject(MofClass metaObject) {
        return
        (metaObject.getName().equals("Package")) ||
        (metaObject.getName().equals("Model")) ||
        (metaObject.getName().equals("UseCase")) ||
        (metaObject.getName().equals("Actor")) ||
        (metaObject.getName().equals("Component")) ||
        (metaObject.getName().equals("Operation")) ||
        (metaObject.getName().equals("Attribute")) ||
        (metaObject.getName().equals("Interface")) ||
        (metaObject.getName().equals("Class"));
    }
}
```

Figura 27: Filtro para Metadados UML

3.3.2.5 *Applet User Interface*

A *SoCManager* foi desenvolvida para a WEB e, segundo esta arquitetura, todo o processamento é realizado no lado do servidor, enquanto que no lado do usuário tem-se apenas as interfaces visuais que permitem o acesso às funcionalidades da ferramenta.

No entanto, alguns dos casos de uso da ferramenta referentes ao GCS necessitam manipular artefatos de software presentes no cliente. Para tornar isso possível, foram utilizadas *Applets* (MICROSYSTEMS, 2004a) que são programas, escritos na linguagem de programação Java, executados no cliente, permitindo inclusive que se tenha acesso ao sistema operacional e aos arquivos do cliente.

Este módulo troca informações com o módulo *"SoCManager Server"* por meio da comunicação WEB e depende dos módulos "*Value Object*", "*Repository*" e "*XMI Explorer*".

Da dependência com o módulo "*Repository*", tem-se o acesso aos repositórios definidos para cada módulo do projeto de software cadastrado na *SoCManager*. Este acesso ocorre em vários momentos do projeto e está relacionado a vários casos de uso definidos na *SoCManager*. A seguir é feita uma descrição de cada um destes acessos realizados pelo módulo "*Applet User Interface*" aos repositórios CVS por meio do módulo "*Repository*":

- **Controlar Artefato do Módulo:** Para este caso de uso, o Engenheiro de Software pode obter um dado artefato do módulo armazenado no repositório, atualizar e remover este artefato. Para tanto, o módulo "*Applet Interface*" utiliza os comandos do CVS para inserir artefatos no repositório (*Import* e *Add*), obter artefatos do repositório (*check-out* e *update*), enviar os artefatos para o repositório (*commit*) e remover os artefatos do repositório (*remove*);

- **Requisitar Aquisição de CIs:** Para este caso de uso, o Engenheiro de Software seleciona os artefatos de software que deseja adquirir dentre aqueles presentes em seu diretório de desenvolvimento. Realizada a aquisição, o módulo "*Applet Interface*" coloca etiquetas (comando *tag* do CVS) nas versões dos artefatos selecionados, indicando que aquelas versões fazem parte de uma acquisição de CI.

- **Auditar Artefatos da Aquisição de CIs:** Para este caso de uso, o Auditor realiza a auditoria sobre os artefatos definidos na aquisição. Para tanto, o módulo "*Applet Interface*" obtém os artefatos do repositório que possuem uma etiqueta referente àquela aquisição que está sendo auditada. Concluída a aquisição, põe-se uma outra etiqueta sobre as versões dos artefatos aprovados, indicando que aquelas versões fazem parte de uma dada *baseline* do projeto (que no caso é a *baseline* associada a Aquisição de CIs);

- **Obter CIs da *Baseline*:** Para este caso de uso, o Engenheiro de Software ou Auditor obtém os CIs de uma dada *baseline*. Para tanto, o módulo "*Applet Interface*" obtém os artefatos do repositório que possuem a etiqueta referente a *baseline* da qual se deseja obter os CIs.

- **Avaliar Planejamento da Modificação:** Para este caso de uso, o Membro do Comitê de Controle de Configuração avalia o planejamento de modificação, aprovando-o ou rejeitando-o. Caso o planejamento seja aprovado, então o módulo "*Applet Interface*" coloca uma etiqueta sobre os CIs que foram selecionados para a modificação (esta seleção é realizada previamente no caso de uso "Planejar Modificação"), sendo que esta etiqueta representa um novo ramo (*branch*) no grafo de versões daquele artefato no repositório.

- **Controlar Artefato em Manutenção:** Para este caso de uso, o Engenheiro de Software que faz parte da equipe de manutenção associada a modificação pode obter e alterar um dado artefato a ser modificado. Para tanto, o módulo "*Applet Interface*" utiliza os comandos do CVS para obter artefatos do repositório (*check-out* e *update*) e enviar os artefatos para o repositório (*commit*); e

- **Auditar Artefatos em Manutenção:** Para este caso de uso, o Auditor realiza a auditoria sobre os artefatos modificados. Para tanto, o módulo "*Applet Interface*" obtém os artefatos do repositório que possuem uma etiqueta referente àquela modificação que está sendo auditada. Aprovada a modificação, efetua-se a atualização dos artefatos no repositório por meio da união das versões dos artefatos referentes à modificação e aos CIs. Esta atualização cria uma nova versão dos artefatos no repositório onde são postas etiquetas referentes à *baseline* do CI previamente modificado (a etiqueta da *baseline* é primeiramente removida da versão anterior do artefato).

É mostrado na Figura 28 (obtida por meio da ferramenta WinCVS[2]) o grafo de versões de um dado artefato manipulado pelo módulo "*Applet Interface*" em resposta às requisições do cliente e por meio do uso do módulo "*Repository*".

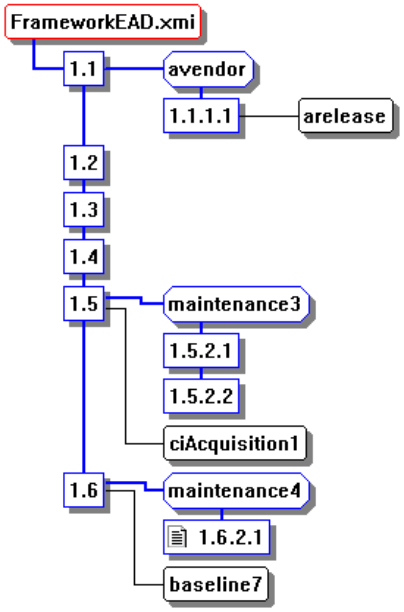

Figura 28: Grafo de Versões do Artefato FrameworkEAD.xmi

Na figura pode-se notar que a versão 1.5 sofreu o processo de aquisição ("ciAcquisition1") e foi aprovado. Deste ponto em diante duas modificações foram realizadas ("maintenance3" e "maintenance4"), sendo que apenas a modificação "maintenance3" foi concluída e aprovada (vê-se por meio da etiqueta "baseline7" associada à versão 1.6 que indica que aquele artefato compõe uma *baseline*). No momento em que a modificação "maintenance4" for concluída, será gerada uma nova versão 1.7, a partir da união da versão 1.6 com a última versão da ramificação representada pela etiqueta "maintenance4" (no caso da figura, a versão 1.6.2.1). Além disso a etiqueta "baseline7" será movida para a versão 1.7.

Por fim, da dependência com o módulo "*XMI Explorer*", tem-se a instrospeção dos artefatos descritos segundo o padrão XMI, auxiliando na identificação dos itens internos deste artefato durante a execução do caso de uso "Requisitar Aquisição de CIs".

3.3.3 Integração da *SoCManager* com o ambiente *Orion*

A *SoCManager* é uma ferramenta de apoio ao processo de GCS, dando suporte à sua adoção em projetos de software. O fato desta ferramenta permitir a introspecção de artefatos descritos

[2]WinCVS (`http:\\www.wincvs.org`, Consultado em Janeiro/2005)- Ferramenta gratuíta que oferece uma interface cliente para acesso às funcionalidades do CVS.

segundo o padrão XMI proporciona a sua integração com outras ferramentas que utilizam o XMI para persistir as informações por elas manipuladas.

No GOES - Grupo de Engenharia de Software do Departamento de Computação da UFSCar - Universidade Federal de São Carlos, onde foi desenvolvida esta pesquisa, existem outras duas ferramentas de auxílio ao desenvolvimento de software.

A ferramenta MVCASE é uma ferramenta de modelagem baseada na UML, que vem sendo desenvolvida desde 1997 (BARRÉRE, 1999) por alunos de mestrado e iniciação científica do Departamento de Computação da Universidade Federal de São Carlos e que atualmente está sendo estendida para persistir suas informações em arquivos descritos em XMI (LUCRÉDIO, 2005). A MVCASE auxilia o Engenheiro de Software nas atividades relativas à análise, projeto, implementação e implantação de sistemas de software orientados a objetos.

A ferramenta C-CORE (NETO et al., 2004) auxilia nas atividades de codificação, depuração e projeto de interface gráfica com o usuário e, assim como a MVCASE, persiste as suas informações em arquivos XMI. Juntas, essas duas ferramentas e a *SoCManager* formam o ambiente *Orion*, ilustrado na figura 29.

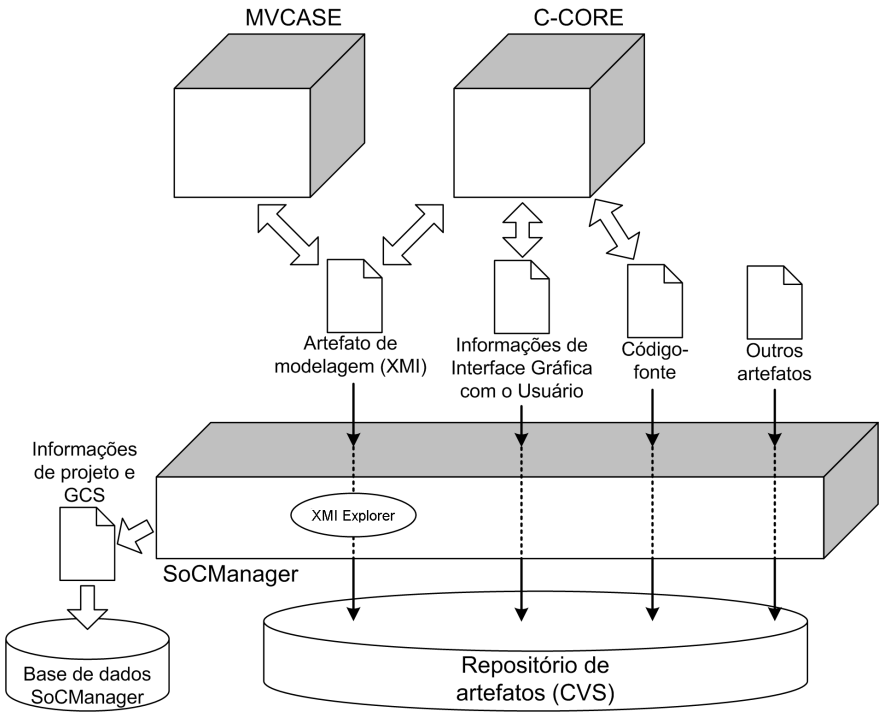

Figura 29: Ambiente *Orion*

A ferramenta MVCASE produz artefatos de modelagem, persistidos em XMI. A ferramenta C-CORE trabalha principalmente com código-fonte e informações relativas à interface gráfica com o usuário, mas também é capaz de persistir as informações contidas no código-fonte em XMI, como estruturas de classes, hierarquias, atributos, métodos e até mesmo o corpo dos métodos. Dessa

forma, artefatos do tipo XMI produzidos pela MVCASE podem ser diretamente manipulados pela C-CORE e vice-versa.

A ferramenta *SoCManager* gerencia projetos e auxilia em algumas tarefas do GCS. Para isso, ela é responsável pelo armazenamento dos artefatos em um repositório, no caso o CVS. A *SoCManager* utiliza uma base de dados própria para armazenar informações de controle de artefatos de qualquer tipo, incluindo XMI, código-fonte ou informações relativas à interface gráfica com o usuário, produzidos pelas ferramentas do *Orion*, ou outros artefatos quaisquer.

Porém, quando o artefato é do tipo XMI, a *SoCManager* pode realizar a introspecção sobre o mesmo por meio do módulo "*XMI Explorer*". Sendo assim, é possível realizar o controle sobre artefatos tanto no nível de arquivo como no nível de sua estrutura interna. A *SoCManager* utiliza para isso o conceito de Item Interno, que é baseado no conceito de elemento controlado (ITAMI, 1997), permitindo um controle mais fino sobre as alterações e os relacionamentos entre os artefatos. Por exemplo, é possível realizar o controle sobre elementos da UML representados dentro de um documento XMI, como atores, casos de uso, classes, entre outros.

A idéia por trás desta arquitetura é tornar possível que:

- Se tenha o uso independente de cada uma das ferramentas relacionadas ao repositório;
- As ferramentas compartilhem as informações dos artefatos armazenados no repositório, garantindo uma maior consistência e evitando a duplicação dessas informações; e
- Novas ferramentas possam ser integradas ao ambiente. O funcionamento e a utilização conjunta das ferramentas do ambiente *Orion* pode ser melhor visualizado no capítulo 5, onde é apresentado um estudo de caso envolvendo o ambiente.

3.4 Resumo da Pesquisa

O principal e mais visível produto desta pesquisa foi a ferramenta *SoCManager*. No entanto, para isto foi necessário criar todo um arcabouço, que envolveu a elaboração de duas abordagens, de tal forma que se pudesse atender ao objetivo da pesquisa.

A abordagem para a implantação de um processo de GCS descreve as atividades que devem ser realizadas para efetuar a implantação deste processo em empresas de software. Esta abordagem é uma instância simplificada do modelo IDEAL e está voltada para a implantação de processos de GCS. Ela define 12 atividades a serem realizadas por um agente, denominado Agente de Mudança, na empresa onde se deseja realizar a implantação do processo.

Já abordagem para o processo de GCS define os conceitos deste processo, os agentes e as atividades. Esta abordagem estabelece os aspectos de GCS com base em uma série de abordagens existentes e define as atividades para Identificação, Controle, Administração de Estado e Auditoria da Configuração. Esta abordagem foi desenvolvida com base em outras abordagens citadas no capítulo 2, tendo em vista as metas definidas pela área de processo de Gerenciamento de Configuração do modelo de capacitação CMMi.

Por fim, a *SoCManager*, ferramenta de apoio ao GCS, automatiza parte dos aspectos definidos na abordagem para este processo, auxiliando a sua adoção em empresas de *software*. Esta ferramenta automatiza as atividades referentes à Identificação e Controle de Configuração e possui recursos que permitem o controle de versões (por meio do módulo "*Repository*") e o gerenciamento de mudanças. Assim sendo, esta ferramenta apóia tanto o Gerenciamento de Configuração como o Gerenciamento de Mudanças (categorias de ferramentas de GCS definidas por Sommerville (SOMMERVILLE, 2003) e citadas na seção 2.5).

