UNIVERSIDADE ESTADUAL DO CEARÁ
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO

VIVIANE ALMEIDA DOS SANTOS

APRENDIZADO ORGANIZACIONAL E MELHORIA CONTÍNUA DE PROCESSOS DE SOFTWARE ATRAVÉS DE REUSO DE PROCESSOS DE SOFTWARE

FORTALEZA
2009

VIVIANE ALMEIDA DOS SANTOS

APRENDIZADO ORGANIZACIONAL E MELHORIA CONTÍNUA DE PROCESSOS DE SOFTWARE ATRAVÉS DE REUSO DE PROCESSOS DE SOFTWARE

Dissertação submetida à Coordenação do Curso de Mestrado em Ciência da Computação da Universidade Estadual do Ceará, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Orientação: Profa. Dra. Mariela Inés Cortés.

FORTALEZA
2009

VIVIANE ALMEIDA DOS SANTOS

APRENDIZADO ORGANIZACIONAL E MELHORIA CONTÍNUA DE PROCESSOS DE SOFTWARE ATRAVÉS DE REUSO DE PROCESSOS DE SOFTWARE

Dissertação submetida à Coordenação do Curso de Mestrado em Ciência da Computação da Universidade Estadual do Ceará, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Orientação: Profa. Dra. Mariela Inés Cortés.

Aprovada em ___/___/_____

BANCA EXAMINADORA

_____

Profa. Dra. Mariela Inés Cortés
Universidade Estadual do Ceará

_____

Prof. Dr. Jerffeson Teixeira de Souza
Universidade Estadual do Ceará

_____

Prof. Dr. Rodrigo Quites Reis
Universidade Federal do Pará

Ao meu esposo, Gerson, por ter sido o maior incentivador deste trabalho.

AGRADECIMENTOS

À Deus, minha grande inspiração, pelo seu amor, misericórdia e capacidade infinita.
Ao Espírito Santo, que tem sublimemente me orientado durante toda a minha vida.

Ao meu esposo, Gerson, pelo amor, carinho e apoio.

Aos meus pais, Enésio e Francineide, pelo amor incondicional, apoio em todos os momentos de minha vida e enorme esforço para me proporcionarem uma formação digna.
Às minhas irmãs, Cristiane e Camila, pelo amor, amizade e por sempre me apoiarem nos meus desafios.
Aos meus familiares, à minha cunhada do coração, Gisele; aos cunhados Alfredo, Jean e Francisco; aos sobrinhos Monique e Alfredinho; à minha sogra, Gilma; ao meu sogro, Mauriene; às tias Lucineide e Marineide; aos primos Júnior, Mário e Marquinho; à comadre Angélica; à minha avó paterna, Maria; à minha finada avó materna, Judith; à minha bisavó Bandeirinha.

Aos grandes amigos de Belém, Dani, Mauro, Sandro, Thayssa, Denise, Anamélia, Andréa, Alano, Mônica, Leonardo e Gleisson.

Aos amigos do mestrado, Alex, Fabiano, Cristiane, Márcia, Tarciane, Karine, Carlos Cidade, Robson, Renato, Vigno, Marcelo e David.

Aos novos amigos de Fortaleza, Márlio, Simone, Roberta, William, Renée, Tuan, Emanuel, Milena, Bruno, Ariadyne e Adriana.

À minha orientadora, Profa. Mariela Cortés, pelo conhecimento, confiança, incentivo, presença marcante e valiosa orientação no desenvolvimento deste trabalho e dos artigos publicados.

A todos os professores do MACC-UECE.

Aos colaboradores da QR Consultoria, Ernani, Breno e Anderson, pelas idéias, críticas construtivas e apoio.
À equipe do LABES-UFPA pela contribuição na simulação deste trabalho.

Ao finado Prof. Belchior da Unifor, pelos ensinamentos que levarei sempre comigo.

"Em tudo dai graças."

Tessalonicenses 5:18

RESUMO

O reuso de processo de software envolve diferentes aspectos do conhecimento obtidos através de modelos de processo abstratos e de experiências anteriores bem sucedidas. O benefício do reuso é alcançado com a definição de um processo efetivo e sistemático para classificar, recuperar, adaptar, avaliar e reter ativos de processo de software para reutilização em outros contextos. Neste trabalho é proposta uma abordagem formal para o reuso de processos de software através de raciocínio baseado em casos, com o objetivo de apoiar o gerenciamento de ativos de processo de software. Com isto, pretende-se facilitar o estabelecimento do aprendizado organizacional e da melhoria contínua dos processos das organizações de software. Esta abordagem foi implementada em um ambiente de desenvolvimento de software centrado em processos, de forma a permitir sua avaliação através de simulações utilizando processos aplicados sobre eventos fictícios.

Palavras-chave: reutilização, raciocínio baseado em casos, gerenciamento de ativos de processo de software, aprendizado organizacional, melhoria contínua.

ABSTRACT

Software process reuse involves different aspects of the knowledge obtained from generic process models and previous successful experiences. The benefit of reuse is reached by the definition of an effective and systematic process to classify, retrieve, adapt, evaluate and retain software process assets for reuse in other contexts. In this work is proposed a formal approach for software process reuse through the *case-based reasoning* technology in order to support the management of the software process assets. Facilitate the establishment of organizational learning and process continuous improvement of the software development organizations is the intention of this work. This approach was implemented in a process-centered software engineering environment to enable its validation through simulations using processes applied to fictitious facts.

Keywords: reuse, case-based reasoning, software process assets management, organizational learning, continuous improvement.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CSCW	Computer Supported Cooperative Work
CSG	Comparação de Similaridades Globais
CMMI	Capability Maturity Model Integration
CMMI-SW	Capability Maturity Model Integration for Software
D-CMM	Dynamic CMM
ES	Engenharia de Software
IA	Inteligência Artificial
IDEAL	Initiating, Diagnosing, Establishing, Acting and Learning
k-NN	k-Nearest Neighbor
KPA	Key Process Area
MPS.BR	Melhoria de Processo de Software Brasileiro
NR	Nível de Reuso
NUM	Numérico
PMBOK	Project Management Book of Knowledge
PML	Process Modeling Language
PRAXIS	Processo para Aplicativos Extensíveis Interativos
PSEE	Process-Centered Software Engineering Environment
PSP	Personal Software Process
QIF	Qualitativo para Itens Fixos
QIV	Qualitativo para Itens Variáveis
RBC	Raciocínio Baseado em Casos
RUP	Rational Unified Process
SCAMPI	Standard CMMI Appraisal Method for Process Improvement
SIM	Similaridade Global
sim	Similaridade Local
SPICE	Software Process Improvement and Capability Determination
TSP	Team Software Process
UML	Unified Modeling Language
XP	Extreme Programming

SUMÁRIO

1 INTRODUÇÃO

Desde o advento do computador, as economias de todo o mundo têm dependido de sistemas baseados em computador. Ao contrário do hardware, que só tem evoluído e reduzido seus custos e tamanhos, o software ainda deixa a desejar principalmente nos quesitos qualidade, custo e tempo de desenvolvimento. Pela dificuldade no acompanhamento das tecnologias de hardware e software, assim como a grande necessidade do mercado por softwares melhores e mais baratos, este problema resultou na conhecida crise do software, a qual ainda não foi superada até hoje [Sommerville, 2003].

A dificuldade do desenvolvimento de software consiste fundamentalmente no fato de produzir algo abstrato e intangível. Por um lado, não envolve limitações físicas para seu desenvolvimento, mas por outro, a falta de restrições pode tornar o software extremamente complexo e muito difícil de entender [Sommerville, 2003]. Assim, a disciplina de Engenharia de Software (ES) surgiu para proporcionar o desenvolvimento de sistemas de software de alta qualidade a um custo satisfatório através de técnicas e métodos necessários para controlar a complexidade inerente aos sistemas.

Mesmo com o aumento da nossa capacidade de produzir software, a complexidade do software aumenta ainda mais. Com o passar do tempo e de diversas experiências, atualmente sabe-se que não há uma única abordagem ideal para o desenvolvimento de software, visto que há diversos tipos de sistemas e organizações. No entanto, noções fundamentais de processos e sistemas constituem a base de todas estas técnicas, tornando-se a essência desta disciplina. Por este motivo chega-se à conclusão de que para se ter qualidade no produto é necessário ter qualidade no processo de desenvolvimento [Pressman, 2005].

Em decorrência da necessidade de produzir e fornecer software de qualidade, dentro dos prazos estabelecidos e ao menor custo possível, a importância da adoção de um processo de software, adequado às características do produto e da organização, torna-se um fator chave para o sucesso de um projeto de software [Pfleeger, 2001].

Desta forma, o uso de processos padrões tem se tornado uma prática essencial nas organizações de software e baseá-los em normas, modelos de qualidade, melhores práticas ou experiências passadas pode poupar bastante esforço inicial, além de proporcionar mais qualidade ao processo gerado.

1.1 Descrição do Problema

Visto que a qualidade do produto depende da qualidade do processo, diversos modelos e normas para a definição e melhoria de processos de software têm sido desenvolvidos. Normas como ISO/IEC 12207 [ISO, 2008] e modelos como o *CMMI* (*Capability Maturity Model Integration*) [Chrissis et al., 2006] [Paulk et al., 1994], o *SPICE* (*Software Process Improvement and Capability dEtermination*, atualmente, ISO/IEC TR 15504) [ISO, 2006] e o MPS.BR (Melhoria de Processo de Software Brasileiro) [Softex, 2007] são cada vez mais conhecidos e utilizados. Porém, mesmo diante de diversos modelos e normas, ainda assim as empresas têm sentido dificuldades em implementá-las, dificultando a visibilidade, de forma quantitativa, dos seus benefícios e provocando problemas como resistência ao uso, queixas de burocracia, etc.

É neste contexto que a pesquisa na área de reutilização de processos de software necessita de avanços significativos para apoiar o reuso e a adaptação de processos, baseados em normas e modelos de qualidade, às características e necessidades da organização, sem influenciar o seu ciclo de vida ou impor processos pré-definidos.

A principal dificuldade em definir um processo padrão da organização encontra-se na impossibilidade de aplicar um determinado processo de software genericamente a qualquer organização [Machado et al., 2000]. Por conseqüência disto, a organização precisa de se autoconhecer, considerando diversos fatores como, por exemplo, o grau de maturidade da equipe em engenharia de software, para então alcançar uma definição de processo de software mais apropriado às suas características.

Além disto, é importante que a organização construa uma base de conhecimentos com ativos que favoreçam o aprendizado coletivo e a disseminação de seus sucessos e falhas, pois qualquer tipo de experiência pode ser reutilizado com o objetivo de evitar retrabalho e melhorar a qualidade [Machado et al., 2000]. Este aprendizado da organização como um todo, através dos acertos e erros nos projetos, tem se mostrado ainda pouco eficiente. Primeiro, porque requer investimento, tempo e pessoal especializado. Segundo, porque há pouco apoio automatizado que ofereça uma forma de sistematização para facilitar esta construção do conhecimento organizacional.

Os ambientes de desenvolvimento de software centrados em processos, do inglês *Process-centered Software Engineering Environment* (*PSEE*) que trabalham de forma integrada fornecendo mecanismos para definição, instanciação e acompanhamento de processos de software. Além de prover melhor comunicação entre os envolvidos e a

consistência do que está sendo feito; realização de algumas tarefas automáticas, livrando os seus usuários da execução de tarefas repetitivas; fornecimento de informações sobre o andamento do processo, quando necessário; possibilidade de reutilização de processo de software e a coleta automática de métricas (importantes para o controle e aperfeiçoamento de processos). Porém há uma carência de mecanismos que possibilitem a sugestão de ativos de processos de acordo com as necessidades da organização, a avaliação efetiva da reutilização de processos e o registro de informações que garantam aprendizado organizacional.

1.2 Trabalhos Relacionados

O problema descrito na seção anterior vem sendo alvo de pesquisas recentes, no entanto, a sua resolução ainda tem se apresentado incompleta, deficiente e improvisada. Alguns *PSEEs* apóiam a reutilização de processos de software, mas pouco aprendizado organizacional é gerado de maneira a sustentar a melhoria contínua de seus processos, além de não fornecerem apoio ao reuso baseado em experiências anteriores.

Dentre estes *PSEEs*, encontra-se o ambiente APSEE-Reuse [Reis, 2002], que consiste em um meta-modelo para apoiar a reutilização de processos de software. Este ambiente é integrado a plataforma de automação de processos, denominada PROSOFT-Java e apresenta deficiências, como a ausência de recomendação de ativos de processos de software.

O WebAPSEE-Pro [Costa e Sales, 2007] é uma evolução do ambiente WebAPSEE, oferece infra-estrutura e funcionalidades específicas para reuso de processos de software. No entanto, não possibilita a avaliação e o registro de aprendizado de um processo reutilizado.

A Estação Taba [Machado et al., 2000] permite a instanciação de um processo pré-definido que atende aos requisitos das áreas de processo do *CMMI*, níveis 2 e 3, e do MR-MPS.BR, níveis G a C. Desta forma, consiste em um repositório de melhores práticas, porém não considera o uso de um processo padrão como base para a definição, mas sim normas/modelos de qualidade e não permite novas configurações sem perda do conhecimento organizacional.

Já o ImPProS [Oliveira, 2007], permite a modelagem, instanciação, simulação, execução e avaliação de processos de software, bem como permite a realimentação e coleta de experiências aprendidas. Contudo, é de difícil utilização e possui pouco formalismo gráfico.

Rational Method Composer [RMC, 2006] é uma plataforma de ferramentas para autoria, configuração, visualização, publicação de métodos e processos, além de prover a

reutilização de componentes. Uma das deficiências desta ferramenta é que não executa processos em tempo real.

Por fim, a ferramenta *ProKnowHow* [Borges e Falbo, 2001] oferece um repositório contendo os conhecimentos formal e informal obtidos nos projetos, organizados de forma a poderem ser reutilizados em outros projetos, no entanto não encontra-se integrado à reutilização e instanciação de processos.

Uma análise mais aprofundada sobre alguns *PSEEs* é apresentada no Capítulo 2.

1.3 Objetivos

A reutilização de processos de software tem o propósito de apoiar a definição de processos com base em experiências passadas bem-sucedidas ou em padrões de processos (baseados em normas ou modelos de qualidade) consagrados para execução em projetos de software.

O objetivo desta abordagem é alcançar o entendimento claro das atividades, controle, disciplina e produtividade de forma que o aumento na qualidade do produto final seja alcançado como conseqüência natural do aumento na qualidade do processo.

De acordo com a descrição do problema mencionada anteriormente, vale a pena se aprofundar no estudo deste problema, pois conforme [Reis, 2002] e [Falbo, 1998], os modelos de processos de software descrevem o conhecimento de uma organização e, portanto, modelos caracterizados por experiências bem sucedidas devem ser continuamente disseminados para reutilização e aprimoramento em outros projetos.

Com base nas limitações detectadas no levantamento bibliográfico de trabalhos relacionados e na dificuldade em definir processos de software, visto que não há como criar um processo possível de ser genericamente aplicado, observa-se a falta de um guia contendo as reais necessidades dos responsáveis pela definição do processo e este, por sua vez indique as melhores práticas a serem instanciadas a partir de um processo padrão.

O objetivo deste trabalho é oferecer uma abordagem para definição e reuso de processos com base em padrões de processos e experiências passadas de acordo com as características e a realidade da organização, contemplando um repositório de ativos de processos de software com terminologias unificadas a partir de classificações que serão recuperadas através do estabelecimento de critérios determinados pelo responsável pela definição de processos da organização.

Estas recomendações darão apoio à modelagem de processos, porém sem restringir a sua composição, permitindo a flexibilidade desejável para a adaptação às

características da organização. Como forma de garantir o aprendizado, esta abordagem proporciona a avaliação do processo executado ao final do projeto, para que a organização analise a satisfação do processo às suas características e opte em realimentar o repositório, com novas representações e modelos de processo para reuso em projetos futuros.

Com o intuito de facilitar a definição da abordagem para construção do conhecimento organizacional, esta proposta aplica o Raciocínio Baseado em Casos (RBC) como metodologia de resolução de problemas. Esta metodologia utiliza uma solução aplicada a um problema anterior similar na solução de novos problemas [Wangenheim e Wangenheim, 2003].

RBC e a reutilização de processos de software apresentam semelhanças em relação ao aproveitamento de melhores práticas à solução de problemas similares.

Como forma de avaliação desta proposta, é também objeto deste trabalho a implementação de um protótipo desta abordagem em um *PSEE* para a realização de um estudo de caso que resulte na avaliação desta proposta. O WebAPSEE-Pro foi o ambiente escolhido para a implementação deste trabalho, visto que possui um meta-modelo mais completo e um formalismo gráfico que permite projetar uma variedade de funcionalidades para a gerência de processos.

Desta forma, dando apoio à construção do conhecimento organizacional aliada à reutilização de processos, espera-se aumentar a maturidade dos processos à medida que vão sendo utilizados e facilitar a implementação da melhoria contínua nas organizações de software.

1.4 Organização da Dissertação

Esta dissertação está organizada conforme a seguir:

- O Capítulo 2 fornece uma fundamentação sobre processos de software, reuso de processos de software, trabalhos correlatos e melhoria contínua;
- O Capítulo 3 aborda o RBC, bem como, seus elementos básicos, seu ciclo de vida, algumas de suas aplicações, suas vantagens e desvantagens;
- O Capítulo 4 descreve a abordagem desta proposta para prover autoconhecimento organizacional e aprimoramento contínuo através da união do reuso de processos de software e do raciocínio baseado em casos;
- O Capítulo 5 fornece a especificação proposta para implementação no ambiente WebAPSEE-Pro. O protótipo é apresentado para ilustrar o ciclo de vida desta abordagem;

- O Capítulo 6 relata a simulação da abordagem proposta no ambiente WebAPSEE-Pro em um projeto com duas etapas distintas para avaliação desta proposta;
- O Capítulo 7 apresenta as conclusões e as perspectivas futuras em torno deste trabalho.

Este trabalho apresenta um estudo aprofundado em relação ao tema, abordando aspectos importantes do reuso de processos de software e do RBC para apoiar o estabelecimento do aprendizado organizacional e a melhoria contínua dos processos das organizações de software.

2 REUSO DE PROCESSO DE SOFTWARE

A abordagem de reutilização de software surgiu para aumentar a produtividade e a qualidade dos produtos de software; reduzir custos, tempo das entregas e riscos dos projetos; proporcionar padronização, interoperabilidade, previsibilidade e confiabilidade durante o ciclo de desenvolvimento de software [Melo, 2004]. Esta abordagem tem sido muito utilizada para tratar a reutilização de processos de software, porém apenas parcialmente, pois os processos de software envolvem elementos relacionados a aspectos sociais, organizacionais, tecnológicos e ambientais.

A reutilização de processos de software tem o propósito de apoiar a definição de processos com base em experiências passadas e em padrões de processos baseados em normas ou modelos de qualidade [Oliveira, 2005]. Considerando a dependência direta entre a qualidade do processo e a qualidade do produto, é de fundamental importância alcançar o entendimento profundo das atividades envolvidas no processo e promover a sua gestão para atingir maior produtividade e qualidade no produto, além de redução de custos. Esta abordagem se destaca por ser uma das principais práticas para dar apoio à melhoria contínua de processos.

Atualmente, muitos modelos de processos de software são instanciados por diferentes organizações de desenvolvimento de software. Esta oportunidade de reutilizar a experiência adquirida durante o planejamento e condução de projetos anteriores possibilita determinar melhores estratégias gerenciais em projetos futuros [Oliveira, 2005].

No presente capítulo é descrita uma visão geral da abordagen de reutilização de processos de software, incluindo uma visão geral sobre o conceito de Processo de Software, Modelo de Processo de Software e Processo Padrão da Organização. Em seguida, são apresentados os Ciclos de Vida do Processo de Software e do Reuso de Processos de Software. Posteriormente, são relatados os trabalhos relacionados a esta proposta. Por fim, é destacada a necessidade do estabelecimento da melhoria contínua através da implantação de um programa de Melhoria de Processo de Software.

2.1 Processo de Software

Um processo de software é um conjunto de atividades e resultados associados que produzem um produto de software [Sommerville, 2003]. É constituído por atividades, métodos, práticas e transformações.

De acordo com [Falbo, 1998] um processo de software consiste em um conjunto de:

- Atividades, que são as tarefas ou trabalhos a serem realizados. Uma atividade requer recursos e pode consumir ou gerar artefatos. Para sua realização, uma atividade pode adotar um procedimento. É possível, ainda, a sua decomposição em sub-atividades. Além disso, atividades podem depender da finalização de outras atividades, denominadas pré-atividades. Em função de sua natureza, atividades podem ser classificadas em: atividades de gerência, atividades de construção e atividades de avaliação da qualidade.
- Artefatos, que são produtos de software gerados ou consumidos por atividades durante a sua execução. Os artefatos podem ser classificados em: artefatos de código, componentes de software e documentos.
- Procedimentos, que são condutas bem estabelecidas e ordenadas para a realização de atividades. Quanto à sua natureza, procedimentos podem ser classificados em: métodos, técnicas e diretrizes.
- Recursos: qualquer fator necessário à execução de uma atividade, mas que não seja um insumo para a atividade. Os recursos podem ser classificados em: recursos de hardware, recursos de software e recursos humanos.

Já conforme [Reis, 1998], um processo de software é formado por um conjunto de passos de processo parcialmente ordenados, relacionados com conjuntos de artefatos, pessoas, recursos e também estruturas organizacionais e restrições.

Na Engenharia de Software, os processos são amplamente utilizados para determinação das atividades que constituem o ciclo de vida do projeto. Há 4 macro-processos fundamentais que são comuns a todos os processos:

- Especificação – o que o sistema deveria fazer e suas restrições;
- Desenvolvimento – produção do software;
- Validação – verificação se o software é o que o cliente solicitou;
- Manutenção – incorporar mudanças ao software em resposta a demandas de mudanças;

O que diferencia um processo de software de outro, é o tipo de aplicação, onde estas atividades genéricas podem ser organizadas de maneira diferente e descritas com diferentes níveis de detalhes [Sommerville, 2003]. Mesmo assim, o uso inadequado de um

processo de software pode reduzir a qualidade ou a utilidade do produto de software a ser desenvolvido, ou pode aumentar os custos de desenvolvimento.

Atividade é um passo de processo que produz mudanças de estado visíveis externamente no produto de software. Atividades incorporam e implementam procedimentos, regras e políticas, e têm como objetivo gerar ou modificar um dado conjunto de artefatos.

Uma atividade aloca recursos e é escalonada, monitorada e atribuída a profissionais, denominados agentes, que podem utilizar ferramentas para executá-la. Toda atividade possui uma descrição, a qual pode especificar os artefatos necessários, as relações de dependência com outras atividades, a data de início e fim planejados, os recursos a serem alocados e os agentes respectivamente responsáveis.

Um agente está relacionado às atividades de um processo e pode estar organizado em cargos. Pode ser uma pessoa ou uma ferramenta automatizada, quando a atividade é automática.

Um artefato é um produto gerado durante um processo. Tal produto é resultado de uma atividade e pode ser utilizado posteriormente como insumo para a mesma ou para outra atividade a fim de gerar novos produtos.

Para garantir o mínimo de falhas de software, é importante ter um processo de software bem definido e reproduzível [Royce, 1998] [Pollice et al., 2004], que independa de habilidades individuais para funcionar corretamente com pessoas diferentes e inclua atividades de verificação e validação significativas.

2.2 Modelo de Processo de Software e Processo Padrão da Organização

Um modelo de processo de software é uma abstração de um processo de software. Sua representação é simplificada, podendo utilizar, por exemplo, modelos de *workflow* ou *dataflow* [Georgakopoulos et al., 1995]. Este modelo inclui atividades que fazem parte do processo de software, produtos de software e agentes envolvidos na engenharia do software [Sommerville, 2003]. Segundo [Franch e Ribó, 2002], há três tipos de modelos de processo de software:

- Modelos Abstratos (*Templates*), que fornecem um modelo de solução para problemas semelhantes, os quais não consideram informações relacionadas a uma organização específica. Um *template* pode ser customizado para atender os requisitos de um determinado processo, ou ainda combinado com outros templates ou modelos instanciados.

- Modelos Instanciados (*Enactable Models*), que são modelos de processo prontos para serem executados. Tais modelos podem ser adaptados (ou instanciados) de um modelo abstrato, ou da combinação de outros modelos de processo, ou terem sido criados de maneira improvisada. O processo de instanciação envolve principalmente a adaptação do modelo abstrato às características e cultura da organização.
- Modelos Executados (*Enacting Models*): que são modelos de processos instanciados contendo o histórico da execução de um processo, incluindo eventos e desvios. Tais modelos podem possuir descrições incompletas ou abstratas que serão completadas em tempo de execução.

Por sua vez, um projeto de software é a instância de um modelo abstrato, tem início e fim bem definidos, com objetivos e restrições específicos, envolvendo uma estrutura organizacional, prazos, orçamentos, recursos e um processo de desenvolvimento. A gerência de projetos tem como responsabilidades o planejamento, controle e monitoração de um processo em execução, enquanto que a gerência de processos preocupa-se em construir, analisar e verificar modelos de processo.

Um processo padrão é um modelo abstrato que possui uma estrutura básica que guiará a definição ou adaptação de qualquer processo de software na organização, seguindo suas características e necessidades [Machado et al., 2000]. Desta forma, o processo padrão deve garantir um conjunto de elementos fundamentais a serem incorporados em qualquer processo definido na organização. Este processo padrão será seguido por todas as equipes envolvidas em um projeto de software, independente das características do software a ser desenvolvido.

Na literatura atual, observa-se uma tendência à utilização de processos padrões na definição ou adaptação de processos de software. A norma ISO/IEC 12207 [ISO, 2008], a norma ISO/IEC 15504 [Salviano, 2006], CMMI [Crissis et al., 2003] [Paulk et al., 1994] e o MPS.BR [Softex, 2007], definem um processo padrão como base para a obtenção de um processo especializado, de acordo com as características de um projeto de software específico. No Processo Unificado, [Jacobson et al., 1999], definem um template para processo que pode ser especializado em instâncias de processos para atender às necessidades das organizações e de projetos específicos. Portanto, ser adaptável e configurável torna-se um importante requisito a ser atingido na definição de um processo padrão. [Humphrey, 1989] define um conjunto de razões para a definição de um processo padrão:

- Redução de problemas relacionados a treinamento, revisões e apoio a ferramentas;
- As experiências adquiridas nos projetos são incorporadas ao processo padrão e contribuem para melhorias em todos os processos de software definidos;
- Possibilidade de medições de processo e de avaliação da qualidade;
- Economia de tempo e esforço em definir novos processos de software adequados a projetos de desenvolvimento.

De acordo com [Reis, 2002] e [Falbo, 1998], os modelos de processos de software descrevem o conhecimento de uma organização e, portanto, modelos caracterizados por experiências bem sucedidas devem ser continuamente disseminados para reutilização em outros projetos. Esta boa prática é promovida por padrões, tais como a norma ISO/IEC 15504 [ISO, 2006] que recomenda o estabelecimento de processos para reuso (REU.1 Gerência de ativos reusáveis, REU.2 Gerência de programa de reuso e REU.3 Engenharia de domínio), a fim de explorar de forma sistemática as oportunidades de reuso na organização. Outro exemplo é o PMBOK [PMI, 2008], que recomenda o uso das informações históricas contidas nos ativos de processos organizacionais como entrada para vários processos da organização. Estes ativos representam, além de políticas e diretrizes, o aprendizado e o conhecimento obtidos em projetos anteriores, e que podem ser utilizados com o objetivo influenciar o sucesso do projeto: a análise do comportamento da organização em projetos anteriores pode melhorar a implementação dos processos de gerenciamento em projetos futuros. [Humphrey, 1989] descreve os cinco princípios da qualidade e produtividade:

- Um processo bem definido e estruturado pode melhorar a eficiência no trabalho;
- O processo pessoal deve ser ajustado ao conhecimento e preferências de cada indivíduo;
- Para que um desenvolvedor se sinta à vontade com um processo, este deve participar na sua definição;
- À medida que o conhecimento e habilidade de um profissional evoluem, o mesmo deve acontecer com o processo utilizado;
- A melhoria contínua fica facilitada com um processo de realimentação permanente.

A definição de um processo de software não é simples, exige experiência e envolve o conhecimento de muitos aspectos da Engenharia de Software. De acordo com

[Machado et al., 2000], a dificuldade em definir processos de software encontra-se na ausência de um processo de software possível de ser genericamente aplicado.

Para aumentar as chances de acerto, deve-se considerar a sua adequação às tecnologias envolvidas, ao tipo de software em questão, ao domínio da aplicação, ao grau de maturidade (ou capacitação) da equipe em engenharia de software, às características próprias da organização e à localização geográfica da equipe.

2.3 Ciclo de Vida do Processo de Software

O ciclo de vida do processo de software (Figura 1) é análogo ao ciclo de vida de produtos de software. As atividades do ciclo de vida do processo de software são chamadas de meta-atividades, e o processo de desenvolvimento e evolução de processo de software é denominado de meta-processo [Humphrey, 1989]. As fases do ciclo de vida do processo de software são representadas em alto nível de abstração, de forma que cada fase pode ser decomposta em subfases de pouca granularidade.

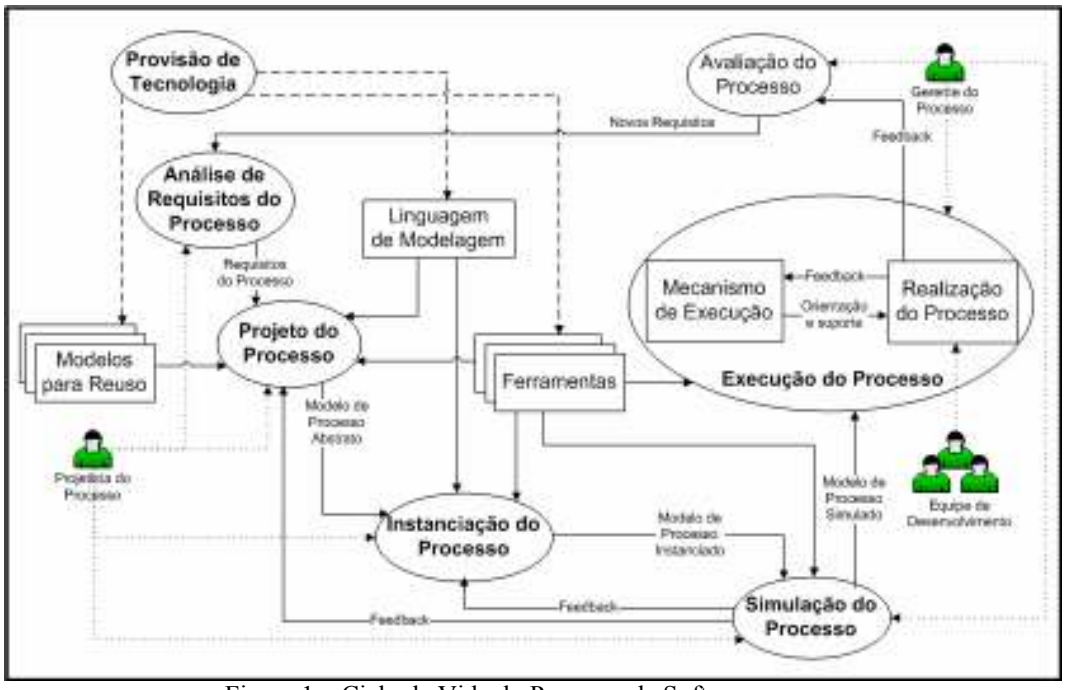

Figura 1 – Ciclo de Vida do Processo de Software.

A seguir são descritas as meta-atividades básicas que compõem o ciclo de vida do processo de software [Oliveira, 2005]:

- Provisão de Tecnologia: a tecnologia de apoio a produção de software e de modelos de processo de software deve ser provida, incluindo as linguagens de modelagem de processo, modelos de processo prontos para reutilização e ferramentas para aquisição, modelagem, análise, projeto, simulação, evolução,

execução e monitoração de modelos de processo. Um exemplo de apoio automatizado para esta tecnologia são os ambientes de desenvolvimento de software centrado em processos;

- Análise de Requisitos do Processo: nesta meta-atividade são identificados requisitos para um novo processo ou novos requisitos para um processo existente. Os requisitos resultantes especificam os recursos e propriedades que o processo deve oferecer. Os métodos utilizados nesta fase podem ser os métodos convencionais de análise de sistemas de informação (por exemplo, modelos orientados a objetos), ou até mesmo técnicas de aquisição de conhecimento. Um exemplo de requisito de processo é a aderência deste modelo de processo a uma determinada norma de qualidade;
- Projeto do Processo: provê a arquitetura geral e detalhada do processo. Nesta etapa as linguagens de modelagem do processo são utilizadas, havendo necessidade de que satisfaçam os requisitos. Pode-se criar, por exemplo, um plano de processo para o projeto ou definí-lo através de regras de gramáticas de grafo [Lima Reis, 2003];
- Implementação ou Instanciação do Processo: implementa a especificação do processo produzida pela meta-atividade anterior. Nesta fase é gerado um modelo de processo instanciado, contendo informações detalhadas sobre prazos, agentes e recursos utilizados por cada atividade definida no processo, aproveitando os recursos disponibilizados pela meta-atividade *Provisão de Tecnologia*;
- Simulação do Processo: a simulação ocupa um papel chave na verificação e validação dos processos definidos. A simulação é uma tarefa que geralmente é acompanhada tanto pelo projetista de processo quanto pelo gerente do desenvolvimento com o objetivo de antever o comportamento do projeto (execução do processo). Um exemplo desta meta-atividade é realizar a simulação do processo a partir de um plano de execução do processo;
- Execução do Modelo de Processo: nesta meta-atividade o modelo de processo instanciado é executado em um projeto real através da invocação de ferramentas, como por exemplo, ambientes de desenvolvimento de software centrado em processos, para guiar e assistir a realização do processo no projeto.

Informações sobre o andamento do processo (*feedback*) são coletadas e analisadas durante a execução;

- Avaliação do Processo: provê informação quantitativa e qualitativa descrevendo o desempenho de todo o processo em execução. Esta fase ocorre simultaneamente à execução do modelo de processo e as informações adquiridas são utilizadas na atividade de análise de requisitos. Nesta fase são usados métodos de avaliação do processo como, por exemplo, SCAMPI para avaliação CMMI [Paulk et al., 1994].

No meta-processo é necessária a participação dos agentes humanos que operam na fase de execução do processo. No caso da modelagem do processo é necessário um projetista de processo, que é o responsável por descrever o processo a ser executado e um gerente do processo que deverá acompanhar a execução e a avaliação do processo, analisando seu desempenho.

2.4 Ciclo de Vida do Reuso de Processos de Software

Um processo de software e seu respectivo modelo têm uma natureza evolucionária, devido à necessidade de melhoria e correção contínua, assim como a instabilidade do ambiente operacional. Haja vista que o modelo de processo de software é primeiro estabelecido para representar o mundo real inicial e durante sua execução, este é exposto a mudanças causadas por eventos planejados e não planejados externos e/ou internos da organização [Oliveira, 2005].