Adicionalmente, a *SoCManager* permite o cadastro de informações sobre a configuração dos artefatos de software em três diferentes categorias: Produto (arquivo relacionado com o artefato); Item Interno (elemento interno ao artefato); e Relacionamento (relacionamento entre Produtos, Itens Internos ou Produtos e Itens Internos). Além disso, o suporte ao padrão XMI oferece uma característica inovadora, permitindo a introspecção sobre este tipo de artefato, cada vez mais utilizado em projetos de desenvolvimento.

4 Avaliação dos Resultados

Apresenta-se neste capítulo um estudo de caso, desenvolvido nos laboratórios do Departamento de Computação da Universidade Federal de São Carlos (UFSCar), para ilustrar a utilização dos recursos oferecidos pela ferramenta *SoCManager* no auxílio à execução do processo de GCS em projetos de software. Adicionalmente, para as atividades relacionadas à modelagem e à implementação, foram utilizados a MVCASE e C-CORE.

Os alunos, que participaram do estudo de caso, utilizaram as ferramentas do ambiente *Orion*, sendo que a *SoCManager* foi utilizada nas tarefas de GCS, a MVCASE nas tarefas de modelagem e construção dos componentes, e a C-CORE nas tarefas de codificação e projeto de interface com o usuário.

Buscou-se avaliar, por meio deste estudo de caso, a utilização da *SoCManager* no desenvolvimento de projetos de software, assim como os aspectos da abordagem do processo por ela implementados e a abordagem da implantação. Esta última foi realizada em primeiro lugar, como preparação do estudo de caso.

O estudo de caso foi desenvolvido nos laboratórios de pesquisa do Departamento de Computação da UFSCar, com a equipe distribuída em duas salas distintas. A comunicação entre os membros da equipe foi realizada por meio de correio eletrônico, e aplicativos de *Instant Messaging* e teleconferência, quando a situação exigia iterações mais imediatas.

4.1 Metodologia para a Realização do Estudo de Caso

Tendo sido realizado em laboratório, o estudo de caso não permitiu avaliar a abordagem da implantação em sua total extensão, como o seria em um ambiente industrial.

Porém, a abordagem da implantação foi adotada como metodologia para a realização do estudo de caso. Para tanto, o papel do Agente de Mudança, previsto na abordagem da implantação, foi desempenhado pelo aluno que desenvolveu esta pesquisa.

Inicialmente, o Agente de Mudança entrou em contato com os pesquisadores do Departamento de Computação, visando ter o seu apoio na execução do estudo de caso (atividade "Obter Patrocínio"). Em seguida, ele fez um breve levantamento de informações sobre o processo de GCS, considerando as informações que já tinham sido obtidas e que foram mostradas na seção 2.1.1.

A próxima atividade executada pelo Agente de Mudança foi a "Estabelecer Objetivos e Metas" (a atividade "Levantar Informações sobre a Empresa" não foi executada considerando o fato de a abordagem estar sendo realizada em laboratório), onde definiu-se, como objetivo, a implantação da abordagem do processo elaborada nesta pesquisa e, como meta, a implantação das atividades de GCS referentes à Identificação e Controle de Configuração (limitou-se apenas a estas duas atividades devido ao fato da *SoCManager* implementar apenas estes aspectos definidos na abordagem do processo). Dessa forma, na atividade "Definir uma Abordagem para o Processo de GCS", teve-se a adoção da abordagem do processo elaborada nesta pesquisa.

Partiu-se então para a atividade "Selecionar Ferramentas de GCS", onde foi selecionada a *SoCManager*, por apoiar a execução da abordagem adotada na implantação.

Em seguida, teve-se a escolha dos projetos nos quais foram aplicados tanto a abordagem do processo como a ferramenta *SoCManager* (atividade "Escolher o Projeto Piloto"). O Agente de Mudança, com o apoio dos patrocinadores (obtidos na atividade "Obter Patrocínio"), selecionou quatro projetos: 1) Construção de um *framework* de componentes para educação à distância; 2) Construção de uma aplicação que reutilizou o *framework* para educação a distância; 3) Construção de um *framework* para aplicações multimídia; e 4) Construção de uma aplicação que reutilizou os componentes do *framework* multimídia. Estes projetos fazem parte de outra pesquisa desenvolvida no Departamento de Computação da UFSCar, denominado DBCM - Desenvolvimento Baseado em Componentes Multimídia (UFSCAR, 2004).

Para o desenvolvimento desses projetos, foi utilizado neste estudo o método *Catalysis* (D'SOUZA; WILLS, 1999). O Catalysis compreende três níveis: Domínio do Problema, Especificação dos Componentes e Projeto Interno dos Componentes. Esses níveis são realizados conforme o modelo espiral de ciclo de vida de software (PRESSMAN, 2001). Sempre que necessário retorna-se aos passos anteriores para refinar ou remover inconsistências dos artefatos produzidos em cada passo.

Seguindo as atividades da abordagem da implantação, teve-se a definição e implantação do plano de GCS (atividades "Definir o Plano de GCS" e "Implementar o Plano de GCS"), por meio da *SoCManager*, onde foi feita uma apresentação da abordagem do processo e da ferramenta, designando os papéis a serem desempenhados pelos alunos que atuaram nos projetos.

O desenvolvimento desses projetos envolveu uma equipe de quatro alunos (A,B,C e D) de

pós-graduação, com experiência na linguagem de programação Java (Linguagem utilizada pae nos domínios EAD e multimídia. A equipe foi dividida da seguinte forma:

- O aluno A foi designado como gerente de projeto, responsável por controlar as informações do projeto, definir a equipe, controlar dos módulos do projeto, controlar as *baselines* que serão adquiridas no decorrer do desenvolvimento, designar tarefas, e definir quais projetos serão reutilizados. Além de gerente de projeto, o aluno A também foi designado como bibliotecário, responsável pela definição dos repositórios do projeto e associação de módulos do projeto aos repositórios.

- O aluno B foi designado como único membro do comitê de controle de configuração. Este comitê é responsável pela definição dos tipos de artefatos a serem controlados, avaliação de requisições de mudança, solicitação de modificações, avaliação de planejamentos de modificação. O aluno B também foi designado como auditor, responsável por avaliar os artefatos entregues para controle, e os artefatos que foram modificados.

- Todos os alunos A,B,C e D também foram designados Engenheiros de Software, responsáveis pelas tarefas de desenvolvimento, envolvendo a modelagem, codificação, testes e empacotamento do software.

A partir deste momento, o Agente de Mudança acompanhou utilização da ferramenta *SoCManager* (atividade “Acompanhamento da Execução do Plano de GCS”) visando garantir a correta execução do processo de GCS, segundo definido na abordagem do processo.

Concluído o desenvolvimento dos projetos, o Agente de Mudança partiu para a atividade “Avaliar a Implementação do Plano de GCS”, onde foram obtidas impressões das pessoas que atuaram nos projetos acerca da utilização tanto da ferramenta *SoCManager* quanto dos aspectos da abordagem do processo por ela implementados. Tanto estas impressões como a avaliação do uso da abordagem da implantação são apresentadas na seção 4.5.

Por fim, a atividade “Estender a Implementação para Outros Projetos de Software” não foi realizada pelo fato deste estudo de caso ter sido feito em laboratório.

4.2 Descrição dos Domínios

Descreve-se brevemente nesta seção os domínios envolvidos com o estudo de caso, que são o domínio de Aplicações Multimídia e o de Educação à Distância. As descrições a seguir foram extraídas de (SILVA et al., 2004).

4.2.1 Domínio de Aplicações Multimídia

Aplicações com recursos multimídia vêm se tornando mais comuns em ambientes computacionais, pois oferecem interfaces áudio-visuais que facilitam a compreensão e permitem uma grande interatividade com o usuário. A utilização de recursos multimídia nas aplicações de diferentes domínios tem exigido novas técnicas, métodos e ferramentas para apoiar seus desenvolvimentos e um maior conhecimento de programação para realizar suas implementações.

Informações multimídia podem ser classificadas com base nos tipos de mídias envolvidas. Mídia se refere ao tipo de informação, como dados alfanuméricos, imagens, áudio e vídeo. Existem muitas formas de classificar mídias. Uma forma proposta por Guojun (GUOJUN, 1996) classifica as mídias conforme a existência ou não da dimensão de tempo para a mídia. Nessa classificação, existem duas classes de mídias: estáticas e dinâmicas.

Mídias estáticas não têm a dimensão de tempo e, portanto, seus conteúdos e significados não são dependentes do tempo de apresentação. Mídias estáticas incluem dados alfanuméricos, gráficos, e imagens. Mídias dinâmicas têm a dimensão de tempo, e seus significados e exatidão dependem da taxa com que são apresentadas. Mídias dinâmicas incluem animação, áudio e vídeo.

Do ponto de vista lingüístico, qualquer sistema capaz de manipular mais de um tipo de mídia pode ser chamado de “sistema multimídia”. Guojun (GUOJUN, 1996) define sistema multimídia como aquele capaz de manipular ao menos um tipo de mídia dinâmica no formato digital, assim como mídia estática. Nessa definição, a combinação sincronizada de múltiplos tipos de mídias com ao menos uma mídia dinâmica denomina-se “objeto multimídia”.

A criação de objetos multimídia envolve vários aspectos relacionados à sincronização entre as mídias. A Figura 30 mostra formas de sincronizações para apresentação das mídias. Em (1) tem-se a representação de uma mídia e em (2) a representação de um bloco que pode conter uma sincronização paralela (4) ou seqüencial (5). Em (3) tem-se a equivalência entre largura e tempo da apresentação e em (6) tem-se a possibilidade de atribuir intervalos entre a apresentação das mídias. É possível também atribuir atraso no início de uma mídia em relação à outra (7) e em (8) é apresentada como é a apresentação de uma sincronização seqüencial em um bloco paralelo. Em (9) e (10) são apresentadas possíveis composições de blocos.

O uso de objetos multimídia em sistemas computacionais tem motivado o aprimoramento da transferência de informação com o envolvimento de dois ou mais sentidos dos seus usuários, se tornando cada vez mais comuns e presentes em vários domínios, como medicina, entretenimento e educação. Para atender estes e outros domínios, as funcionalidades do domínio Multimídia devem ser direcionadas para criar aplicações e encapsular a complexidade de manipulação de objetos

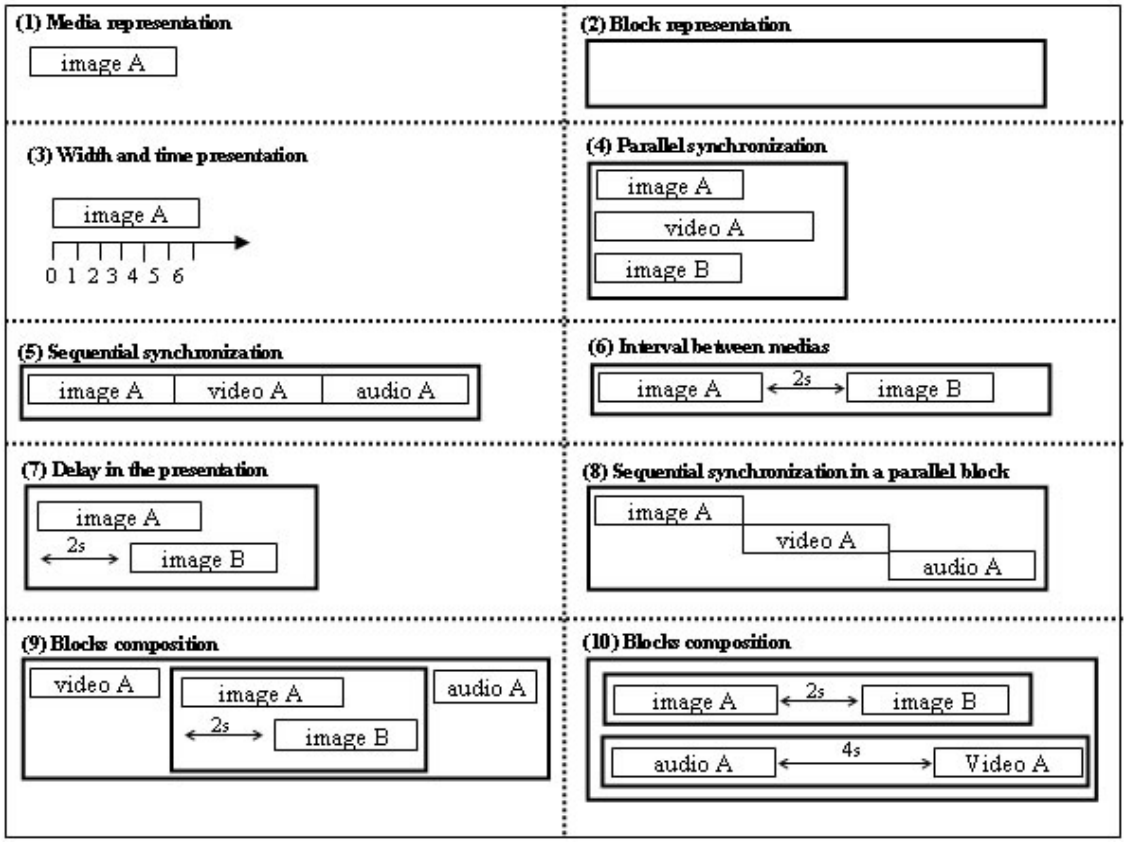

Figura 30: Formas de sincronização.

multimídia, considerando os seguintes pontos:

- Visualização de mídias;
- Armazenamento e recuperação de mídias;
- Criação e edição de objetos multimídia; e
- Interação com os objetos multimídia durante a sua execução.

Além das funcionalidades, é preciso identificar as informações que farão parte de um objeto multimídia. A Figura 31 mostra um *Snapshot* (D'SOUZA; WILLS, 1999) representando um objeto multimídia por meio de uma composição de cenas (*ScenesComposition 01*) onde, em determinado instante, cada cena (*Scene 01*) é composta por mídias (*Media*). As mídias podem ser imagens (*Image*), textos (*Text*), áudios (*Audio*) ou vídeos (*Video*) e são armazenadas em seqüências de números binários (0 ou 1) ou em seqüências de caracteres.