Neste contexto, Jørgensen propôs um ciclo de vida de processos de software no cenário de reutilização [Jørgensen, 2001]. Nesta proposta (Figura 2), as setas são etapas e os vértices, em itálico, são estados. A etapa "Modelagem de Processos Visando a Reutilização" tem como objetivo definir, através de uma Linguagem de Modelagem de Processos (*Process Modeling Language – PML*), modelos de processos e componentes genéricos e abstratos que possam ser reutilizados em diferentes contextos. Já a etapa "Recuperação e Adaptação de Processos" define critérios, estratégias e mecanismos para auxiliar um projetista na seleção, adaptação e instanciação de processos genéricos que forneçam soluções para o novo problema. Na etapa "Execução de Processos" são descritas todas as ocorrências de um modelo de processo desde o início de sua execução, incluindo as modificações referentes à sua evolução gerando processos específicos. E finalmente, a etapa "Generalização e Avaliação de Processos Encerrados" realiza a extração de informações a partir da experiência

adquirida de processos bem sucedidos para alimentar o repositório de modelos de processos reutilizáveis.

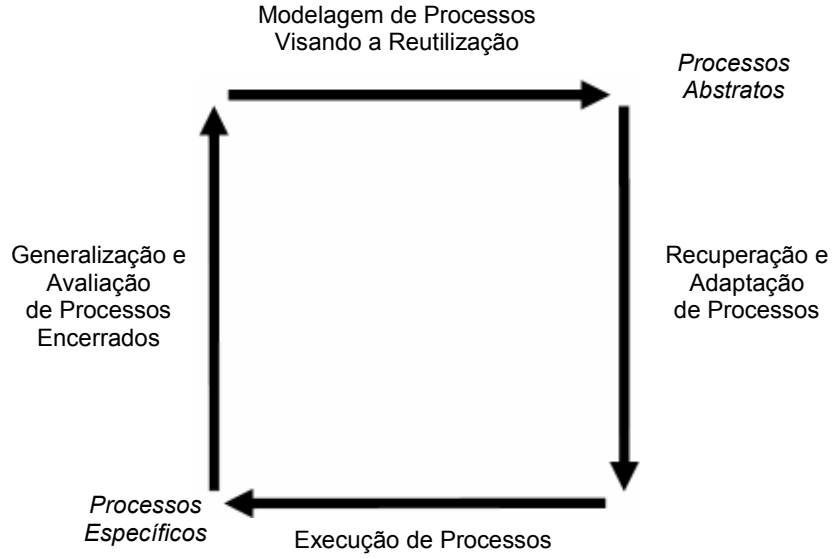

Figura 2 – Ciclo de Vida do Reuso de Processos de Software.

Este ciclo de vida proposto por Jørgensen estabelece etapas muito similares às descritas no ciclo de vida de processos de software, abordado na Seção 2.3. Sua principal diferença encontra-se no foco de generalizar e abstrair processos de software para a definição de modelos de processos de software (que são os processos abstratos, conforme a Figura 2) melhores e mais adaptados aos contextos da organização de software.

De acordo com [Reis, 2002], a reutilização de processos de software define uma ampla área de estudo e prática relacionada aos diferentes aspectos envolvidos com a reutilização do conhecimento adquirido na condução de projetos passados. Historicamente, essa área vem sendo norteada pelos avanços nos estudos da reutilização de produtos de software e da tecnologia de processo de software, em que se destacam as áreas de Trabalho cooperativo apoiado por computador (*Computer Supported Cooperative Work* – CSCW) e *Workflow* [Poltrock e Grudin, 1999].

2.5 Trabalhos Relacionados

Há diversas pesquisas com foco na automação do reuso de processos. Nesta seção serão relatadas algumas das principais ferramentas disponíveis para esta finalidade, através da descrição de seus objetivos, principais características e limitações. Ao final desta seção, será apresentado um comparativo entre elas para fornecer uma perspectiva do que cada uma atende em relação aos principais requisitos [Reis, 2002] para a automação do reuso de processos.

2.5.1 APSEE-Reuse

Este ambiente foi originado a partir de uma experiência no desenvolvimento de um *PSEE* (*Process-centered Software Engineering Environment*) chamado APSEE [Lima Reis, 2003] [Lima Reis, 1998] e evoluído para atender a reutilização de processos de software.

2.5.1.1 Objetivo

De acordo com [LABES-UFPA, 2007] [Lima Reis, 2003] [Reis, 2002] [Reis, 2001], este trabalho objetiva apoiar a reutilização de processos de software através de um meta-modelo que fornece uma série de construtores sintáticos que permitem que os diferentes aspectos desse contexto sejam descritos segundo múltiplas perspectivas, complementares entre si, contribuindo para diminuir a complexidade do modelo geral. Este ambiente é integrado a uma infra-estrutura de automação de processos de software denominada PROSOFT-Java [Lima Reis et al., 1998].

2.5.1.2 Principais Características

- O ambiente destaca-se por fornecer um formalismo gráfico para modelagem de processos visando a reutilização de processos utilizando Gramática de Grafos baseado em regras na chamada APSEE-PML, que é composta de um hierarquia de tipos;
- Execução de processos para obter apoio automatizado na condução do desenvolvimento de software. Como os modelos são dinâmicos, estes podem sofrer alterações estruturais durante a sua adaptação;
- Generalização e avaliação de processos encerrados ocorrem através da coleta de métricas de produto e processo durante a sua execução.

2.5.1.3 Limitações

- O autor sugere que a recuperação e adaptação de processos se baseiem na estruturação de repositórios de processos compostos das propriedades dos processos para definir critérios de buscas especificados pelo usuário, porém não detalha formas de classificar e selecionar processos reutilizáveis;

- Este estudo não fornece maneiras de realizar recomendações de ativos de processos. Há a busca por processos instanciados e não por modelos de processos;
- Não constrói uma base de conhecimento para apoio a decisão sobre processos;
- Não realiza avaliação do modelo de processo proposto em relação a modelagem de processos de software.

2.5.2 Ambiente WebAPSEE-Pro

Este ambiente foi originado a partir de uma experiência no desenvolvimento de um *PSEE* (*Process-centerd Software Engineering Environment*) chamado APSEE [Lima Reis, 2003], o qual foi evoluído para WebAPSEE (versão para software livre). O WebAPSEE-Pro (versão comercial) possui diversas evoluções do ambiente, inclusive fornece as funcionalidades para reuso de processos.

Este ambiente consiste em um meta-modelo projetado para permitir a integração de vários serviços de gerência de processos, incluindo modelagem, execução, reutilização, adaptação, simulação, visualização, descoberta de conhecimento, coleta automática de métricas, instanciação e resposta a eventos da execução [Costa e Sales, 2007].

O WebAPSEE-Pro baseou-se no modelo APSEE-Reuse, Seção 2.5.1, que auxilia o reuso de processos de software através de construtores sintáticos e mecanismos definidos na linguagem APSEE-PML [Lima Reis, 2003] para a modelagem de processos abstratos, denominados templates. Esta linguagem permite a descrição de tipos abstratos de dados, através de um paradigma algébrico baseado em objetos e Gramática de Grafos que proporciona a transformação de um grafo em função de um conjunto de regras previamente definidas.

2.5.2.1 Objetivo

Prover uma infra-estrutura de apoio à reutilização de processos através de uma hierarquia de tipos, dimensões de um modelo reutilizável e seus componentes, e as funcionalidades da reutilização através de *templates*.

2.5.2.2 Principais Características

- Permite a construção de *templates* (representação abstrata de um processo de software) através da hierarquia de tipos;

- Oferece funcionalidades específicas para prover reuso de processos de software: *Process Instantiation*, que permite reutilizar um modelo abstrato através da definição de processos como instâncias do mesmo; *Process Composition*, que consiste na reutilização de um ou mais templates, sendo que o template é um fragmento do modelo de processo abstrato que está sendo definido; e *Process Distilling*, que proporciona a generalização de um modelo de processo encerrado, gerando um *template* correspondente;
- Fornece o histórico de modificações (versões) e instanciações de *templates* como processos executáveis.

2.5.2.3 Limitações

- O ambiente não oferece a sugestão de ativos de processo com base em características (por exemplo, modelo de ciclo de vida, modelo de maturidade, complexidade do projeto) e/ou no *feedback* do usuário, ocorrendo apenas através da representação do processo conforme a hierarquia de tipos e dimensões do modelo;
- O ambiente não realiza a construção de uma base de conhecimento para apoio à decisão sobre processos instanciados por *templates*;
- Não oferece avaliação de processos instanciados em relação ao *template*;
- Ao generalizar um processo encerrado, não possibilita o registro de aprendizado sobre o processo instanciado;
- Não aborda a adaptação de um processo de forma automática.

2.5.3 Estação TABA

Este ambiente apóia a execução das atividades de um processo de software, através de um conjunto de ferramentas integradas e repositórios contendo informações adquiridas durante a execução do processo do projeto [Berger, 2003].

2.5.3.1 Objetivo

A Estação TABA foi desenvolvida para apoiar a execução das atividades a serem desempenhadas em um processo de software, através de um conjunto de ferramentas integradas e repositórios contendo informações adquiridas durante a execução do processo do projeto.

2.5.3.2 Principais Características

- Permite a instanciação de um processo pré-definido que atende aos requisitos das áreas de processo do CMMI, níveis 2 e 3, e do MR MPS.BR, níveis G a C;
- É um repositório de melhores práticas;
- Possui ferramentas integradas.

2.5.3.3 Limitações

- Não considera o uso de um processo padrão como base para a definição, mas sim a norma de qualidade ISO/IEC 12207 e os modelos de qualidade CMMI e MPS.BR;
- Possui atividades sugeridas pela ferramenta, não permite a adaptação do processo da organização na ferramenta, mas sim, a eliminação de atividades pré-definidas;
- Não permite a modelagem de processos abstratos através de um formalismo gráfico que minimize a complexidade da sua visualização;
- Não possui apoio total para a avaliação dos processos nos projetos;
- Não permite novas configurações sem perda do conhecimento organizacional;
- Ferramenta muito dependente de consultoria da própria empresa que a desenvolveu.

2.5.4 ImPProS (Ambiente de Implementação Progressiva de Processos de Software)

Ambiente que oferece uma abordagem para definição de processos de software em uma organização de forma progressiva, ou seja, é aperfeiçoada com as experiências aprendidas na sua definição, simulação, execução e avaliação [Oliveira, 2007].

2.5.4.1 Objetivo

De acordo com [Oliveira et al., 2006] [Oliveira, 2007] [Vasconcelos et al., 2005] [Vasconcelos et al., 2006], o ambiente visa contribuir para o amadurecimento da tecnologia de processos de software através do uso de modelos e padrões de qualidade, além de mecanismos para auxiliar os usuários durante a definição progressiva de seus processos.

2.5.4.2 Principais Características

- Possui um meta-modelo de processo de software com uma terminologia única entre os vários modelos de qualidade de processo de software existentes, para uso do ambiente em seus serviços providos;
- Apóia a definição de um processo de software para organização;
- Permite a modelagem e instanciação deste processo;
- Permite a simulação do processo a partir das características instanciadas para um projeto específico;
- Apóia a execução do processo de software tomando como base uma máquina de inferência;
- Possibilita a avaliação dos critérios do processo de software;
- Apóia a melhoria contínua do processo de software e o reuso através da realimentação e coleta das experiências aprendidas.

2.5.4.3 Limitações

- Ambiente ainda em desenvolvimento, possui diversas ferramentas integradas, porém muito textual, tornando-se de difícil utilização;
- A ferramenta propõe-se em sugerir ativos de processos com base em normas e modelos de qualidade, porém ainda está muito sob consulta, não oferece integração no momento da definição do processo padrão.

2.5.5 Comparativo

Com a avaliação relatada acima, é possível realizar um comparativo entre os *PSEE*s descritos e os principais requisitos relativos a automação do reuso de processos [Reis, 2002], conforme ilustrado na Tabela 1. Percebe-se que nenhum dos ambientes contempla todos os requisitos enumerados.

Analisando a Tabela 1 é possível identificar alguns pontos fracos (atendem, no geral, de maneira limitada ou não atendem) que envolvem todos estes *PSEE*s, são eles: “Separação de Detalhes em Múltiplas Perspectivas”, “Recuperação de Modelos de Processo”, “Avaliação de Processos Encerrados”, “Estruturação/ Realimentação de Grandes Bases de Modelos de Processos” e “Registro da Aplicabilidade de Modelos de Processos”.

Tabela 1 – Comparativo entre *PSEE*s

Requisitos	APSEE-REUSE	WEBAPSEE-PRO	IMPPROS	TABA
Modelagem de Processos Visando Reuso	Sim	Sim	Sim	Não
Separação de Detalhes em Múltiplas Perspectivas	Limitado	Limitado	Não	Não
Recuperação de Modelos de Processos	Limitado	Limitado	Limitado	Não
Composição de Processos	Sim	Sim	Não	Não
Adaptação de Processos	Limitado	Sim	Sim	Sim
Execução de Processos	Sim	Sim	Sim	Sim
Generalização e Especialização	Sim	Sim	Não	Não
Avaliação de Processos Encerrados	Não	Não	Não	Não
Estruturação/ Realimentação de Grandes Bases de Modelos de Processos	Não	Limitado	Limitado	Limitado
Conexão de Instâncias aos Modelos de Processos Generalizados	Sim	Sim	Não	Não
Registro da Aplicabilidade de Modelos de Processos	Não	Não	Não	Não

Desta forma, conforme será visto no Capítulo 4, este trabalho visa contribuir com uma abordagem que possibilite atender os pontos fracos detectados, facilitando o alcance de um processo padrão conforme as características do projeto/organização e seu aprimoramento contínuo.

2.6 Melhoria de Processo de Software

De acordo com [Paula Filho, 2003] os principais problemas da produção de software não são causados por deficiências das pessoas. As pessoas geralmente erram por uma das seguintes razões:

- Têm informação imprecisa, confusa ou incompleta;
- Não têm os recursos necessários;
- Têm métodos e procedimentos mal definidos;
- Não foram treinadas adequadamente;
- Não sabem seguir os procedimentos que têm.

Para tornar uma organização mais madura e capacitada, é preciso melhorar a qualidade dos seus processos. A maturidade de uma organização em Engenharia de Software mede o grau de competência, técnica e gerencial, que esta organização possui para produzir software de boa qualidade, dentro de prazos e custos razoáveis e previsíveis [Paula Filho,

2003]. Em contrapartida, sintomas de imaturidade de uma organização se baseiam nas seguintes características:

- Os projetos não são definidos com clareza. Atividades de desenvolvimento de software são disfarçadas de manutenção, ou mesmo realizadas sem nenhum marco formal. Às vezes não se sabe ao certo quem é o responsável por um projeto, ou mesmo se uma atividade faz parte de algum projeto. Os clientes e usuários não sabem exatamente a quem se dirigir. Os gerentes têm dúvidas sobre o quê cobrar de quem.
- As pessoas não recebem o treinamento necessário. Ou não existe disponibilidade de tempo para treinamento, ou as pessoas se inscrevem no treinamento que bem entendem. Os treinamentos são avaliados apenas quanto à satisfação dos treinandos, se tanto. Não se avalia o treinamento quanto ao benefício trazido para os projetos, e não se comparam benefícios com custos. As pessoas trabalham em ambientes inadequados. Não existem planos claros de recrutamento, remuneração e avaliação do desempenho. Não existe boa comunicação entre as pessoas.
- As ferramentas não ajudam realmente a resolver os problemas. As pessoas não têm acesso a ferramentas compatíveis com o estado da arte, ou têm as ferramentas que bem entendem. Os usuários das ferramentas não recebem treinamento e orientação em grau satisfatório. Ferramentas são escolhidas de forma política, sem considerar avaliações técnicas e necessidades de padronização.
- Os procedimentos e padrões, quando existem, são definidos e seguidos de forma burocrática. Muitas vezes, o processo oficial, que existe nos documentos escritos, é rígido demais. Por outro lado, o processo real praticado é, com freqüência, muito diferente do processo oficial, e muito mais relaxado que este. Os gerentes são os primeiros a não levar os processos a sério.

Processos não melhoram simplesmente por estarem de acordo com um padrão externo. O critério de verdadeiro êxito dos processos é a medida de quanto eles contribuem para que os produtos sejam entregues aos clientes e usuários com melhor qualidade, por menor custo e em prazo mais curto, ou seja, bons processos devem ajudar a produzir software melhor, mais barato e mais rápido.

Um programa de melhoria de processos requer muitos cuidados. Em organizações onde este tipo de programa nunca foi realizado existem muitas oportunidades de fracasso. Um programa fracassado de melhoria de processos pode ser pior do que nenhum programa, porque desmoraliza o próprio conceito. Para evitar o fracasso, algumas precondições importantes devem ser observadas:

- O programa deve ter forte apoio da alta direção da organização;
- Os gerentes dos projetos devem ter participação ativa no programa;
- Todos os que usarão os processos devem ser envolvidos;
- Os processos atuais devem ser conhecidos;
- Investimentos significativos devem ser feitos;
- O programa deve ter estágios intermediários bem definidos, onde são localizados os pontos de controle.

Para garantir a estabilidade dos novos processos, os programas de melhoria introduzem práticas de institucionalização [Santos et al., 2007]. Estas práticas visam criar infra-estrutura e cultura que apóiam as práticas fins dos novos processos. Aliando-as à cultura da organização, elas tornam a execução dos processos independente de pessoas específicas. Algumas práticas contribuem para a estabilidade da mudança de processos.

- Formação de grupos de melhoria dos processos.
- Formação de estruturas organizacionais adequadas e estáveis.
- Desenvolvimento de recursos de apoio a processos, como padrões, modelos, exemplos, documentação de referência e bases de dados.
- Realização de programas de treinamento, orientação, aconselhamento e comunicação.
- Estabelecimento de medições para avaliação da melhoria de processos.
- Estabelecimento de mecanismos de controle e verificação.

Com a implantação de um programa de melhoria de processos, a organização precisa quantificar sua realidade para elaborar uma avaliação mais objetiva sobre sua evolução. Desta forma, as medições possuem um papel fundamental, pois é através delas que a organização percebe o aumento ou diminuição da qualidade do software produzido. A medição de software tem se tornado crucial na busca da melhoria do processo de software, na avaliação do andamento de projetos e também tem auxiliado na tomada de decisões estratégicas da organização. [Pressman, 2005] propõe diversas métricas para estes fins, como

métricas de produtividade, de qualidade, orientadas a tamanho e à função, de confiabilidade, etc.

Em [Borges e Falbo, 2001] é apresentado um relato de experiência de uma organização de software ao adotar um programa de melhoria contínua. Neste relato, os pontos mais problemáticos relacionados aos esforços de melhoria foram os seguintes:

- Instanciação do processo para projetos específicos;
- Divulgação do conhecimento adquirido durante os projetos;
- Melhoria contínua do processo de software através de realimentação.

Para melhorar os resultados do programa de melhoria contínua, conforme abordado por [Broomé e Runeson, 1999] e [Wangenheim et al., 1999], é necessário estabelecer um procedimento de alteração do processo padrão da organização, visto que procedimentos imaturos precisavam ser alterados à medida que fossem utilizados. Além da atualização do processo da organização, a disseminação de lições aprendidas, através de apoio automatizado, durante os projetos de desenvolvimento da organização foi considerada uma etapa fundamental para registrar os detalhes que envolvem o sucesso ou o fracasso dos projetos e evitar a repetição de falhas.

A seguir será brevemente apresentado um modelo de melhoria de processo de software denominado Modelo IDEAL.

2.6.1 Modelo IDEAL

O Modelo IDEAL [Mcfeeley, 1996], ilustrada na Figura 3, fornece uma simples abordagem para o aprimoramento contínuo, visto que mostra os passos necessários para se estabelecer um programa de melhoria de processos bem sucedido. Seguir as fases, atividades e princípios deste modelo têm se provado benéfico em muitos esforços de melhoria. O modelo fornece uma abordagem disciplinada para o aprimoramento, foca no gerenciamento do programa de aprimoramento e estabelece a base para uma estratégia de aprimoramento de longo prazo.

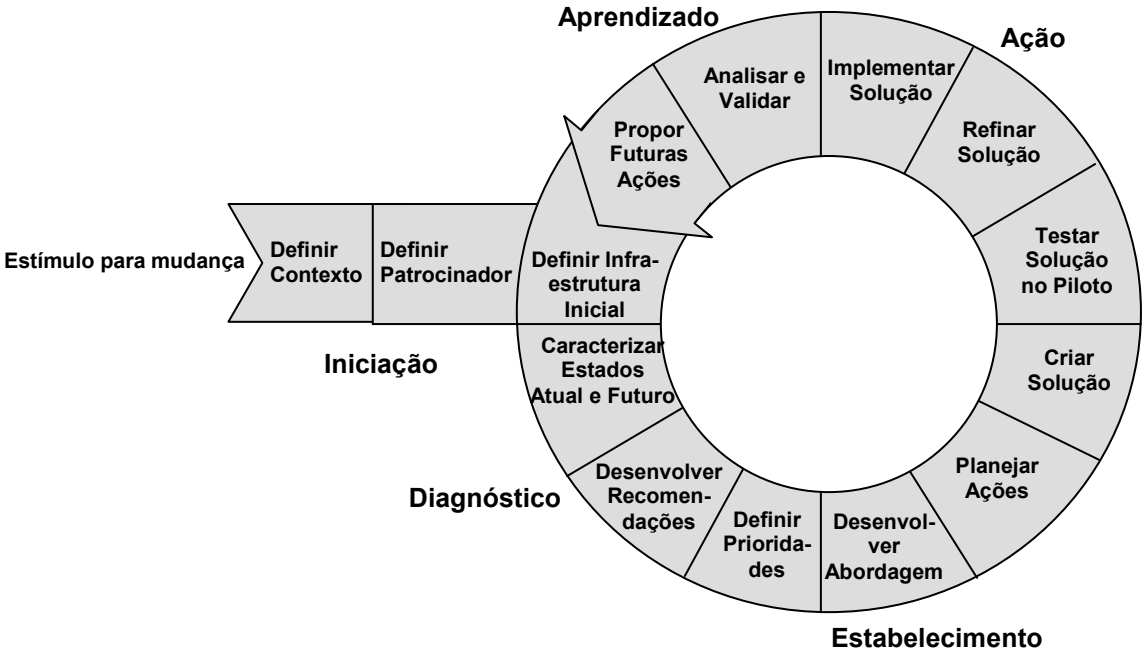

Figura 3 - Modelo IDEAL [Mcfeeley, 1996].

O modelo consiste em cinco fases, são elas:

- Iniciação: esta é a fase onde a infra-estrutura inicial da melhoria é estabelecida, os papéis e as responsabilidades são inicialmente definidos, e os recursos iniciais são atribuídos. Esta fase busca determinar o estímulo (justificativa) para a melhoria do processo de software, a definição do contexto e do patrocinador;
- Diagnóstico: nesta fase, a organização precisa realizar um entendimento sobre o trabalho de melhoria desejado. Durante esta fase duas estratégias para a melhoria do processo da organização são definidas: o estado atual e o estado futuro desejado. Estes estados organizacionais são usados para desenvolver uma abordagem para a prática de melhoria do negócio, desenvolvendo recomendações de como proceder nas fases subseqüentes;
- Estabelecimento: o objetivo desta fase é desenvolver um plano de trabalho detalhado. As prioridades de quais práticas organizacionais serão melhoradas são ajustadas para refletir as recomendações feitas durante a fase anterior. Este passo tem a finalidade de colocar prioridades para as alterações, desenvolver uma estratégia para realização do trabalho, identificar recursos disponíveis, e desenvolver um plano de implementação detalhado (onde as ações, marcos de referência e as responsabilidades são incorporados em um plano de ação);

- Ação: as atividades desta fase servem para ajudar uma organização a implementar o trabalho realizado nas três fases anteriores (Iniciação, Diagnóstico e Estabelecimento). Assim, o foco é definir processo e medições de forma que atenda as necessidades organizacionais identificadas, planejar e testar a solução criada através de um projeto piloto, modificar e acompanhar a solução para refletir o conhecimento, experiências e lições aprendidas no projeto piloto, e aplicar a solução em toda a organização;
- Aprendizado: nesta fase a experiência adquirida na execução do modelo é revisada para determinar se os esforços conseguiram atingir os objetivos pretendidos, e como a organização pode executar mudanças de maneira mais eficaz e/ou eficiente nos próximos ciclos de melhoria. Para isso, as lições são coletadas, analisadas e documentadas. As necessidades identificadas na fase Iniciação são reexaminadas para verificar se foram atendidas, e propostas de alterações para melhoria futura devem ser fornecidas.

2.7 Considerações Finais

As organizações de desenvolvimento de software são pressionadas constantemente a aprimorar os seus processos de desenvolvimento, de forma a reduzir custos no desenvolvimento de produtos e serviços, com qualidade crescente. Desta forma, o reuso de processos vem auxiliar, de forma decisiva, neste aprimoramento contínuo, fornecendo meios para minimizar tais problemas.

3 RACIOCÍNIO BASEADO EM CASOS

O modelo de cognição e comportamento humanos tem inspirado uma área da Inteligência Artificial (IA) denominada Raciocínio Baseado em Casos (RBC) [Pal e Shiu, 2004]. Esta área tem procurado desvendar como as pessoas aprendem uma nova habilidade e como os seres humanos geram hipóteses sobre novas situações, baseadas em suas experiências passadas.

O objetivo principal destas pesquisas é evoluir de uma área de pesquisa isolada e específica da IA para um campo de interesse amplo no desenvolvimento de Sistemas de Gestão do Conhecimento que ajudem às pessoas a aprender.

RBC é uma metodologia de resolução de problemas que utiliza uma solução aplicada a um problema anterior similar [Wangenheim e Wangenheim, 2003]. Esta metodologia é utilizada para modelar o raciocínio e o pensamento humanos, assim como para construir de sistemas computacionais inteligentes.

No RBC, situações anteriores similares a uma situação atual são usadas para solucionar problemas atrelados a esta nova situação. Casos passados são utilizados para sugerir formas de resolver problemas, de adaptar soluções que não são totalmente adequadas, de alertar sobre possíveis falhas e de interpretar uma situação [Kolodner, 1993].

Um caso é composto do par *problema*, que descreve o contexto em que o caso ocorreu, e *solução* que postula como o problema foi resolvido, podendo ser uma ação, um plano ou uma informação útil ao usuário [Wangenheim e Wangenheim, 2003].

Neste capítulo, esta metodologia será apresentada, através da conceituação dos elementos básicos do RBC, do seu ciclo de vida, das suas aplicações e das vantagens e desvantagens de seu uso.

3.1 Elementos básicos do Raciocínio Baseado em Casos

Um sistema de RBC possui quatro elementos básicos [Kolodner, 1993], descritos nas subseções a seguir.

3.1.1 Representação do conhecimento

A representação do conhecimento consiste na descrição das informações relevantes dos casos, utilizando formalismos para que o conhecimento seja reutilizado. Estas informações relevantes podem incluir [Wangenheim e Wangenheim, 2003]:

- O problema, que descreve o estado do mundo real quando o caso ocorreu;

- A solução, que postula uma solução derivada para aquele problema, podendo ser uma ação, um plano ou uma informação útil ao usuário, e/ou;
- O resultado, que descreve o estado do mundo real após a implementação a solução proposta, ou, ainda, quão bem a solução resolveu o problema.

Para que um caso seja útil são necessários, pelo menos, os seguintes componentes: descrição do problema e da solução. Com estes componentes, novos problemas poderão ser resolvidos procurando um caso relevante na base de casos e adaptando-o ao problema.

A descrição do problema pode reunir as seguintes informações:

- Objetivos, os quais devem ser atingidos pela solução do problema (por exemplo, sugerir um diagnóstico de falha de um componente).
- Restrições, às quais os objetivos estão sujeitos (por exemplo, evitar ensaios destrutivos).
- Atributos, do problema e o relacionamento entre suas partes (por exemplo, taxa de falhas do componente, temperatura de trabalho, característica do ambiente, etc.).

De modo geral, todas as informações são consideradas essenciais para se atingir os objetivos. Nesta ótica, o conteúdo das soluções deverá variar com o tipo de aplicação do sistema RBC, e pode incluir:

- Uma solução em si;
- Um conjunto de etapas de raciocínio necessárias para resolver um problema;
- Explicações para as decisões tomadas durante a solução do problema;
- As soluções alternativas viáveis, mas não sugeridas e suas respectivas explicações;
- As soluções inviáveis que foram excluídas do processo e suas respectivas explicações;
- Expectativas a respeito do ambiente após a implementação da solução sugerida e possíveis problemas.

Da mesma forma, o conteúdo do resultado pode incluir informações detalhadas do mundo real com respeito à implementação da solução sugerida entre outras:

- O resultado em si;
- Se o resultado atingiu ou violou as expectativas e as devidas explicações;
- O êxito ou não da solução sugerida e as devidas explicações;
- As possíveis causas das falhas e alternativas para correção.

Por outro lado, a representação do caso significa codificar o conhecimento contido nos casos mediante uma ampla gama de formalismos representacionais ou linguagens desenvolvidas no âmbito da IA, permitindo armazená-lo em uma forma simples, concisa, completa e clara. Isto é, eliminando suposições ou elementos de forma ambíguos [Russel e Norvig, 2004].

Os formalismos tradicionais incluem grafos, representações atributo-valor, orientação a objetos, predicados, etc. Já os formalismos mais modernos envolvem técnicas de computação, como lógica fuzzy [Pal e Shiu, 2004]. A seguir é apresentado um dos principais formalismos relatados na literatura: a representação por vetores de atributo-valor. Este é o formalismo utilizado neste trabalho devido ser o mais simples e o que resolve grande parte dos problemas de aplicação de RBC, conforme [Watson, 2003] [Wangenheim e Wangenheim, 2003].

3.1.1.1 Representação por vetores de atributo-valor

Neste formalismo de representação, um objeto (instância) do mundo real é descrito mediante um vetor de valores de atributos, onde um atributo é uma característica ou uma informação que visa descrever o objeto, como mostra a Tabela 2.

Tabela 2 – Exemplo de representação por vetores de atributo-valor

Objetivo do sistema: Especificação de motor elétrico de indução	
Caso 001	
Atributo	**Valor**
Faixa de altitude que o motor deverá operar	1500 (m)
Temperatura máxima de trabalho Temperatura mínima de trabalho	60 (ºC) -40 (ºC)
Ambiente de operação	Agressivo

A escolha dos pares atributo e valor deve levar em conta sua relevância em relação ao objeto (caso) representado e tem como vantagens: simplicidade na representação, simplificação das medidas de similaridade, facilidade de armazenar em bases de dados com fácil recuperação. Entretanto, apesar desta representação ser a mais utilizada nos sistemas RBC, outras representações mais complexas contendo modelos de relações entre atributos ou conjunto de atributos podem ser necessárias para representar estruturas de informação mais complexas [Wangenheim e Wangenheim, 2003].

Analisando a representação dos casos do ponto de vista destes componentes, percebe-se que os mesmos fornecem uma maneira definida de representar casos. Estes componentes nos dão formas variadas para representar diferentes tipos de conhecimento utilizados em diversas tarefas. Por exemplo, casos que contêm apenas o problema e a solução podem ser usados na derivação de soluções para novos problemas. Casos que contêm a descrição do problema e os resultados são utilizados na avaliação de novas situações. Se além da descrição e dos resultados o sistema tiver a solução, este pode ser utilizado na avaliação da solução proposta e na antecipação de potenciais problemas.

3.1.2 Medida de similaridade

A medida de similaridade define como um caso é similar a um novo problema. Esta medida considera um método heurístico[1] [Wangenheim e Wangenheim, 2003] que analisa um caso e julga sua utilidade para o novo problema. A utilidade de um caso é um conceito intuitivo sensível ao contexto, que depende da aplicação e do objetivo específicos para os quais os casos são recuperados da base de casos. Um caso é considerado útil para a solução de um problema, em relação ao esforço necessário para adaptá-lo ao problema atual.

A recuperação, através da medida de similaridade, resulta em um conjunto de casos úteis para a solução do problema atual. No RBC, o foco não é somente em casos que correspondam completamente a uma consulta dada, assim como um sistema de banco de dados tradicional, mas também em *ranking* de casos potencialmente úteis, que possam corresponder apenas parcialmente à consulta.

Mesmo de posse do conceito de similaridade, não há uma teoria universal de determinação da similaridade, tendo em vista os muitos aspectos que devem ser considerados, a incerteza associada ao processo e ao cenário de aplicação [Wangenheim e Wangenheim, 2003].

Das abordagens de recuperação existentes, há duas que são principais e amplamente conhecidas. A primeira é baseada na distância computacional entre casos, onde o caso mais similar é determinado pela avaliação da medida de similaridade. A segunda abordagem é mais relacionada a estruturas de indexação de casos, onde esta estrutura pode servir como pilar na busca de um caso similar [Pal e Shiu, 2004].

---

[1]Envolve estratégias, procedimentos ou métodos de aproximação tentativa/erro na busca da melhor maneira de atingir um determinado fim. Os métodos heurísticos aproximam-se mais da forma como o ser humano raciocina e chegam à resolução de problemas, garantindo soluções eficientes.

Há diversas medidas de distância computacional, porém a mais comum é baseada na localização dos objetos no espaço, denominada Distância Euclidiana Ponderada, onde a distância é calculada pela raiz quadrada da soma dos quadrados das diferenças aritméticas entre as coordenadas correspondentes dos dois objetos.

Como exemplo, [Mikos, 2008] apresenta um caso simples para considerar o risco de um empréstimo em um contexto bancário indexado por rendimento mensal líquido (R$) e pagamento mensal do empréstimo (R$), como mostram as figuras 4, 5 e 6.

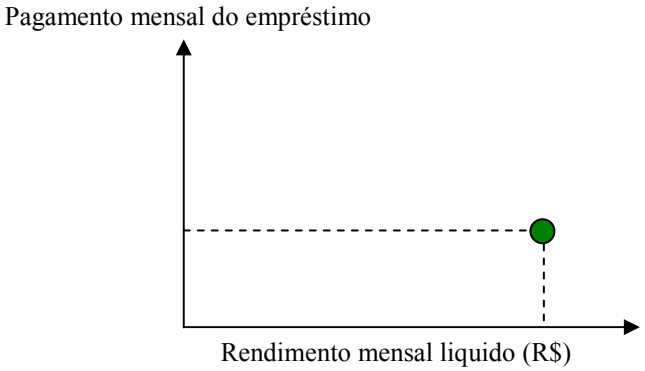

Figura 4 – Localização do objeto (caso) no espaço bi-dimensional.

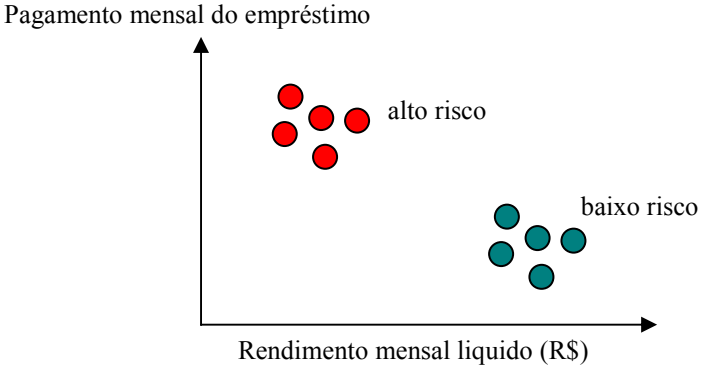

Figura 5 – Agrupamento de casos no espaço bi-dimensional.

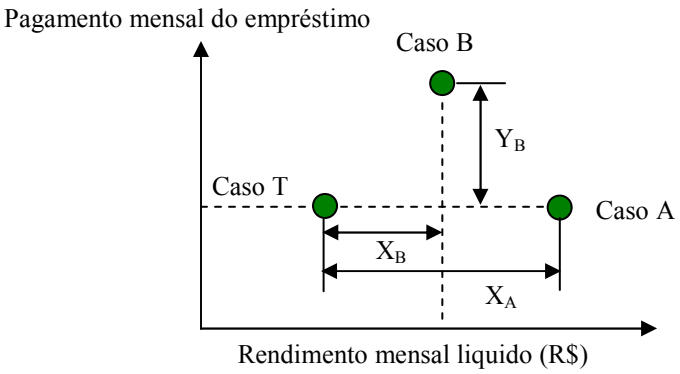

Figura 6 – Distância dos casos no espaço bi-dimensional.

Na Figura 6, o cálculo da medida de similaridade é realizado considerando a distância do Caso T (novo caso) até o Caso A e Caso B, respectivamente:

$$d_A = X_A + Y_A \tag{1}$$

$$d_B = X_B + Y_B \tag{2}$$

Nesta técnica, o caso que apresentar a menor distância (d) em relação à descrição do novo caso será considerado o *vizinho-mais-próximo* (*nearest-neighbor*, em inglês). Não obstante a simplicidade da técnica, a mesma pode ser aplicada em situações com múltiplos índices, implicando em um espaço multidimensional, e ainda suportar atributos com valores simbólicos, booleanos e textuais.

Conforme [Pal e Shiu, 2004], denote por $CB = \{e_1, e_2, ..., e_N\}$ uma biblioteca de casos com N casos. Cada caso nesta biblioteca pode ser identificado por um índice das suas características correspondentes. Uma coleção de características $F_j (j = 1,2,...,n)\}$ é usada para indexar casos. O *i*ésimo caso e_i na biblioteca pode ser representado como um vetor dimensional $(\mathrm{n}+1)$, que é $e_i = (x_{i1}, x_{i2}, ..., x_{in})$, onde x_{ij} corresponde ao valor da característica $F_j (1 \le j \le n)$.

Suponha que para cada característica $F_j (1 \le j \le n)$, um peso $w_j (w_j \in [0,1])$ seja atribuído a *j*ésima característica para indicar a sua importância. Para cada par de casos e_p e e_q na biblioteca, a métrica da distância ponderada pode ser definida como:

$$d_{pq}^{(w)} = d^{(w)}(e_p, e_q) = \left[\sum_{j=1}^{n} w^2 (x_{pj} - x_{qj})^2 \right]^{\frac{1}{2}} = \left(\sum_{j=1}^{n} w_j^2 \chi_j^2 \right)^{\frac{1}{2}} \tag{3}$$

onde $\chi_j^2 = \left(x_{pj} - x_{qj}\right)^2$. Quando todos os pesos possuem valor 1, a métrica da distância ponderada definida acima fica resumida por d_{pj}. Usando a distância ponderada, a medida de similaridade entre dois casos, sim_{pq}^{w}, pode ser definida como:

$$sim_{pq}^{w} = \frac{1}{1 + \alpha d_{pq}^{(w)}} \tag{4}$$

onde α é uma constante positiva. Quanto maior o valor de $d_{pq}^{(w)}$, menor a similaridade entre e_p e e_q. Quando todos os pesos possuem valor 1, a medida de similaridade é denominada $sim_{pq}^{(1)}, sim_{pq}^{(1)} \in [0,1]$.

A partir do conceito formal de similaridade e das medidas de similaridade apresentadas acima, os modelos para a determinação de similaridade entre descrição de um novo caso e os casos já indexados e armazenados no repositório de casos podem ser estabelecidos, entre os quais os modelos:

- Para a determinação da similaridade global (*SIM*) que envolve o cálculo da similaridade entre a descrição do novo caso e um caso da base, levando em consideração todos os atributos dos casos envolvidos;
- Para determinação da similaridade local (*sim*) que consiste no cálculo da similaridade entre atributos específicos.

No contexto da similaridade local (*sim*), diversas funções podem ser estabelecidas para o cálculo, levando em consideração os diversos tipos possíveis de valores para os atributos específicos, entre eles [Mikos, 2008]:

- Tipo numérico: funções escada, funções lineares, funções assintóticas polinomiais;
- Tipos não-numéricos: escalas ordinais para símbolos ordenados (por exemplo, temperatura baixa, média e alta), matrizes de similaridade para símbolos não-ordenados (por exemplo: comando Siemens, Fagor, Fanuc) e símbolos taxionômicos (por exemplo, valores similaridade para nós internos de árvores ou tamanho do caminho entre objetos da árvore);
- Tipo conjuntos: elementos máximos, intervalos;
- Tipo *string*: correspondência exata, correção ortográfica, contagem de palavras e taxa de maior *substring*.

No RBC, um caso, mesmo com o valor de um atributo específico diferente da descrição do novo caso, poderia ainda ser considerado como similar. Como exemplo, a Tabela 3 apresenta os valores para alguns atributos considerados, onde se pode observar que o atributo "Modelo" conta com valor de similaridade local, o qual pode ser integrado, então, ao cálculo da medida de similaridade global, tornado-a muito mais expressiva [Mikos, 2008].

Tabela 3 – Exemplos de valores de similaridade local

Atributos	Caso *p*	Caso *q*	sim_{pq}
Descrição da falha	Máquina não fecha placa auto-centrante	Máquina não liga	0
Modelo	Cosmos 10 G	Cosmos 10 U	0,6
Luz indicadora do estado do comando	Apagada	Apagada	1
SIM_{pq}			1,6

3.1.3 Adaptação

A adaptação é o processo de transformar a solução recuperada em uma solução apropriada para o problema atual. Este processo refere-se à reutilização do conhecimento de solução de problemas por meio da transferência de conhecimento do caso anterior para o caso atual.

Há diversos métodos de adaptação de casos [Wangenheim e Wangenheim, 2003], dentre eles, adaptação nula, onde nada é adaptado; transformacional, que aplica conhecimento específico para transformar a solução do caso recuperado em uma nova solução satisfazendo o problema atual; gerativa-derivacional, que requer um solucionador gerativo que seja integrado ao raciocinador baseado em casos para inferir sobre o caso recuperado. Na última abordagem, além da descrição da solução, é necessário o registro dos passos do processo derivacional que levou à determinada solução. Na adaptação composicional, novos componentes de solução, adaptados de vários casos anteriores, são combinados para produzir uma nova solução composta. Finalmente, na hierárquica, a adaptação é realizada de cima para baixo, visto que os casos são armazenados em vários níveis de abstração.

3.1.4 Aprendizado

Um sistema de RBC possui a habilidade de aprender a partir das próprias experiências. Para o aprimoramento da sua aprendizagem, o sistema necessita de um retorno sobre a corretude da interpretação elaborada para o caso atual [Pal e Shiu, 2004]. Aprendizado efetivo implica em um conjunto de métodos para extrair conhecimento relevante da experiência passada, indexar este conhecimento para uso posterior e integrar casos em uma estrutura de conhecimento existente. É através deste aprendizado que o sistema aumenta a sua capacidade, em relação ao aprimoramento nas suas interpretações, tornando-o um solucionador de problemas cada vez mais poderoso.

O Raciocínio Baseado em Casos pode se tornar mais competente ao longo do tempo na medida em que retorna respostas melhores das retornadas quando tinha menos experiência. Um dos usos do Raciocínio baseado em casos é antecipar e conseqüentemente evitar que os mesmos erros acontecidos no passado possam acontecer novamente [Wangenheim e Wangenheim, 2003].

3.2 Ciclo de Vida do RBC

Nesta seção será descrito o ciclo de vida utilizado no RBC. O ciclo é formado pelos seguintes processos (4R) [Aamodt e Plaza, 1994]:

- Recuperação de caso(s) mais similar(es);
- Reutilização da informação ou conhecimento do caso para resolver o problema;
- Revisão da solução proposta;
- Retenção de partes desta experiência com mais probabilidade de serem úteis na solução de futuros problemas.

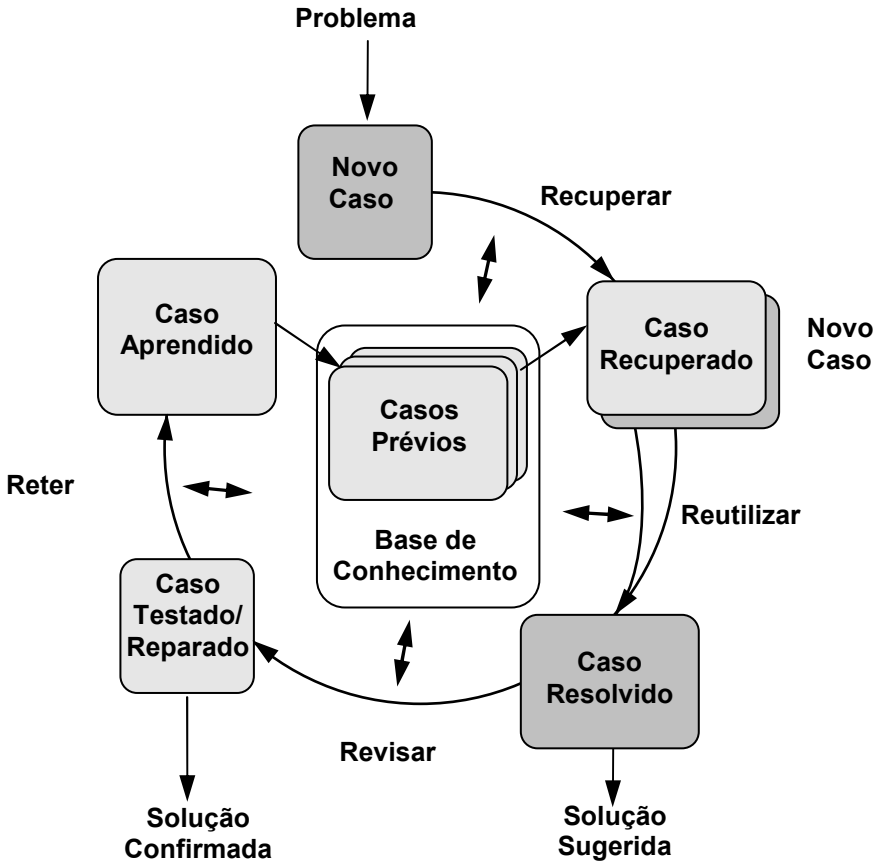

Figura 7 – Ciclo de vida do RBC [Aamodt e Plaza, 1994].

Neste modelo dinâmico, apresentado na Figura 7, uma descrição inicial de um problema define um novo caso, o qual deverá ser usado pelo processo de recuperação para pesquisar e identificar um caso similar em uma coleção de casos prévios denominada de base de casos prévios. O caso recuperado é combinado, então, com o novo caso, por meio do processo de reutilização, tornando-se assim um caso resolvido, isto é, uma solução proposta para o problema inicial. Posteriormente, o processo de revisão deverá testar a solução proposta no ambiente do mundo real, de modo a avaliar a sua pertinência e reparar a solução quando esta falhar. Por fim, o processo de retenção deve armazenar a informação útil para futuras reutilizações e a base de casos prévios deve ser atualizada com o novo caso aprendido.

Neste modelo, a base de conhecimento geral (modelos causais) indicada na figura, normalmente contribui para o desenvolvimento do ciclo RBC. Esta contribuição pode ser

representada em um *continuum*, tendo em um extremo uma contribuição insignificante ou mesmo nenhuma e, em outro extremo, uma contribuição essencial em função do método de RBC em questão.

Neste contexto, este conhecimento significa um conhecimento genérico e dependente do domínio, em contraste ao conhecimento específico incorporado nos casos.

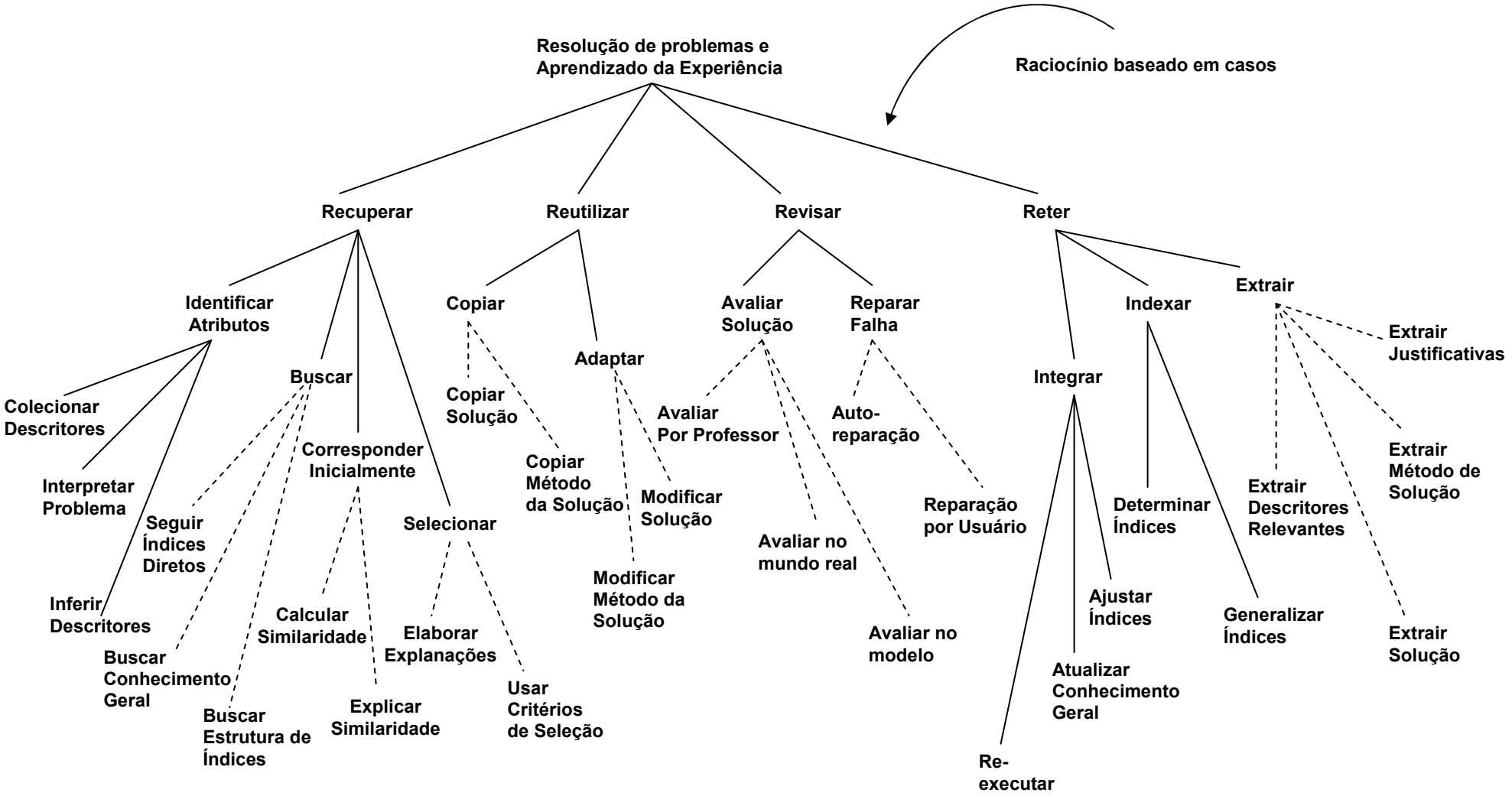

Figura 8 – Modelo de decomposição do método de tarefa de RBC [Aamodt e Plaza, 1994].

3.2.1 Recuperação de Casos

O processo de recuperar casos geralmente consiste em 2 passos, que são: recuperar casos antigos e selecionar o melhor subconjunto dos casos recuperados.

O principal desafio do processo de recuperar casos é recuperar bons casos da base de casos. Bons casos são aqueles que têm um bom potencial de fazer boas predições sobre o novo caso. Recuperação é feita utilizando as características do novo caso como índices na base de casos.

É importante ressaltar que os métodos apresentados na Figura 8 representam, na verdade, uma superclasse de métodos. E, portanto, um ou mais métodos podem ser necessários para completar a tarefa ou, ainda, algumas tarefas de nível mais baixo na hierarquia podem ser completadas, diretamente, por um método de execução de tarefa.

De acordo com [Aamodt e Plaza, 1994], a tarefa de recuperação é dividida em quatro sub-tarefas como mostra a Figura 8: identificar atributos, pesquisar índices, encontrar uma correspondência inicial de casos potencialmente candidatos, e selecionar a melhor correspondência.

Assim como em banco de dados, os algoritmos de recuperação de dados têm grande importância na eficiência da base de casos. Os algoritmos de recuperação de casos podem não trazer exatamente um caso igual ao solicitado pelo usuário e sim os mais próximos, os quais são considerados relevantes em relação ao caso solicitado pelo usuário. Abaixo são citados alguns algoritmos utilizados na recuperação de casos [Kolodner, 1993]:

- Busca Serial;
- Busca em Profundidade de Grafos para Redes de Características Compartilhadas;
- Busca em Largura de Grafos para Redes de Discriminação Prioritária;
- Busca Paralela;
- Memória Hierárquica.

A habilidade de distinguir dentre todos os casos retornados pelos algoritmos de recuperação de casos quais serão mais úteis para o usuário é um dos tópicos chave em RBC. Escolher os melhores casos ou os mais úteis é primeiramente um processo de comparação parcial. O processo começa quando os algoritmos de recuperação estão executando. No momento da procura, algoritmos irão chamar funções de comparação

para computar o grau de similaridade de acordo com certas dimensões representadas pelos índices. Baseadas nesta série de índices, funções de busca coletarão um conjunto de casos que, parcialmente, conferem com a nova situação. Depois que este conjunto é coletado é feita uma avaliação comparativa levando em conta a importância de cada dimensão na similaridade dos casos. Este processo é conhecido como *ranking* dos casos [Kolodner, 1993].

Um dos grandes desafios no RBC é recuperar os casos apropriados. Este problema é chamado de indexação de casos e possui duas partes. A primeira é como determinar rótulos aos casos no momento que eles são adicionados à base. A segunda parte é como organizar os casos. A organização dos casos facilita a recuperação eficiente e precisa dos casos na base.

Para que os índices sejam considerados bons, estes devem possuir algumas características importantes. A primeira é que os índices devem antecipar o vocabulário que o sistema deve usar. Os índices devem descrever os itens que estão indexados. Estas descrições podem ser abstratas ou especializadas.

3.2.2 Reutilização de Casos

O processo de propor uma solução inicial consiste em selecionar porções relevantes dos casos selecionados no processo de recuperação dos casos. No problema de interpretação de novos casos este processo particiona os casos recuperados de acordo com as interpretações ou soluções que eles podem predizer.

Uma vez que um caso adequado é recuperado da base de casos, a solução sugerida por este caso é objeto de uma tentativa de reutilização para a solução do problema atual. Durante este passo, reutiliza-se o conhecimento da solução por meio da transferência de conhecimento (a descrição da solução) do caso anterior, conhecido, para o caso atual, ainda não solucionado. Assim, o passo da reutilização consiste na capacidade que um sistema de RBC possui de aplicar ao problema atual, com ou sem interação com o usuário, a solução descrita para um problema similar recuperado da base de casos.

A reutilização consiste principalmente na adaptação da solução do caso anterior ao caso atual, e as técnicas tratadas na reutilização de casos tentam resolver os problemas envolvidos na adaptação de casos, que são: quais aspectos da situação devem ser adaptados, quais modificações devem ser realizadas para esta adaptação, que método aplicar para realizar a adaptação e como controlar este processo.

Adaptação tem um papel fundamental na flexibilidade dos sistemas de RBC, e a sua capacidade de resolver novos problemas depende de sua habilidade em adaptar casos recuperados a novas circunstâncias e em sua habilidade de consertar soluções que falham ao serem aplicadas. A dificuldade maior surge quando tentamos definir como realizar a adaptação. Há muitas maneiras de se adaptar um caso. A adaptação efetiva depende do conhecimento sobre possíveis modificações válidas e das maneiras de se selecionar quais serão apropriadas e, efetivas em uma determinada situação. Questões centrais para a adaptação de casos são:

- quais são os aspectos de uma situação (descrita por meio de um caso) que devem ser adaptados;
- quais modificações são razoáveis de serem realizadas para adaptar o caso;
- quais são os métodos de adaptação aplicáveis para modificar estes aspectos;
- como controlar o processo de adaptação para saber que as modificações estão sendo realizadas de modo correto.

Para tornar possível a adaptação automática de casos, as técnicas desenvolvidas enfocam dois aspectos: (a) as diferenças entre o caso passado (cuja solução é conhecida e deve ser adaptada) e o caso corrente, e (b) qual parte do caso recuperado pode se transferida para o caso novo. Com base nesta informação, a solução proposta é consertada de forma a satisfazer completamente os requisitos da situação presente. Para esta modificação orientada à solução existe uma série de estratégias básicas de adaptação, variando desde técnicas muito simples até algumas bastante complexas.

3.2.3 Revisão

Na solução de problemas baseada em casos, soluções antigas são usadas como inspiração para a solução de novos problemas. Geralmente novas situações não conferem exatamente com casos antigos e é necessário adaptar as soluções dos casos recuperados para resolver novos problemas.

A solução ou interpretação é justificada, geralmente antes de ser apresentada ao usuário. Quando todo o conhecimento sobre o domínio é conhecido, este processo se torna um processo de validação desta nova solução. Este processo consiste em justificar e criticar soluções com base nas técnicas de interpretação utilizadas no RBC,

determinando qual das soluções derivadas é a melhor alternativa. Isto é feito comparando e contrastando a solução proposta com soluções similares. Também é possível propor situações hipotéticas para testar o quanto as soluções são robustas. Outro modo de criticar é usar um simulador e comparar os resultados.

Neste contexto, se a solução gerada não é adequada, surge uma oportunidade para a correção da falha e, conseqüentemente, a possibilidade da aprendizagem sustentada no sistema de RBC.

A primeira etapa do processo de revisão [Aamodt e Plaza, 1994] envolve a sub-tarefa de avaliação, que em geral é externa ao sistema de RBC, cujo resultado decorre da aplicação da solução no ambiente do mundo real. Estes resultados, inclusive, podem não ser obtidos de forma imediata, dependendo do domínio de aplicação. Por exemplo, em aplicações de diagnóstico médico, a confirmação de uma solução pode demorar dias, meses ou mesmo anos; logo, este caso pode permanecer na base de casos, mas deve ser identificado como um caso não validado.

A segunda etapa concentra-se em reparar a falha detectada, que pode envolver a identificação de parte da solução que contém a falha, e a recuperação ou a geração de uma explicação coerente para a sua ocorrência.

Um excelente exemplo deste procedimento foi apresentado no sistema *CHEF* [Kolodner, 1993], onde um conhecimento causal é usado para gerar uma explicação de porquê certos objetivos do plano de solução não foram atingidos. Neste sentido, o sistema identifica a situação geral que irá causar a falha, usando uma técnica de aprendizagem baseada em explicações. Assim, é incluída, internamente, uma memória de falha que é usada na tarefa de reutilização para predizer possíveis deficiências de planos de solução. Desta forma, a detecção de erros é usada na fase de adaptação como uma ação preventiva, pois tais erros podem ser: previstos, tratados e evitados.

3.2.4 Retenção de Casos

A tarefa de retenção de novos casos tem por objetivo incorporar, de forma contínua e sustentada, os conhecimentos úteis revelados a partir do processo de solução de um novo caso.

O processo de aprendizagem pode ocorrer a partir do sucesso ou do fracasso de uma solução gerada, tendo início com as saídas das sub-tarefas de revisão de casos: avaliação e reparo de solução. Este processo envolve a seleção das informações da

descrição do novo caso e sua respectiva solução validada, que devem ser retidas, bem como estabelecer a forma de retenção, isto é, a forma de indexação do novo caso, visando futuras recuperações e como integrar este novo caso na estrutura do repositório de casos.

O novo caso é então armazenado apropriadamente na base de casos para o uso futuro. O caso consiste no problema, na solução e no raciocínio de apoio que o sistema necessite ao utilizá-lo. Um processo importante é como escolher os índices apropriados para o novo caso. Estes índices serão utilizados futuramente na recuperação desse novo caso. Bons índices irão ajudar o sistema a recuperar este caso quando este for parecido com uma nova situação.

3.3 Aplicações do RBC

As aplicações do RBC são constituídas de tarefas analíticas, que envolvem classificação, diagnóstico, apoio à decisão, tutoriais, previsão e avaliação [Wangenheim e Wangenheim, 2003]. Esta metodologia pode ser utilizada como meio para as mais diversas áreas, como:

- Medicina - diagnósticos médicos, classificação e recuperação de casos clínicos. Exemplos: *Protos* (classificação de desordens auditivas) [Bareiss, 1989], *ProtoISIS* (recuperação de imagens radiológicas através de indicações clínicas) [Kahn e Anderson, 1994], *Casey* (diagnóstico de disfunções cardiovasculares) [Koton, 1988];
- Atendimento - centrais de atendimento ao cliente, suporte técnico e suporte interativo. Exemplos: *Homer* (Help desk de CAD/CAM na DaimlerChrysler) [Pantleon, 1998];
- Vendas - comércio eletrônico envolvendo pré-venda, venda e pós-venda. Exemplos: *Analog Devices* (catálogo online de amplificadores operacionais) [Lenz et al., 1998];
- Direito - inferência sobre argumentações legais. Exemplos citados em [Kolodner, 1993]: *HYPO* (modelagem da argumentação na área da legislação de direitos autorais) e *JUDGE* (sentenciamento criminal em casos de assassinato, lesão corporal e assalto).

3.4 Vantagens e Desvantagens do RBC

Como todas as metodologias, o RBC precisa ser avaliado antes de ser aplicado em um determinado sistema da organização. Os métodos de RBC implementados como comportamento dos agentes de recursos mostraram-se capazes de fornecer respostas corretas em relação aos casos de testes.

3.4.1 Vantagens

- O RBC permite ao usuário propor soluções para problemas complexos mais rapidamente, minimizando o tempo necessário para derivar respostas;
- O RBC permite ao usuário propor soluções em um domínio que não é totalmente entendido pelo usuário;
- O RBC oferece ao usuário uma maneira de avaliar as soluções quando nenhum algoritmo é disponível para a avaliação;
- Casos são úteis na interpretação de conceitos abertos e em conceitos bem definidos;
- Experiências anteriores são particularmente úteis para alertar problemas potenciais que ocorreram no passado, auxiliar o usuário a tomar decisões, evitar que erros anteriores ocorram;
- Casos ajudam ao usuário a raciocinar apenas nas partes importantes dos problemas identificando quais características dos problemas são importantes;
- Transferência de experiência de especialistas para inexperientes;
- Permite a construção de memória corporativa pelo compartilhamento de experiências individuais, apoiando soluções poderosas de suporte a decisão.

3.4.2 Desvantagens

- O sistema de RBC pode estar usando casos antigos cegamente, sem validá-los na nova situação;
- Um usuário do RBC pode aceitar casos viciados na resolução de um novo problema;

- Geralmente pessoas inexperientes não relembram casos mais apropriados quando estão raciocinando.

3.5 Considerações Finais

A metodologia de RBC tem se disseminado por todo o mundo. O número de trabalhos de pesquisa e de aplicações comerciais na área está aumentando rapidamente. O número de sistemas de RBC comerciais, principalmente, cresceu muito nos últimos anos. Vários grupos de pesquisa e empresas que trabalham com RBC foram criados, e atualmente realizam-se conferências anuais e também workshops sobre o assunto.

O RBC tem amadurecido como metodologia de resolução de problemas e tem se tornado uma disciplina de engenharia, com a utilização crescente de técnicas de engenharia de conhecimento e de engenharia de software para o desenvolvimento de aplicações comerciais reais, em lugar de enfoques de pesquisa imaturos.

O grau de maturidade tecnológica alcançado pelo RBC tem sido percebido pela comunidade acadêmica através do grande número de artigos publicados nesta área e também pelo mercado através do número crescente de aplicações de RBC comerciais que obtiveram sucesso, garantindo ainda mais interesse pelo RBC, no futuro.

4 ABORDAGEM PARA GERENCIAMENTO DINÂMICO DE ATIVOS DE PROCESSOS DE SOFTWARE

A abordagem apresentada neste capítulo corresponde à contribuição deste trabalho para a área de reuso de processos de software. Esta abordagem leva em consideração o estado da arte tecnológico e a dificuldade das organizações de desenvolvimento de software em definir seus processos de software. A principal finalidade deste trabalho é prover autoconhecimento organizacional e aprimoramento contínuo através do reuso de processo de software e do raciocínio baseado em casos.

Este capítulo está organizado da seguinte maneira. Primeiramente é apresentada uma visão geral da abordagem proposta e os requisitos deste trabalho. Em seguida, são descritos os principais elementos desta abordagem, que são: Representação de Contextos dos Ativos de Processo de Software, Recuperação de Processos de Software, Avaliação de Processos Encerrados e Retenção.

4.1 Visão Geral

A abordagem proposta utiliza a metodologia de Raciocínio Baseado em Casos (RBC) para a construção do aprendizado organizacional. A justificativa de escolha do RBC é que consiste em um modelo cognitivo e um método de construção de sistemas inteligentes que melhor se adequa às práticas de reutilização promovidas pela Engenharia de Software, a qual preconiza o aproveitamento de melhores práticas difundidas na definição de processos de software da organização [Paulk et al., 1994], [ISO, 2008], [Crissis et al., 2003], [ISO, 2006], [Softex, 2007], [PMI, 2008].

A Figura 9 esquematiza a abordagem para gerenciamento de ativos de processos de software [Santos et al., 2009a] [Santos e Cortés. 2009] [Márcia e Cortés, 2008], baseada nas propostas de [Jørgensen, 2001], para o ciclo de vida do reuso de processos de software, e de [Aamodt e Plaza, 1994], para o ciclo de vida do RBC (4R´s): recuperação, reutilização, revisão e retenção.

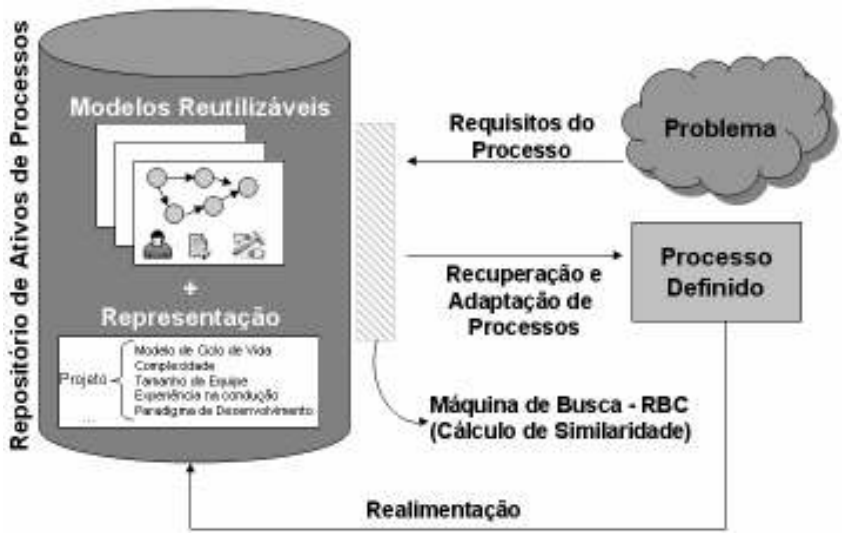

Figura 9 – Abordagem para gerenciamento de ativos de processos de software.

O componente principal é o Repositório de Ativos de Processos, onde são armazenados os modelos de processo de software reutilizáveis e suas representações atributo-valor. Estas representações envolvem uma série de propriedades relevantes que contextualizam cada caso (modelo de processo de software) através de atributos de projeto e processo. Os valores para estas propriedades podem ser numéricos, alfa-numéricos, termos pré-definidos, etc. Estas representações indicam a utilidade de um determinado modelo do repositório para um caso real. Considerando que modelos de processo de software são abstratos, a inclusão destes no repositório requer a existência de, pelo menos, uma instância em um caso específico. É importante ressaltar que o repositório de ativos de processos deve conter uma grande variedade de modelos que satisfaçam às mais diversas situações a fim de atender, mesmo que parcialmente, o problema descrito pelo usuário.

A Máquina de Busca utiliza RBC [Wangenheim e Wangenheim, 2003] para a recuperação de casos similares através da medida de similaridade da representação de cada caso. O usuário fornece os requisitos de processo através dos atributos de projeto e processo, e a máquina de busca realiza o cálculo de similaridade para os casos do repositório.

Através do modelo de decomposição de [Aamodt e Plaza, 1994], descrito no Capítulo 3, a reutilização consiste na cópia (copiar solução ou método da solução) ou adaptação (modificar solução ou método da solução) de uma solução prévia para um caso similar utilizando um método específico.

Após a reutilização e execução do caso-base no novo projeto, a instância do processo reutilizado é avaliada em relação à sua utilidade perante os requisitos de

processo do projeto e, por conseguinte, realimentar o repositório, de maneira a incrementar as representações atributo-valor para o modelo ou, em casos de adaptação, generalizar ou abstrair o processo executado, de maneira a incluir um novo modelo de processo.

Este mecanismo favorece a construção do autoconhecimento organizacional, visto que registra, à medida que ocorre o ciclo de reutilização de modelos de processo, quais modelos mais se adéquam às características da organização para que os esforços dos ciclos de melhoria de processo de software sejam minimizados.

4.1.1 Requisitos deste Trabalho

A abordagem proposta neste trabalho [Santos et al. 2009b] considera alguns dos requisitos propostos por [Reis, 2002] e que atualmente são questões em aberto, tais como:

- Representação de Contextos de Ativos de Processo de Software: esta representação, conforme mostrada na Figura 9, é pré-requisito para a etapa seguinte, que é a recuperação. Só é possível realizar a recuperação de modelos de processo de software se o repositório estiver apropriadamente classificado. Esta etapa trata da representação dos ativos de processo de software armazenados em função de uma determinada abstração. Neste trabalho, a abstração utilizada para a representação dos modelos de processo reutilizáveis consiste nos contextos em que são submetidos: Projeto e Processo;
- Recuperação de Processos de Software: conforme descrito no Capítulo 1, com a ausência de uma abordagem satisfatória para estabelecer a recuperação de processos de software de um repositório, este trabalho propõe, de maneira normalizada, uma abordagem para recuperação de processos de software através de uma máquina de busca, ilustrada na Figura 9, com RBC;
- Avaliação de um Processo Executado: após a recuperação e a adaptação do(s) processo(s), conforme Figura 9, um processo é definido para atender às necessidades de um determinado projeto da organização. Ao encerrar a execução do processo, a avaliação é fundamental para que a organização possa aferir sobre a reutilização do processo executado.

Nesta etapa, a representação de contextos para o processo executado é armazenada e serão fornecidos índices que irão auxiliar na avaliação final da organização quanto ao processo executado;

- Retenção de Processos: na realimentação a memória organizacional precisa de ser constantemente atualizada para refletir o aprendizado e aprimoramento obtido a cada ciclo de execução dos processos reutilizados. Nesta etapa, o registro do sucesso ou do fracasso do processo ajuda a melhorar o resultado das futuras recuperações em novas situações. Além disto, a organização pode escolher atualizar a versão do processo reutilizado ou criar um novo processo reutilizável.

Na Figura 10, são destacadas, em negrito e sublinhadas, as etapas do meta-processo de Jørgensen, explicado no Capítulo 2, em que as contribuições descritas acima estão associadas.