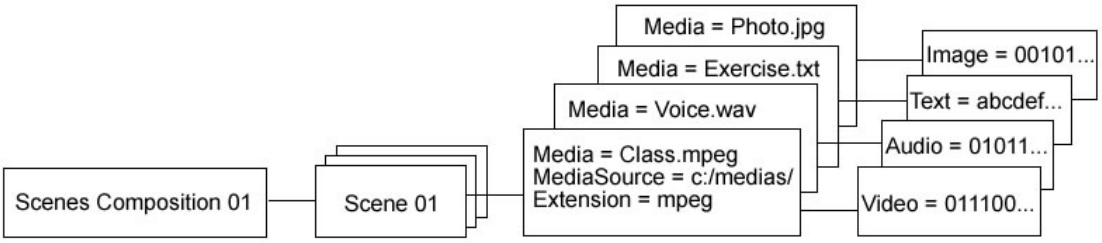

Figura 31: *Snapshot* de objetos multimídia.

4.2.2 Domínio de Educação à Distância

Com o avanço tecnológico, a Educação a Distância é vista hoje não só como um sistema especial, mas como uma parte integrante do aprendizado que prepara profissionais nas diversas áreas de ensino e pesquisa. Muitas Instituições de Ensino estão aperfeiçoando e desenvolvendo seus programas de EAD. Recursos como redes de trabalho, videoconferência, computação colaborativa, Internet, objetos multimídia e outros são utilizados para maior interação entre estudantes e professores, melhoria da qualidade do ensino e difusão da educação em diferentes locais e regiões do mundo (SILVA et al., 2004).

A EAD envolve três atores principais: Administrador, Professor e Estudante. Na tabela 16 são apresentadas as principais funcionalidades que estudantes, professores e administradores dos cursos podem realizar.

Tabela 16: Funcionalidades do domínio EAD.

Ator	**Ação (funcionalidade)**
Administrador	Cria cursos
Administrador	Gerencia cursos
Professor	Cria disciplinas
Professor	Cria aulas (para determinada disciplina)
Professor	Cria exercícios e provas (para determinada disciplina)
Professor	Armazena mídias (como material de aula)
Professor	Cria objetos multimídia (como material de aula)
Professor	Gerencia disciplinas
Estudante	Matricula-se no curso
Estudante	Realiza aulas
Estudante	Realiza exercícios
Estudante	Realiza provas
Estudante	Acessa materiais (de determinada aula)

Como discutido anteriormente, o estudo de caso envolveu quatro projetos destes domínios de Multimídia e EAD. Com o propósito de ilustrar a execução deste estudo, apresenta-se a seguir dois desses projetos: construção do *framework* para educação à distância e de uma aplicação que

reutiliza este *framework*. Os outros dois projetos foram desenvolvidos de forma similar.

4.3 Construção do *framework* para Educação à Distância

Inicialmente, todos os alunos se cadastraram na *SoCManager* para que estes pudessem estar atuando na execução do processo de GCS apoiado por esta ferramenta.

Em seguida, o aluno A, que foi designado como gerente do projeto no estudo de caso, criou um novo projeto na *SoCManager* com a sigla (*alias*) "*FrameworkDE*". Em decorrência disso, o membro correspondente ao aluno A passou a ter acesso, para aquele projeto, à perspectiva de Gerente do Projeto (Figura 32).

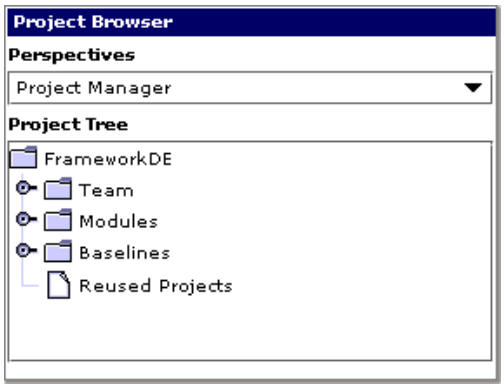

Figura 32: Perspectiva do Gerente do Projeto na SoCManager

Dentre as atividades desempenhadas pelo membro nesta perspectiva, cita-se a de definição da equipe de desenvolvimento do projeto de software e controle dos módulos e *baselines* do projeto.

Para a atividade de definição da equipe de desenvolvimento, o Gerente do Projeto selecionou os 4 membros correspondentes aos alunos A, B, C e D e, para cada membro, foram atribuídos os papéis da *SoCManager* (*Librarian*, *Software Engineer*, *Auditor* e *Configuration Control Board Member*) (Figura 33) conforme descrito na seção 4.1.

Para a atividade de controle dos módulos do projeto, o Gerente do Projeto definiu apenas um módulo ao qual foram designados todos os membros do projeto.

Já para a atividade de controle das *baselines*, o Gerente do Projeto definiu duas *baselines*: uma *baseline* de requisitos, composta pelos documentos de requisitos e de análise do domínio de Educação a Distância (engloba artefatos tanto das atividades de Domínio do Problema como de Especificação dos Componentes); e uma *baseline* de implementação, que irá ser composta pelos componentes de software (engloba tanto artefatos das atividades de Especificação dos Componentes como de Projeto Interno dos Componentes).

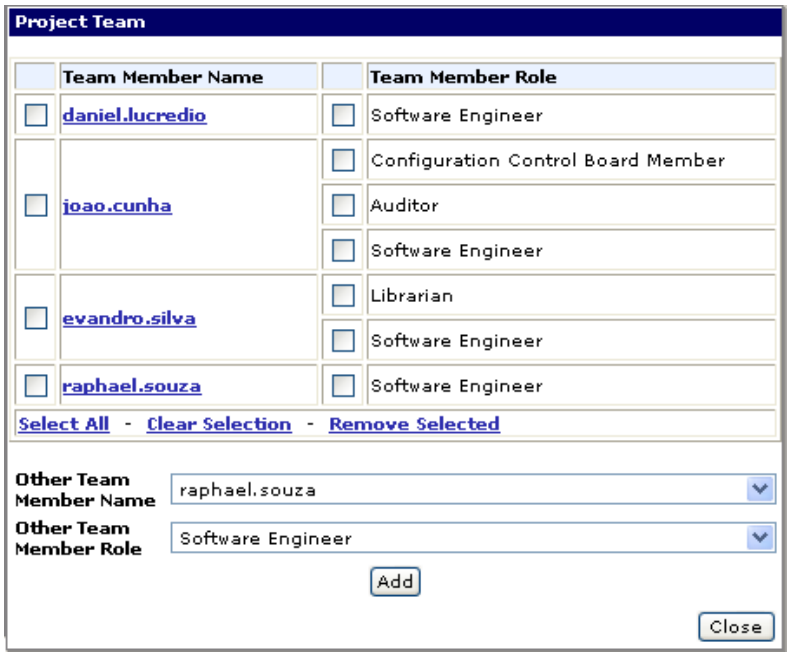

Figura 33: Tela da ferramenta *SoCManager* para cadastro de equipe.

Após a execução destas atividades, o aluno A, por meio da perspectiva de Bibliotecário, criou um repositório CVS para armazenar os artefatos do projeto, associando-o ao módulo do projeto.

A partir deste momento e seguindo o *Catalysis*, os Engenheiros de Software realizaram o levantamento de requisitos, com base em experiências anteriores (SILVA, 2002a; SILVA; VIEIRA, 2001; SILVA, 2002b; CATARINO, 2002) envolvendo o próprio grupo de pesquisa onde foi desenvolvido o presente trabalho. Foram então definidos documentos de requisitos, que foram armazenados no repositório correspondente ao módulo do projeto "*FrameworkDE*" por meio do apoio da *SoCManager*. Terminado este levantamento, todos os artefatos do módulo gerados nesta fase foram adquiridos e passaram a compor a *baseline* de requisitos do projeto (Figura 34).

Após a aquisição, estes artefatos passam a ser controlados, e só podem ser alterados após a execução das atividades de Controle de Configuração, definidas na abordagem do processo e automatizadas pela *SoCManager*.

Em seguida, estes requisitos foram refinados em modelos de casos de uso e modelos de tipos. A Figura 35 mostra o modelo de tipos construído na ferramenta MVCASE. Nesta figura não são mostrados os atributos de cada tipo, para enfatizar os relacionamentos entre eles.

Um professor (*Teacher*) pode criar várias disciplinas (*Discipline*) para determinado curso (*Course*) e definir várias aulas (*Lesson*) e provas (*Test*) para cada disciplina. Se o professor desejar, pode também definir exercícios (*Exercise*) de múltipla escolha (*Option*) para cada aula, com até cinco alternativas. As provas da disciplina agregam várias questões (*Question*) e cada questão

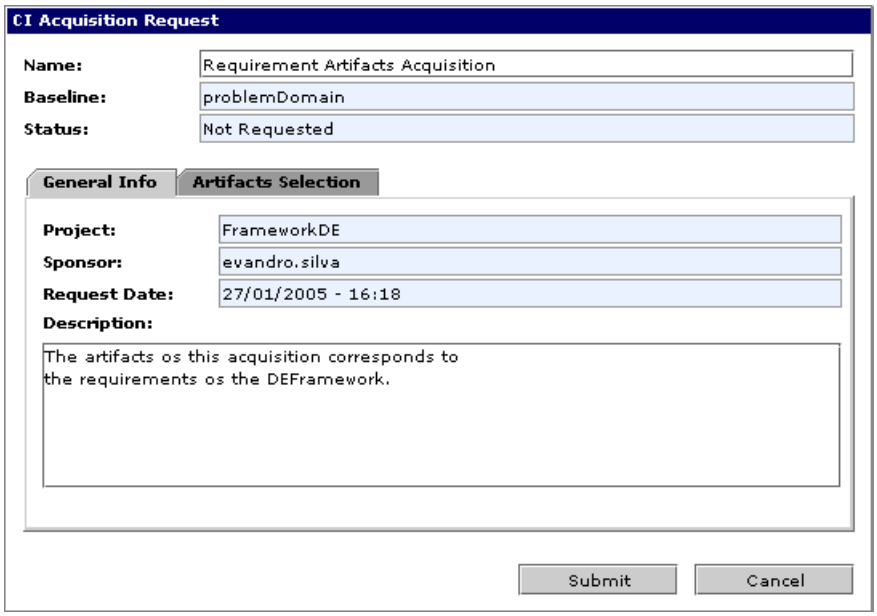

Figura 34: Acquisição de CIs na SoCManager

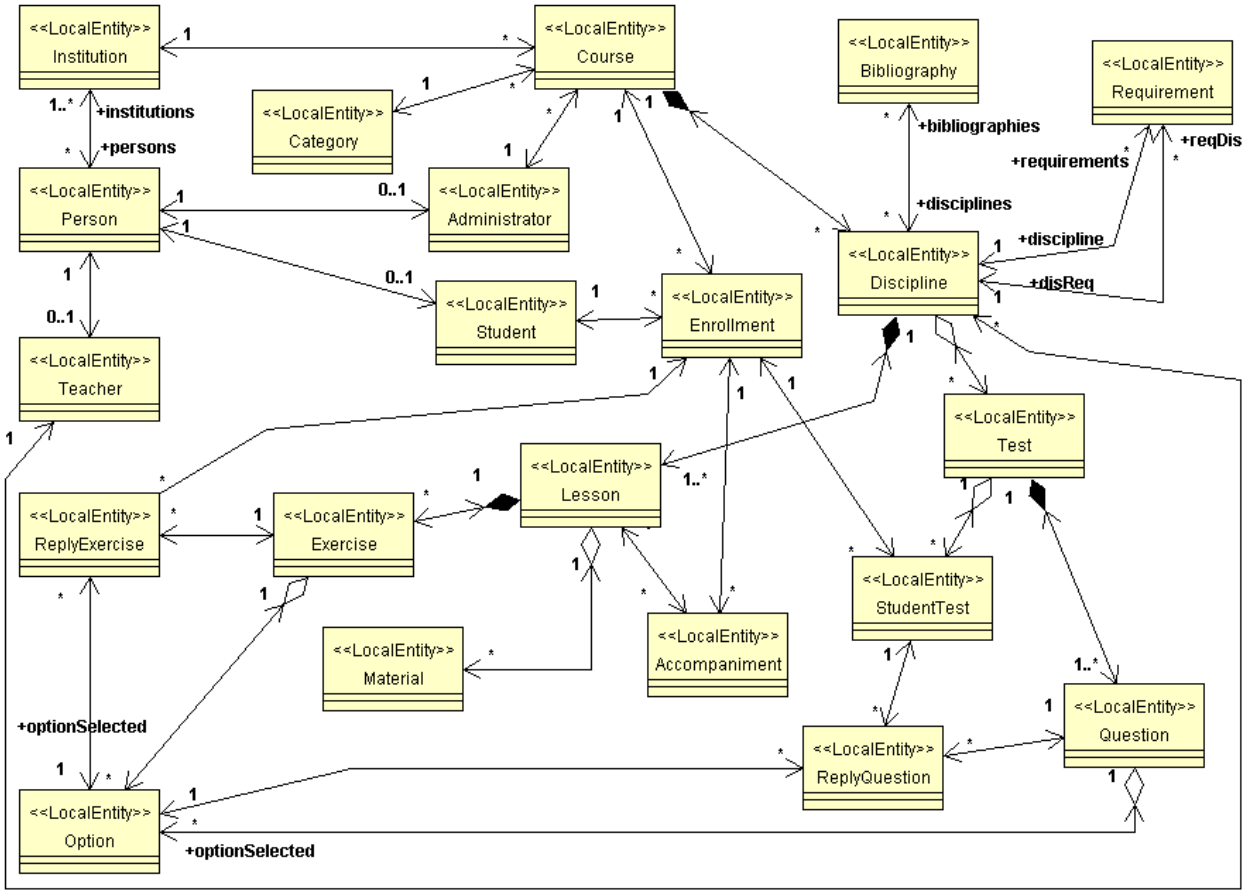

Figura 35: Tela da ferramenta MVCASE com o modelo de tipos (Notação UML (OMG, 2003))

pode agregar até cinco alternativas (*Option*). As respostas dos exercícios (*ReplyExercise*) e das questões das provas (*Question*) são armazenadas para que o professor possa definir uma média final da disciplina para o estudante. Em relação à prova, o professor pode definir várias questões

e atribuir um número de questões para que cada estudante responda. Por exemplo, um professor define trinta questões para uma prova de determinada disciplina e define que os estudantes precisam responder apenas dez questões. Quando o estudante acessar a prova são escolhidas, aleatoriamente, dez questões entre as trinta definidas pelo professor. A prova gerada para o estudante é armazenada em uma nova classe.