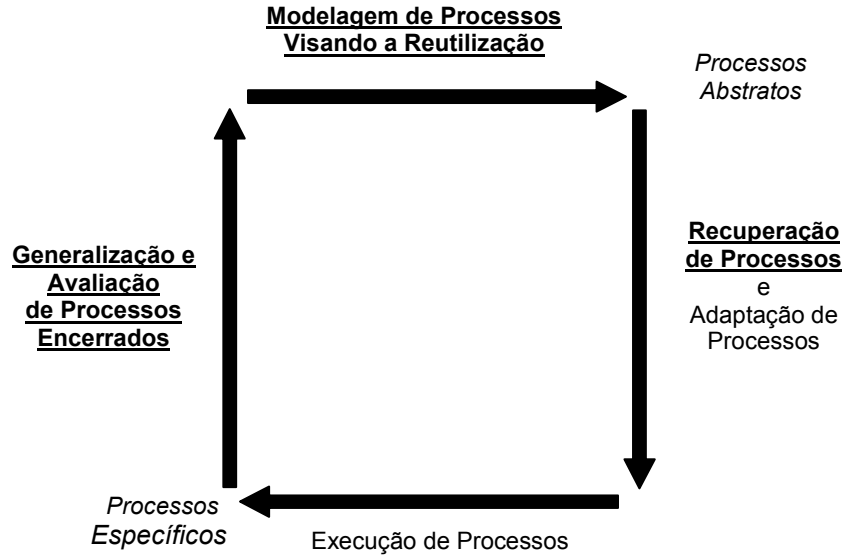

Figura 10 – Etapas do Meta-Processo de Jørgensen associadas aos requisitos deste trabalho.

A representação de contextos de ativos de processo de software está vinculada à etapa de *Modelagem de Processos Visando a Reutilização* para possibilitar que modelos de processo de software sejam caracterizados para a etapa de posterior, denominada *Recuperação de Processos*. Não faz parte do escopo deste trabalho a modelagem de processos, visto que esta tarefa já foi atendida por algumas pesquisas, conforme exposto no Capítulo 1.

Já o mecanismo de recuperação de processos baseado em RBC está associado à etapa de *Recuperação de Processos*. As etapas de *Adaptação* e *Execução de Processos* também não fazem parte do escopo deste trabalho. Por fim, os requisitos de

avaliação e retenção de processos executados estão associados à etapa de *Generalização e Avaliação de Processos Encerrados*. Nas seções a seguir é detalhado cada um destes elementos.

4.2 Representação de Contextos dos Ativos de Processo de Software

A reutilização dos casos é possível sempre que estiverem apropriadamente organizados e armazenados no repositório, de forma a facilitar a sua recuperação posterior. A representação adequada dos ativos de processo é um fator crítico para o sucesso do método, visto que o grau de similaridade para a correta recuperação dos casos é calculado com base nesta representação.

O repositório contém experiências que descrevem estratégias de solução que contribuíram com sucesso para resolver problemas anteriores. O conceito de similaridade consiste em estabelecer uma estimativa da utilidade de um caso-base armazenado no repositório, com base na similaridade observada entre a descrição do problema atual e a contida no caso-base [Wangenheim e Wangenheim, 2003].

De acordo com [Sales et al., 2006], os tipos de similaridade são referenciais aplicados aos atributos de representação, para estabelecer sua correspondência ou co-ocorrência entre casos. Na Tabela 4 são descritos os tipos de similaridade utilizados neste trabalho.

Tabela 4 - Tipos de Similaridade propostos para a Representação de Contextos

Tipos de Similaridade	Abreviatura	Valores Possíveis	Cálculo Aplicado
Numérico	**NUM**	Números inteiros ou reais positivos	Proporcionalidade
Qualitativo para Itens Fixos	**QIF**	Termos pré-definidos para o atributo	Comparação com valores possíveis
Qualitativo para Itens Variáveis	**QIV**	Termos registrados, com possibilidade de novos itens	Distância do caminho entre termos na árvore de termos

Para encontrar casos similares no repositório, é necessário definir quais atributos serão utilizados para realizar a comparação entre um caso-base e a situação atual. Diversos autores desenvolveram estudos relacionados à classificação dos ativos de processo para reuso em outros contextos [Reis, 2002] [Bertollo et al., 2003] [Cunha, 2005] [Oliveira, 2005] [Sales et al., 2006] [Machado et al., 2000] [Freitas Júnior e Reis, 2007] [Barbosa, 2005].

No presente trabalho, a classificação proposta para a representação dos ativos no repositório (Tabela 5) considera os atributos mais relevantes e presentes na maioria das referências citadas acima. Tais atributos foram organizados em contextos

relativos ao projeto e ao processo. Novos atributos podem ser incorporados pela organização, dependendo da necessidade. Esta flexibilidade facilita a adaptação da representação de ativos às necessidades da organização.

Tabela 5 - Representação dos ativos no repositório

Contexto	j	Atributo	Descrição	Tipo de Similaridade
Projeto	1	Modelo de Ciclo de Vida	Descreve o modelo de ciclo de vida utilizado no projeto (Cascata, Iterativo Incremental, Evolutivo, Espiral).	QIV
	2	Complexidade	Descreve a complexidade do projeto: Alta (possui funcionalidades críticas e avançadas), Média (necessita de análises, porém suas funcionalidades são factíveis), Baixa (possui funcionalidades simples).	QIF
	3	Tamanho do Projeto	Descreve o porte do projeto em relação à quantidade de funcionalidades: Grande, Médio ou Pequeno.	QIF
	4	Tamanho da Equipe	Identifica o número de integrantes do projeto.	NUM
	5	Prazo	Identifica a duração do projeto em meses.	NUM
	6	Conhecimento em Engenharia de Software	Descreve o nível de conhecimento em engenharia de software: Alto (teórico e prático), Médio (apenas teórico), Baixo (não possui conhecimento).	QIF
	7	Paradigma de Desenvolvimento	Restringe a busca por paradigmas de desenvolvimento específicos (Estruturado, Orientado a Objetos, etc.).	QIV
Processo	8	Modelo de Desenvolvimento	Restringe a busca por modelos de desenvolvimento de software, como RUP, XP, SCRUM, etc.	QIV
	9	Modelo de Maturidade	Restringe a busca por modelos de maturidade específicos, por exemplo, CMMI, MPS.BR, etc.	QIV
	10	Nível de Maturidade	Restringe a busca por níveis de maturidade específicos do modelo de maturidade escolhido anteriormente. Pode ser, por exemplo, 1 a 5 (CMMI e ISO/IEC 15504) ou G a A (MPS.BR).	QIV
	11	Complexidade	Descreve a complexidade do processo com base nos níveis de maturidade: Alta (níveis avançados), Média (níveis intermediários), Baixa (níveis baixos).	QIF
	12	Processo	Restringe a busca por processos específicos, que são um conjunto de atividades inter-relacionadas as quais transformam entradas em saídas, tais como Gerência de Requisitos, Planejamento de Projeto, Garantia da Qualidade, Gerência de Configuração, etc.	QVI
	13	Experiência no uso de processos	Descreve a experiência da equipe no uso de processos de software: Alta (processos utilizados em mais de 15 projetos), Média (processos utilizados na faixa de 5 a 14 projetos), Baixa (nenhum ou até 4 projetos).	QFI

4.3 Recuperação de Processos de Software

A solução mais adequada para o problema especificado pelo usuário deve ser recuperada do repositório através de uma medida de similaridade. Esta medida indica o grau de semelhança entre o problema atual (consulta) e um caso específico da base de casos (caso-base). O maior valor desta medida indica maior similaridade entre os casos.

Em RBC diversas técnicas podem ser aplicadas para a recuperação de dados. Em [Pal e Shiu, 2004] o algoritmo para calcular a similaridade é baseado na técnica do vizinho mais próximo (k-Nearest Neighbor, *k-NN)*, onde a similaridade global (*SIM*) entre dois casos, *a* (caso-base) e *b* (caso-consulta), é definida pela soma ponderada das similaridades locais (sim_j) de cada atributo (A_j), conforme apresentado na fórmula (5).

$$SIM(a,b) = \sum_{j=1}^{n} w_j \times sim_j(A_j(a), A_j(b)) \tag{5}$$

O peso (w_j) reflete a relevância do atributo (A_j) de acordo com a similaridade dos casos. Este fator é determinado pelo usuário e é medido pelos valores: Alto (100), Médio (50) e Baixo (10). Os atributos considerados mais importantes na visão do usuário para a solução do problema atual possuem pesos maiores.

Os casos-base considerados suficientemente similares são propostos ao usuário como candidatos ao reuso. Note que se o mesmo peso é especificado para todos os atributos, o caso-base que atende o maior número de atributos deve ser o sugerido.

4.3.1 Similaridade Local para Atributo do Tipo de Similaridade NUM e QIF

A similaridade local é calculada de acordo com o tipo de similaridade de cada atributo (Tabela 5). Para atributos dos tipos de similaridade NUM e QIF, a similaridade local é calculada a partir da distância (d_j) entre cada valor de atributo nos casos *a* e *b*, conforme apresentado na fórmula (6).

$$sim_j = \frac{1}{1 + d_j(a,b)} \tag{6}$$

Esta medida de similaridade local deve ser normalizada [Ricci et al., 2002] para evitar que uma dimensão se sobreponha em relação às outras, bem como padronizar faixas de valores para permitir comparações de uma métrica através do

grande intervalo de valores dos atributos. O processo de normalização utiliza o menor e o maior valor contidos no repositório para um determinado atributo de maneira a linearmente produzir valores entre 0 e 1.

A distância entre dois atributos de tipo de similaridade Numérica (NUM) é calculada com base na relação de proporcionalidade entre os valores. Assim, a similaridade local neste caso é expressa conforme a seguir:

$$d_j(a,b) = \left| \left(\frac{A_j(a) - \min(A_j)}{\max(A_j) - \min(A_j)} \right) - \left(\frac{A_j(b) - \min(A_j)}{\max(A_j) - \min(A_j)} \right) \right| \quad (7)$$

Se max(A_j) e min(A_j) são iguais, a fórmula (7) não se aplica. Portanto, neste caso, esta fórmula muda para:

$$d_j(a,b) = \left| \frac{A_j(a) - A_j(b)}{\min(A_j)} \right| \quad (8)$$

Para calcular a similaridade de atributos de tipo de similaridade Qualitativo para Itens Fixos (QIF), a distância é calculada pelo estabelecimento da proporção entre valores através dos itens fixos: Alto/Grande (9), Médio (6) e Baixo/Pequeno (3). A expressão da distância para atributos do tipo QIF consiste em:

$$d_j(a,b) = \left| \left(\frac{A_j(a) - 3}{9 - 3} \right) - \left(\frac{A_j(b) - 3}{9 - 3} \right) \right| \quad (9)$$

Desta forma, a expressão acima pode ser resumida a:

$$d_j(a,b) = \left| \left(\frac{A_j(a) - 3}{6} \right) - \left(\frac{A_j(b) - 3}{6} \right) \right| \quad (10)$$

4.3.2 Similaridade Local para Atributo do Tipo de Similaridade QIV

Para o cálculo da distância entre os atributos de tipo de similaridade Qualitativo para Itens Variáveis (QIV), é usada uma taxonomia [Wangenheim e Wangenheim, 2003], a qual representa arranjos ordenados de entidades de acordo com seus relacionamentos presumidos. Neste trabalho, as taxonomias utilizadas neste tipo de similaridade, representam hierarquicamente os relacionamentos entre os termos. Uma organização hierárquica implica uma relação de similaridade muito mais complexa, que leva em consideração a posição de um objeto na hierarquia de seu tipo.

Em uma taxonomia, quanto mais fundo os nodos estiverem localizados na hierarquia, tanto maior é a similaridade. Sejam n_a e n_b nodos diferentes em uma taxonomia, a similaridade conceitual entre estes nodos, $sim_j(n_a, n_b)$, proposta por [Wu e Palmer, 1994] consiste em:

$$sim_j(n_a, n_b) = \frac{2 \times A(n_{pai}, n_{raiz})}{A(n_a, n_{pai}) + A(n_b, n_{pai}) + 2 \times A(n_{pai}, n_{raiz})} \quad (11)$$

onde A é uma função que calcula o número de arestas no caminho entre dois nodos, portanto $A(n_a, n_{pai})$ e $A(n_b, n_{pai})$ representam o número de arestas entre os nodos correspondentes (n_a) e (n_b) e o nodo-pai comum na hierarquia (n_{pai}). Finalmente, $A(n_{pai}, n_{raiz})$ determina o número de arestas entre o nodo-pai comum e a raiz (n_{raiz}) da taxonomia. Esta medida é interessante, pois considera o nodo-pai comum e a raiz da taxonomia para normalizar a distância entre os nodos n_a e n_b. A fórmula (11) não calcula corretamente o valor da similaridade local entre nodos iguais, portanto, sejam n_a e n_b nodos iguais, o valor de $sim_j(n_a, n_b)$ é definido igual a 1, que consiste no máximo valor de similaridade local.

As Figuras 11, 12, 13, 14, 15 e 16 apresentam as taxonomias elaboradas no contexto da presente proposta para os atributos do tipo QIV. Para a construção destas taxonomias foi considerado o conhecimento descrito na literatura, conforme as referências apresentadas.

Figura 11 - Taxonomia para o atributo Modelo de Ciclo de Vida (Projeto).

Figura 12 - Taxonomia para o atributo Paradigma de Desenvolvimento (Projeto).

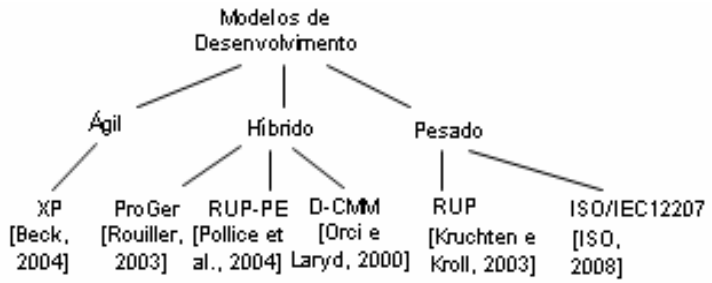

Figura 13 - Taxonomia para o atributo Modelo de Desenvolvimento (Processo).

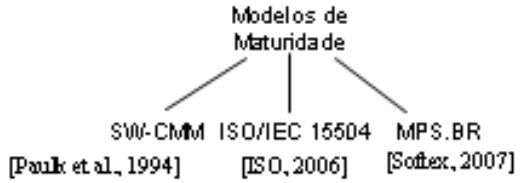

Figura 14 - Taxonomia para o atributo Modelo de Maturidade (Processo).

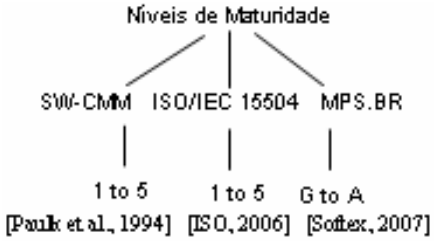

Figura 15 - Taxonomia para o atributo Nível de Maturidade (Processo).

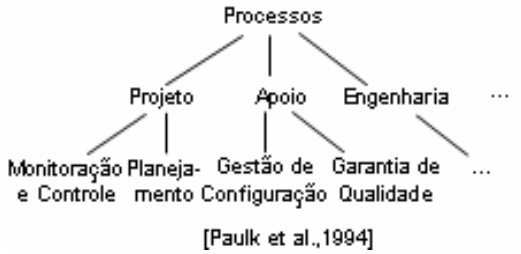

Figura 16 - Taxonomia para o atributo Processo (Processo).

Para ilustrar, como exemplo, na fórmula (12) é apresentado o cálculo da similaridade local dos modelos *ProGer* e *RUP-PE* integrantes da taxonomia *Modelos de Desenvolvimento*, conforme a Figura 13. O nodo-pai comum dos nodos *ProGer* e *RUP-PE* é o nodo *Híbrido*, portanto o número de arestas entre estes é 1 (um). Já o número de arestas deste nodo-pai comum para a raiz, *Modelos de Desenvolvimento*, também é 1 (um). Portanto, segue abaixo a similaridade destes nodos:

$$sim_j(n_a, n_b) = \frac{2 \times 1}{1 + 1 + 2 \times 1} = \frac{2}{4} = 0{,}5 \qquad (12)$$

4.4 Avaliação de Processos Encerrados

O processo de aprendizado de um sistema de RBC ocorre através da revisão dos casos sugeridos como respostas a consultas [Wangenheim e Wangenheim, 2003]. No contexto do reuso de processos de software, esta revisão possibilita avaliar o desempenho e a efetividade da nova instância de modelo de processo para a organização antes de armazenar no repositório, quando o projeto é encerrado.

Nesta proposta, o processo de avaliação consiste em três passos, que são: Comparação da Similaridade Global dos Contextos Preliminar e Real, Avaliação da Reutilização do Modelo de Processo Reutilizável e Avaliação do Nível de Sucesso. Estes passos serão detalhados a seguir.

4.4.1 Comparação das Similaridades Globais dos Contextos Preliminar e Real

A recuperação de processos similares a partir do repositório acontece a partir da caracterização do projeto de forma preliminar, de acordo com as informações relativas ao projeto e o processo. Estas informações constituem o contexto preliminar.

Uma vez que o modelo de processo selecionado é executado no contexto do projeto, haverá um processo executado e a representação de um novo contexto (*real*) que reflete a realidade do projeto, e que, possivelmente, difere do contexto preliminar inicialmente descrito. De posse da representação de ambos contextos: preliminar e real, é possível estabelecer uma métrica com base nos valores de similaridade global de ambas representações em relação ao caso-base selecionado. Esta métrica visa avaliar o grau de alteração dos contextos, refletindo as mudanças no contexto após a execução do projeto. Este índice permite a avaliação do contexto inicialmente estabelecido frente ao contexto real em que o projeto foi executado e pode ser útil na avaliação do nível de aderência do modelo de processo adotado.

A comparação entre ambas as representações, chamada Comparação de Similaridades Globais (*CSG*), é calculada com base nos valores de similaridade globais dos contextos preliminar e real. A similaridade global do contexto preliminar é calculada entre o caso-base *a* e o caso-atual *b* recuperado na hora da consulta *SIM(a,b).* Após o encerramento do projeto, são solicitadas novamente as informações de representação, no entanto, do contexto real do projeto para que este seja comparado novamente ao caso-base selecionado inicialmente e obtido o valor da similaridade global para o contexto real, este valor é armazenado como *SIM(a,b')*. A comparação

entre os valores de similaridade global *SIM(a,b)* e *SIM(a,b')* corresponde a fórmula abaixo:

$$CSG = \left(100 \times \frac{SIM(a,b')}{SIM(a,b)} \right) - 100 \quad (13)$$

Esta métrica demonstra o grau de mudança nos contextos do novo caso e serve de apoio para a avaliação subjetiva do sucesso da escolha do caso-base. Portanto, seu resultado não permite concluir que o caso-base é apropriado ou não para o novo caso, visto que esta é uma avaliação que fica a cargo do usuário. No entanto, sua utilidade encontra-se na identificação da mudança de contexto ao final do projeto e, com base nisto, indica se a escolha inicialmente feita pelo caso-base deveria se manter ou não até o encerramento do projeto. A Tabela 6 explica a avaliação desta métrica.

Tabela 6 – Avaliação da métrica *CSG*

CSG	Interpretação	Indicação
= 0	Este resultado ocorre quando os valores de *SIM(a,b)* e *SIM(a,b')* permanecem os mesmos, o que significa que não houve alteração nos contextos preliminar e real.	Neste caso, o usuário não precisaria mudar a sua escolha pelo caso-base, porém deve contribuir com o processo de aprendizado da organização através da avaliação do sucesso do caso-base em relação ao novo caso.
> 0	Significa dizer que o contexto real mudou em relação ao preliminar, porém encontra-se ainda mais similar ao caso-base selecionado do que o contexto preliminar.	Este é considerado o melhor resultado desta métrica, visto que indica que o usuário possivelmente fez uma boa escolha e reutilizou um modelo de processo mais indicado às suas necessidades reais. Da mesma forma, também resta ao usuário avaliar o sucesso do caso-base em relação ao novo caso.
< 0	Significa dizer que o contexto real é menos similar ao caso-base selecionado em relação ao contexto preliminar.	Neste cenário, possivelmente o usuário fez uma má escolha pelo caso-base e reutilizou um modelo de processo pouco indicado às suas necessidades de projeto. É bastante provável que diversas modificações foram demandadas para adequar o processo para a sua execução.

O *CSG* não demonstra a variação das similaridades locais dos atributos nos contextos preliminar e real. Desta forma, é possível que as similaridades globais tenham se mantido, mesmo com esta variação. Para facilitar esta identificação, é desejável que sejam apresentadas também as similaridades locais de cada uma das representações. Assim, o usuário poderá constatar o que foi de fato modificado e melhorar a sua inferência sobre o modelo de processo reutilizado. Portanto, o gráfico de barras da Figura 17 ilustra um exemplo de variação das similaridades locais nos contextos preliminar e real de cada atributo da representação.

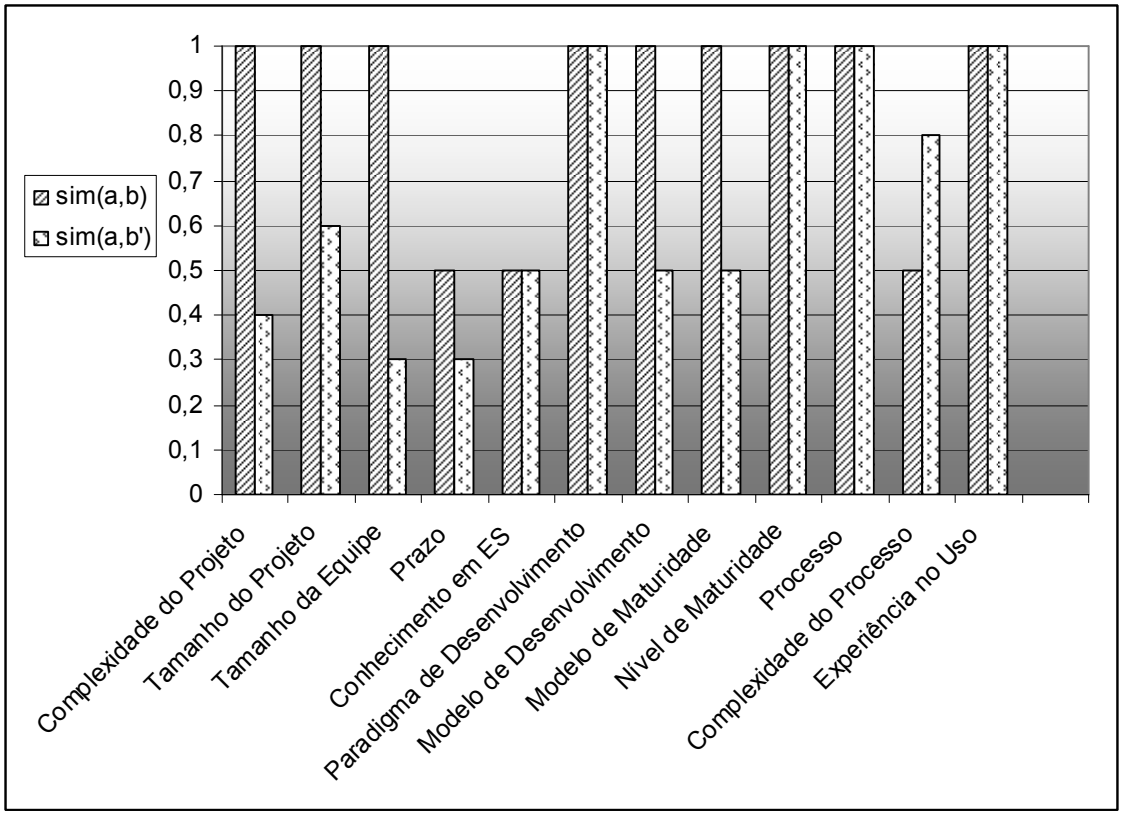

Figura 17 – Exemplo de um gráfico de variação das similaridades locais dos atributos nos contextos preliminar e real.

4.4.2 Avaliação da Reutilização do Modelo de Processo Reutilizável

Esta métrica de avaliação demonstra o percentual de reutilização do caso-base selecionado (*a*) frente ao caso executado (*b'*). Para obtê-la é necessário realizar o mapeamento de todos os componentes das atividades existentes nos casos *a* e *b'*, tais como nome, tipo, artefatos, conexões, papéis, recursos, etc. Na Figura 18 é apresentado o mapeamento do caso *b'*, bem como a identificação de quais componentes do caso *a* foram reutilizados (ilustrados pelo ícone ✔) ou não (ilustrados pelo ícone ✘). Quando uma atividade não é completamente reutilizada (ilustrada pelo ícone ⚠), é necessário considerar os componentes reutilizados para garantir um índice que reflita a realidade da reutilização dos componentes do modelo de processo adotado. A partir deste mapeamento, um cálculo de proporcionalidade é realizado para estabelecer o nível de reutilização do caso *a* no caso *b'*.

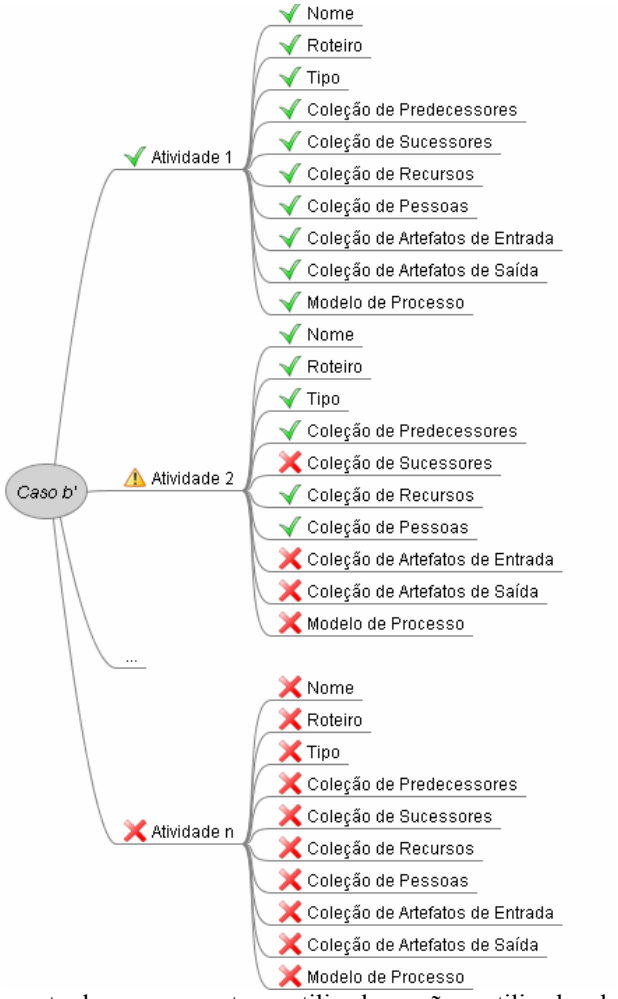

Figura 18 – Mapeamento dos componentes reutilizados e não reutilizados do caso *a* no caso *b'*.

O Nível de Reuso (*NR*) do Modelo de Processo é estabelecido pela fórmula abaixo:

$$NR = \frac{\sum_{q=1}^{n} N_{CSim_q(a,b')}}{m \times N_C} \tag{14}$$

onde *n* é o número de atividades de *b'* e *m* é o número de atividade de *a*. N_{CSim} é uma função que retorna o número de componentes similares de uma atividade específica (*q*) entre *a* e *b'*. Para considerar um componente similar é necessário estabelecer um limiar de similaridade[2]. Nesta proposta, o limiar de similaridade é considerado um valor maior ou igual a 90%, para que a métrica represente de maneira satisfatória o grau de

[2] Limite que determina se um componente é similar ou não [França, 2007]. Por exemplo, se o limiar de similaridade estabelecido é de 90% e um componente encontra-se 85% similar, então neste caso não é suficientemente similar para ser considerado como reutilizado.

reutilização do caso-base. Como o intuito desta métrica é avaliar o nível de reuso, qualquer variação no novo caso, deve ser representado de maneira que o usuário consiga identificar o quanto reutilizou do caso-base. Quanto maior o valor desta métrica, maior é a reutilização do caso-base e menor é a adaptação realizada para o caso atual.

A função N_{CSim} utiliza três tipos algoritmos de similaridade, são eles: comparação de textos, distância entre nós de uma hierarquia e comparação de coleções de elementos. Com estas comparações é possível considerar que um determinado componente da atividade foi reutilizado (ícone ✔ da Figura 18) ou não (ícone ✘ da Figura 18), conforme o limiar estabelecido. Este limiar pode ser calibrado pelo usuário.

O N_C é o número de componentes considerados nas atividades de um caso-base. Este número de componentes é um valor fixo e nesta proposta são considerados 10 componentes presentes em uma atividade de um modelo de processo (Figura 18).

Conforme [Costa e Sales, 2007], os componentes de uma atividade abstrata estão descritos na Tabela 7. A coluna "Função Utilizada" corresponde ao tipo de comparação utilizada para determinar se um componente é similar ou não de acordo com o limiar de similaridade estabelecido. Existem três tipos de função:

- *Token* – compara textos [Wangenheim e Wangenheim, 2003];
- Hierárquica - calcula a distância entre os nós de uma determinada hierarquia [Wangenheim e Wangenheim, 2003]. A cada salto na hierarquia é diminuído em duas unidades o grau de similaridade;
- *BackTracking* – esta função gera todas as possíveis combinações entre os elementos da coleção pertencente ao componente. A combinação mais semelhante à coleção da atividade a ser comparada é a considerada no cálculo. Função adaptada de [Skiena e Revilla, 2003] e [França, 2007].

A coluna "Peso" da Tabela 7 expressa a ponderação dada a cada componente, nesta proposta os componentes possuem os mesmos pesos, devido à importância de cada componente no reuso de um modelo de processo. Os sub-componentes (6.1, 7.1, 7.2, 7.2.1, 7.2.2, 7.3, 8.1, 8.2, 9.1 e 9.2) podem possuir pesos de 1 a 10, no entanto, o somatório dos sub-componentes de mesmo nível deve sempre resultar em 10. Há um caso específico para o componente 7, onde cada elemento da coleção de pessoas pode ser ou uma pessoa isolada, ou um grupo (sendo que este "ou" é

exclusivo). Sendo assim, só pode haver os subcomponentes 7.1 (peso 9) e 7.2 (peso 1), que resultam em 10 ou apenas o sub-componente 7.3 (peso 10).

Tabela 7 – Componentes presentes em uma atividade de um modelo de processo

i	Componente	Descrição	Função Utilizada	Peso
1	Nome da Atividade	Representa o nome da atividade	*Token*	1
2	Roteiro da Atividade	Descreve os passos para a execução da atividade	*Token*	1
3	Tipo da Atividade	Descreve o tipo de hierarquia da atividade	Hierárquica	1
4	Coleção de Predecessores	Reúne todas as conexões estabelecidas como predecessores da atividade	*BackTracking*	1
5	Coleção de Sucessores	Reúne todas as conexões estabelecidas como sucessores da atividade	*BackTracking*	1
6	Coleção de Recursos Necessários	Reúne os recursos de apoio da atividade	*BackTracking*	1
6.1	Tipo de Recurso	Especifica o tipo de recurso utilizado pela atividade de acordo com a hierarquia de tipos	Hierárquica	10
7	Coleção de Pessoas Envolvidas	Reúne os papéis (cargos) e os grupos envolvidos na atividade	*BackTracking*	1
7.1	Papel	Papel que desempenha a atividade	Hierárquica	9
7.2	Coleção de Habilidades	Reúne as habilidades necessárias para a execução da atividade	*BackTracking*	1
7.2.1	Habilidade	Especifica a habilidade para desempenhar a atividade	*Token*	5
7.2.2	Tipo de Habilidade	Tipo de habilidade necessária para desempenhar a atividade de acordo com a hierarquia de tipos	Hierárquica	5
7.3	Tipo de Grupo	Hierarquia do grupo de pessoas envolvidas na atividade	Hierárquica	10
8	Coleção de Artefatos de Entrada	Artefatos definidos como entrada para a atividade	*BackTracking*	1
8.1	Artefato	Especifica o artefato de entrada	*Token*	5
8.2	Tipo de Artefato	Tipo de artefato de entrada conforme a sua hierarquia de tipos	Hierárquica	5
9	Coleção de Artefatos de Saída	Artefatos definidos como saída para a atividade	*BackTracking*	1
9.1	Artefato	Especifica o artefato de saída	*Token*	5
9.2	Tipo de Artefato	Tipo de artefato de saída conforme a sua hierarquia de tipos	Hierárquica	5
10	Modelo de Processo	Modelo de processo ou subprocesso em que a atividade está inserida.	*Token*	1

Para exemplificar, considerando as três atividades da Figura 18 (*Atividade 1*, *Atividade 2* e *Atividade n*) como sendo atividades do modelo de processo e suas indicações de similaridade ilustradas pelos ícones ✔ e ✘, o Nível de Reuso para o processo desta figura é calculado conforme abaixo:

$$NR = \frac{10+6+0}{3\times 10} = \frac{16}{30} = 0{,}53 \quad (15)$$

A *Atividade 1* reutilizou todos os componentes do modelo de processo, portanto, resultou em dez componentes similares. Já a *Atividade 2*, apenas reutilizou seis componentes. E finalmente, a *Atividade n* não reutilizou nenhum dos componentes do modelo de processo, por isto seu resultado foi zero. Desta forma, o *NR* é 53%, reutilização quase pela metade, que é considerado pouco reutilizado.

O resultado desta métrica pode ser útil no apoio à decisão do novo caso para a organização. A partir da identificação do nível de reuso do modelo de processo, a organização pode concluir que realizou diversas modificações devido o modelo de processo não ser adequado às necessidades do projeto, ou também pode constatar que descartou componentes importantes para a execução do projeto e que aprendeu que mantê-los é a melhor opção.

4.4.3 Avaliação do Nível de Sucesso do Processo Executado

O nível de sucesso é uma métrica subjetiva alimentada pelo usuário sempre que um processo é avaliado, corresponde ao registro de *feedback* do usuário em relação ao caso-base. Nesta avaliação, o usuário fornece uma nota dentro do intervalo de 0 a 10 para o processo adotado, para refletir a sua utilidade e efetividade no repositório.

Esta informação é útil para posterior adoção do modelo de processo e contribui na busca da melhoria continua do processo na organização, visto que modelos de processo com os maiores níveis de sucesso serão priorizados nos resultados da máquina de busca.

4.5 Retenção

A retenção é o processo de incorporação ao repositório de conhecimento de elementos de reuso a partir do novo episódio de solução de um problema [Wangenheim e Wangenheim, 2003]. Reter continuamente é fundamental para que o repositório seja incrementado com novas soluções.

No âmbito deste trabalho, esta etapa ocorre após a avaliação do processo executado, de maneira a registrar conhecimento para uso posterior e incorporar novos casos na estrutura de representação existente.

Dependendo da avaliação realizada sobre o processo executado, conforme a Seção 4.4, o usuário poderá escolher entre generalizá-lo ou não. O ato de generalizar consiste em transformar o processo executado em um modelo de processo abstrato [Costa e Sales, 2007], isto significa que os detalhes específicos do processo são retirados para deixá-lo apropriado para a reutilização em diversos projetos. A partir desta escolha, duas opções são oferecidas para o usuário, que são:

- Gerar nova versão do caso-base reutilizado: esta opção provê a oportunidade de evoluir um modelo de processo que foi reutilizado na organização;
- Gerar um novo caso-base: consiste na criação de um novo modelo de processo reutilizável a partir de um processo executado.

No caso do usuário escolher não generalizar o processo, a representação do novo caso não será armazenada e não haverá nenhuma alteração no caso-base.

5 ESPECIFICAÇÃO DA ABORDAGEM PROPOSTA PARA O AMBIENTE WEBAPSEE-PRO

No presente capítulo é especificada a abordagem proposta, projeto e implementação no contexto do ambiente WebAPSEE-Pro, ferramenta escolhida para a implementação do protótipo desta proposta.

Este capítulo está organizado da seguinte maneira. Primeiramente são apresentados os requisitos desta abordagem de acordo com os propostos por [Reis, 2002]. A seguir é apresentada a especificação das extensões necessárias na ferramenta WebAPSEE-Pro para implementação da abordagem proposta. Depois são descritos os algoritmos dos mecanismos de recuperação, de comparação das similaridades globais e de avaliação da reutilização dos processos. Finalmente são ilustradas as telas da implementação no ambiente.

5.1 Requisitos Propostos

Os requisitos propostos por este trabalho, conforme a Seção 4.1.1, estão relacionados aos requisitos do trabalho de [Reis, 2002] (Tabela 8) e consistem em questões ainda em aberto nesta área.

Diversas pesquisas têm tratado de assuntos como modelagem visando reutilização, recuperação, adaptação e execução de processos, mas ainda há pouco a oferecer em relação à avaliação de processos e registro de *feedback* sobre o reuso com o objetivo de construir aprendizado.

Tabela 8 – Requisitos desta proposta e sua correspondência com os requisitos do trabalho de [Reis, 2002]

Requisitos propostos	**Requisitos correspondentes [Reis, 2002]**		
	Categoria	**Id**	**Nome**
Representação de Contextos de Ativos de Processo de Software	1. Modelagem de Processos visando à reutilização	R1.2	Separação de Detalhes em Múltiplas Perspectivas
Recuperação de Processos de Software	2. Recuperação de Processos e Adaptação de Processos	R2.1	Estruturação de Grandes Bases de Processos Abstratos
Avaliação de um Processo Executado	4. Generalização e Avaliação de Processos Encerrados	R4.2	Comparação de Modelos de Processo
Retenção de Processos		R4.3	Conexão de Instâncias aos Processos Abstratos Generalizados

5.2 Especificação dos Requisitos Propostos

A especificação dos requisitos deste trabalho possibilita compreender o que deve ser feito para estender a ferramenta WebAPSEE-Pro e o que se espera como resultado para que esta ferramenta ofereça, dentro da reutilização de processos de software, a gestão de ativos de processos de software para a construção do conhecimento organizacional.

A especificação da solução proposta foi definida com base nos diagramas da *UML* (*Unified Modelling Language*) [OMG, 2009] [Booch et al., 1998]: casos de uso, classes e atividades, para descrever os detalhes de projeto da implementação. A máquina de estados, existente no ambiente, é utilizada neste trabalho.

5.2.1 Diagrama de Casos de Uso

Os casos de uso foram especificados de acordo com a abordagem descrita no Capítulo 4. A Figura 19 apresenta este diagrama que é detalhado a seguir.

O principal ator do sistema é o usuário do ambiente, chamado *Manager Console User*.

Os casos de uso "Cadastrar Escopo" e "Cadastrar Atributo" são pré-requisitos para a representação de contextos dos ativos de processos de software (Seção 4.2) a qual está representada pelo caso de uso "Caracterizar *Template*".

A recuperação de modelos de processo de software (Seção 4.3) é representada pelo caso de uso "Recuperar *Templates*".

Já a avaliação de um processo executado (Seção 4.4) é representada pelo caso de uso "Avaliar Processo Encerrado" o qual inclui mais quatro casos de uso denominados "Caracterizar Processo Encerrado", "Avaliar Reuso", "Avaliar Contextos" e "Avaliar Nível de Sucesso".

A retenção do processo encerrado (Seção 4.5) é representada pelo caso de uso "Reter Processo Encerrado" o qual inclui o caso de uso "Generalizar Processo Encerrado" e estende os casos de uso "Criar Novo *Template*" e "Criar Nova Versão de *Template*". Ambos, por sua vez, incluem os casos de uso "Atribuir Nível de Sucesso" e "Atribuir Características".

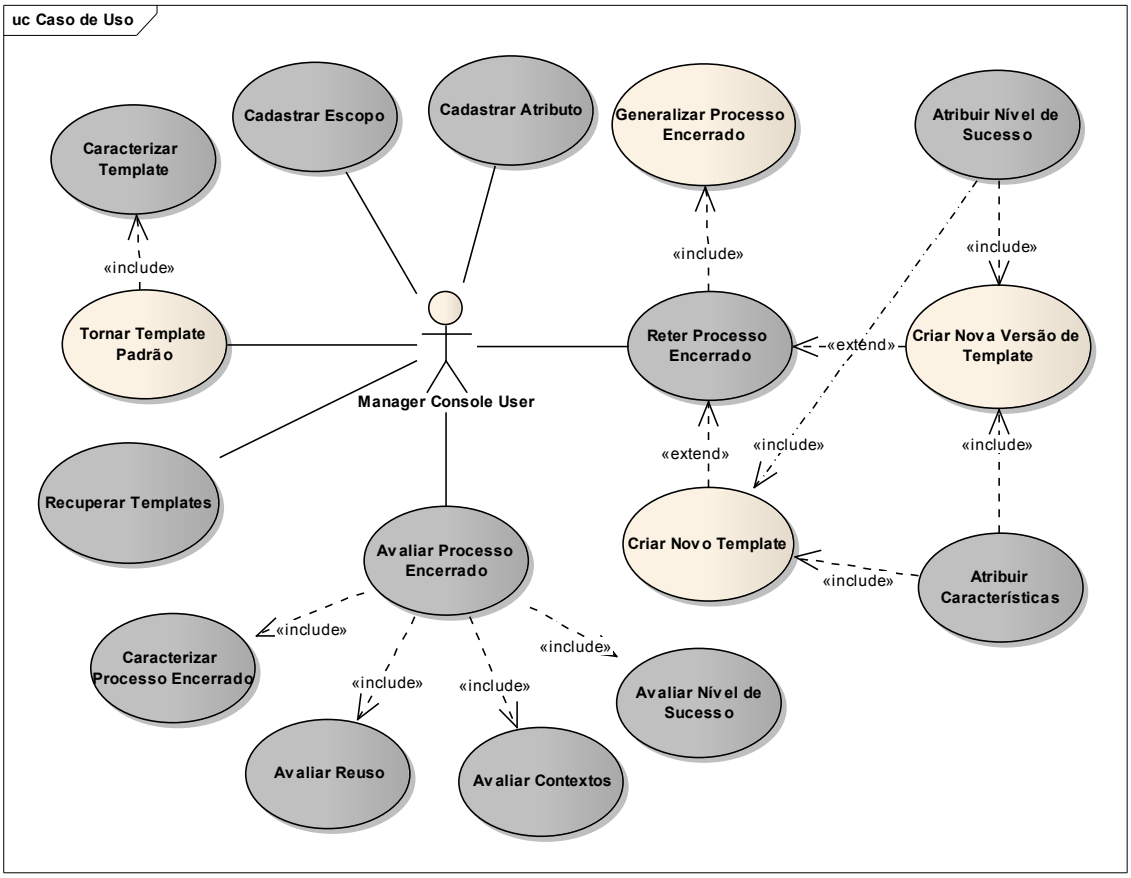

Figura 19 – Diagrama de casos de uso destacando as funcionalidades deste trabalho e sua relação com as funcionalidades pré-existentes no ambiente WebAPSEE-Pro.

A Tabela 9 descreve cada funcionalidade da Figura 19, indicando se a funcionalidade é *nova*, *alterada* ou *mantida*. As funcionalidades *mantidas* foram extraídas de [Costa e Sales, 2007]. As funcionalidades *nova*s surgiram como conseqüência da abordagem proposta. Já as funcionalidades *alteradas*, envolvem funcionalidades existentes que foram modificadas para atender aos novos requisitos.

Tabela 9 – Detalhamento das funcionalidades integrantes deste trabalho

Caso de Uso	Descrição	Situação
Cadastrar Escopo	O usuário cadastra o escopo que serve para agrupar os atributos da representação de contextos dos ativos de processo de software. Os escopos pré-definidos para este trabalho são Processo e Projeto, no entanto, é possível criar novos escopos.	Nova
Cadastrar Atributo	O usuário cadastra o atributo da representação de contextos dos ativos de processo de software.	Nova
Tornar *Template* padrão	O usuário seleciona o *template* que deseja tornar padrão. Isto ocorre com a mudança de estado do *template* (de *draft* para *defined*, de acordo com a Seção 5.2.3). Esta funcionalidade já existe no ambiente [Costa e Sales, 2007].	Mantida
Caracterizar *Template*	O usuário registra a classificação dos contextos de processo e projeto (Tabela 4) de cada caso-base contido no repositório.	Nova
Recuperar *Templates*	O sistema executa o mecanismo de busca automatizado para recuperação de casos-base no repositório de ativos de processos. Esta funcionalidade foi alterada para realizar a busca com base na representação de contextos proposta.	Alterada
Avaliar Processo Encerrado	O usuário realiza a avaliação do processo executado após o seu encerramento.	Nova
Caracterizar Processo Encerrado	Após o encerramento do projeto, a representação de contextos real é registrada pelo usuário.	Nova
Avaliar Reuso	O sistema executa a avaliação de reutilização do processo executado em relação modelo de processo selecionado.	Nova
Avaliar Contextos	O sistema executa a avaliação de contextos, comparando os valores de similaridade global dos contextos preliminar e real em relação ao *template* selecionado.	Nova
Avaliar Nível de Sucesso	O usuário avalia o processo executado fornecendo uma nota no intervalo de 1 a 10.	Nova
Reter Processo Encerrado	O usuário opta por reter o processo encerrado	Nova
Generalizar Processo Encerrado	O sistema gera um *template* do processo executado, removendo detalhes e instâncias de classes [Costa e Sales, 2007].	Mantida
Criar Nova Versão de *Template*	O usuário pode optar por criar uma nova versão do *template* que utilizou. O ambiente já trata este controle de versão [Costa e Sales, 2007].	Mantida
Criar Novo *Template*	O usuário pode optar por criar um novo *template* separadamente [Costa e Sales, 2007].	Mantida
Atribuir Características	O sistema salva as características do *template* gerado.	Nova
Atribuir Nível de Sucesso	O sistema salva o nível de sucesso para o *template* gerado.	Nova

5.2.2 Diagrama de Classes

No contexto da especificação da ferramenta WebAPSEE-Pro, os conceitos centrais de *processo abstrato* e *processo concreto* são modelados através das abstrações *Template* e *Process,* respectivamente, onde a classe *Template* é definida como uma especialização da classe *Process* [Costa e Sales, 2007].A classe *Project* representa os dados do projeto que executa um determinado processo.

A classe *ProcessModel* representa um modelo de processo, onde um processo pode ser composto de vários modelos de processo (Requisitos, Análise, Projeto, Testes, etc.). O relacionamento bidirecional entre as classes *Template* e *ProcessModel*, tem o objetivo de guardar a referência de origem de um processo ou modelo de processo para um *template* e vice-versa. A Figura 20 apresenta o diagrama de classes de acordo com a especificação descrita acima.

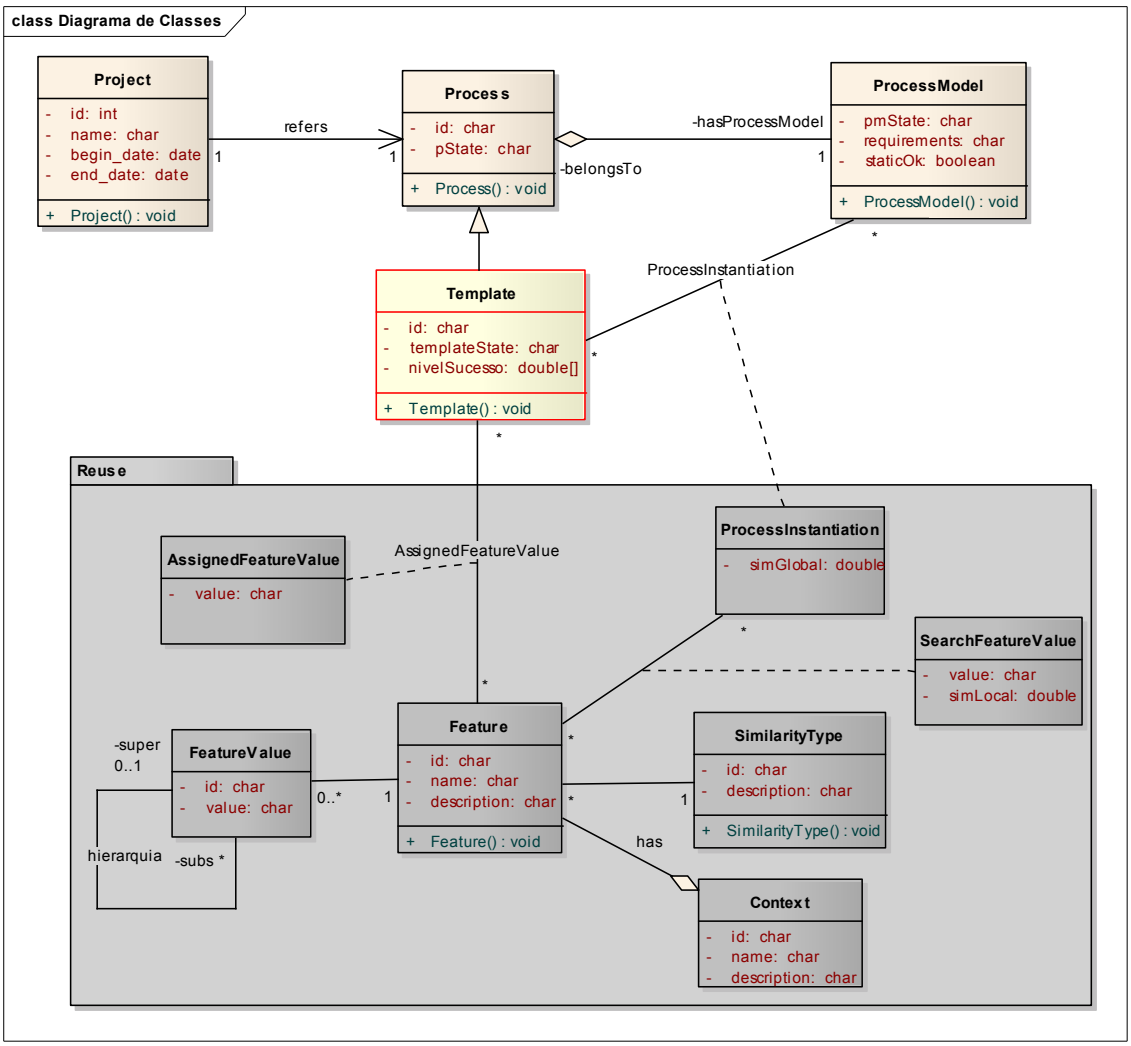

Figura 20 – Diagrama de classes apresentando as classes criadas e sua relação com as classes do modelo WebAPSEE-Pro.

Para viabilizar a implementação das extensões propostas neste trabalho novas classes e atributos foram adicionados ao diagrama original. A classe *Template* foi enriquecida com um novo atributo, denominado *nivelSucesso,* que tem por objetivo armazenar a avaliação atribuída pelo usuário ao *template*. Esta avaliação é utilizada para o cálculo da média do nível de sucesso do *template* nos processos que o instanciaram. As classes *SimilarityType* (classe que representa os tipos de similaridade da Tabela 4), *Feature* (classe que representa os atributos da Tabela 5), *FeatureValue* (classe que

representa os valores possíveis para os atributos), *AssignedFeatureValue* (classe associativa que representa os valores registrados aos atributos para um determinado *Template*), *ProcessInstantiation* (classe associativa que representa o valor da similaridade global do caso-consulta com o caso-base selecionado) e *SearchFeatureValue* (classe associativa que representa o valor do caso-consulta de cada atributo e de sua similaridade local com o caso-base selecionado) foram criadas e agrupadas no pacote *Reuse*.

5.2.3 Máquina de Estados para Templates

Para auxiliar na definição de *templates* no ambiente, [Costa e Sales, 2007] especificaram estados possíveis para um *template*. Esta máquina de estados existente é utilizada nesta proposta e é apresentada na Figura 21.

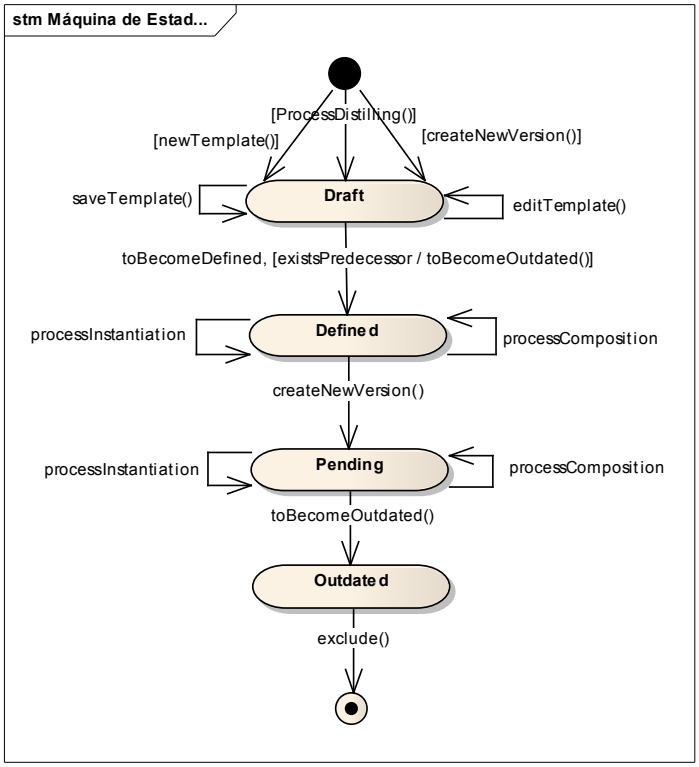

Figura 21 – Máquina de estados de um *template*.

Como esta máquina de estados já atende a abordagem proposta, não houve necessidade de alteração destes estados, portanto, foram mantidos e são descritos a seguir:

- *Draft* (Rascunho): estado de um *template* criado, possibilitando sua modelagem e alteração;
- *Defined* (Definido): estado de um *template* padronizado para a organização, impedindo sua alteração e permitindo sua reutilização;

- *Pending* (Pendente): estado de uma antiga versão de um *template*, que não pode mais ser versionada, quando a nova versão está sendo modelada (no estado *Draft*);
- *OutDated* (Desatualizado): estado de uma antiga versão de um *template* quando uma nova versão de um *template* é padronizada (no estado *Defined*), indicando que não pode ser mais reutilizado.

5.2.4 Diagrama de Atividades

A Figura 22 apresenta o diagrama de atividades das funcionalidades propostas por este trabalho (Tabela 9), destacadas de cinza, e suas respectivas mudanças de estados em *templates* do ambiente WebAPSEE-Pro.

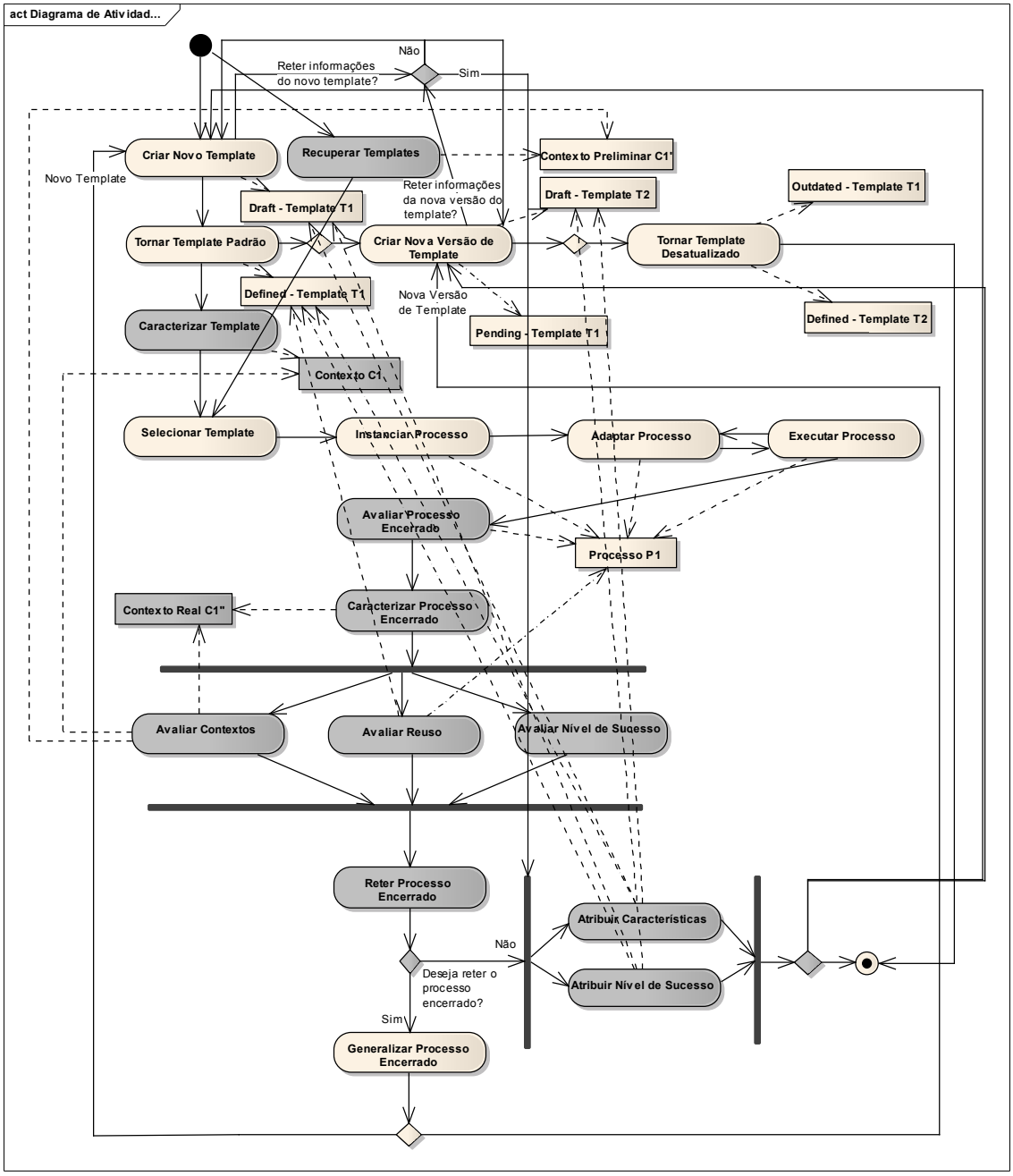

Figura 22 – Diagrama de atividades destacando as funcionalidades deste trabalho.

5.3 Algoritmos dos Principais Mecanismos desta Abordagem

5.3.1 Mecanismo de Recuperação através do Cálculo de Similaridade

O ambiente WebAPSEE-Pro possui uma máquina de busca, chamada *Search Engine*, a qual utiliza RBC para recuperar modelos de processo reutilizáveis similares [Sales et al., 2006]. A busca utiliza como atributos dados referentes às cinco dimensões do meta-modelo, que são os componentes do processo, como por exemplo, artefato de entrada (dimensão Software).

No contexto da extensão proposta neste trabalho, a máquina de busca do WebAPSEE-Pro é alterada para contemplar a busca de modelos de processo reutilizáveis com base em características pré-definidas e níveis de sucesso. Desta forma, a organização pode reutilizar processos mais bem sucedidos a partir dos ciclos de execução dos projetos. Para que isto ocorra, cada *template* deve ter registrada sua representação para os contextos de projeto e processo.

O mecanismo de recuperação adotado utiliza o cálculo de similaridade apresentado na Seção 4.3. Este mecanismo é implementado a partir de uma função principal, denominada *Similaridade_Global*, que realiza o cálculo da similaridade global entre a representação de contextos dos requisitos fornecidos pelo usuário (caso-atual) e as representações de contexto dos modelos de processo do repositório (casos-base). Nesta função, são chamadas mais três funções, denominadas *CalcularSimLocalNUM*, *CalcularSimLocalQIF* e *CalcularSimLocalQIV*, que calculam as similaridades locais dos atributos de acordo com o tipo de similaridade (NUM, QIF e QIV). A seguir são apresentados os algoritmos de cada uma das funções mencionadas.

```
ALGORITMO Similaridade_Global
Variáveis:
        Vetor CASO_ATUAL[2, 14]: alfanumérico;
        //Primeira linha deste vetor armazena os pesos atribuídos a cada atributo
        //correspondente ao índice do vetor para o caso atual.
        //Segunda linha deste vetor armazena o valor de cada atributo para o caso atual.

        Vetor CASO_BASE[2, 14]: alfanumérico;
        //Primeira linha deste vetor armazena o tipo de similaridade de cada atributo
        //correspondente ao índice do vetor para o caso atual.
        //Segunda linha deste vetor armazena o valor de cada atributo para o caso-base.

        Vetor SIM_LOCAL[1, 14]: real;
        //Primeira linha do vetor armazena o valor da similaridade encontrada para o
        //atributo correspondente ao índice do vetor.

        Vetor SIM_GLOBAL[2, n]: real;
        //Primeira linha deste vetor armazena o identificador do caso do repositório em
        //relação ao caso atual.
        //Segunda linha deste vetor armazena o valor da similaridade global de cada caso
        //do repositório em relação ao caso atual.

        real: val_sim_local, val_sim_global;
        inteiro: i, j;
Início
        Para cada caso do repositório faça
                Para i de 1 até 14 faça
                Se CASO_BASE[1, i] = "NUM"
                Então
                                val_sim_local   = CalcularSimLocalNUM (i, CASO_BASE[2,i],
                                CASO_ATUAL[2, i]);
                Se CASO_BASE(1, i) = "QIF"
                Então
                                val_sim_local    =  CalcularSimLocalQIF  (CASO_BASE[2,i],
                                CASO_ATUAL[2, i]);
                Se CASO_BASE(1, i) = "QIV"
                Então
                                val_sim_local   = CalcularSimLocalQIV (i, CASO_BASE[2,i],
                                CASO_ATUAL[2, i]);
                        SIM_LOCAL[i] = val_sim_local;

                val_sim_global = 0;

                Para j de 1 até 14 faça
                        val_sim_global   =   val_sim_global   +   (CASO_ATUAL[1,   j]   *
                        SIM_LOCAL[j]);
                SIM_GLOBAL[j] = (id_caso, val_sim_global);

                 Retorna SIM_GLOBAL;
Fim
```

O algoritmo *CalcularSimLocalNUM*, é baseado nas fórmulas (7) e (8) da Seção 4.3.

```
ALGORITMO CalcularSimLocalNUM (i, val_caso_atual, val_caso_base)
Variáveis:
        real: val_caso_atual, val_caso_base, val_sim_local, val_min, val_max, d;
        inteiro: i;
Início
        val_min = ObterValorMinimo(i);
        val_max = ObterValorMaximo(i);
        Se val_min = val_max
        Então
             d = |(val_caso_atual - val_caso_base)/(val_min)|;
        Senão
              d = |((val_caso_base - val_min)/(val_max - val_min)) -
                      ((val_caso_atual - val_min)/(val_max - val_min))|;

        val_sim_local = (1/1+d);

        Retorna val_sim_local;
Fim
```

O algoritmo *CalcularSimLocalQIF*, implementa a fórmula (10) da Seção 4.3.

```
ALGORITMO CalcularSimLocalQIF (val_caso_atual, val_caso_base)
Variáveis:
       real: val_caso_atual, val_caso_base, val_sim_local, val_min, val_max, d;
Início
       d = |((val_caso_base - 3)/6) - ((val_caso_atual - 3)/6)|;
              val_sim_local = (1/1+d);

              Retorna val_sim_local;
Fim
```

O algoritmo *CalcularSimLocalQIV*, implementa a fórmula (11) da Seção 4.3.

```
ALGORITMO CalcularSimLocalQIV (na, val_caso_atual, val_caso_base)
Variáveis:
       real: val_caso_atual, val_caso_base, val_sim_local;
       inteiro: Apr, Aap, Abp, npai;
Início
       Se (mesmoCaso(val_caso_atual, val_caso_base))
       Então
           val_sim_local = 1;
       Senão
           //Obter o pai comum entre caso atual e o caso-base na taxonomia do atributo
           npai = ObterPaiComum(val_caso_atual, val_caso_base);

           //Contar nº arestas do caso base para o pai comum
           Aap = ObterDistanciaAtePaiComum(val_caso_base, npai);

           //Contar nº arestas do caso atual para o pai comum
           Abp = ObterDistanciaAtePaiComum(val_caso_atual, npai);

           //Contar nº arestas do pai comum para a raiz da taxomonia
           Apr = ObterDistanciaAteRaiz(npai);

           val_sim_local = (2*Apr/(Aap+Abp+(2*Apr)));

       Retorna val_sim_local;
Fim
```

5.3.2 Mecanismo de Comparação da Similaridade Global dos Contextos Preliminar e Real

Este mecanismo realiza a comparação da similaridade global do contexto preliminar fornecido pelo usuário como requisito de obtenção do modelo reutilizável com a similaridade global do contexto real após o encerramento do projeto. O algoritmo é baseado na fórmula (13) da Seção 4.4.1.

```
ALGORITMO Comparacao_Similaridades_Globais

Variáveis:

       real: simC1, simC2, indice;
Início
       simC1 = ObterSimilaridadeGlobalC1;
       simC2 = ObterSimilaridadeGlobalC2;

       indice = (100*simC2/simC1)-100;

       Retorna indice;

Fim
```

5.3.3 Mecanismo de Avaliação da Reutilização do Modelo de Processo

Este mecanismo fornece o nível de reuso de um modelo de processo em um processo executado, resultando em um percentual de reuso. O algoritmo é baseado na fórmula (14) da Seção 4.4.2.

```
ALGORITMO Avaliacao_Nivel_Reuso(id_processo, id_template, limiar)
Variáveis:
        Real: nR;
        Inteiro: i, totalCSim, totalAt, nCSim, id_atividade, id_processo, id_template;
Início
        totalCSim = 0;
        i = 10;
        totalAt = ObterTotalAtividades(id_template);

        Para j = 1 até totalAt faça
             nCSim    =    ObterQtde_Componentes_Similares(id_atividade,    id_processo,
id_template, limiar);
             totalCSim = totalCSim  + nCSim;

        nR = totalCSim /totalAt*i;

        Retorna nR;
Fim
```

O algoritmo *ObterQtde_Componentes_Similares* retorna o total de componentes similares de acordo com as funções da Tabela 7 (Seção 4.4.2).

```
ALGORITMO ObterQtde_Componentes_Similares(atividadeProc, atividadeTemp, limiar)
Variáveis:
        Inteiro: nCSim;
Início
        nCSim = 0;
        nCSim = nCSim +
            Token(atividadeProc.Nome, atividadeTemp.Nome, pesoNome, limiar);

        nCSim = nCSim +
            Token(atividadeProc.Roteiro, atividadeTemp.Roteiro, pesoRoteiro, limiar);

        nCSim = nCSim +
            Hierarquica(atividadeProc.Tipo, atividadeTemp.Tipo, pesoTipo, limiar);

        nCSim = nCSim +
            simCon(atividadeProc.Pred, atividadeTemp.Pred, pesoPred, limiar);

        nCSim = nCSim +
            simCon(atividadeProc.Suc, atividadeTemp.Suc, pesoSuc, limiar);

        nCSim = nCSim +
            simRecursos(atividadeProc.Recursos, atividadeTemp.Recursos, limiar);

        nCSim = nCSim +
            simPessoas(atividadeProc.Pessoas, atividadeTemp.Pessoas, limiar);

        nCSim = nCSim +
            simArtefatos(atividadeProc.Artef_In, atividadeTemp.Artef_In, limiar);

        nCSim = nCSim +
            simArtefatos(atividadeProc.Artef_Out, atividadeTemp.Artef_Out, limiar);

        nCSim = nCSim +
            Token(atividadeProc.Mod_Proc, atividadeTemp.Mod_Proc, limiar);

        Retorna nCSim;
Fim
```

5.4 Protótipo para o Ambiente WebAPSEE-Pro

A abordagem descrita no Capítulo 4, a sua especificação e os algoritmos dos principais mecanismos descritos nas seções anteriores deste capítulo servem de base para a implementação do protótipo para o ambiente WebAPSEE-Pro. Nesta seção são apresentadas as telas deste protótipo que permitem a visualização do resultado desta especificação.

Para permitir a representação de contextos de ativos de processo de software são necessários os cadastros de escopo e atributo, ilustrados nas Figuras 23 e 24, conforme Seção 4.2.

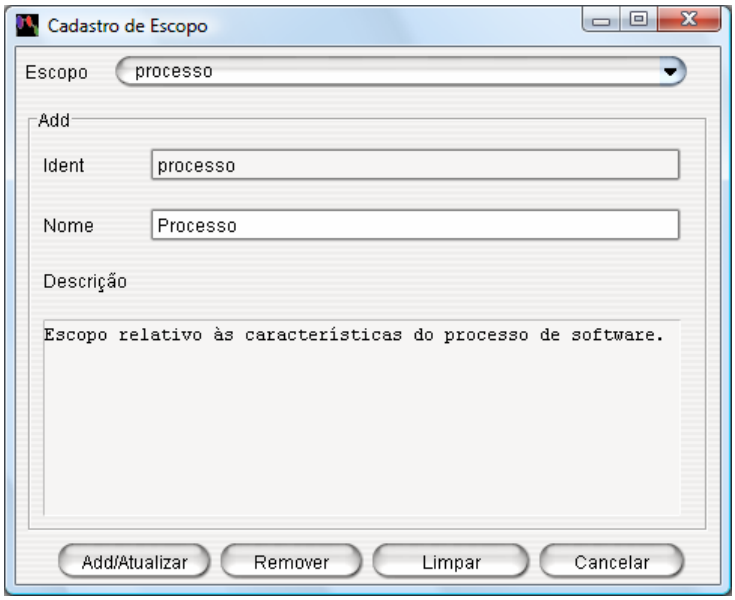

Figura 23 – Tela para cadastrar o escopo da representação de ativos de processo de software.

Figura 24 – Tela para cadastrar atributos da representação de ativos de processo de software.

Ao criar um *template* seu estado fica inicialmente no estado *Draft* (Figura 21). Após a conclusão da modelagem do *template*, este apenas fica disponível para instanciação de processos de software quando o usuário o torna um processo padrão, isto significa que o *template* muda para o estado *Defined* e, assim, está apto à instanciação. Na Figura 25 é apresentado o *template* ProGer se tornando processo padrão. O WebAPSEE-Pro permite a criação de diversos processos padrão contemplando contextos diferenciados. Isto significa que uma organização pode utilizar mais de um processo padrão [Costa e Sales, 2007].