Ao definir uma disciplina, podem ser indicadas várias bibliografias (*Bibliography*). Cada disciplina também pode estar associada a requisitos (*Requirements*), que são outras disciplinas obrigatórias para realização da disciplina. Quando um estudante matriculado acessa e realiza os exercícios de uma aula de determinada disciplina, é gerado um acompanhamento (*Accompaniment*) do estudante para a aula. Uma aula pode agregar vários materiais (*Material*). O material pode ser uma mídia ou uma composição de cenas (objeto multimídia). Uma composição de cenas pode agregar uma ou mais cenas, e cada cena agrega uma ou mais mídias, que podem ser do tipo imagem, texto, áudio ou vídeo.

Após a criação do modelo de tipos, a modelagem do *framework* foi adquirida pelos Engenheiros de Software por meio da *SoCManager*. Para esta aquisição, os Engenheiros de Software contaram com o apoio da introspecção de arquivos XMI, já que a MVCASE persistiu a modelagem do framework neste formato. Isso se deve ao fato da MVCASE realizar a persistência dos modelos UML em artefatos XMI. Dessa forma, os Engenheiros de Software puderam especificar mais precisamente os itens internos ao artefato que deveriam também estar sendo controlados. A Figura 36 mostra uma tela da ferramenta *SoCManager*, com um artefato do tipo XMI sendo visualizado. É possível portanto controlar os elementos da UML, como casos de uso, classes, atributos e métodos, entre outros.

Na Figura 36, tem-se no lado esquerdo um *browser* com as informações internas do artefato. No lado direito tem-se a interface para cadastramento de relacionamentos entre itens de configuração. No caso da Figura 36, o componente UML **StudentBean** está sendo relacionado às classes Java **Student.java**, **StudentBean.java** e **StudentHome.java**. Dessa forma, é possível realizar uma análise de impacto mais precisa, auxiliando no planejamento e execução das mudanças, além de se ter um controle mais fino sobre os artefatos.

Concluída esta aquisição, o Gerente do Projeto adquiriu a *baseline* de requisitos, não permitindo assim que novos artefatos sejam adicionadas a ela.

Em seguida, a *baseline* de implementação foi aberta, indicando o inicio da fase de implementação dos componentes referente principalmente a atividade de Projeto Interno dos Componentes, do método *Catalysis*.

Nesta fase, foram gerados na MVCASE os componentes persistentes e transientes, segundo

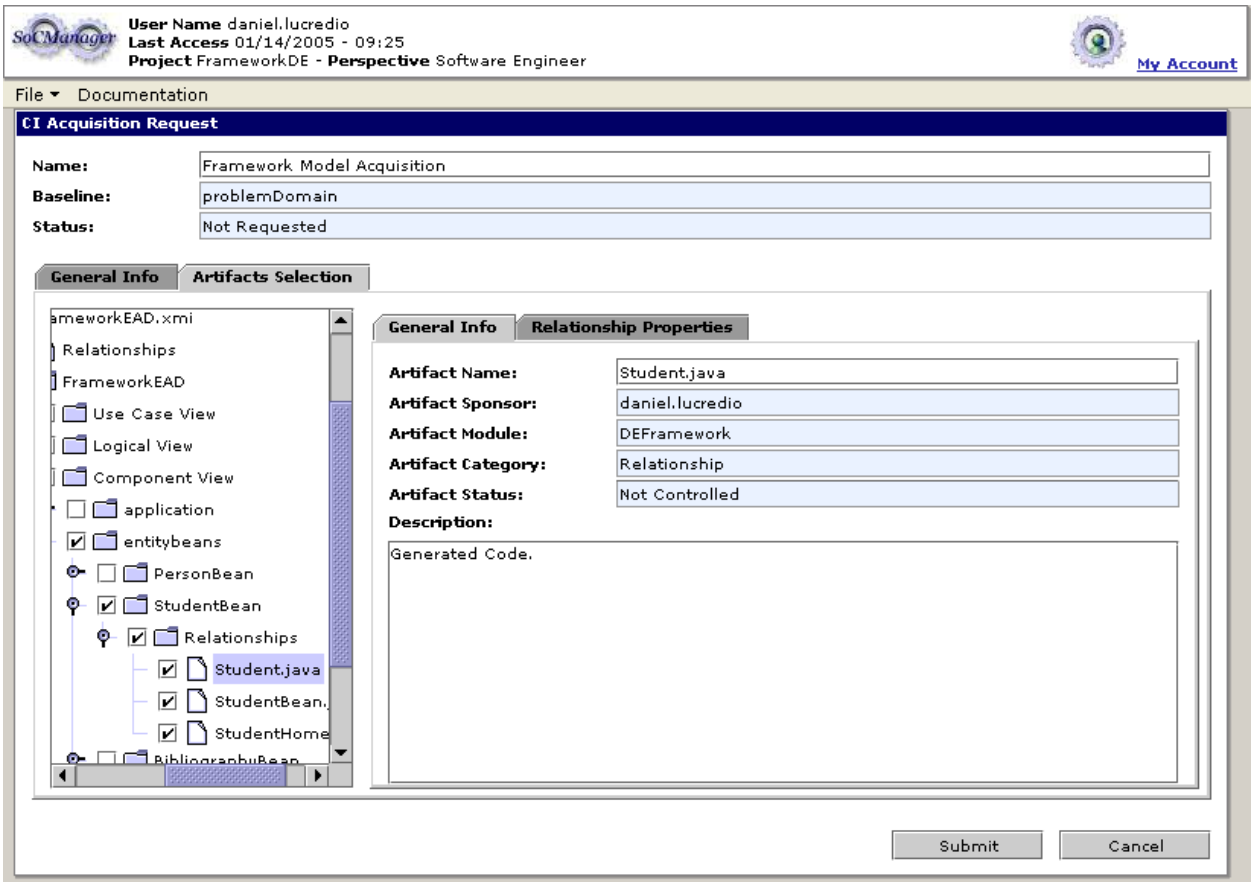

Figura 36: Tela da ferramenta *SocManager* onde um artefato XMI é visualizado internamente.

a tecnologia EJB (*Enterprise Java Beans*) (DEMICHIEL, 2002). O código gerado foi compilado e executado. Foi utilizado o servidor de componentes JBoss (JBOSS, 2004) para execução. Testes foram realizados, utilizando a *C-CORE* para auxiliar nas tarefas de depuração. Correções e alterações foram realizadas diretamente na MVCASE, gerando-se o código e testando-o novamente até que não fossem detectados mais erros.

Finalizada a implementação, os novos artefatos produzidos foram adquiridos pela *SoCManager* e passaram a compor a *baseline* de implementação, que foi adquirida pelo Gerente do Projeto.

Uma vez finalizado, esse *framework* pode ser então reutilizado na construção de aplicações do domínio de educação à distância. Para tanto, a *SoCManager* permite que um projeto de software possa reutilizar outros projetos cadastrados. Este reuso, no entanto, limita-se ao acesso às *baselines* adquiridas do projeto reutilizado, e à obtenção de seus CIs.

Na próxima seção é apresentada a segunda parte deste estudo de caso, onde uma aplicação que reutiliza os componentes deste *framework* foi desenvolvida.

4.4 Construção de uma aplicação para Educação à Distância

Construiu-se uma aplicação onde administradores podem criar cursos e professores podem criar disciplinas para os cursos. Ao criar as disciplinas, os professores devem também criar aulas e definir materiais de apoio às aulas. Os materiais definidos podem ser objetos multimídia que o professor também deverá criar. Os estudantes acessam as disciplinas e visualizam os materiais disponíveis.

A aplicação foi desenvolvida reutilizando os componentes EJB do projeto "*FrameworkDE*", e as tecnologias JSP (*Java Server Pages*) para construção de páginas *Web*.

Para tanto, o projeto de desenvolvimento da aplicação foi cadastrado no *SoCManager*. Além disso, foi definido que este projeto reutiliza outro projeto cadastrado na ferramenta, o "*FrameworkDE*", descrito anteriormente (Figura 37).

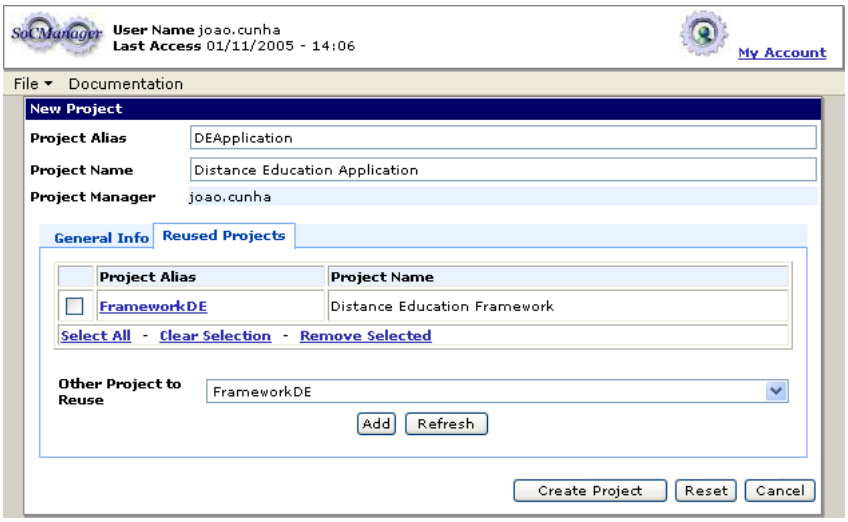

Figura 37: Cadastro do Projeto para Desenvolvimento da Aplicação EAD na *SoCManager*

Após a criação deste projeto, foram desempenhadas as atividades de definição da equipe, controle de módulos e *baselines* pelo Gerente do Projeto, e o controle de bibliotecas do projeto pelo Bibliotecário, de forma similar ao que foi descrito na seção 4.3.

Além disso, os Engenheiros de Software, definidos na equipe de desenvolvimento do projeto, puderam recuperar os componentes do domínio EAD, que no caso compõem a *baseline* de implementação do projeto "*FrameworkDE*". Uma vez recuperados esses componentes, foram construídas páginas JSP que implementam as funcionalidades requeridas para a aplicação. A Figura 38 ilustra uma tela da aplicação implementada, para cadastro de aulas. Neste exemplo, está sendo cadastrada uma aula introdutória ao assunto de Desenvolvimento Baseado em Componentes.

Durante a construção da aplicação, foram observadas algumas situações onde surgiu a neces-

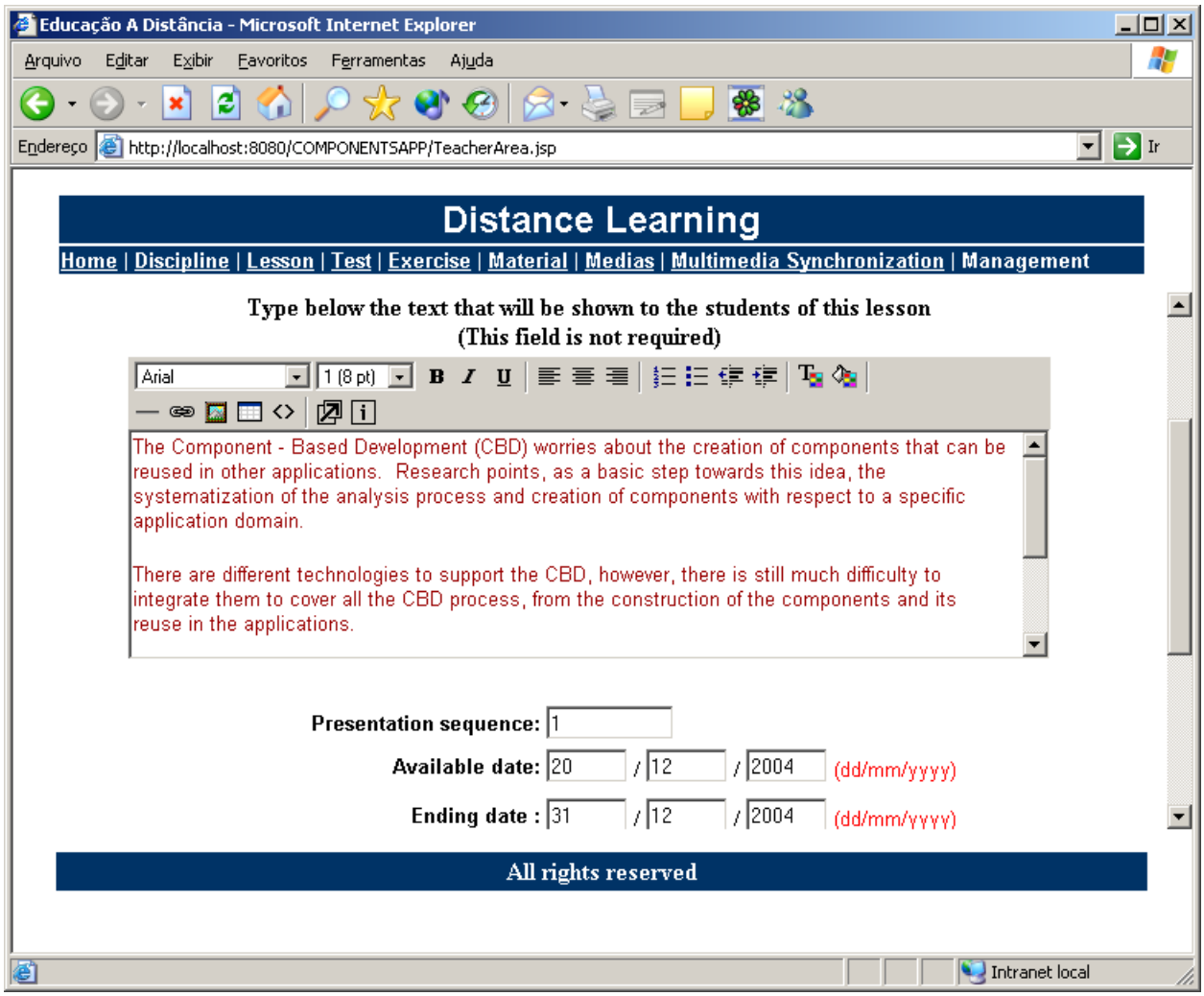

Figura 38: Tela da aplicação construída.