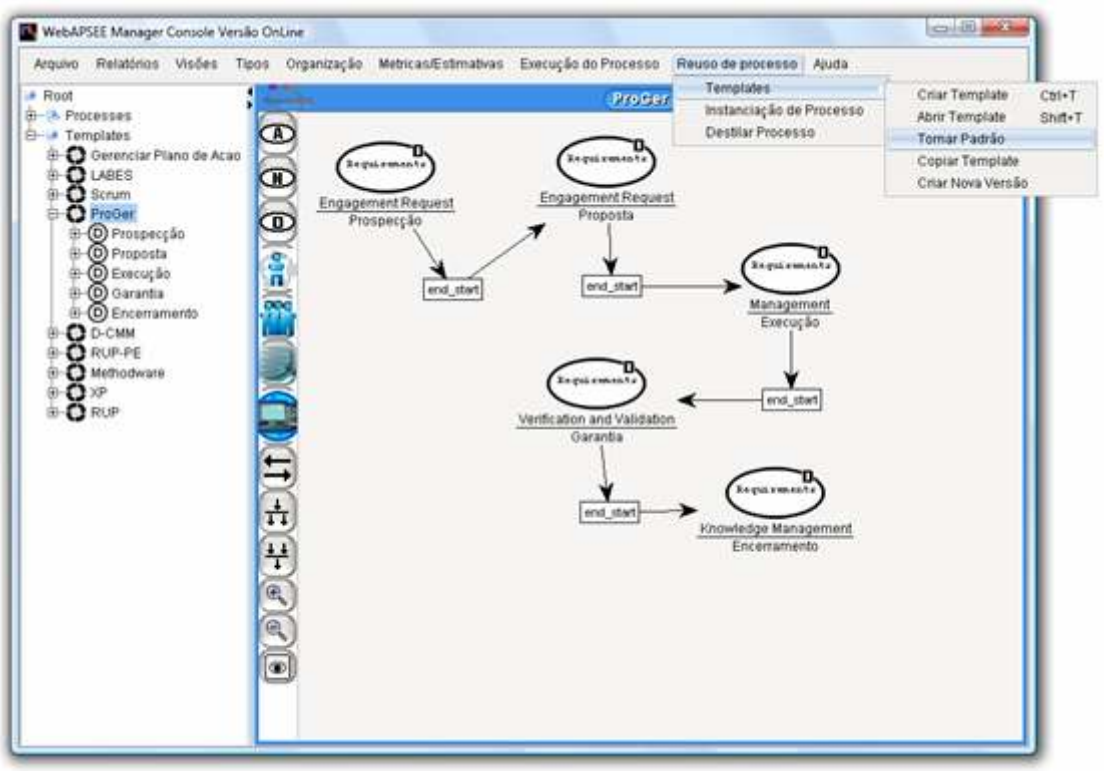

Figura 25 – Tela que exibe o *template* ProGer se tornando padrão (mudança para o estado *Defined*).

Um *template,* que se encontra no estado *Draft* ou *Defined,* pode ser caracterizado dentro dos atributos e escopos estabelecidos. As figuras 26 e 27 apresentam a caracterização do *template* ProGer para os escopos *Projeto* e *Processo*. Estes atributos de caracterização são opcionais.

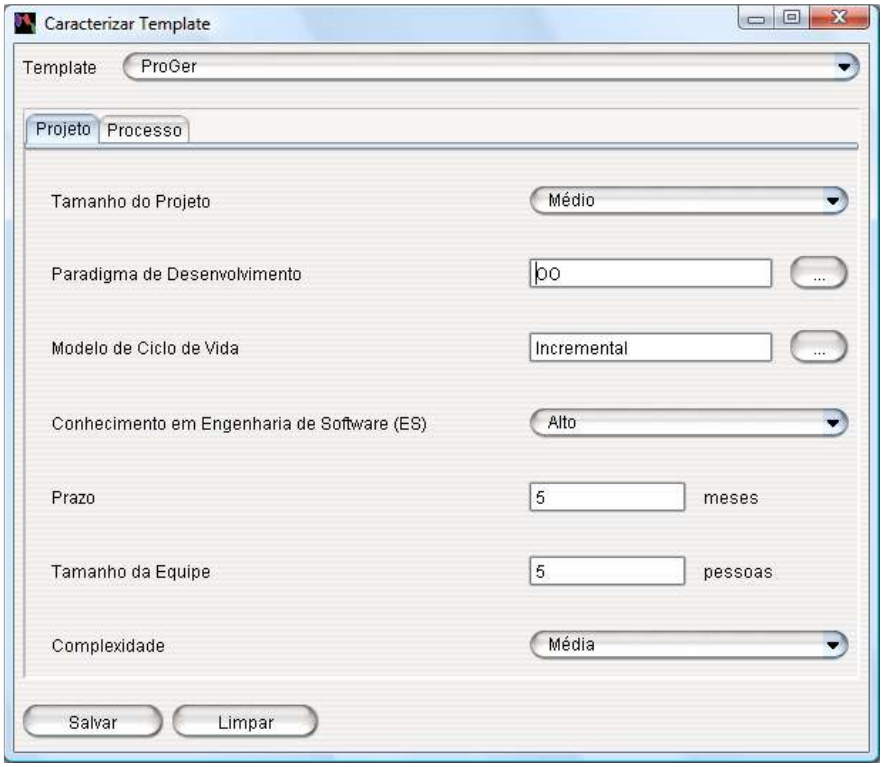

Figura 26 – Tela para caracterização do *template* ProGer para o escopo *Projeto*.

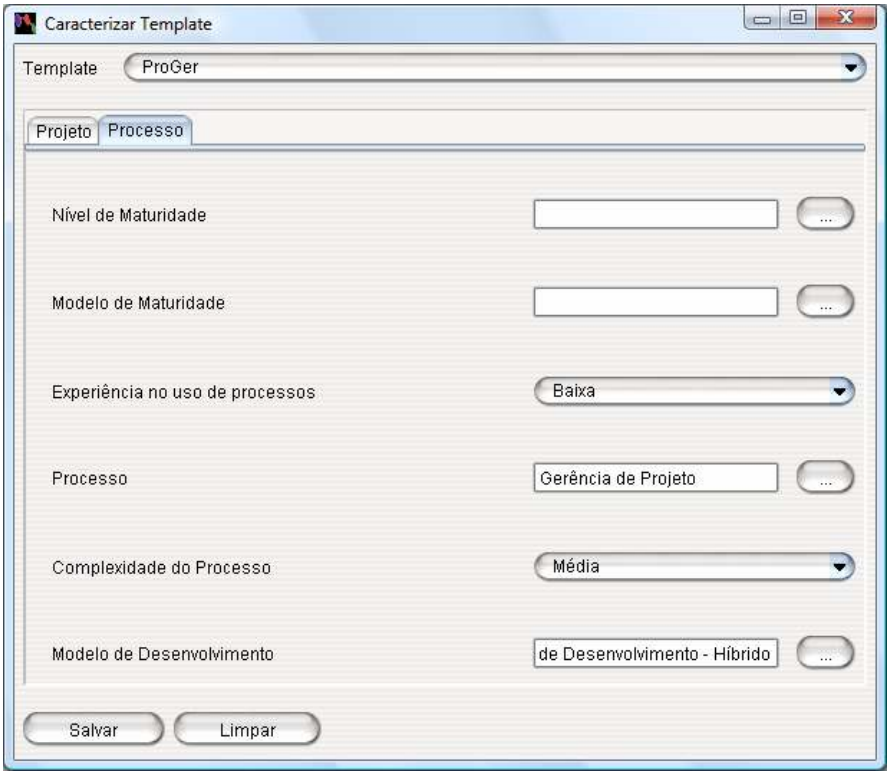

Figura 27 – Tela para caracterização do *template* ProGer para o escopo *Processo*.

Após a caracterização dos *templates* no estado *Defined*, o usuário está apto a realizar o processo de recuperação de *templates* mais similares às suas características e necessidades. Nas Figura 28 e 29 são apresentadas as telas de busca de *templates* de acordo com o escopo, onde o usuário preenche as informações dos requisitos do *template* desejado.

O campo *Nível de Sucesso* serve de filtro para o resultado da busca. Se o usuário preencher este campo, o resultado da busca apenas retornará os *templates* mais similares com nível de sucesso acima do valor fornecido. Caso contrário, o sistema retorna todos os *templates* existentes no repositório, bem como suas respectivas similaridades globais e nível de sucesso, se houver.

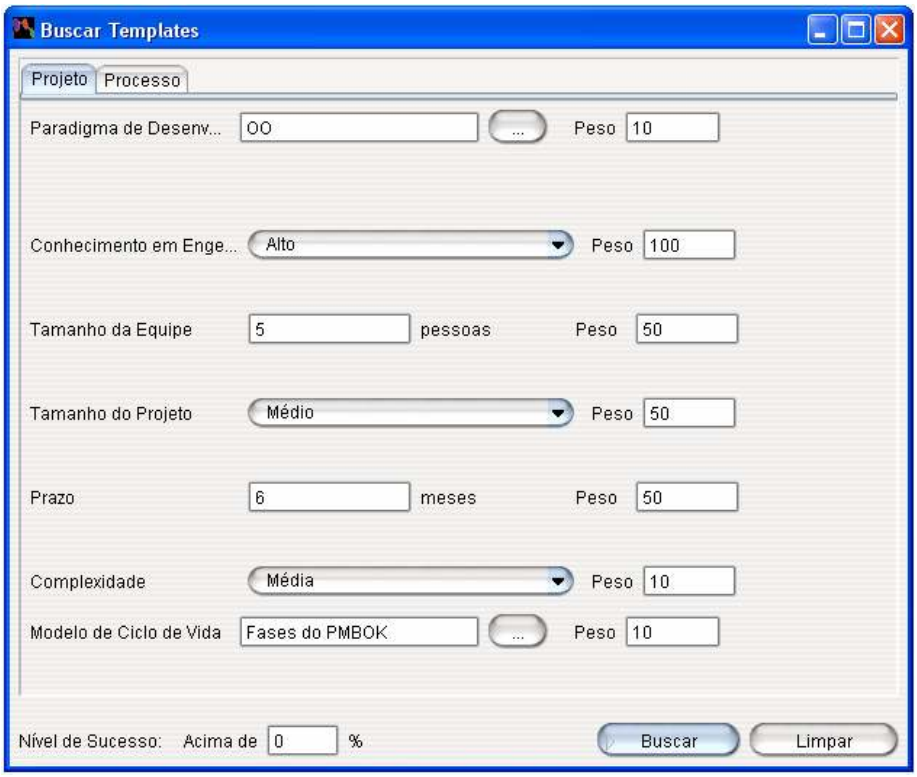

Figura 28 – Tela de busca de *templates* escopo *Projeto*.

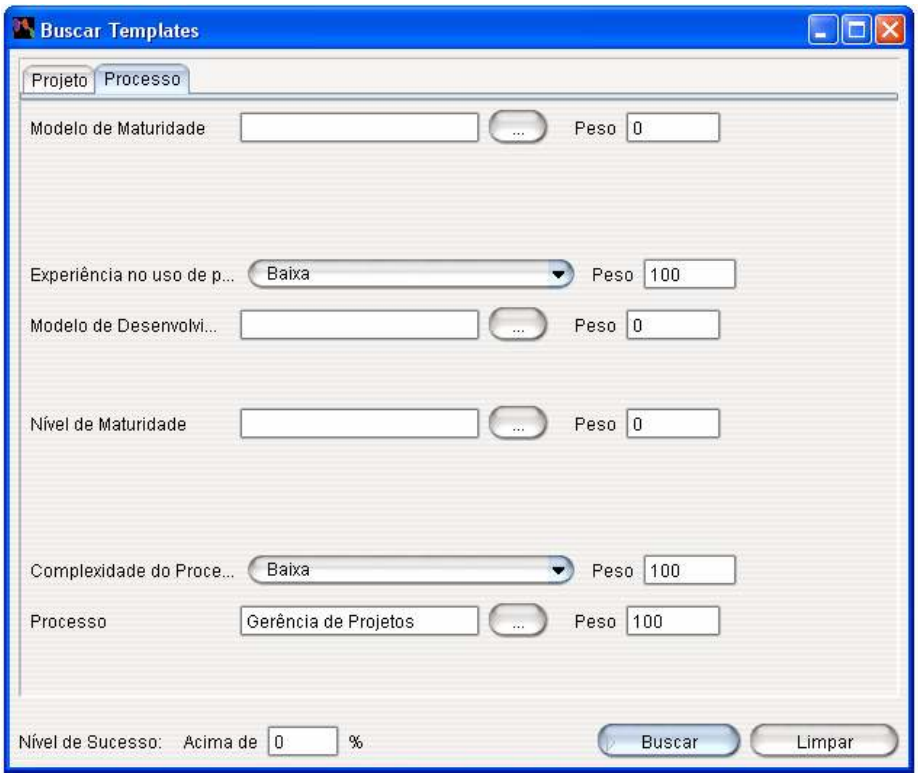

Figura 29 – Tela de busca de *templates* escopo *Processo*.

A Figura 30 ilustra o resultado da recuperação de *templates* de acordo com os valores definidos para os atributos da representação de contextos e nível de sucesso, denominado *Ranking de Templates*. Este *ranking* é pré-ordenado em ordem decrescente

dos valores de similaridade global de cada *template,* retornado e filtrado pelo nível de sucesso fornecido na busca.

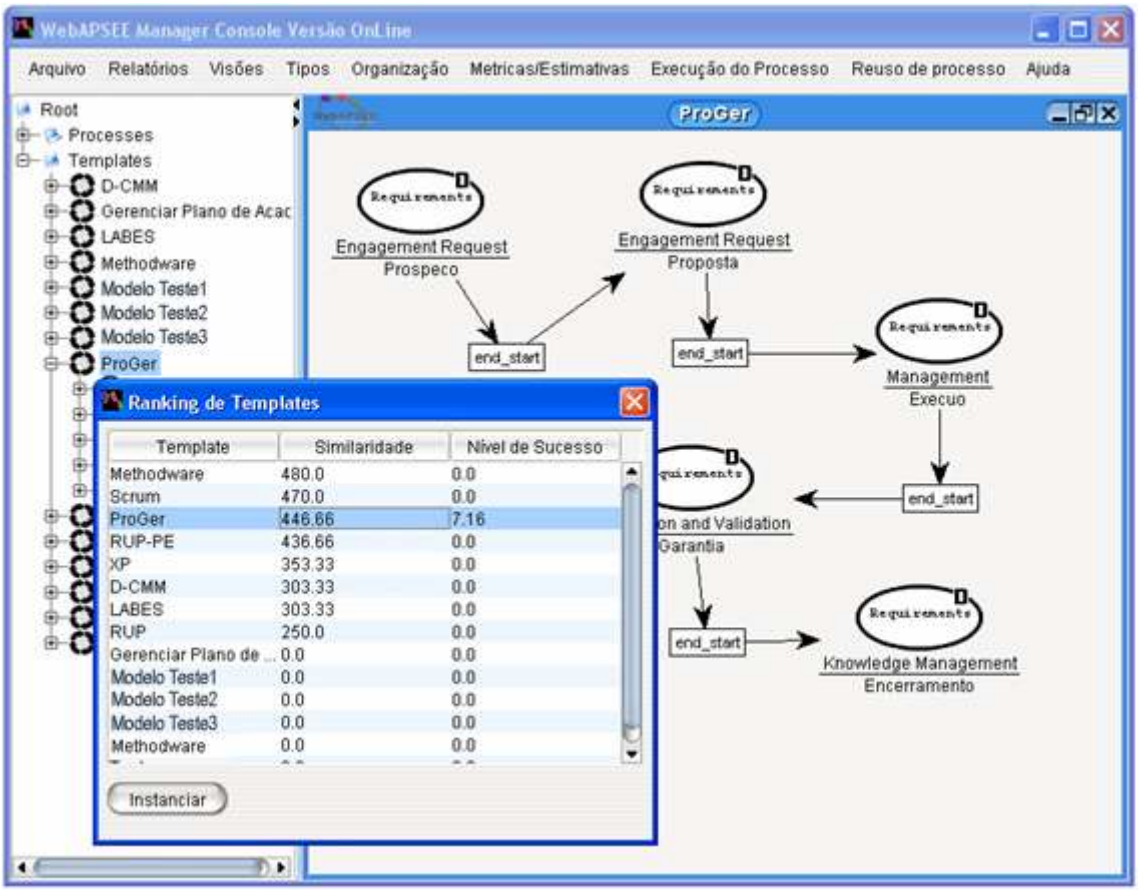

Figura 30 – Tela que exibe o resultado da recuperação de *templates* e visualização do *template* selecionado.

Ao selecionar o *template* mais adequado, o usuário instancia o processo (Figura 31) e a partir daí pode efetuar modificações antes ou durante a sua execução, como adaptação/composição de processos (funcionalidades do próprio ambiente [Costa e Sales, 2007]).

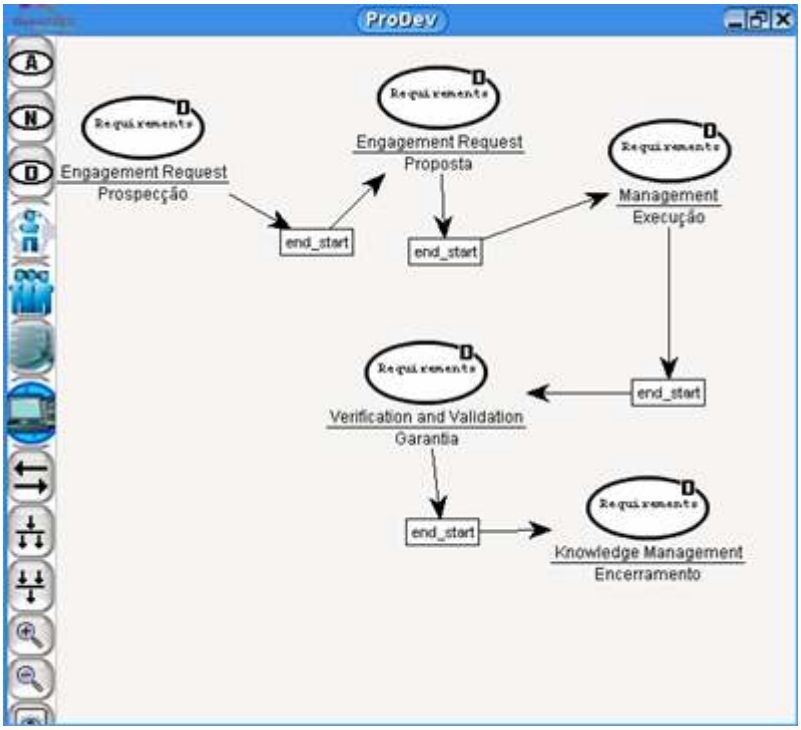

Figura 31 – Processo ProDev como uma instância do *template* ProGer.

Após o encerramento do projeto, o usuário pode realizar a avaliação do processo encerrado. Para isto, ele é solicitado a caracterizar o processo encerrado (representação de contexto real), conforme as Figuras 32 e 33.

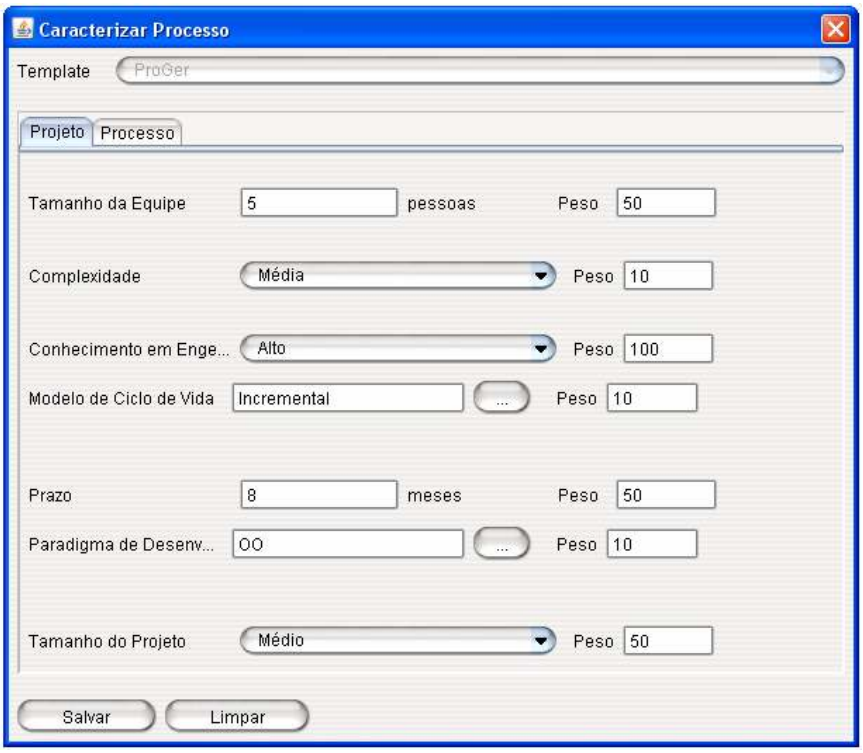

Figura 32 – Representação do contexto de *Projeto* para o processo ProDev.

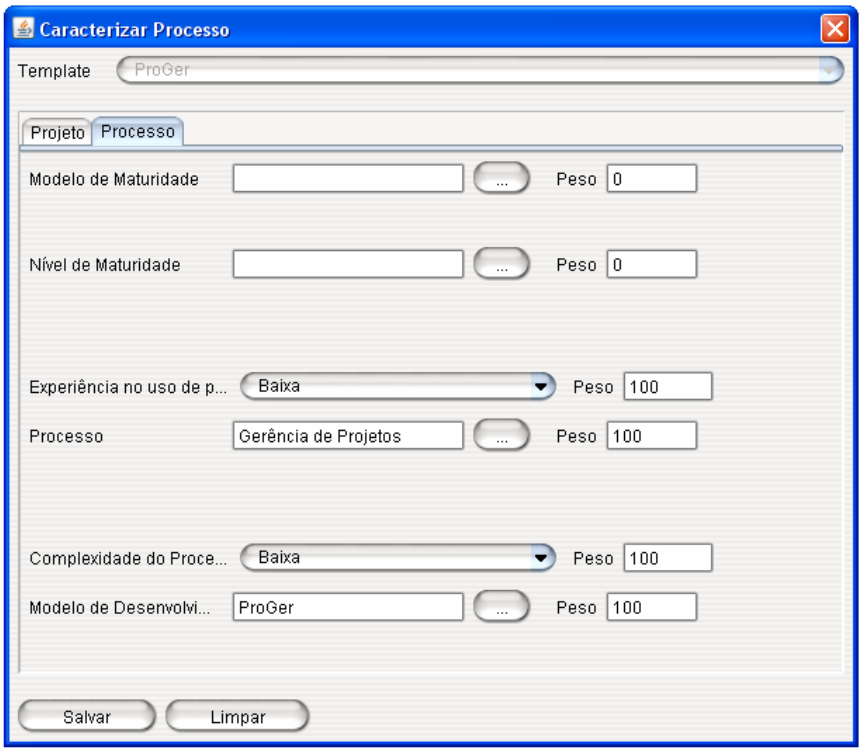

Figura 33 – Representação do contexto de *Processo* para o ProDev.

Após a caracterização do processo encerrado, a tela de avaliação e retenção do processo é apresentada. A Figura 34 ilustra as métricas de avaliação, onde a *Comparação de Similaridades Globais* e o *Nível de Reutilização do Template* são gerados automaticamente, tornando-se visível o campo texto para o usuário fornecer o *Nível de Sucesso* do processo encerrado, conforme Seção 4.4. Para a métrica *Nível de Reutilização do Template* é possível o usuário especificar o limiar de similaridade, conforme explicado na Seção 4.4.2. No entanto, este precisa conhecer o meta-modelo do ambiente para calibrar o limiar de similaridade para um valor satisfatório para a sua organização [França, 2007].

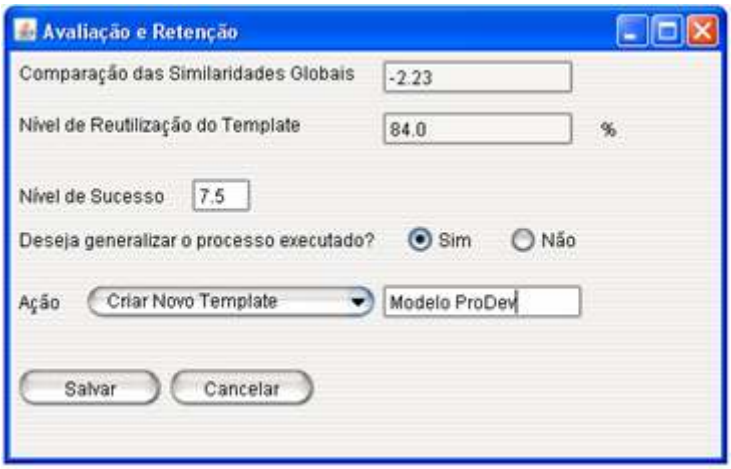

Figura 34 – Tela de avaliação e retenção do processo encerrado.

Da mesma forma, a Figura 34 também disponibiliza as opções de retenção do processo encerrado. Se o usuário solicitar *Criar Novo Template* (Figura 35), um campo de texto é habilitado para a inclusão do nome do novo *template*, a ser criado no estado *Draft*, bem como são atribuídas a representação do contexto real e o nível de sucesso para este novo *template*. Porém, se o usuário selecionar *Criar Nova Versão do Template*, este campo fica desabilitado e uma nova versão, no estado *Draft*, é gerada para o *template.* Neste caso, a representação do contexto real e o nível de sucesso são atribuídos a esta nova versão. Quando o usuário seleciona não reter o processo encerrado, o nível de sucesso deste *template* é armazenado juntamente com as notas previamente definidas.

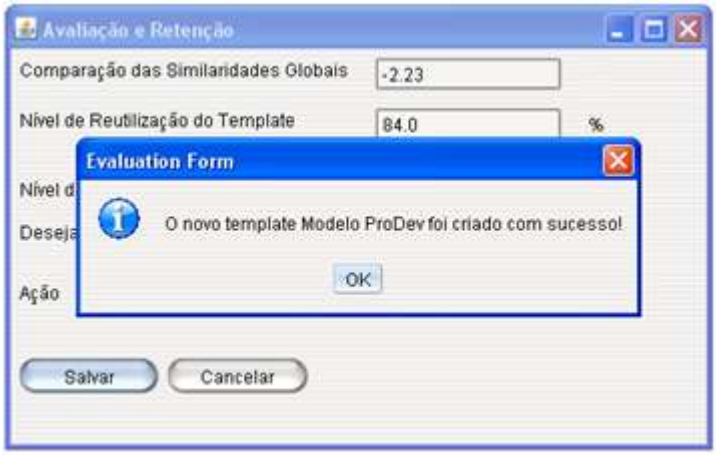

Figura 35 – Tela que exibe a criação de um novo *template*.

A Figura 36 ilustra que a retenção de um novo *template* ou de uma nova versão do *template*, a partir do processo, possibilita disponibilizar a sua representação de contextos e o seu nível de sucesso em novas recuperações de *templates*.

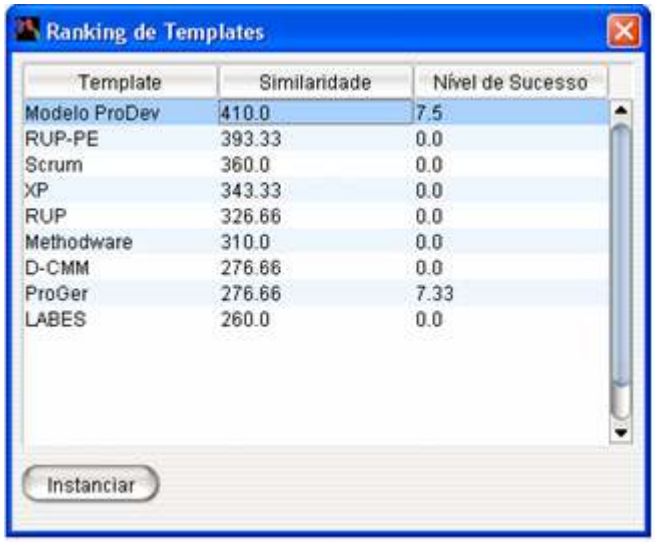

Figura 36 – Tela que exibe o Modelo ProDev fazendo parte do resultado da busca após a sua retenção no repositório.

Por fim, dependendo dos requisitos fornecidos na busca e do filtro de nível de sucesso, os *templates* mais similares e mais bem sucedidos são priorizados, promovendo, desta forma, o reuso de experiências bem sucedidas e a melhoria contínua dos processos da organização.

6 SIMULAÇÃO DA PROPOSTA

A simulação é um processo de planejamento de um modelo para um sistema real, onde são conduzidos experimentos para um propósito qualquer de compreensão do comportamento do sistema ou para o propósito de avaliar várias estratégias para a operação do sistema [Shannon, 1975].

Na área de processos de software, a simulação é uma ferramenta de análise útil no refinamento de modelos de processos, visto que com a percepção da execução do modelo possibilita enriquecer o modelo executável através de informações obtidas da simulação e antecipar resultados da realização do projeto. Conseqüentemente pode-se aumentar o grau de entendimento e gerenciamento de um processo de software [Lins e Chwif, 2008].

Este capítulo apresenta uma simulação da abordagem proposta no Capítulo 4 a partir de um projeto pertencente ao Laboratório de Engenharia de Software da Universidade Federal do Pará (LABES-UFPA), denominado SIGAP (Sistema Integrado de Gestão de Grupos e Atividades de Pesquisa do Estado do Pará), relatado em [Lemos et al., 2009]. Por ser uma simulação, a maioria das informações apresentadas é fictícia.

O projeto SIGAP possui o propósito de permitir a coleta e gerência de informações sobre os pesquisadores, laboratórios e grupos de pesquisa em atuação no Pará. Além de possibilitar a geração de relatórios e indicadores úteis aos órgãos de fomento à pesquisa do Governo do Estado do Pará.

Este projeto possui duas etapas, a primeira trata do módulo Formulário e a segunda trata dos módulos de Coleta e Geração de Relatórios e Indicadores [Lemos et al., 2009]. Cada etapa deste projeto possui requisitos de processo específicos, no entanto, por ser um processo de software do LABES-UFPA, seu principal requisito consiste em seguir um processo padrão da instituição aderente ao nível G do MPS.BR [Softex, 2007].

Apesar da adoção de um *template*, o ambiente permite mudanças durante a execução. Com o surgimento de eventos específicos, ações são tomadas para que o processo de software adotado se adapte às necessidades no decorrer da sua execução. Em ambos os casos é possível identificar os eventos ocorridos durante o projeto que favorecem mudanças nos processos e, a partir disto, é fundamental avaliar a adaptação realizada, a reutilização do *template* e seu nível de sucesso de acordo com as suas características para apoiar a retroalimentação do repositório.

De acordo com [Lins e Chwif, 2008], o diagrama de causas e efeitos representa o mecanismo mais simples para representação de modelos de simulação. A Figura 37 ilustra o diagrama de causas e efeitos proposto neste trabalho para a modelagem de um processo de software e são utilizados neste capítulo para explicar o impacto dos eventos ocorridos durante a execução de ambos os projetos.

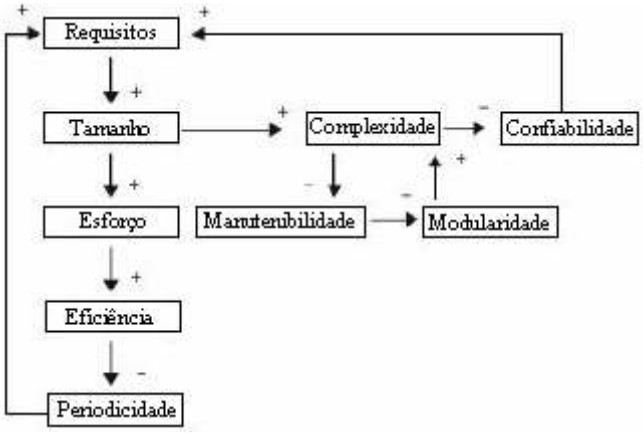

Figura 37 - Diagrama de causas e efeitos para a modelagem de um processo de software.

Neste diagrama há três ciclos de *feedback*, seguindo o sentido das setas:

- Requisitos – Tamanho – Esforço – Eficiência – Periodicidade: à medida que surgem novos requisitos, o tamanho do sistema aumenta, resultando em um maior esforço de desenvolvimento, tornando necessário o aumento da eficiência da equipe, o que contribui para a diminuição da periodicidade entre as versões do sistema em evolução, fazendo com que mais requisitos possam ser atendidos;
- Requisitos – Tamanho – Complexidade – Confiabilidade: à medida que surgem novos requisitos, aumenta o tamanho do sistema, aumenta também a complexidade estrutural do mesmo, contribuindo para a diminuição da confiabilidade do sistema e, conseqüentemente, faz com que novas manutenções sejam necessárias no sentido de atender requisitos de qualidade do sistema;
- Complexidade – Manutenibilidade – Modularidade: um aumento de complexidade estrutural no sistema acarreta a diminuição de sua manutenibilidade, o que faz com que a modularidade do sistema diminua, ocasionando aumento na complexidade do sistema.

Neste capítulo é apresentado o *template* LABES-UFPA, bem como sua representação de contexto fictícia. Em seguida são descritas as duas etapas do projeto, abordando seus requisitos (fictícios) de processo, a recuperação que levou a instanciação do *template* LABES-UFPA, os principais eventos que levaram às mudanças realizadas em cada processo, a avaliação e a retenção, e por fim, serão apresentados os resultados obtidos com este projeto.

6.1 Template LABES-UFPA

Este *template* foi definido pelo LABES-UFPA no ambiente WebAPSEE-Pro, para ser utilizado no desenvolvimento de seus projetos de software. O *template* é totalmente aderente às práticas do nível G do MPS.BR [Lemos et al., 2009], incrementado com o processo de Medição e outros processos de níveis mais altos, como Engenharia de Requisitos, Projeto Arquitetural, Revisões e Testes e algumas práticas de Gerência de Conhecimento e Definição e Avaliação do Processo Organizacional.

A Figura 38 ilustra o *template* LABES-UFPA, as elipses representam as atividades, o *D*, nas cinco elipses da figura, significa atividade decomposta, ou seja, um novo subprocesso é definido para a atividade, enquanto as setas e caixas fornecem informação sobre a seqüência de execução, de acordo com a notação WebAPSEE [LABES-UFPA, 2007].

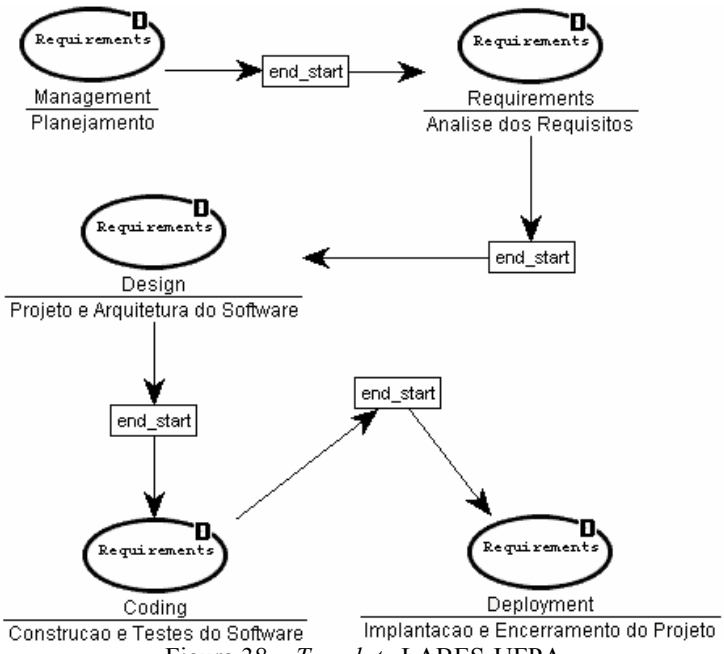

Figura 38 – *Template* LABES-UFPA.

As Figuras 39, 40, 41, 42 e 43 representam os subprocessos deste *template*, que são Planejamento, Análise de Requisitos, Projeto e Arquitetura do Software, Construção e Testes do Software, e Implantação e Encerramento.

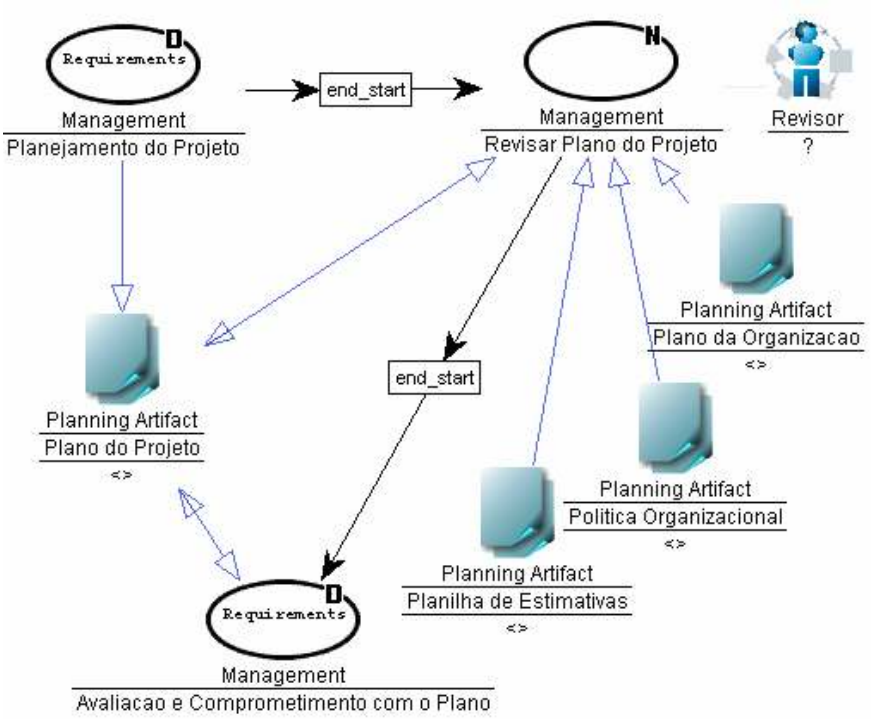

Figura 39 – Subprocesso *Planejamento* do *template* LABES-UFPA.

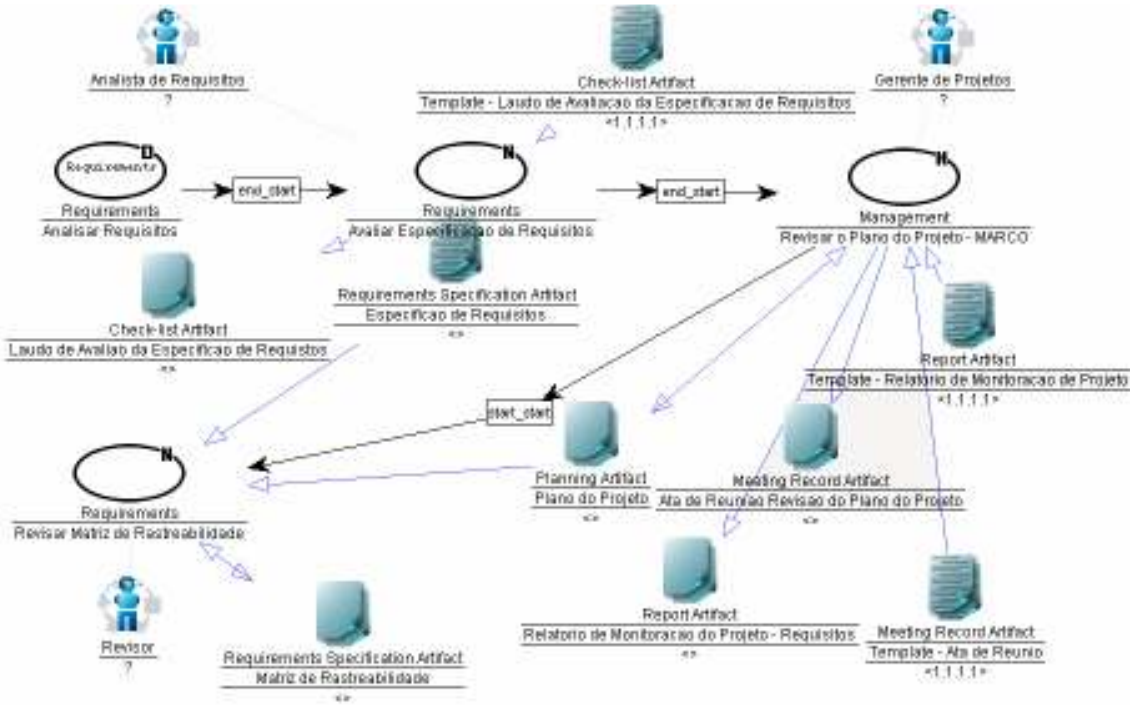

Figura 40 – Subprocesso *Análise dos Requisitos* do *template* LABES-UFPA.

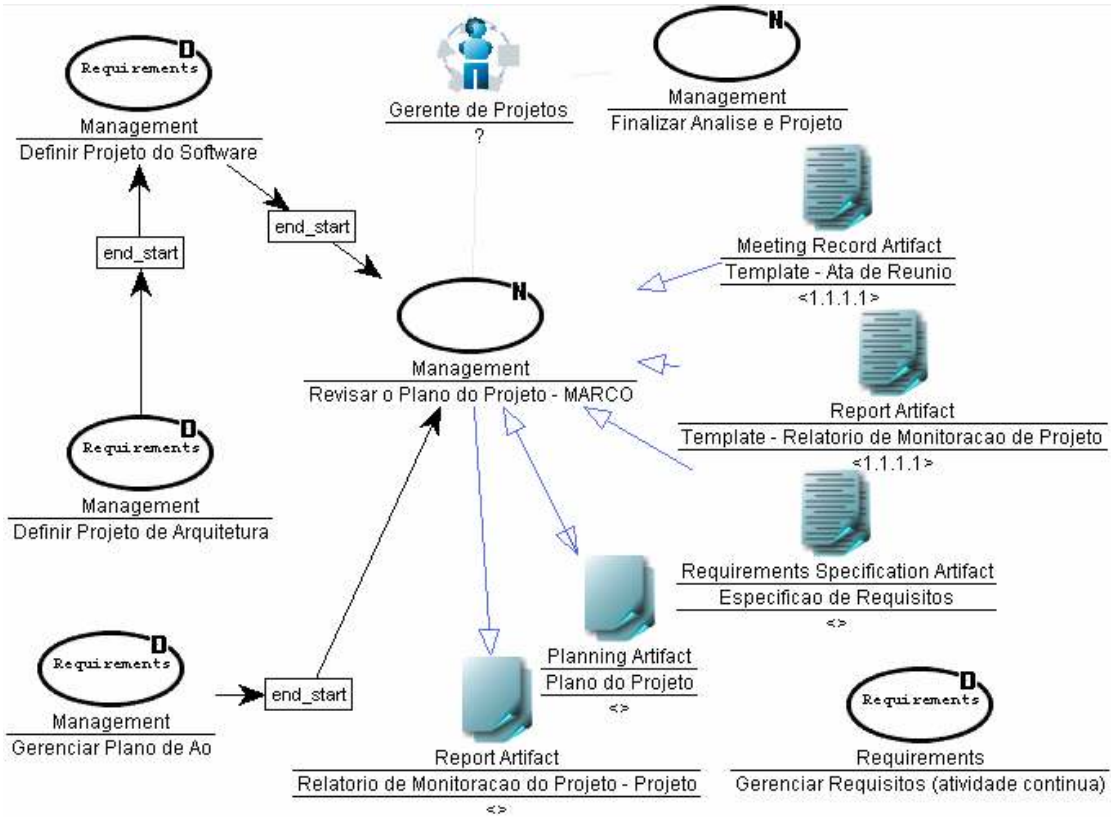

Figura 41 – Subprocesso *Projeto e Arquitetura do Software* do *template* LABES-UFPA.

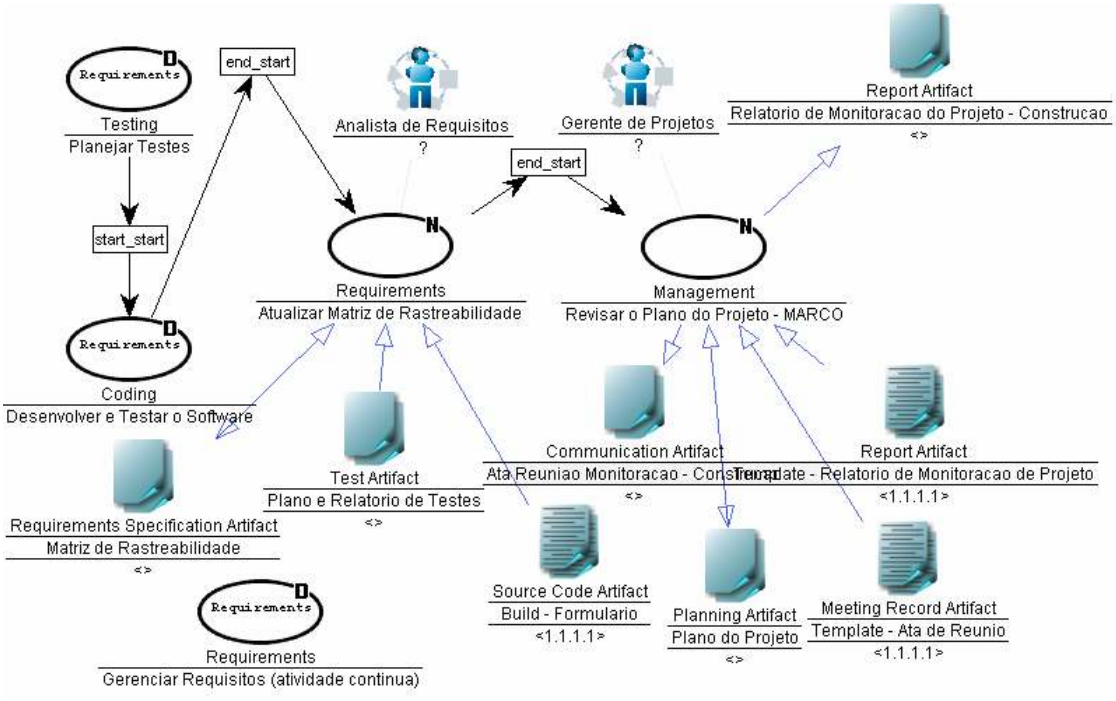

Figura 42 – Subprocesso *Construção e Testes do Software* do *template* LABES-UFPA.

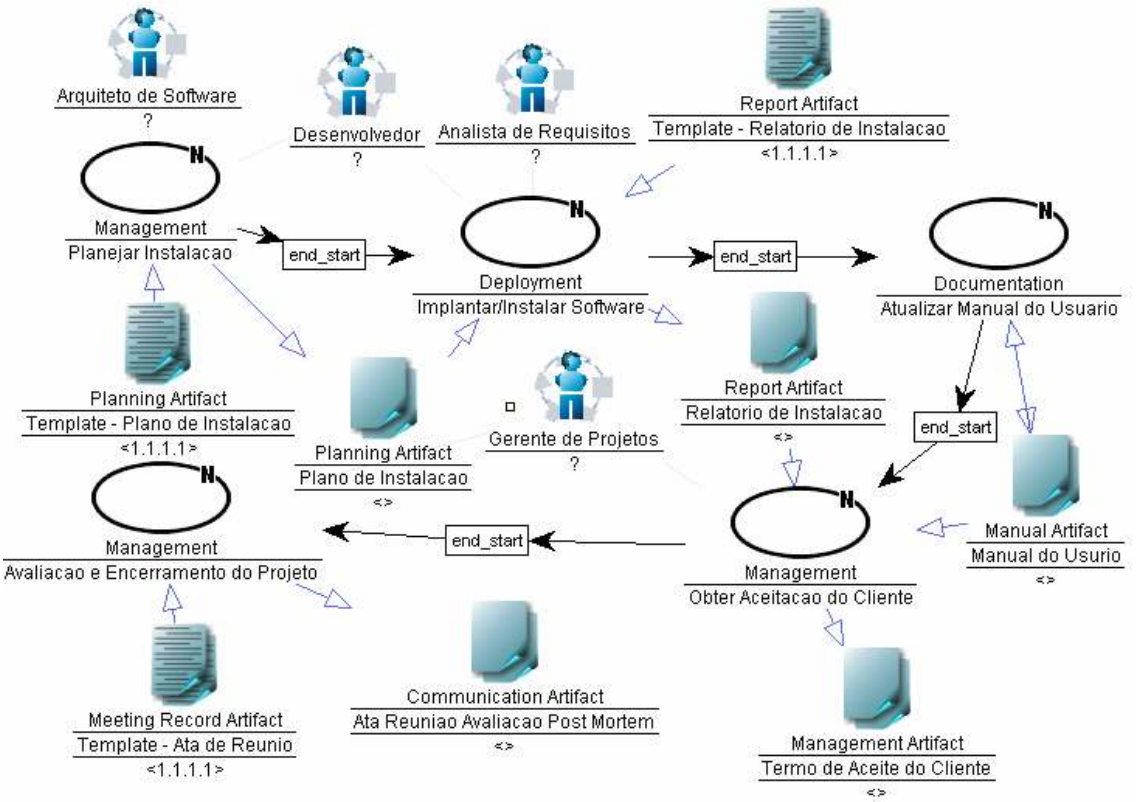

Figura 43 – Subprocesso *Implantação e Encerramento do Projeto* do *Template* LABES-UFPA.

6.1.1 Representação de Contextos do Template LABES-UFPA

Seguindo o processo de reuso proposto no presente trabalho, uma representação do contexto preliminar do projeto é elaborada com base nas propriedades relativas ao projeto específico e ao processo a ser utilizado. O *template* LABES-UFPA possui uma representação de contexto fictícia, conforme descrita na Tabela 10. Esta é a representação utilizada no cálculo de similaridade quando o *template* é submetido ao processo de recuperação.

Tabela 10 - Representação de contextos do *template* LABES-UFPA

Contexto	j	Atributo	Valor
Projeto	1	Modelo de Ciclo de Vida	Cascata
	2	Complexidade	Média
	3	Tamanho do Projeto	Pequeno
	4	Tamanho da Equipe	5 pessoas
	5	Prazo	3 meses
	6	Conhecimento em Engenharia de Software	Alto
	7	Paradigma de Desenvolvimento	OO
Processo	8	Modelo de Desenvolvimento	Híbrido
	9	Modelo de Maturidade	MPS.BR
	10	Nível de Maturidade	G
	11	Complexidade	Baixa
	12	Processo	-
	13	Experiência no uso de processos	Baixa

Nas Figuras 44 e 45 é ilustrado o ingresso dos dados referentes à representação de contextos preliminar de projeto e processo para este *template,* utilizando a interface gráfica do ambiente.

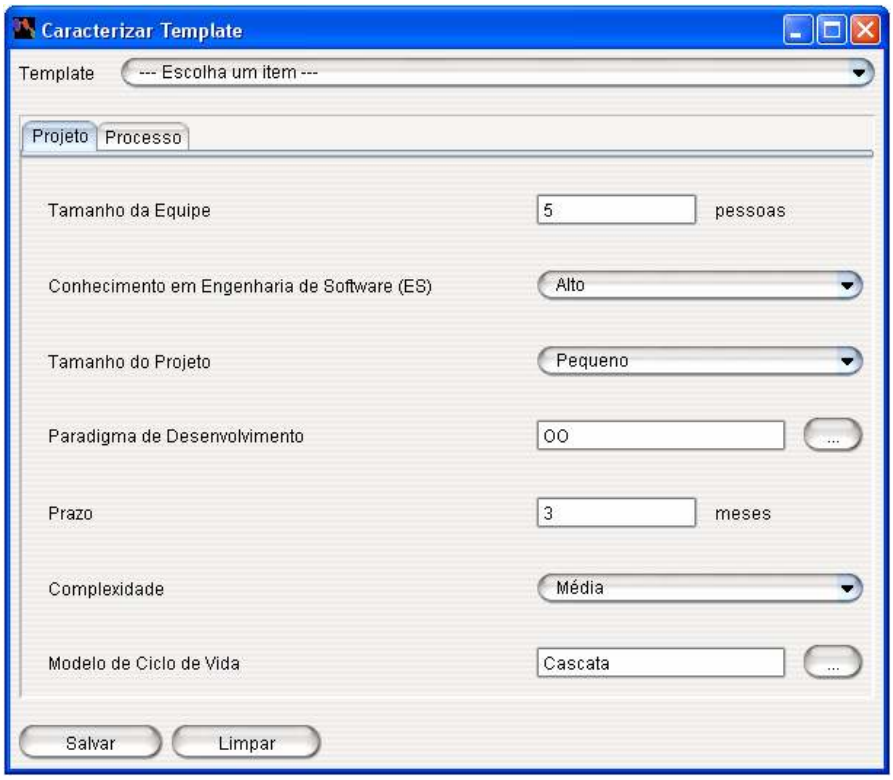

Figura 44 – Representação de contexto de projeto do *template* LABES-UFPA.

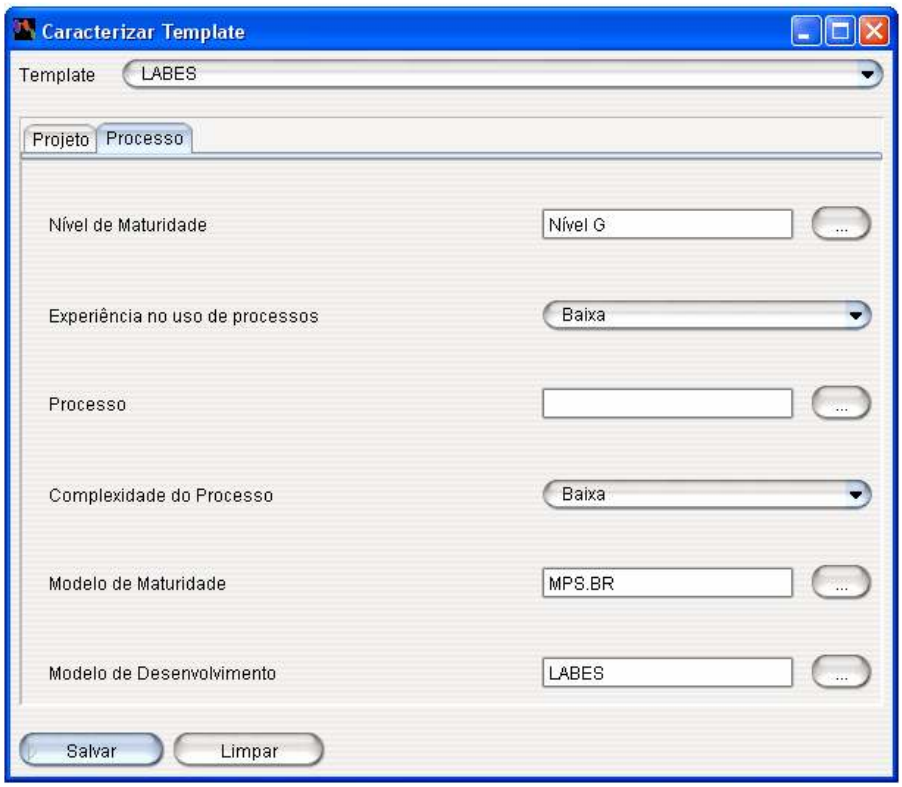

Figura 45 – Representação de contexto de processo do *template* LABES-UFPA.

6.2 Projeto SIGAP

Este projeto de software foi criado com o objetivo de desenvolver, implantar e manter o SIGAP utilizando recursos humanos, materiais e tecnológicos do LABES-UFPA e seguindo um processo de desenvolvimento aderente às práticas recomendadas pelo nível G do modelo MPS.BR [Lemos et al., 2009].

A primeira etapa deste projeto consiste no desenvolvimento de um protótipo para o Instituto de Ciências Biológicas (ICB) da UFPA, denominado módulo Formulário. Este módulo trata basicamente um formulário para o cadastro dos laboratórios do ICB e suas informações (informações gerais, equipamentos, equipes, serviços prestados, potencialidades e necessidades). Já a segunda etapa, refere-se à implementação dos módulos de Coleta e de Relatórios do SIGAP.

Cada etapa deste projeto adota um processo de software de acordo com as suas características, no entanto, por ser um processo de software do LABES-UFPA, seu principal requisito consiste em seguir um processo padrão do LABES-UFPA aderente ao nível G do MPS.BR. Por este motivo, as seções 6.2.1.1 e 6.2.2.1, as quais relatam os requisitos de cada processo a ser instanciado, apresentam como *template* mais similar o *template* LABES-UFPA.

6.2.1 Módulo Formulário

O foco desta primeira etapa é desenvolver um protótipo com o objetivo de determinar a viabilidade e construir uma visão clara sobre os requisitos do sistema, visto que, inicialmente, os clientes do SIGAP não tinham uma noção clara das suas necessidades [Lemos et al., 2009].

6.2.1.1 Recuperação de Processos de Software

A abordagem proposta no Capítulo 4 prevê a recuperação de *templates* do repositório, que possam ser adequados para utilização no novo contexto, propiciando a sua reutilização. Para realizar a recuperação de processos, é necessário o estabelecimento de requisitos em nível de projeto e processo. Estes requisitos correspondem aos atributos da representação de contexto proposta e permitem a visualização do *ranking* de *templates* mais similares à consulta.

Nesta simulação é considerado como principal requisito, o uso de um processo de desenvolvimento aderente às práticas recomendadas pelo nível G do modelo MPS.BR. Os outros atributos e a importância de cada um (peso), representadas

de maneira fictícia, estão listadas na Tabela 11, posteriormente, nas Figuras 46 e 47 são ilustradas as telas desenvolvidas para o ingresso destas informações.

Tabela 11 – Requisitos de para a recuperação de *templates* para o módulo Formulário

Contexto	j	Atributo	Valor	Peso
Projeto	1	Modelo de Ciclo de Vida	Cascata	Médio
	2	Complexidade	Baixa	Médio
	3	Tamanho do Projeto	Pequeno	Médio
	4	Tamanho da Equipe	8 pessoas	Baixo
	5	Prazo	2 meses	Alto
	6	Conhecimento em Engenharia de Software	Alto	Alto
	7	Paradigma de Desenvolvimento	OO	Alto
Processo	8	Modelo de Desenvolvimento	-	Baixo
	9	Modelo de Maturidade	MPS.BR	Alto
	10	Nível de Maturidade	G	Alto
	11	Complexidade	Baixa	Médio
	12	Processo	-	Baixo
	13	Experiência no uso de processos	Baixa	Alto

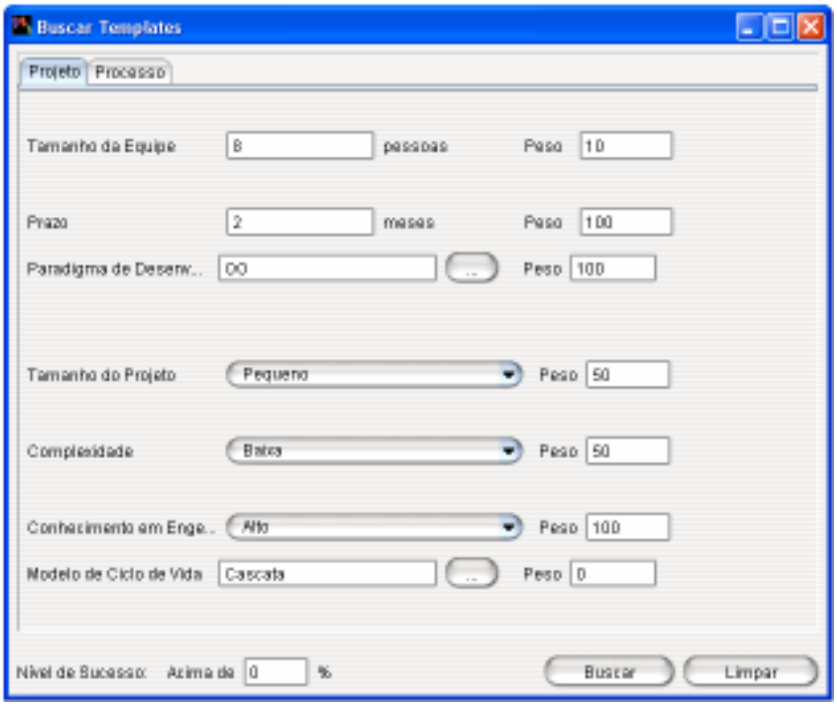

Figura 46 – Requisitos de contexto de projeto para o módulo Formulário.

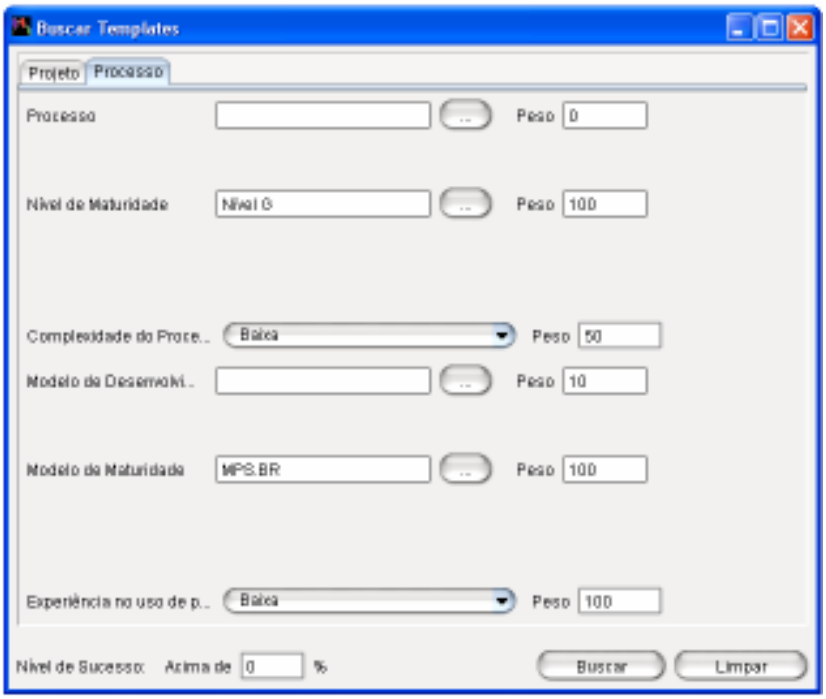

Figura 47 – Requisitos de contexto de processo para o módulo Formulário.

Ao executar o processo de recuperação proposto no Capítulo 4, o *template* considerado mais similar aparece no topo do *ranking* de *templates*. Trata-se do LABES-UFPA (Figura 48), o qual é instanciado no processo denominado SIGDCT. A Tabela 12 apresenta os valores de similaridade local de cada atributo, bem como o produto com o peso definido pelo usuário.

Tabela 12 – Cálculo de similaridade do caso-base LABES-UFPA e o caso-consulta

Contexto	**j**	**Atributo**	***sim*$_i$**	***sim*$_i$ x *peso***
Projeto	1	Modelo de Ciclo de Vida	1	50
	2	Complexidade	0,67	33,33
	3	Tamanho do Projeto	1	50
	4	Tamanho da Equipe	0,75	7,5
	5	Prazo	0,94	94
	6	Conhecimento em Engenharia de Software	1	100
	7	Paradigma de Desenvolvimento	1	100
Processo	8	Modelo de Desenvolvimento	0	0
	9	Modelo de Maturidade	1	100
	10	Nível de Maturidade	1	100
	11	Complexidade	1	50
	12	Processo	0	0
	13	Experiência no uso de processos	1	100
***SIM*(LABES-UFPA, caso-consulta)**				784,83

Figura 48 – Resultado da busca por *templates* similares aos requisitos do módulo Formulário.

6.2.1.2 Principais Alterações do Processo

Durante a execução do processo instanciado, surgem situações específicas do projeto que influenciam no seguimento do fluxo do processo. Estas situações aleatórias resultam em mudanças no processo que está sendo executado.

Adaptações em processos são necessárias para adequar o reuso de um modelo de processo e alcançar a definição de um processo apropriado às características do projeto ou da organização. Na simulação desta primeira etapa, os eventos que levam às alterações no processo são fictícios e aleatórios. A Tabela 13 lista os principais eventos ocorridos na simulação deste processo.

Tabela 13 – Lista de principais eventos do projeto A e suas causas e efeitos no processo

Evento	Categoria	Causa	Efeito
1	Eficiência	Necessidade de alcançar rapidamente a fase de construção do protótipo.	Reduzir as atividades da fase de planejamento com a remoção da atividade *Revisar Plano de Projeto* (Figura 48).
2	Eficiência	Necessidade de alcançar rapidamente a fase de construção do protótipo.	Remoção da atividade *Revisar matriz de rastreabilidade* do subprocesso *Análise de Requisitos*.
3	Requisito	Necessidade de avaliar a conclusão da fase de construção e testes do software e produzir um parecer sobre a viabilidade do projeto.	Inclusão de uma nova atividade no subprocesso *Construção e Testes do Software*, chamada *Finalizar Construção e Testes do Software*.
4	Requisito	Necessidade de correção do projeto do protótipo devido a um erro na interpretação de um requisito.	Inclusão de subprocesso chamado *Gerenciar Plano de Ação 1* na fase de *Projeto e Arquitetura do Software*.
5	Requisito	A identificação de uma mudança de requisito na última etapa do projeto.	A ação tomada para solucionar o problema foi a inclusão do subprocesso chamado *Gerenciar Plano de Ação 2* na fase de *Implantação e Encerramento do Projeto*.

A coluna *Categoria* corresponde ao tipo de evento e é especificada, conforme o diagrama de causas e efeitos da Figura 37, como Requisitos, Tamanho, Esforço, Eficiência, Periodicidade, Complexidade, Confiabilidade, Modularidade e Manutenibilidade. Já a coluna *Causa* apresenta o motivo que deu origem à alteração no processo, cada motivo foi estabelecido de acordo com as características desta etapa do projeto SIGAP. Por fim, a coluna *Efeito* apresenta a alteração (fictícia) realizada no processo.

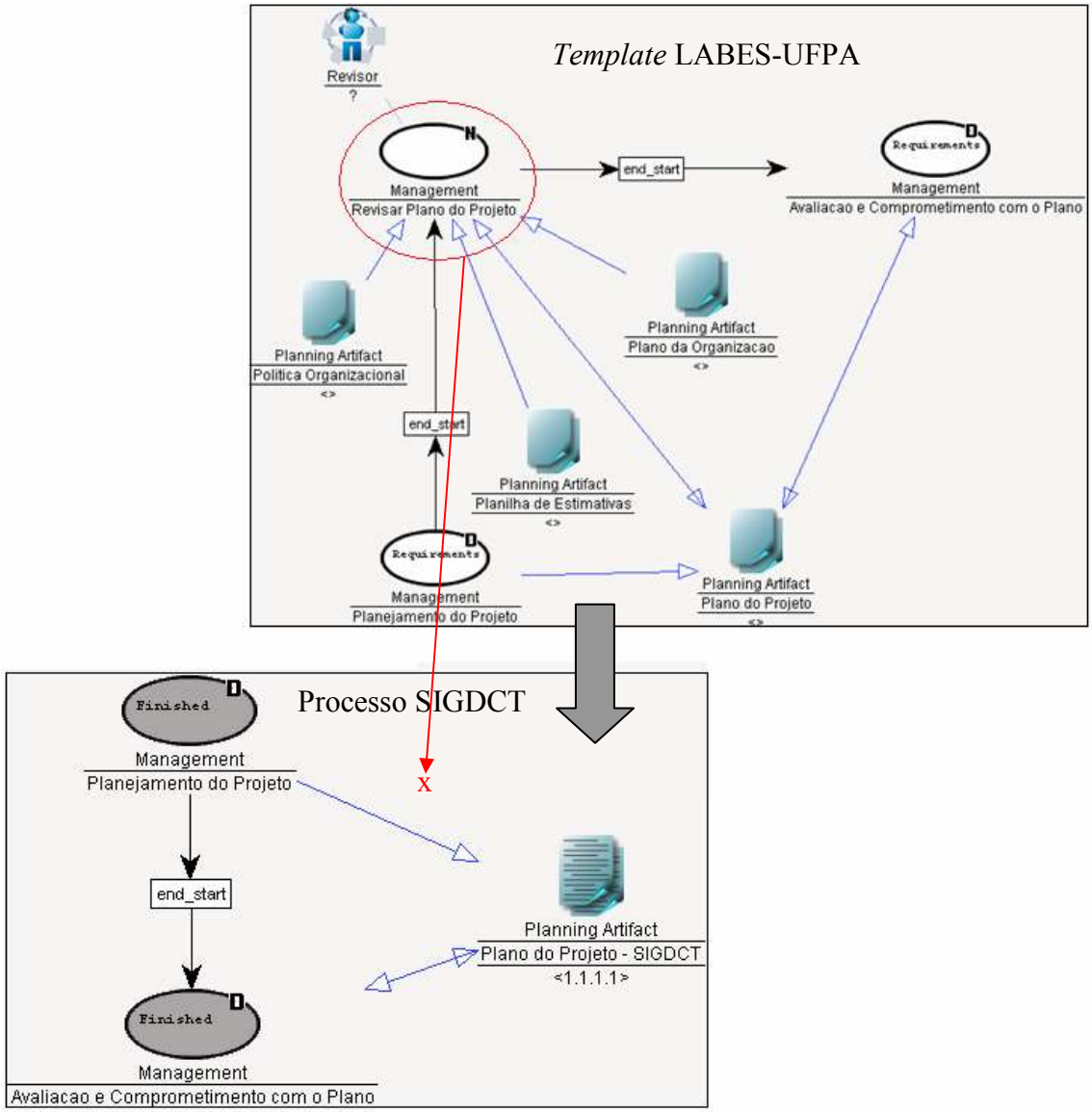

Figura 49 – Remoção da atividade *Revisar Plano de Projeto* do subprocesso *Planejamento*.