sidade de se alterar o *framework*, pois seus componentes nem sempre atendiam exatamente aos requisitos da aplicação. Porém, como o *framework* já havia adquirido o *status* de *baseline*, alterações sobre o mesmo só poderiam ser efetuadas após solicitadas, avaliadas e aprovadas, conforme definido pela abordagem da *SoCManager*.

Para exemplificar esse processo, apresenta-se uma dessas situações de mudança: um dos Engenheiros de Software identificou que o acompanhamento das aulas considerava uma aula concluída apenas quando o aluno havia respondido a todos os exercícios. Não era feita nenhuma verificação se o aluno havia visto todos os materiais disponíveis.

Seguindo o processo da *SoCManager*, o Engenheiro de Software que identificou a necessidade de mudança solicitou, por meio da ferramenta *SoCManager* (Figura 39 (a)), uma alteração sobre as classes de análise **Accompaniment** (Figura 39 (b)) e **AccomplishLesson** (Figura 39 (c)), responsáveis pelo comportamento em questão, localizadas dentro do artefato **FrameworkEAD.xmi**.

Após a requisição da mudança, esta foi analisada e aprovada por um dos membros do Comitê

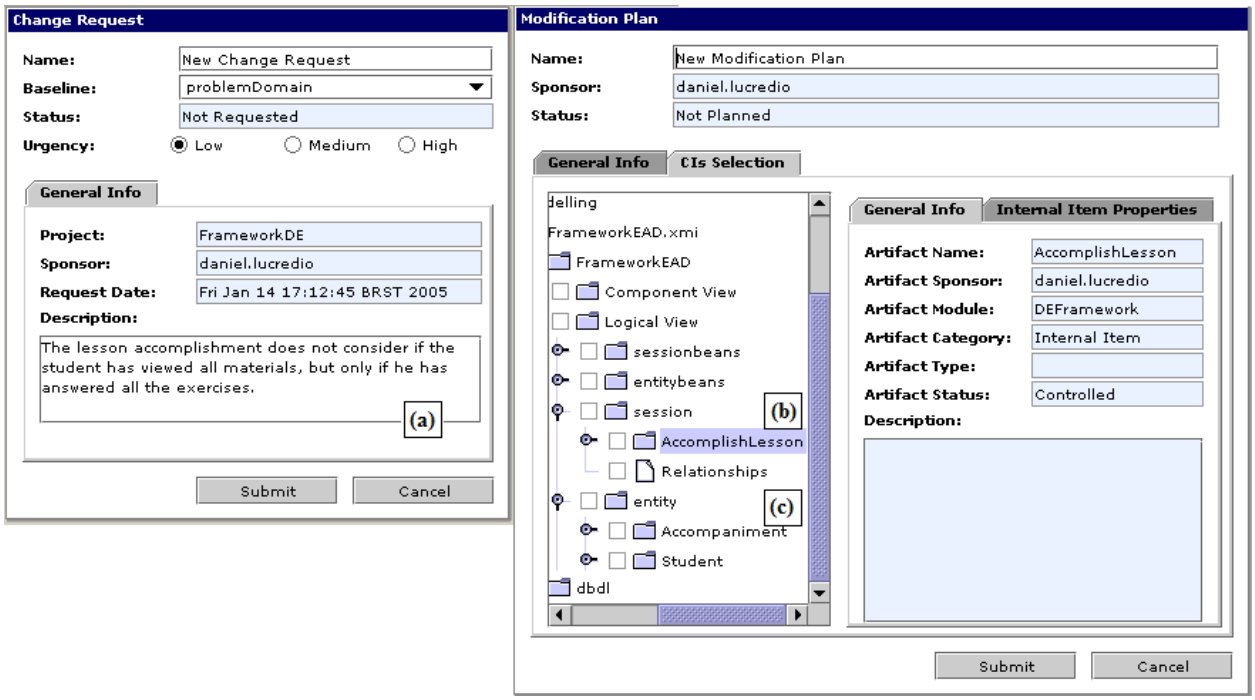

Figura 39: Solicitação de mudança sendo realizada na *SoCManager*.

de Controle de Configuração que, em seguida realizou uma solicitação de modificação sobre a *baseline* de implementação definindo, dentre os Engenheiros de Software que compõem a equipe de desenvolvimento do projeto "*FrameworkDE*", uma equipe de manutenção responsável pela realização da modificação.

Esta equipe de manutenção realizou então o planejamento da modificação, que envolveu a análise de impacto das modificações citadas na solicitação. A Figura 40 mostra uma tela da *SoC-Manager* que auxilia nessa análise, mostrando os relacionamentos da classe **AccomplishLesson**. Neste exemplo, identificou-se que uma alteração sobre a classe **AccomplishLesson** implicaria em alterações sobre a classe EJB gerada, **AccomplishLessonBean** (a), sobre as interfaces EJB geradas, **AccomplishLesson** e **AccomplishLessonHome** (b), e sobre o componente EJB gerado **AccomplishLessonBean** (c), todos dentro do mesmo XMI. Continuando com a análise, seriam necessárias também alterações sobre os artefatos Java **AccomplishLesson.java**, **AccomplishLessonBean.java** e **AccomplishLessonHome.java**.

Realizado o planejamento da modificação, este plano é avaliado pelos membros do Comitê de Controle de Configuração que, dependendo dos impactos da mudança e do tempo requerido para a modificação, aprovam ou rejeitam a modificação. Neste caso, o plano foi aprovado e, a partir deste momento, os Engenheiros de Software integrantes da equipe de manutenção da modificação, puderam alterar o artefato XMI correspondente ao modelo do *framework*, importando-o para a MVCASE (Figura 41). No caso, foi inserido um novo atributo na classe **Accompaniment** (a)

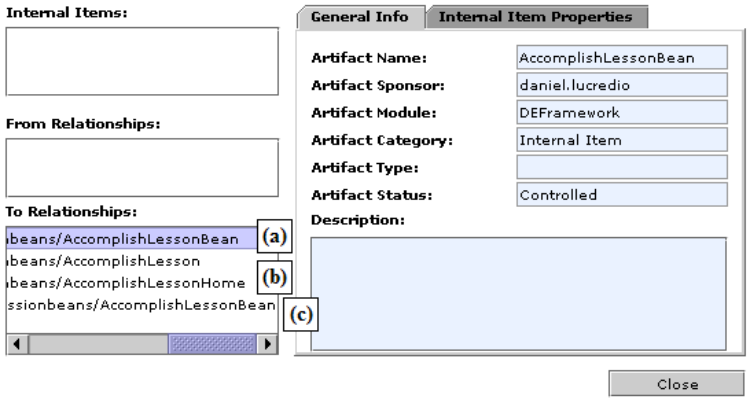

Figura 40: Análise de impacto na ferramenta *SoCManager*.

para armazenar quais materiais foram vistos. Além disso, o método **verifyCompletedLesson()** da classe **AccomplishLesson**, que verifica se uma aula foi completada, foi modificado para consultar lista de materiais vistos pelo aluno além dos exercícios respondidos.

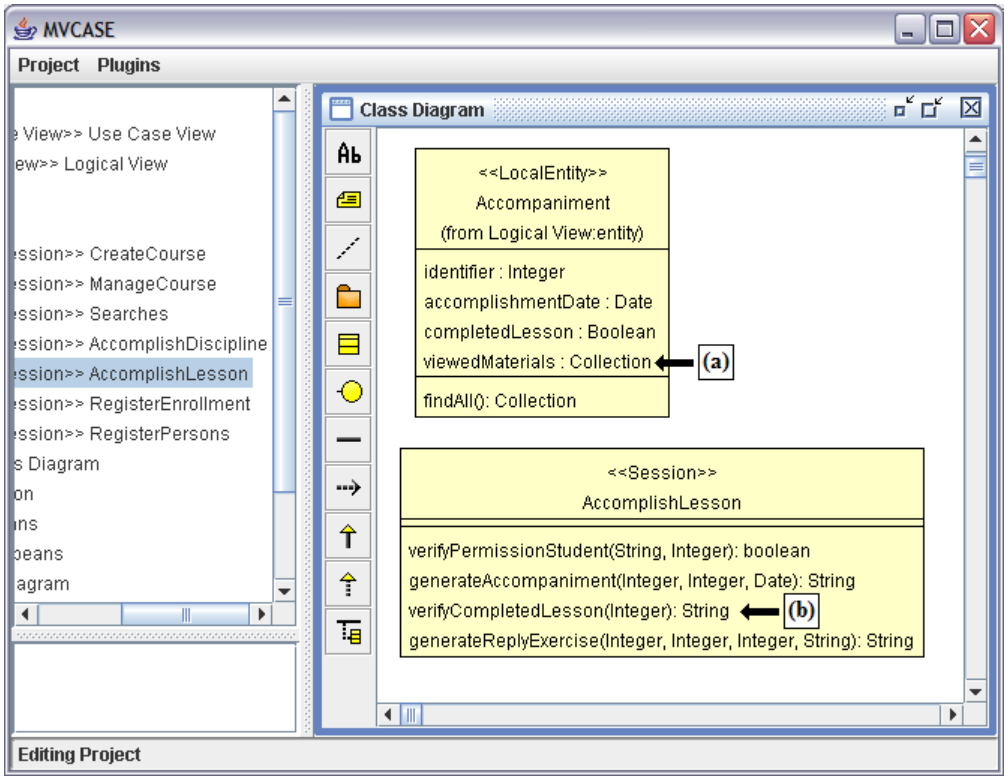

Figura 41: Alteração sendo efetuada na MVCASE.

A propagação das alterações foi efetuada por meio do *plugin* EJB da MVCASE [1], que regerou os artefatos EJB, e por meio do *plugin* para geração de código Java, que regerou os arquivos Java correspondentes. Dessa forma, propagou-se as alterações conforme definido na análise de impacto.

Terminada a alteração sobre os artefatos, a equipe de manutenção requisitou a conclusão da

[1]Mais informações sobre a arquitetura em plug-ins da MVCASE podem ser vistas em (LUCRÉDIO, 2005)

modificação que em seguida foi aprovado pelo Auditor. Após a aprovação, a *SoCManager* atualiza os CIs da *baseline*, permitindo que as novas versões estejam disponibilizados para reuso.

4.5 Avaliação do Estudo de Caso

O estudo de caso durou aproximadamente 4 meses e durante este período puderam ser avaliados os resultados desenvolvidos nesta pesquisa. Em relação à abordagem da implantação, a sua utilização como metodologia para o desenvolvimento do estudo de caso facilitou o controle sobre a aplicação do processo de GCS nos projetos. Isso se deve principalmente ao fato desta abordagem estar direcionada especificamente à implantação do processo de GCS. No entanto, faz-se necessária a sua aplicação no meio industrial para que possam ser feitas análises mais aprofundadas sobre o impacto de sua utilização na implantação de processos de GCS.

Outro benefício observado na aplicação da abordagem da implantação vem da atividade de "Avaliação dos Resultados Obtidos na Implantação do Processo de GCS", que permitiu avaliar os pontos fortes e fracos da abordagem do processo e da ferramenta desenvolvidas nesta pesquisa. Os alunos foram questionados sobre os possíveis benefícios observados. Dentre eles, destacam-se:

- facilidade de acesso ao CVS utilizando a ferramenta *SoCManager*;
- maior conhecimento sobre a configuração do projeto;
- controle de alteração sobre os componentes de software reutilizados pelas aplicações;
- divisão dos artefatos desenvolvidos no projeto em *baselines*, permitindo um maior enfoque no desenvolvimento dessas *baselines* e uma visão clara sobre a evolução do projeto;
- possibilidade de visualizar elementos da UML diretamente durante o processo de GCS, facilitando o seu controle;
- facilidade na reutilização dos artefatos desenvolvidos em outros projetos de software cadastrados na *SoCManager*.

A *SoCManager* se mostrou muito útil em relação a redução da complexidade no desenvolvimento de software, pois, por meio dela os membros do projeto puderam desempenhar atividades específicas, de tal forma que o Engenheiro de Software, por exemplo, se preocupava apenas com o desenvolvimento e manutenção dos artefatos, sem precisar configurar a conexão com o repositório CVS.

Além disso, a *SoCManager* ofereceu por meio de suas funcionalidades, acesso a recursos do CVS como, por exemplo, o uso de etiquetas, de forma transparente. Dessa forma, sempre que o Engenheiro de Software solicitar a obtenção dos CIs de uma dada *baseline*, a conexão com o repositório do CVS será transparente e ele terá acesso a uma cópia destes CIs.

Os alunos também foram questionados quanto a possíveis limitações e outras sugestões. Foram citados a necessidade de se ter um apoio, por parte da *SoCManager*, para o gerenciamento das tarefas de cada membro da equipe do projeto. Também foi identificada a necessidade de se agilizar o gerenciamento de mudanças referente à atividade de Controle de Configuração da abordagem do processo, implementada na *SoCManager*. Para isso, seriam necessários meios mais flexíveis com menos iterações, facilitando principalmente as alterações relacionadas à correção de erros do software. No que diz respeito ao seu desempenho, a *SoCManager* apresentou uma certa lentidão (ocasionado pelo grande tráfego na rede e no gerenciamento das informações persistidas no banco de dados), principalmente quando o número de artefatos referentes a uma dada *baseline*, aquisição de CIs ou planejamento da modificação é muito grande.

Estes e outros pontos relacionados à evolução e melhoria da *SoCManager* são discutidos na seção 5.3 de trabalhos futuros.

4.6 Resumo da Avaliação

Neste capítulo foi apresentado o estudo de caso realizado para avaliar os resultados desta pesquisa.

Devido ao fato do estudo de caso ter sido realizado em laboratório, não foi possível avaliar os resultados em toda a sua extensão.

A abordagem da implantação, por exemplo, foi utilizada como metodologia para a realização do estudo de caso, porém há a necessidade de se ter a sua aplicação no meio industrial de tal forma que se possa avaliar mais amplamente a sua contribuição na implantação de processos de GCS.

Já para a *SoCManager*, foram utilizadas todas as suas funcionalidades, de tal forma que pode-se obter as impressões dos alunos que atuaram no estudo de caso, o que contribuiu para a concepção de trabalhos futuros referentes a esta pesquisa.

Em relação à abordagem do processo, apenas seus aspectos implementados pela *SoCManager* foram avaliados neste estudo de caso, devido à limitação de tempo e ao fato do foco desta pesquisa ter sido o desenvolvimento da ferramenta.

No estudo de caso foram desenvolvidos quatro projetos: dois projetos para construção de com-

ponentes para os domínios de Educação à Distância (EAD) e Multimídia; e outros dois projetos para desenvolver duas aplicações que reutilizassem os componentes desenvolvidos pelos dois primeiros. A construção dos *frameworks* EAD e multimídia envolveu a geração 41 classes de análise, que deram origem a 41 componentes EJB. Foram gerados, para esses componentes, 123 arquivos Java. A construção das aplicações envolveu 49 páginas JSP, reutilizando estes componentes. O tempo decorrido na execução da abordagem foi de 4 meses, sendo que nos três primeiros meses teve-se a construção dos domínios, e no último mês teve-se a construção das aplicações.

Para o desenvolvimento dos projetos foram utilizadas as ferramentas do ambiente *Orion*: a *SoCManager* (para apoio ao GCS); a MVCASE (para modelagem e geração de código); e a C-CORE (para a geração da interface e depuração do código gerado pela MVCASE).

Do estudo de caso foram identificadas várias necessidades que devem ser atendidas pelas ferramentas. Para a *SoCManager* em particular, foram feitas várias críticas e observações que são discutidas no capítulo 5.

5 *Considerações Finais e Trabalhos Futuros*

O objetivo desta pesquisa é de auxiliar empresas de software na implantação do processo de GCS. Dessa forma, apresentou-se nesta dissertação a ferramenta *SoCManager*, que apóia a execução deste processo em projetos de software. Adicionalmente, foram apresentados outros resultados que foram realizados para que se pudesse atender ao objetivo do projeto e construir a *SoCManager*. São eles: uma abordagem para a implantação do processo de GCS; e uma abordagem para o processo de GCS.

Apresenta-se neste capítulo os trabalhos relacionados à ferramenta *SoCManager*, as contribuições desta pesquisa e os trabalhos futuros para cada resultado apresentado nesta dissertação.

5.1 Trabalhos Relacionados

Apresentou-se na seção 2.5 algumas ferramentas de GCS disponíveis no mercado e que auxiliam na execução do processo de GCS. Adicionalmente, apresenta-se nesta seção ferramentas desenvolvidas em pesquisas relacionadas com este processo.

(CRNOKOVIC, 1998) apresenta o SDE (*Software Development Environment* ou Ambiente de Desenvolvimento de Software), uma ferramenta de GCS que oferece recursos como o gerenciamento de versões, de configuração e de manutenção de códigos. Nesta ferramenta, tanto a criação de artefatos como alterações são realizadas no contexto de requisição de mudança. Ela também oferece um módulo para mensurar as alterações de GCS. Os resultados dessas medidas podem ser apresentados por meio de gráficos onde é possível efetuar uma análise de alguns pontos relativos à atividade de alteração dos artefatos mostrando, por exemplo, o reflexo do uso de diferentes modelos de ciclo de vida durante o desenvolvimento do projeto.

Völzer e outros (VÖLZER et al., 2002) apresentam uma ferramenta de GCS, denominada SubCM (*Subsystem Configuration Management*), que é baseada em um *framework* para o gerenciamento de configuração em sub-sistemas (LINDSAY et al., 2001). Estes subsistemas são definidos como

uma coleção de artefatos de software incluindo código, documentação e testes. Esta ferramenta gerencia uma estrutura hierárquica de um conjunto de artefatos de software, não estando restrito apenas a códigos fonte. Permite o controle de mudanças realizadas por diferentes usuários em diferentes equipes de desenvolvimento, tendo em vista o fato de um artefato de software poder estar sendo utilizado em diferentes projetos.

Estes dois trabalhos possuem características peculiares de grande valor para o apoio ao processo de GCS. No entanto, elas não realizam a distinção das atividades dos diferentes agentes que atuam no processo, implicando num maior grau de dificuldade na sua execução.

Neste sentido, cita-se o trabalho de Figueiredo e outros (FIGUEIREDO; SANTOS; ROCHA, 2004) que apresentam um processo de GCS para ambientes de desenvolvimento de software orientados à organização. Segundo Figueiredo e outros, estas organizações enfatizam a necessidade da gestão do conhecimento relacionado com a produção de software de uma determinada organização. Neste contexto, foi definido um processo de GCS que foi baseado na norma ISO/IEC 12207 e SWEBOK (IEEE, 2004), apoiando as definições para a área de processo de Gerenciamento de Configuração do CMMi. Este processo define os gerentes de projeto e desenvolvedores como agentes que colaboram na execução de uma série de atividades: Planejar Gerência da Configuração; Identificar Configuração; Controlar Configuração; Relatar Situação; Auditoria da Configuração; e Gerência de Liberação e Entrega. Com base nesta abordagem foi desenvolvida uma ferramenta, denominada GConf que apóia a sua execução por meio de três módulos: Módulo do Gerente do Projeto; Módulo dos Responsáveis pelo Item de Configuração; Módulo da Equipe do Projeto.

Contudo, o GConf não permite a definição de detalhes sobre a configuração dos artefatos do projeto e não realizam a análise de impacto de possíveis mudanças sobre os artefatos de software. Isto é realizado na *SoCManager* por meio do catálogo de itens internos e relacionamentos e da verificação destas informações ao analisar o impacto sobre a mudança de um dado CI do projeto.

Com base nestas informações e em análises realizadas sobre as ferramentas de GCS citadas neste trabalho, apresenta-se na tabela 17 a comparação de algumas dessas ferramentas com a *SoCManager*. Foram utilizados na comparação alguns dos critérios definidos por Burrows e outros (BURROWS; GEORGE; DART, 1996) (citados na seção 2.5) e, para cada critério, foi definido um grau de satisfação que parte do 0 (não atende) até 5 (atende totalmente).

A ferramenta comercial *ChangeMan*, seguida pela *ClearCase*, atende de forma satisfatória os critérios analisados nesta comparação. Estas ferramentas já são comercializadas a vários anos e possuem uma equipe de desenvolvimento dedicada no seu aperfeiçoamento, que ocorre de acordo com as necessidades de seus clientes. No entanto, ressalta-se que a *SoCManager*, cujo desenvolvimento foi realizado neste projeto de pesquisa, é uma ferramenta computacional gratuita que

Tabela 17: Comparação da *SoCManager* com outras ferramentas de GCS utilizando os critérios de Burrows (BURROWS; GEORGE; DART, 1996)

Critérios	***SoCManager***	***ClearCase***	***ChangeMan***	**SubCM**	**GConf**
Suporte à Equipe de Desenvolvimento	2	4	5	2	1
Suporte ao Desenvolvimento Remoto	3	4	4	3	1
Suporte à Configurações	3	4	4	3	0
Suporte à Gerencia de Mudanças	3	2	4	2	3
Suporte à Construção e Distribuição de Produto	1	4	5	0	2
Suporte ao Processo	4	2	4	2	3
Personalização	0	2	3	0	1

contempla boa parte dos critérios analisados, com exceção do critério “Personalização” que será tratado mais adiante na seção 5.3 de trabalhos futuros. Além disso, observa-se por meio da tabela 17 que tanto a *SoCManager* como a *SubCM*, salvo as suas peculiariedades, obtiveram resultados similares nessa análise comparativa.

Adicionalmente, na tabela 18 são mostrados outros critérios utilizadas na análise comparativa das ferramentas, buscando ressaltar as contribuições da *SoCManager*. São eles: a) Apoio a uma abordagem para o processo de GCS; b) Gerenciamento de mudanças e de configuração em granularidade fina, isto é, controle de artefatos em níveis de abstração maiores do que arquivos; c) Integração com ferramentas de desenvolvimento de software; d) Divisão do projeto em unidades menores, permitindo a aplicação das atividades de GCS sob cada unidade. Utilizou-se nesta tabela a mesma técnica de grau de satisfação adotada na tabela 17.

Observa-se a partir da tabela 18 que a *SoCManager* é a ferramenta que, em geral, melhor atende aos critérios analisados. Segue a análise de cada critério:

- **Apoio às atividades do processo de GCS:** A *SoCManager* apóia parte das atividades de GCS definidas na abordagem do processo referentes a Identificação e Controle de Configuração. Já a ferramenta *GConf* se destaca por atender a todas as atividades de GCS (inclusive Gerenciamento e Entrega de Liberações) definidas na abordagem de GCS adotada para a sua implementação. Por fim, as ferramentas *ChangeMan* e *ClearCase* apóiam atividades específicas de GCS, como controle de versões, mas que não estão dentro do contexto de um processo apoiado pela ferramenta;

- **Gerenciamento de mudanças e de configuração em granularidade fina:** Dentre as fer-

Tabela 18: Comparação da *SoCManager* com outras ferramentas de GCS utilizando critérios para ressaltar as contribuições da *SoCManager*

Critérios	***SoCManager***	***ClearCase***	***ChangeMan***	**SubCM**	**GConf**
Apoio às atividades do processo de GCS	3	1	2	1	5
Gerenciamento de mudanças e de configuração em granularidade fina	3	0	0	0	0
Integração com ferramentas de desenvolvimento de software	3	4	2	1	1
Divisão do projeto em unidades menores	5	1	3	4	0

ramentas analisadas, apenas a *SoCManager* atende a este critério por meio da definição dos itens internos dos artefatos, realizada durante a execução da funcionalidade de aquisição de CIs e, posteriormente, da funcionalidade de implementação de mudanças. Além disso, um das propostas apresentadas na seção 5.3 de trabalhos futuros envolve o controle de versões de metadados, que permitiria o melhor atendimento a este critério. Já as demais ferramentas são centradas em arquivos e portanto não atendem a este critério.

- **Integração com ferramentas de desenvolvimento de software:** A *SoCManager* está integrada a outras ferramentas de desenvolvimento de software por meio da introspecção de artefatos descritos segundo o padrão XMI, facilitando a sua aquisição e controle. Já a ferramenta *ClearCase* se destaca em relação a este critério por apresentar uma grande integração com outras ferramentas computacionais da IBM Rational [1].

- **Divisão do projeto em unidades menores:** Tanto a *SoCManager* (por meio dos módulos do projeto) como a *SubCM* (por meio dos subsistemas) atendem de forma satisfatória este critério, com destaque a primeira, por permitir por meio da funcionalidade "Controlar Designações do Módulo" (seção 3.3.1), onde o Gerente do Projeto pode definir quais membros da equipe atuam sobre um dado módulo. Além disso, na *SoCManager* é possível definir um repositório para cada módulo do projeto. Já as ferramentas *ChangeMan* e *ClearCase* apresentam algumas funcionalidades que permitem a divisão do projeto em unidades menores, mas não de forma direta. Por fim, a *GConf* não atende a este critério, tratando todos os artefatos do projeto como integrantes de uma mesma unidade de gestão.

Na seção 5.3 de trabalhos futuros são propostos trabalhos, como o de controle de versões de

[1] `http://www.ibm.com/`, Consultado em Dezembro/2004

metadados, que irão aprimorar as funcionalidades da *SoCManager* de tal forma que esta ferramenta atenda de forma mais satisfatória tanto os critérios analisados na tabela 17 como os critérios analisados na tabela 18

5.2 Principais Contribuições

Este trabalho oferece diferentes contribuições principalmente para a comunidade de desenvolvimento em geral. Estas contribuições estão relacionadas com cada um dos três resultados apresentados neste documento, sendo que cada um destes resultados traz contribuições que atendem ao objetivo desta pesquisa em auxiliar empresas na implantação do processo de GCS.

A abordagem da implantação define um conjunto de atividades que podem ser aplicadas em empresas de software que desejam realizar a implantação deste processo em seus projetos. Esta abordagem, que é uma instância simplificada do modelo IDEAL, permite que se tenha todo o foco voltado para a implantação do processo de GCS, levantando aspectos particularmente referentes a este processo. O intuito é que esta abordagem possa ser aplicada por empresas de pequeno e médio portes.

Além disso, esta abordagem pode estar sendo aplicada isoladamente ou em conjunto com os demais resultados deste projeto de pesquisa, pois define as atividades como: “Definição de uma Abordagem para o Processo de GCS”, onde se tem a elaboração ou adoção de uma abordagem para o processo de GCS (no caso pode ser adotada a abordagem do processo definida neste projeto de pesquisa); e “Seleção de Ferramentas de GCS”, onde se tem a escolha de ferramentas que apóiem a execução da abordagem (no caso pode ser escolhida a ferramenta *SoCManager*).

A abordagem do processo define atividades que englobam os principais aspectos de GCS citados por (IEEE, 1998) e (IEEE, 2004), dando suporte ao cumprimento das metas definidas pela área de processo de Gerenciamento de Configuração do modelo de capacitação CMMi. Esta abordagem pode estar sendo aplicada nas empresas de software tanto em conjunto com os demais resultados deste projeto de pesquisa como isoladamente. Para este segundo caso, deve-se obter a necessidade de obter ferramentas de GCS que ofereçam o apoio necessário para a realização das atividades da abordagem.

Já a ferramenta *SoCManager* oferece apoio à execução das atividades da abordagem do processo, auxiliando as equipes de desenvolvimento na execução do processo de GCS nos projetos de software. Por meio da ferramenta, tem-se o suporte à Identificação e Controle da Configuração por diferentes perspectivas associadas a cada agente que atua no projeto de software, segundo definido na abordagem de GCS. Por meio destas perspectivas, tem-se a divisão das responsabilidades dos

agentes relacionados com o processo, de tal forma que cada um tenha acesso apenas às informações necessárias para a execução de suas atividades, implicando numa redução das dificuldades em se executar o processo de GCS. Além disso, pelo motivo desta ferramenta estar baseada em uma abordagem, tem-se uma maior facilidade em compreender suas funcionalidades e aplicá-las da melhor forma no desenvolvimento de projetos de software.

A *SoCManager* efetua o GCS do projeto, sendo apoiada pela ferramenta CVS para controle de versões dos artefatos. Por este motivo, esta ferramenta pode ser facilmente aplicada a empresas que já utilizam o CVS como ferramenta para controle de versões. Além disso, cada módulo de um projeto cadastrado na *SoCManager* pode estar sendo associado a um repositório diferente, permitindo que se tenha os artefatos distribuídos geograficamente.

A *SoCManager* também permite que se tenha um cadastro de informações mais detalhadas sobre a configuração do projeto, por meio da adoção dos conceitos de Item Interno e Relacionamento (CUNHA; NAKANISH, 1993). A partir destas informações detalhadas, a ferramenta apóia uma maior análise de impacto sobre a configuração de software, auxiliando na previsão de esforços necessários para a realização de uma dada modificação.

Outra característica presente na SoCManager é o fato dela permitir a introspecção de artefatos de software descritos segundo o padrão XMI (OMG, 2002b), facilitando o detalhamento da configuração dos artefatos descritos segundo este padrão. Além disso, devido a esta característica, a *SoCManager* pode estar sendo mais facilmente integrada com ferramentas que utilizam o XMI para representar as suas informações (como, por exemplo, as ferramentas do ambiente *Orion*).

5.3 Trabalhos Futuros

Outra forma de contribuição importante em um projeto de pesquisa é a geração de trabalhos futuros. Com relação à presente dissertação, uma série de possibilidades foi identificada.

Muitas empresas de software de pequeno e médio porte atualmente ainda não possuem o processo de GCS sendo executando no desenvolvimento de seus projetos. Neste caso, tem-se a possibilidade de estarem sendo realizadas pesquisas ação (realizadas nas empresas de software) que realizem a implantação deste processo com base na abordagem da implantação, definido neste projeto de pesquisa. Como possíveis contribuições para estes projetos, cita-se a própria melhoria da abordagem, obtendo informações concisas sobre a duração na execução de cada atividade da abordagem, como também o fortalecimento das empresas do setor de software.

Além de projetos para a aplicação da abordagem para a implantação do processo de GCS, outros projetos de pesquisa podem estar sendo focados para a aplicação da abordagem para o pro-

cesso de GCS, buscando com isso detalhar e aperfeiçoar os aspectos deste processo nela definidos.

Outro ponto relevante seria a adequação da abordagem para o processo de GCS, visando atender mais especificamente a paradigmas como o Desenvolvimento Baseado em Componentes (DBC) (PRESSMAN, 2001) e o Desenvolvimento de Software Orientado Aspectos (KICZALES et al., 1997). Tem-se, por exemplo, a necessidade de se definir quais tipos de CIs, baseando-se nos conceitos da abordagem, que poderiam estar sendo controlados para cada um destes paradigmas, a exemplo do projeto realizado por Itami (ITAMI, 1997).

Além disso, no caso de se realizar o trabalho voltado para o DBC, cita-se, como um trabalho relacionado, o projeto proposto por Murta (MURTA, 2004) para a definição da abordagem Odyssey-SCM para o GCS voltado para o DBC.

Em relação à ferramenta *SoCManager*, tem-se naturalmente o trabalho que visa atender aos demais aspectos de GCS definidos na abordagem proposta neste projeto de pesquisa. Estes aspectos estão relacionados com as atividades de Administração de Estado e Auditoria da Configuração.

Além disso, citam-se outras possíveis evoluções na ferramenta. Primeiramente, tem-se a definição do Módulo Administrativo da *SoCManager*. A proposta é que a *SoCManager* tenha um módulo administrativo que seja acessível a membros que tenham o papel CMO (com base neste agente definido na abordagem do processo). Para tanto, deve-se implementar o suporte a este papel, que será associado a pessoa que irá instalar a *SoCManager* e a membros da ferramenta que posteriormente estarão sendo selecionados por ele. A partir da criação deste módulo, pode-se ter:

- **Gerenciamento de Membros:** Esta proposta visa criar e realizar o caso de uso “Gerenciar Membros” (*Manager Members*) onde o CMO poderá inserir, alterar e remover o cadastro de membros da *SoCManager*;

- **Requisição de Cadastro de Membros:** Esta proposta visa alterar o caso de uso de “Requisitar Nova Conta” (*Request New Account*) (Figura 42), de tal forma que a pessoa que deseja se tornar membro da *SoCManager* (*Not Member* ou Não Membro) deva realizar uma prévia requisição a ser aprovada pelo CMO.

- **Membro do Comitê de Engenharia de Processo:** Esta proposta visa adicionar o papel “Membro do Comitê de Engenharia de Processo” (referente ao agente PEB da abordagem do processo) na ferramenta. Este papel é independente de projetos cadastrados na ferramenta e estaria sendo associado aos membros por meio do módulo administrativo pelo CMO. Esta proposta engloba também a alteração do caso de uso “Controlar de Tipos de CIs” para que este possa ser realizado por este membro (atualmente esse caso de uso é realizado pelo “Membro do Comitê de Controle de Configuração”).

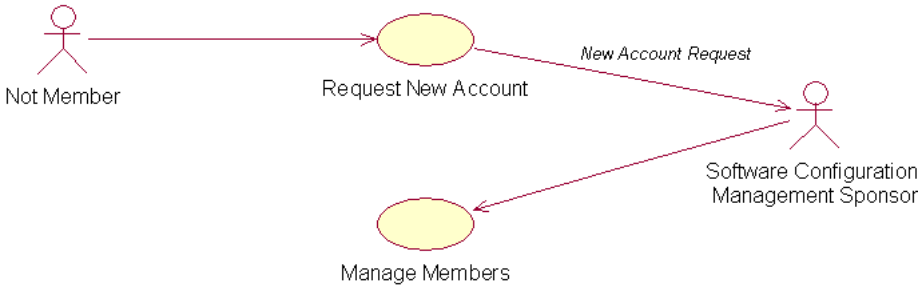

Figura 42: Casos de Uso para registrar nova conta na *SoCManager*

- **Membro do Comitê de Garantia da Qualidade:** Esta proposta visa adicionar o papel "Membro do Comitê de Garantia da Qualidade" (referente ao agente QAB da abordagem do processo) na ferramenta. Este papel é independente de projetos cadastrados na ferramenta e estaria sendo associado aos membros por meio do módulo administrativo pelo CMO. Esta proposta engloba também a criação e realização de dois novos casos de uso (Figura 43): "Controlar os Critérios de Aprovação dos Tipos de CIs", onde o Membro do Comitê de Garantia da Qualidade estaria definindo para cada tipo de CI, definido pelo caso de uso "Controlar Tipos de CIs", os critérios de aprovação que serão utilizados pelo Auditor para avaliar os artefatos relacionados com aquele tipo no momento da aquisição e no momento de conclusão de uma modificação; e "Controlar os Critérios de Aprovação de Baselines" (*Control Baselines Approval Criteria*), onde o Membro do Comitê de Garantia da Qualidade estaria definindo para cada *baseline*, definida pelo caso de uso "Controlar *Baselines*", os critérios de aprovação que serão utilizados pelo Auditor para avaliar as *baselines* no momento em que o Gerente do Projeto solicitar a sua aquisição. Tanto os critérios de aprovação de CIs como os critérios de aprovação de *baselines* podem também estarem sendo utilizados pelo auditor nas atividades referentes à Auditoria da Configuração.

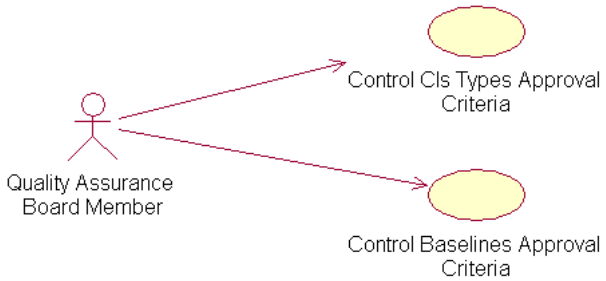

Figura 43: Casos de Uso referentes ao Membro do Comitê de Garantia da Qualidade

- ***Stakeholder*:** Esta proposta visa adicionar o papel "*Stakeholder*"[2] na *SoCManager*. O *Sta-*

[2]Entidade (pessoa física ou jurídica) que utiliza os resultados do projeto (PMI, 2000). Pode ser alguém que esteja utilizando uma aplicação resultante de um projeto ou também algum profissional ou departamento da própria organização, atuante em outro projeto, que o esteja reutilizando.

keholder poderá realizar o acompanhamento do projeto e requisitar mudanças. Adicionalmente, tem-se a alteração do caso de uso “Gerenciar Projeto” permitindo ao Gerente do Projeto definir os *Stakeholders* associados àquele projeto.

Outro módulo que poderia estar sendo adicionado na ferramenta é o de Gerenciamento de Tarefas. A proposta é que toda a atividade de GCS definida na *SoCManager* ocorra sob o contexto de uma tarefa. Dessa forma, os membros da ferramenta podem ter um controle sobre suas atividades que devem ser executadas como, por exemplo, a avaliação das requisições de mudança, planejamentos de modificação, entre outros. Além disso, o Gerente do Projeto poderá estar acompanhando a execução das tarefas de cada membro da sua equipe de desenvolvimento.

Cita-se também a proposta de evoluir o módulo “*Repository*”. Atualmente, a *SoCManager* está integrada com a ferramenta CVS, sendo que os repositórios do CVS atuam como bibliotecas de software dos projetos cadastrados na ferramenta. No entanto, a proposta é que se tenha a possibilidade de cadastrar diferentes tipos de bibliotecas de software, incluindo até sistemas de arquivo, onde serão armazenados os artefatos cuja evolução não precisa ser controlada por um sistema de controle de versões. É mostrado na Figura 44 o módulo “*Repository*” acessando vários repositórios diferentes para armazenar as informações.

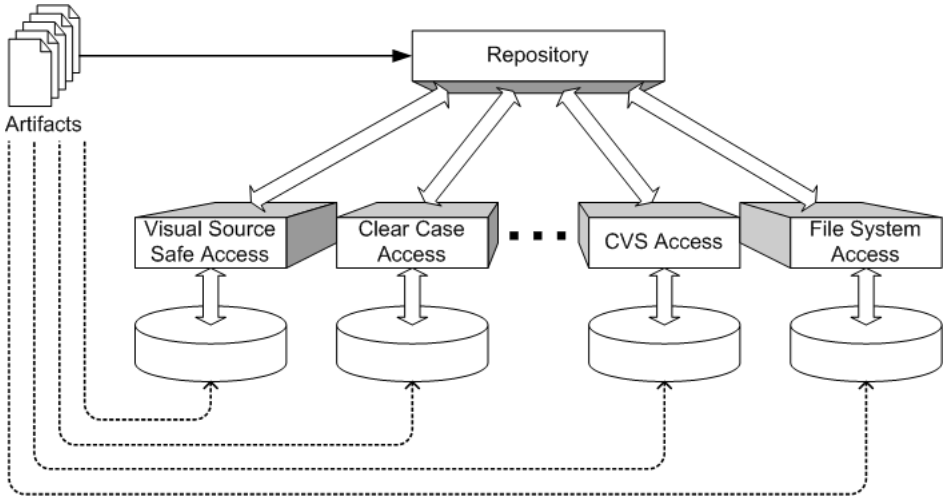

Figura 44: Evolução do Módulo Repositorio

Além disso, tem-se a proposta da *SoCManager* apoiar o controle de versões de Metadados. Atualmente a ferramenta permite a introspecção de artefatos descritos segundo o padrão XMI, auxiliando na sua aquisição e no cadastramento de sua configuração. No entanto, o controle de versões ocorre sobre o arquivo, pois o CVS, ferramenta utilizada na *SoCManager* para controle de versões dos artefatos, é aplicável apenas a arquivos. Porém há a necessidade de se ter um controle de versões em um nível de abstração menor, sobre os elementos internos destes arquivos. Dittrich e outros (DITTRICH; TOMBROS; GEPPERT, 2000) destacam que, em sistemas baseados em classes

e outras unidades de composição menores do que arquivos, o tratamento em nível de arquivo não é adequado a um processo efetivo de GCS. Adicionalmente, tem-se a necessidade de desenvolver um trabalho voltado para o controle de versões de metadados para atender as necessidades específicas de usuários de ferramentas de modelagem, como a MVCASE. A Figura 45 mostra o esquema referente ao controle de versões de artefatos XMI. Segundo a figura, seria implementado um acesso a um repositório de metadados (*Metadata Repository Access*) que iria interagir com o módulo "*Repository*" e realizar o controle de versões dos elementos internos do XMI com base nas informações obtidas pelo módulo "*XMI Explorer*". Nesta mesma linha de pesquisa, cita-se o trabalho apresentado por Oliveira e outros (OLIVEIRA; MURTA; WERNER, 2004) que consiste em uma ferramenta que efetua o controle de versões sobre os artefatos descritos em XMI por meio do uso do MDR e JMI, também utilizados na *SoCManager* para realizar a introspecção destes artefatos.

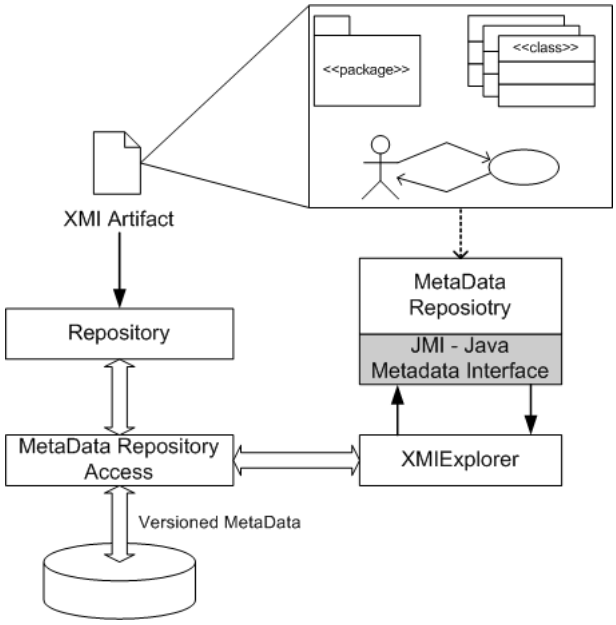

Figura 45: Controle de Versões de Metadados

Um outro aspecto que pode ser implementado na *SoCManager* é a criação da biblioteca de suporte. Atualmente a ferramenta possui apenas bibliotecas de software para desenvolvimento que são representadas pelos repositórios do CVS. No entanto, a proposta é que a ferramenta tenha também bibliotecas de software para suporte. Nesta biblioteca, os membros que atuam num dado projeto poderão colocar artefatos que armazenam conhecimentos que contribuam no melhor entendimento do projeto, seus objetivos, atividades e resultados. Como exemplo de artefatos para a biblioteca de suporte, pode-se citar entrevistas, depoimentos, notícias, entre outros. Devido ao fato de não haver a necessidade de controlar a evolução destes artefatos, esta biblioteca seria aplicável a repositórios que não realizam o controle de versões como, por exemplo, os sistemas de arquivos.

Por fim, cita-se também o trabalho referente à flexibilização do processo de GCS. A *SoCMa-*

nager implementa as atividades de GCS com base na abordagem para este processo, elaborada neste projeto de pesquisa. No entanto, isto limita o uso desta ferramenta por outras empresas que queiram utilizar outras abordagens. Além disso, foi identificada a necessidade de se ter outras formas de realizar modificações sobre os CIs, sem ter que passar pelas atividades de Requisição de Mudança, Solicitação de Modificação e Planejamento da Modificação (cita-se como exemplo o Engenheiro de Software que sabe qual alteração fazer e deseja efetuar o Planejamento da Modificação diretamente). Uma forma de se realizar esta flexibilização é a transformação da *SoCManager* em uma ferramenta centrada no processo, tendo todo o seu processo sob o contexto de Mecanismos de Fluxo de Trabalho (*workflow engines*) (MANOLESCU; PAUL, 2004). O controle sobre o processo naturalmente deve ser realizado pelo Membro do Comitê de Engenharia de Processo e, portanto, deve-se ter a inserção deste papel na *SoCManager* antes de desenvolver esta proposta.

Referências

ABNT. *Tecnologia de Informação - Processos de Ciclo de Vida de Software*. [S.l.], 1998.

AMBLER, S. W.; NALBONE, J. *Enterprise Unified Process(EUP): Enhancing the Rational Unified Process (RUP) to Meet the Real-World Needs of Your Organization*. [S.l.]: Ronin International, 2004.

ANSI/IEEE. *IEEE Guide to Software Configuration Management*. [S.l.], 1987.

ARGOUML. *ArgoUML tool*. ArgoUML, 2004. Disponível em: <http://argouml.tigris.org (17 June 2004)>. Acesso em: 17 June 2004.

BARRÉRE, T. S. *CASE com Múltiplas Visões de Requisitos de Software e Implementação Automática em Java - MVCASE*. Tese (Dissertação de Mestrado) — UFSCar - Universidade Federal de São Carlos, São Carlos - SP - Brazil, 1999.

BORLAND. *Borland Together*. Borland Software Corporation, 2004. Disponível em: <http://www.borland.com/together/ (17 June 2004)>. Acesso em: 17 June 2004.

BURBECK, S. *How to use Model-View-Controller (MVC)*. 2002. Disponível em: <http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html (9 November 2004)>. Acesso em: 9 November 2004.

BURROWS, C.; GEORGE, W. G.; DART, S. A. *OVUM Evaluates: Configuration Management*. Burlington - Massachussets: Ovum, 1996.

CASSIDY, A. *A Practical Guide to Information Systems Strategic Planning*. [S.l.]: St. Lucie Press, 1998.

CATARINO, I. C. S. *Framework para Ensino a Distância via Web*. Tese (Dissertação de Mestrado) — UFSCar - Universidade Federal de São Carlos, São Carlos, 2002.

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S. *CMMI®: Guidelines for Process Integration and Product Improvement*. [S.l.]: Addison Wesley, 2003.

CONRADI, R.; WESTFECHTEL, B. Version Models for Software Configuration Management. *ACM Computing Surveys*, v. 30, n. 2, p. 232–282, 1998.

CRNOKOVIC, I. A Change Process Model in an SCM Tool. In: *Euromicro Conference*. Suécia: [s.n.], 1998.

CUGOLA, G. et al. An Experience in Setting-Up a Configuration Management Environment Software Technology and Engineering Practice. In: *8th IEEE International Workshop on [incorporating Computer Aided Software Engineering]*. [S.l.]: IEEE Computer Society Press, 1997.

CUNHA, J. a. B. S. *Uma Abordagem de Qualidade e Produtividade para o Desenvolvimento de Sistemas de Software Complexos Utilizando a Arquitetura de Placa de Software - SOFTBOARD*. Tese (Tese de Doutorado) — INPE - Instituto Nacional de Pesquisas Especiais, São José dos Campos - SP - Brazil, 1997.

CUNHA, J. a. B. S.; NAKANISH, T. O Controle do Desenvolvimento e Manutenção de Sistemas de Software. In: *XIV Congresso Ibero Latino-Americano de Métodos Computacionais em Engenharia*. [S.l.: s.n.], 1993.

CUNHA, J. R. D. D.; PRADO, A. F. d.; SANTOS, A. C. d. Uma Abordagem para o Processo de Gerenciamento de Configuração de Software. *Revista Eletrônica de Sistemas de Informação (RESI)*, Numero 1, n. Novembro de 2004, 2004.

DART, S. A. *The Past, Present and Future of Configuration Management*. [S.l.], 1992.

DEMICHIEL, L. G. *Enterprise JavaBeans Specification, Version 2.1*. [S.l.], 2002.

DIRCKZE, R. *Java Metadata Interface (JMI) Specification*. [S.l.], 2002.

DITTRICH, K. R.; TOMBROS, D.; GEPPERT, A. Databases in Software Engineering: A Roadmap. In: *The Future of Software Engineering*. [S.l.]: ACM Press, 2000. p. 291–302.

D'SOUZA, D.; WILLS, A. *Objects, Components and Frameworks with UML: The Catalysis Approach*. [S.l.]: Addison-Wesley, 1999. (Object Technology Series).

FIGUEIREDO, S.; SANTOS, G.; ROCHA, A. R. Gerência de Configuração em Ambientes de Desenvolvimento de Software Orientados a Organização. In: *III Simpósio Brasileiro de Qualidade de Software (SBQS2004)*. Brasília: [s.n.], 2004.

GREMBA, J.; MYERS, C. *The IDEAL model: A practical guide to improvement.* [S.l.]: Software Engineer Institute, 1997.

GUOJUN, L. *Communication and Computing for Distributed Multimedia Systems*. [S.l.]: Artech House, 1996.

HEDGE, S. S. Introducing a Configuration Management Solution Corporate-Wide: An Experience Report. In: *Fifth International Workshop on Computer-Aided Software Engineering*. [S.l.]: IEEE Computer Society, 1992.

HIBERNATE. *Hibernate*. 2004. Disponível em: <http://www.hibernate.org/ (08 June 2004)>. Acesso em: 08 June 2004.

HOPE-ROSS, D.; DISBROW, J. B.; WOOD, B. *Choose the Right Business Application Software Solution*. [S.l.]: Gartner, Agosto, 2003.

HUSTED, T. *Struts in Action: Building Web Applications with the Leading Java Fram*. [S.l.]: Manning Publications, 2002.

IEEE. *IEEE Standard Glossary of Software Engineering Terminology*. [S.l.], 1990.

IEEE. *IEEE Standard for Software Configuration Management Plans*. [S.l.], 1998.

IEEE. *SWEBOK: Guide to the Software Engineering Body of Knowledge*. 2004 version. ed. Los Alamitos, California: IEEE Computer Society, 2004.

ISO/IEC. *Software Process Assessment - Part 1: Concepts and introductory guide. Version 1.00*. [S.l.], 1998.

ITAMI, S. N. *Proposta de um Esquema de Controle de Elementos de Software para a Abordagem Orientada a Objetos.* Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais (INPE), São José Dos Campos - SP, 1997.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. *The Unified Software Development Process*. [S.l.]: Addison-Wesley, 1999.

JBOSS. *JBoss*. 2004. Disponível em: <http://www.jboss.org/index.html (17 June 2004)>. Acesso em: 17 June 2004.

KICZALES, G. et al. Aspect-Oriented Programming. In: *11st European Conference Object-Oriented Programming (ECOOP'97)*. Finland: Springer Verlag, 1997. (LNCS, v. 1241), p. 220–242.

KRUCHTEN, P. *Introdução ao RUP: Rational Unified Process*. 1. ed. [S.l.]: Addison-Wesley, 2003.

LINDSAY, P. et al. A Framework for Subsystem-based Configuration Management. In: *Software Engineering Conference*. Australia: [s.n.], 2001.

LUCRÉDIO, D. *Extensão da Ferramenta MVCASE com Serviços Remotos de Armazenamento e Busca de Artefatos de Software*. Tese (Mestrado) — UFSCar - Universidade Federal de São Carlos, São Carlos - SP - Brasil, 2005.

LUCRÉDIO, D. et al. Orion - A Component Based Software Engineering Environment. *JOT - Journal of Object Technology*, v. 3, n. 4, p. 51–74, 2004.

MANOLESCU, D. A.; PAUL, S. *Workflow Engine Evaluation*. 2004. Disponível em: <http://micro-workflow.com/PDF/WorkflowEngineEvaluation.pdf (12 January 2005)>. Acesso em: 12 January 2005.

MCFEELEY, R. *IDEAL: A user guide for software process improvement*. [S.l.], 1996.

MCT. *Qualidade e Produtividade no Setor de Software Brasileiro*. [S.l.]: Ministério da Ciência e Tecnologia, 2002 2002. 1-260 p. Brasilia.

MICROSYSTEMS, S. *Applets*. 2004. Disponível em: <http://java.sun.com/applets/ (07 June 2004)>. Acesso em: 07 June 2004.

MICROSYSTEMS, S. *Java 2 Platform, Standard Edition, v 1.4.2 - API Specification*. Sun Microsystems, 2004. Disponível em: <http://java.sun.com/j2se/1.4.2/docs/api/ (17 June 2004)>. Acesso em: 17 June 2004.

MICROSYSTEMS, S. *Java BluePrints: Model-View-Controller*. 2004. Disponível em: <http://java.sun.com/blueprints/patterns/MVC-detailed.html (6 November 2004)>. Acesso em: 6 November 2004.

MICROSYSTEMS, S. *Sun Java Center J2EE Patterns: Value Objects*. 2004. Disponível em: <http://java.sun.com/j2ee/patterns/ValueObject.html (15 June 2004)>. Acesso em: 15 June 2004.

MURTA, L. *Odyssey-SCM: Uma Abordagem de Gerência de Configuração de Software para o Desenvolvimento Baseado em Componentes*. Tese (Doutorado (em andamento)) — Universidade Federal do Rio de Janeiro, Rio de Janeiro - RJ - Brasil, 2004.

NASA. *Software Configuration Management Guidebook*. [S.l.]: NASA (National Aeronautics and Space Administration), 1995.

NETBEANS. *Javacvs*. NetBeans project, 2004. Disponível em: <http://javacvs.netbeans.org (28 October 2004) (28 October 2004)>. Acesso em: 28 October 2004.

NETBEANS. *The Netbeans project*. NetBeans project, 2004. Disponível em: <http://www.netbeans.org (28 October 2004) (28 October 2004)>. Acesso em: 28 October 2004.

NETO, R. M. S. et al. Component-Based Software Development Environment (CBDE). In: *6th ICEIS - International Conference on Enterprise Information Systems*. Porto - Portugal: [s.n.], 2004.

OLIVEIRA, A. A. C. P. et al. *Gerência de Configuração de Software: Evolução de Software sob Controle*. [S.l.]: Instituto Nacional de Tecnologia da Informação (ITI), 2001.

OLIVEIRA, H.; MURTA, L.; WERNER, C. Odyssey-VCS: Um Sistema de Controle de Versões para Modelos Baseados no MOF. In: *XVIII Simpósio Brasileiro de Engenharia de Software - XI Sessão de Ferramentas*. Brasília - DF - Brasil: SBC, 2004.

OMG. *Meta Object Facility (MOF) Specification*. [S.l.], 2002.

OMG. *XML Metadata Interchange (XMI) Specification*. [S.l.], 2002.

OMG. *Unified Modeling Language Specification*. [S.l.], 2003.

OMONDO. *Eclipse UML Tool*. Omondo, 2004. Disponível em: <http://www.eclipseuml.com/ (17 June 2004)>. Acesso em: 17 June 2004.

PÁDUA, W. d. *Engenharia de Software: Fundamentos, Métodos e Padrões*. 2. ed. [S.l.]: LTC, 2003.

PAULK, M. C. et al. *Capability Maturity Model for Software, Version 1.1*. [S.l.], 1993.

PMI. *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*. [S.l.]: Project Management Institute, 2000.

PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach*. [S.l.]: McGraw-Hill, 2001.

ROSS, D. T. Structured analysis (SA): A language for communicating ideas. *IEEE Transactions on Software Engineering*, v. 3, n. 1, p. 16–34, 1977.

SANT'ANNA, N. *Um Ambiente Integrado para o Apoio ao Desenvolvimento e Gestão de Projetos de Software para Sistemas de Controle de Satélites*. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos - SP - Brasil, 2001.

SEI. *CMMI : Capability Maturity Model Integration (CMMISM) Version 1.1*. [S.l.], 2002.

SILVA, D. *Uma Ferramenta para Descoberta de Conhecimento com Suporte de Data Warehousing e sua Aplicação para Acompanhamento do Aluno em Educação a Distância*. Tese (Dissertação de mestrado) — UFSCar - Universidade Federal de São Carlos, São Carlos, 2002.

SILVA, D. Using Data Warehouse and Data Mining Resources for Ongoing Assessment of Distance Learning. In: *IEEE International Conference on Advanced Learning Technologies*. Kazan, Tatarstan, Russia: IEEE/CS Press, 2002.

SILVA, D.; VIEIRA, M. T. P. An Ongoing assessment model in distance learning. In: *Internet and Multimedia Systems and Applications*. Honolulu, USA: [s.n.], 2001.

SILVA, E. A. d. et al. *Tutorial Reutilização de Componentes (Projeto DBCM)*. 2004. Projeto DBCM - Desenvolvimento Baseado em Componentes Multimídia - DC - UFSCar.

SOMMERVILLE, I. *Software Engineering*. 6. ed. [S.l.]: Prentice-Hall, 2003.

UFSCAR. *DBCM - Desenvolvimento Baseado em Componentes Multimídia*. 2004. Projeto financiado pelo CNPq. Vigência: 12/2003 a 12/2004.

VÖLZER, H. et al. A Tool for Subsystem Configuration Management. In: *International Conference on Software Maintenance*. [S.l.: s.n.], 2002.

W3C. *Extensible Markup Language (XML)*. World Wide Web Consortium, 2004. Disponível em: <http://www.w3.org/XML/ (17 June 2004)>. Acesso em: 17 June 2004.

YOURDON, E. *Declínio e queda de analistas e programadores*. [S.l.]: Makron Books, 1995.