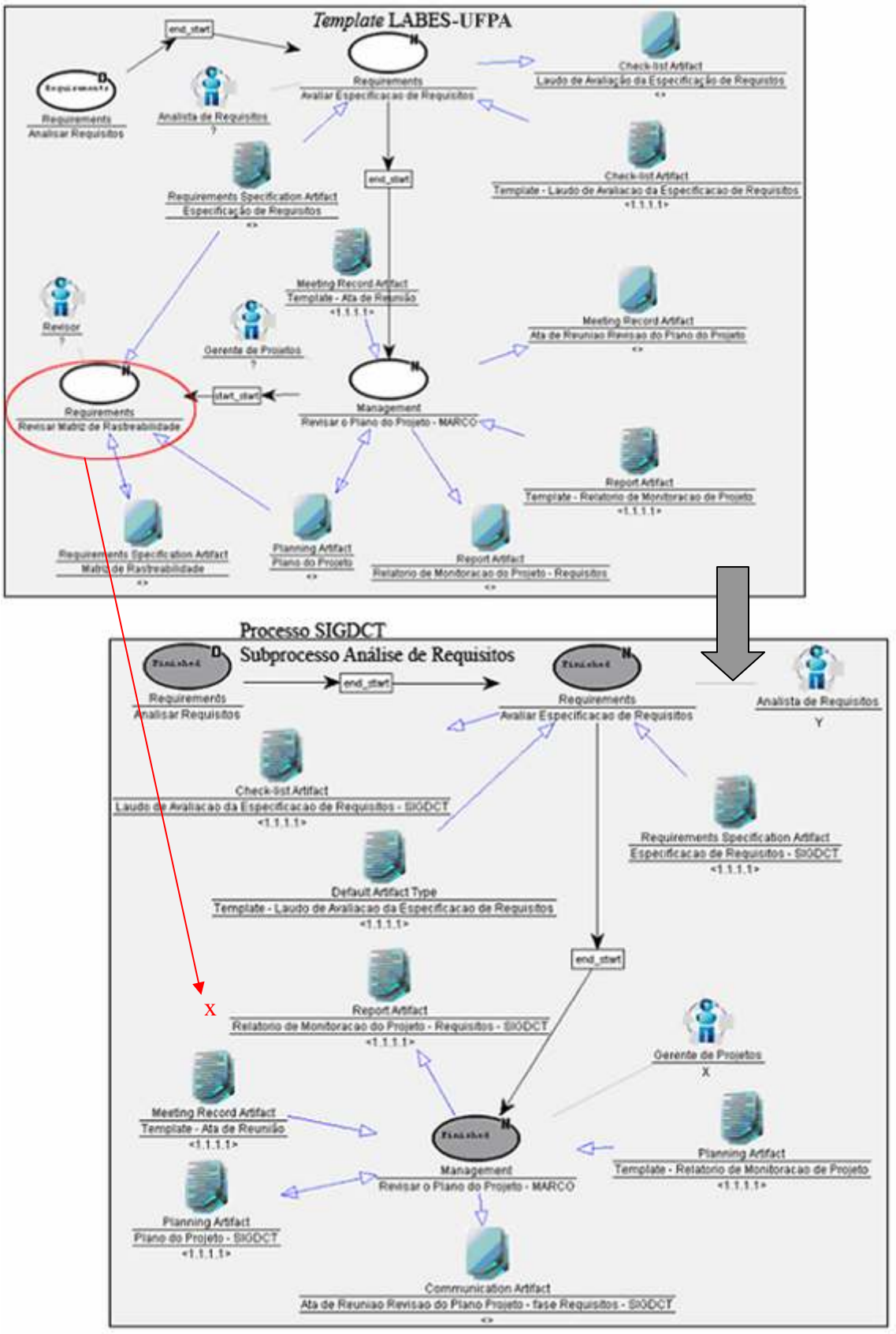

Figura 50 – Remoção da atividade *Revisar matriz de rastreabilidade* do subprocesso *Análise de Requisitos*.

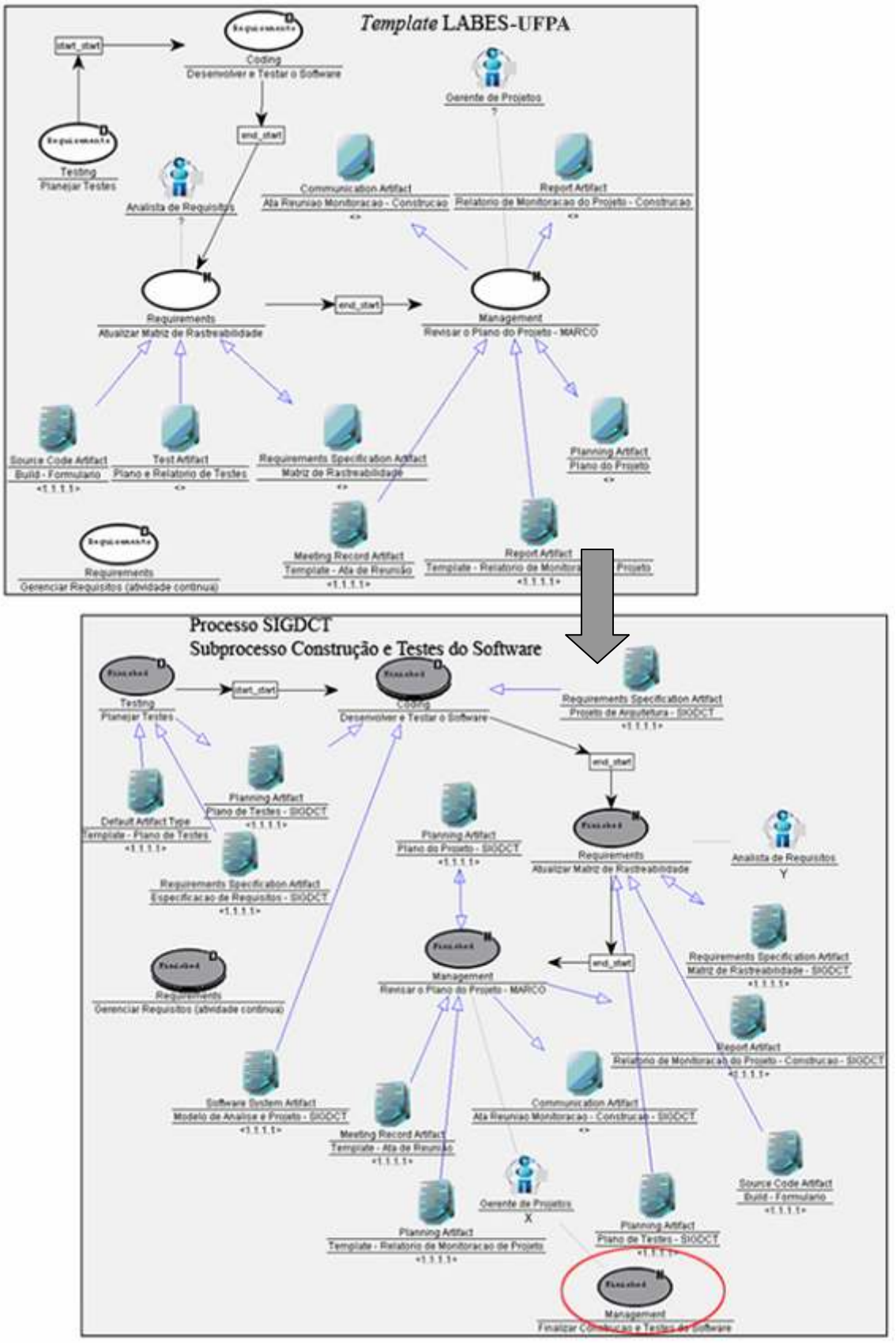

Figura 51 – Inclusão de uma nova atividade no subprocesso *Construção e Testes do Software*.

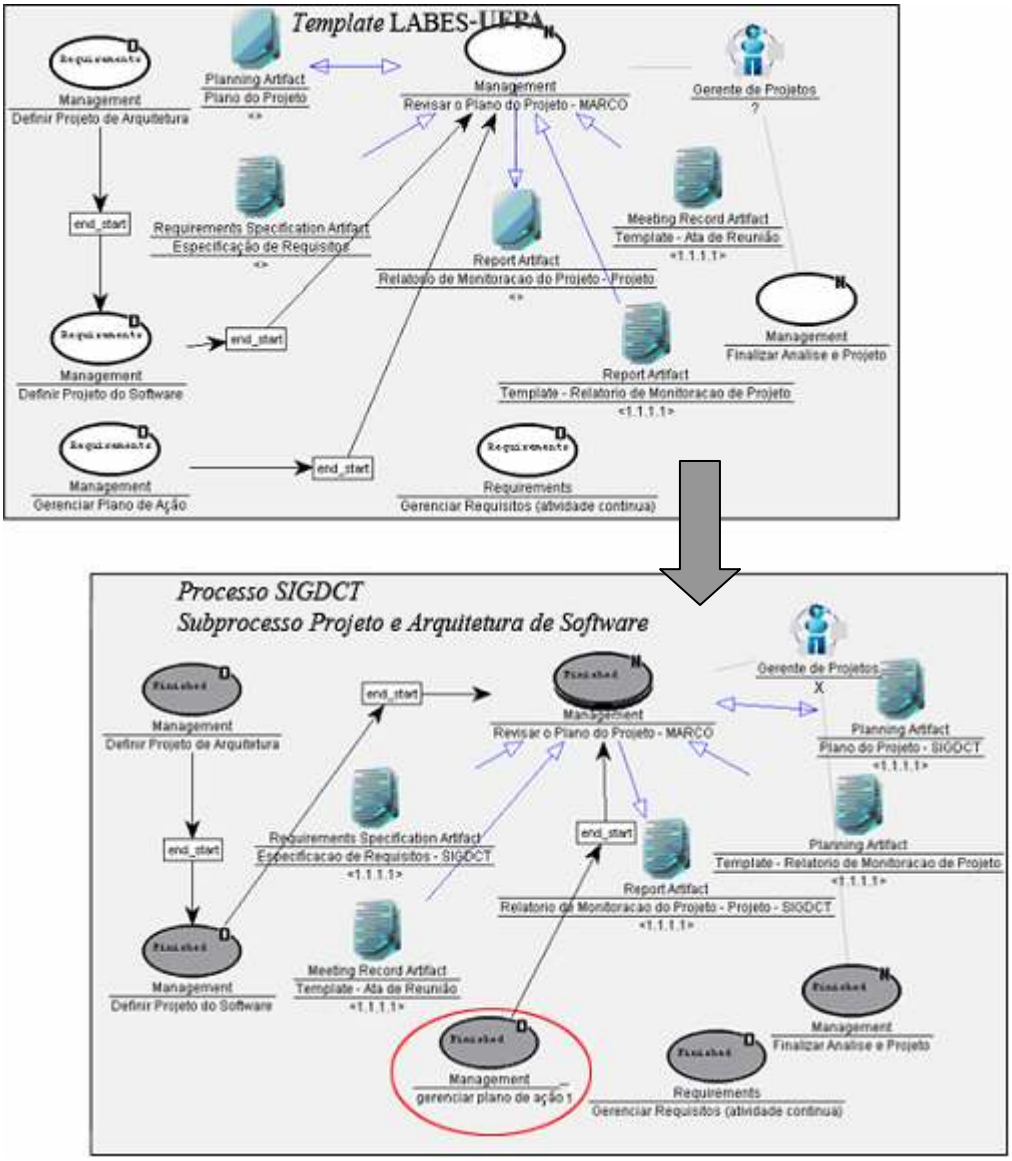

Figura 52 - Inclusão de subprocesso chamado “Gerenciar Plano de Ação 1” na fase de Projeto e Arquitetura do Software.

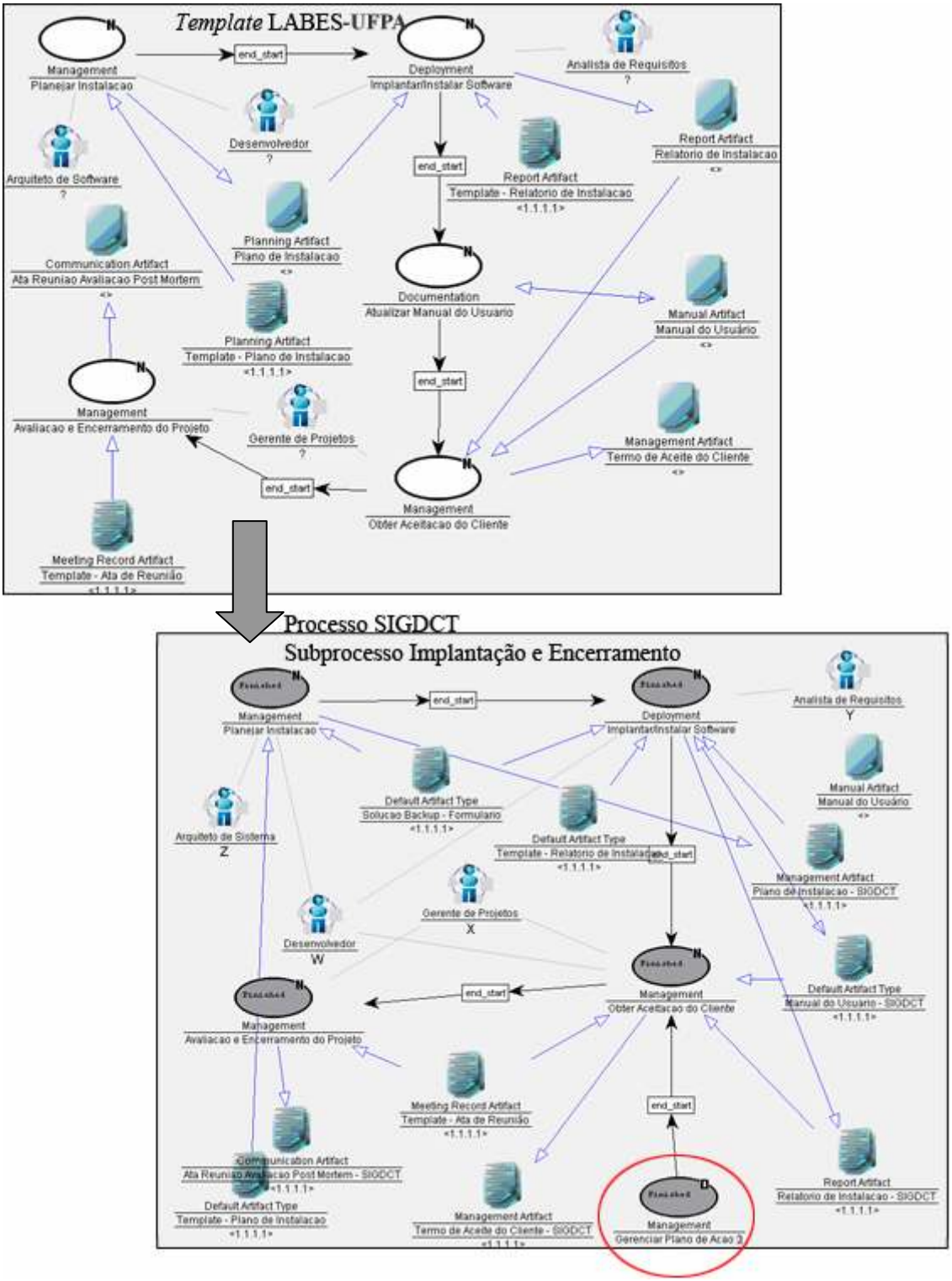

Figura 53 - Inclusão do subprocesso chamado “Gerenciar Plano de Ação 2” na fase de Implantação e Encerramento do Projeto.

6.2.1.3 Avaliação e Retenção de Processo Encerrado

Após a execução do processo, é fundamental avaliá-lo para que as alterações realizadas sirvam de aprendizado para a organização de software. Conforme o Capítulo 4, o passo seguinte consiste na avaliação do processo encerrado, e posteriormente, a

decisão sobre a retenção do mesmo. Na Figura 54, é ilustrado o processo executado, onde suas atividades estão destacadas de cinza no estado *finished* (concluídas), conforme notação WebAPSEE [LABES-UFPA, 2007].

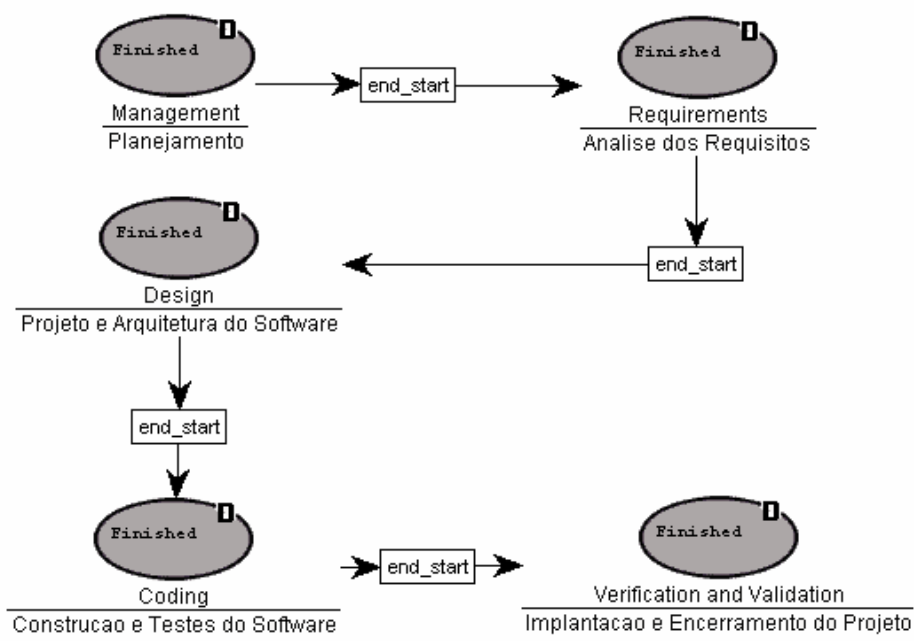

Figura 54 – Atividades decompostas do processo SIGDCT no estado *finished*.

De acordo com a Seção 4.4, para realizar a avaliação de processos encerrados, o usuário precisa fornecer a representação real de contextos do processo para subsidiar a comparação das similaridades globais (preliminar e real). A representação de contextos real é apresentada na Tabela 14 e também ilustrada nas Figuras 55 e 56.

Tabela 14 – Representação real de contextos do processo SIGDCT

Contexto	j	Atributo	Valor	Peso
Projeto	1	Modelo de Ciclo de Vida	Cascata	Médio
	2	Complexidade	Baixa	Médio
	3	Tamanho do Projeto	Pequeno	Médio
	4	Tamanho da Equipe	8 pessoas	Baixo
	5	Prazo	2 meses	Alto
	6	Conhecimento em Engenharia de Software	Alto	Alto
	7	Paradigma de Desenvolvimento	OO	Alto
Processo	8	Modelo de Desenvolvimento	-	Baixo
	9	Modelo de Maturidade	MPS.BR	Alto
	10	Nível de Maturidade	G	Alto
	11	Complexidade	Baixa	Médio
	12	Processo	-	Baixo
	13	Experiência no uso de processos	Baixa	Alto

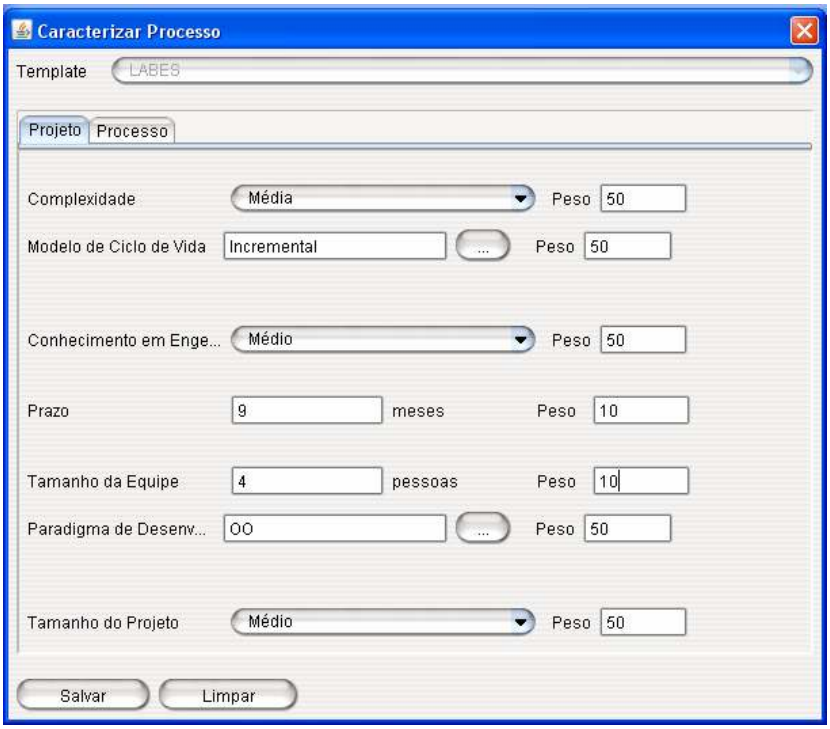

Figura 55 – Representação real de contexto de projeto do SIGDCT.

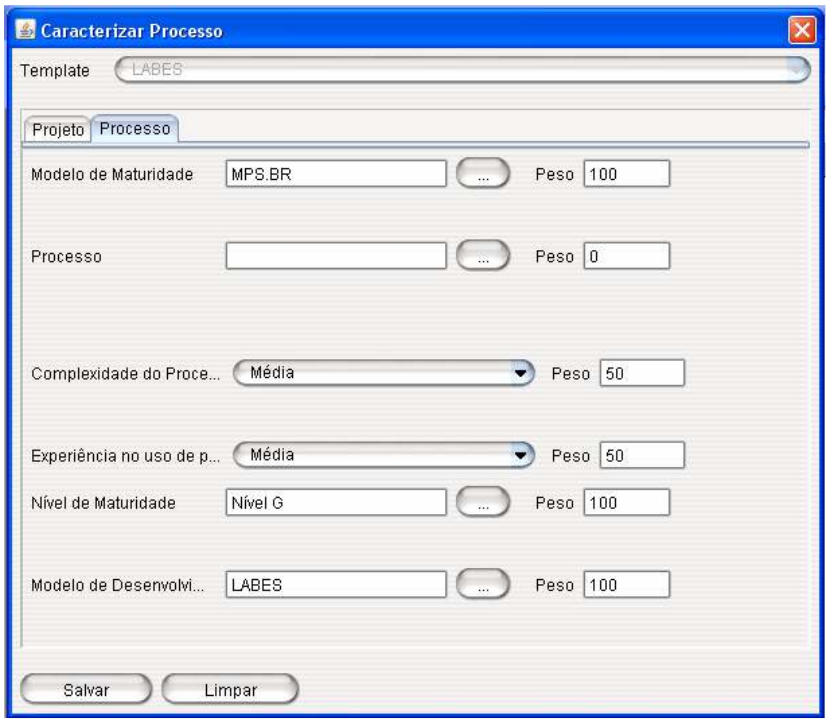

Figura 56 - Representação real de contexto de processo do SIGDCT.

A Tabela 15 apresenta os valores de similaridade local de cada atributo na representação de contextos real, bem como a sua multiplicação com o peso definido pelo usuário. Após o fornecimento da representação real dos contextos do processo SIGDCT (Tabela 14) e a realização do cálculo de similaridade global real deste processo com o *template* escolhido para o reuso.

Tabela 15 – Representação real de contextos do processo SIGDCT

Contexto	j	Atributo	sim_j	sim_j x *peso*
Projeto	1	Modelo de Ciclo de Vida	1	50
	2	Complexidade	0,67	33,33
	3	Tamanho do Projeto	1	50
	4	Tamanho da Equipe	0,75	7,5
	5	Prazo	0,94	94
	6	Conhecimento em Engenharia de Software	1	100
	7	Paradigma de Desenvolvimento	1	100
Processo	8	Modelo de Desenvolvimento	0	0
	9	Modelo de Maturidade	1	100
	10	Nível de Maturidade	1	100
	11	Complexidade	1	50
	12	Processo	0	0
	13	Experiência no uso de processos	1	100
***SIM*(LABES-UFPA, SIGDCT)**				784,83

A partir do cálculo foi obtido o mesmo valor da similaridade global preliminar (784,83), sendo possível calcular a comparação das similaridades globais, de acordo a Seção 4.4.1. Este cálculo é apresentado na fórmula (12) e também ilustrado na Figura 58.

$$CSG = \left(100 \times \frac{784{,}83}{784{,}83}\right) - 100 = 0 \tag{12}$$

Neste processo, conforme apresentado na Tabela 14, não houve mudança nos requisitos de processo preliminar e real, o que significa que, neste cenário, o *template* LABES-UFPA, sugerido inicialmente como o mais similar, continua no topo do *ranking* de *templates* para este contexto.

A Figura 57 ilustra o mapeamento da reutilização dos componentes das atividades do subprocesso Planejamento. Neste diagrama é possível verificar que 83 componentes, de um total de 100, foram reutilizados neste subprocesso. O cálculo do nível de reuso (Seção 4.4.2) utiliza esta contagem de componentes para prover um percentual de reutilização do *template* no processo.

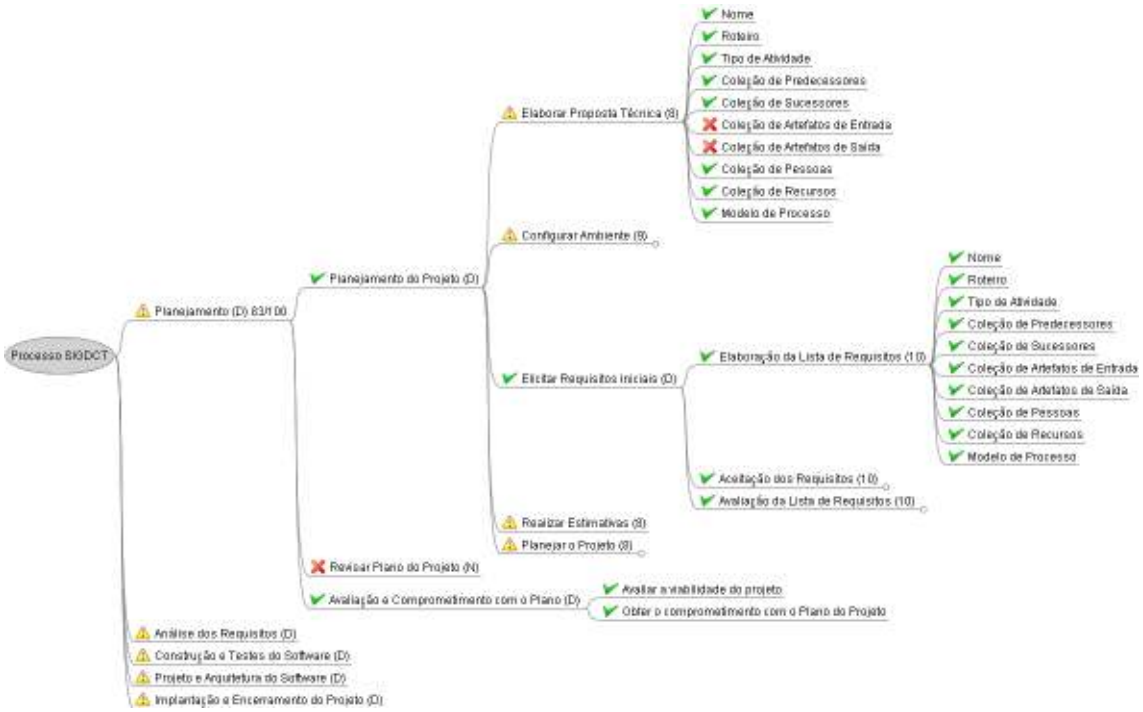

Figura 57 – Mapeamento da reutilização de componentes das atividades do subprocesso *Planejamento* (Figura 49).

O Nível de Reuso do *template* LABES-UFPA no processo SIGDCT é obtido pela fórmula abaixo:

$$NR = \frac{251}{36 \times 10} = 0{,}69722222 \tag{13}$$

De acordo com o resultado desta métrica (Figura 58), poucas são as alterações do processo em relação ao *template* LABES-UFPA, isto significa que a adaptação realizada no processo pouco influencia na reutilização do *template*.

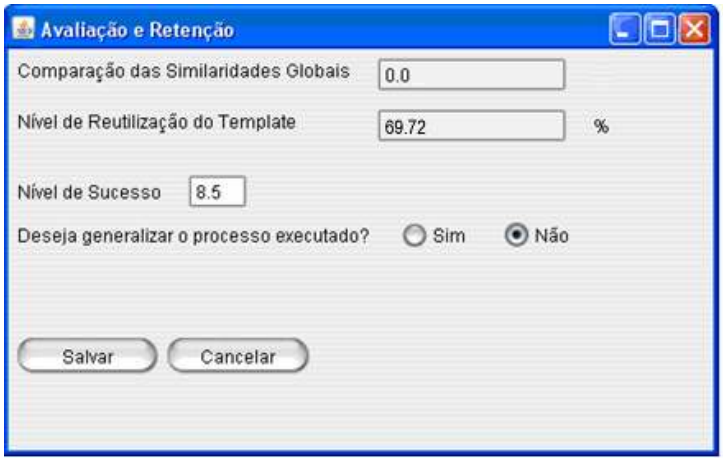

Figura 58 – Tela de avaliação e retenção do processo SIGDCT.

Considerando que o módulo teve sucesso satisfatório com a reutilização do *template* LABES-UFPA, o seu Nível de Sucesso resultou na nota apresentada na Figura

58 e a opção por não generalizar o processo, atribui esta nota à matriz de notas do *template*. De posse desta avaliação é possível decidir sobre a retenção do processo através de uma das seguintes ações: generalizá-lo (criar novo *template* ou nova versão de *template*), ou não (apenas registrar o nível de sucesso do *template* para o processo).

Considerando o bom desempenho do *template* no processo instanciado, a decisão de retenção deste processo consiste em não generalizá-lo para o cenário apresentado. Desta forma, a nota atribuída ao processo encerrado é incrementada na matriz de notas do *template* LABES-UFPA.

6.2.1.4 Análise dos Resultados

De acordo com as características do módulo e as poucas alterações demandadas no processo, percebe-se que o *template* escolhido contempla as atividades fundamentais para o desenvolvimento do protótipo com sucesso, as quais se configuram boas práticas. O protótipo serviu para aprimorar o entendimento dos requisitos por parte dos clientes, da mesma forma, serviu também para experimentar o *template* pela primeira vez nesta simulação.

O mais importante após a execução do processo SIGDCT, conforme apresentado na Figura 59, é que para as próximas buscas, o *template* LABES-UFPA apresentará um nível sucesso que antes não possuía. Este *feedback* sobre a execução deste *template* contribui para a sua escolha no futuro.

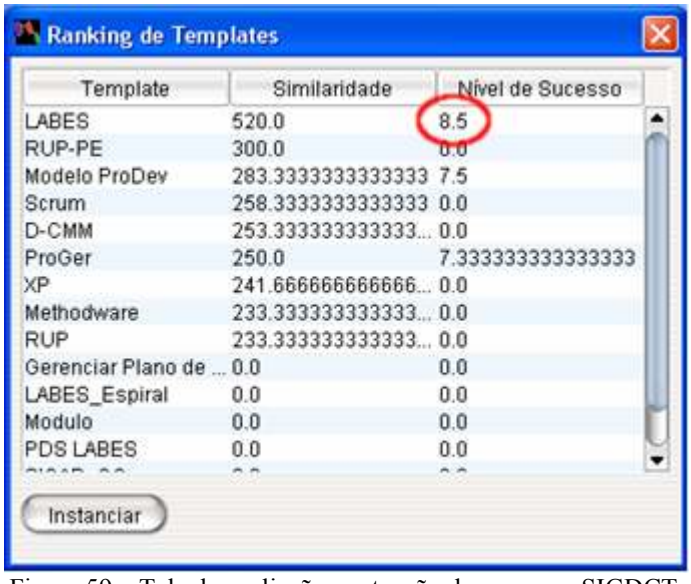

Figura 59 – Tela de avaliação e retenção do processo SIGDCT.

6.2.2 Módulos de Coleta e de Relatórios

Esta segunda etapa do projeto SIGAP consiste na continuação da etapa anterior. Enquanto o módulo Formulário foi utilizado unicamente como fonte de requisitos junto ao cliente, os módulos desta etapa correspondem ao sistema propriamente dito e tratam do controle de acesso, administração do sistema, gerenciamento do laboratório de pesquisa, gerenciamento do grupo de pesquisa, gerenciamento da rede de pesquisa e relatórios [Lemos et al., 2009].

6.2.2.1 Recuperação de Processos de Software

Esta etapa apresenta necessidades de processo diferenciadas da etapa anterior. Além de precisar ser aderente às práticas do nível G do MPS-BR, necessita realizar medições e aplicar práticas da engenharia de requisitos, do projeto arquitetural e revisões e testes. O ciclo de vida a ser utilizado é o iterativo incremental, devido o tamanho do projeto e da necessidade de verificações e monitorações em períodos mais curtos. A Tabela 16 apresenta os requisitos de processo desta etapa e a Tabela 17 apresenta o cálculo de similaridade em relação ao *template* LABES-UFPA. A Figura 60 ilustra o *ranking* de *templates* para o cenário apresentado.

Tabela 16 – Requisitos de processo dos módulos Coleta e Relatórios

Contexto	j	Atributo	Valor	Peso
Projeto	1	Modelo de Ciclo de Vida	Iterativo Incremental	Médio
	2	Complexidade	Média	Médio
	3	Tamanho do Projeto	Médio	Médio
	4	Tamanho da Equipe	7 pessoas	Baixo
	5	Prazo	18 meses	Alto
	6	Conhecimento em Engenharia de Software	Alto	Alto
	7	Paradigma de Desenvolvimento	OO	Alto
Processo	8	Modelo de Desenvolvimento	-	Baixo
	9	Modelo de Maturidade	MPS.BR	Alto
	10	Nível de Maturidade	G	Alto
	11	Complexidade	Média	Médio
	12	Processo	-	Baixo
	13	Experiência no uso de processos	Baixa	Alto

Tabela 17 – Cálculo de similaridade do caso-base LABES-UFPA e o caso-consulta.

Contexto	j	Atributo	sim_j	sim_j x *Peso*
Projeto	1	Modelo de Ciclo de Vida	0	0
	2	Complexidade	1	50
	3	Tamanho do Projeto	0,67	33,33
	4	Tamanho da Equipe	0,82	8,2
	5	Prazo	0,58	58
	6	Conhecimento em Engenharia de Software	1	100
	7	Paradigma de Desenvolvimento	1	100
Processo	8	Modelo de Desenvolvimento	0	0
	9	Modelo de Maturidade	1	100
	10	Nível de Maturidade	1	100
	11	Complexidade	0,67	33,33
	12	Processo	0	0
	13	Experiência no uso de processos	1	100
***SIM*(LABES-UFPA, caso-consulta)**				682,86

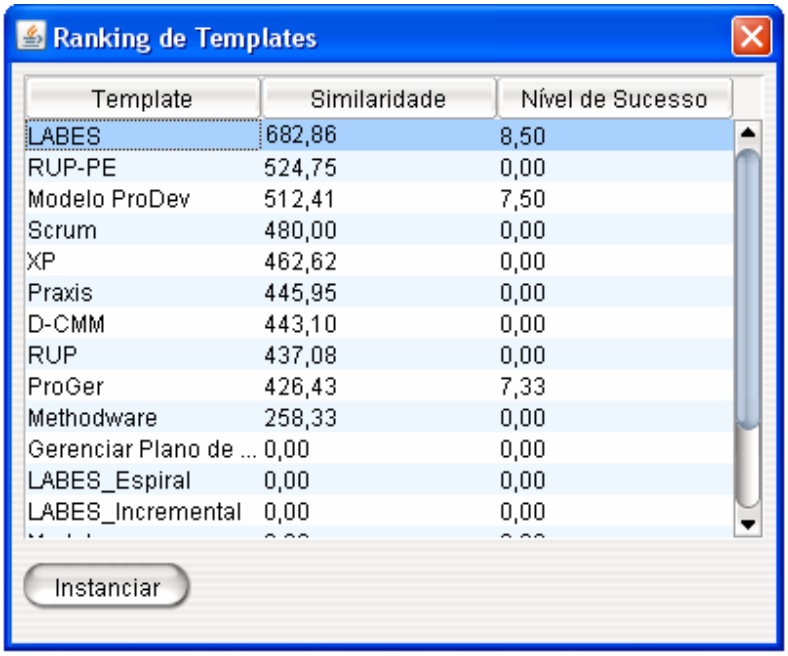

Figura 60 – Ranking de *templates* para os requisitos de processo dos módulos Coleta e Relatórios.

O *template* escolhido foi o LABES-UFPA por estar no topo do *ranking* de *templates*, instanciado no processo SIGAP Coleta. A gerência seguiu um processo definido para os módulos e o adaptou sempre que houve necessidade. As figuras 61 a 70 ilustram a modelagem deste processo.

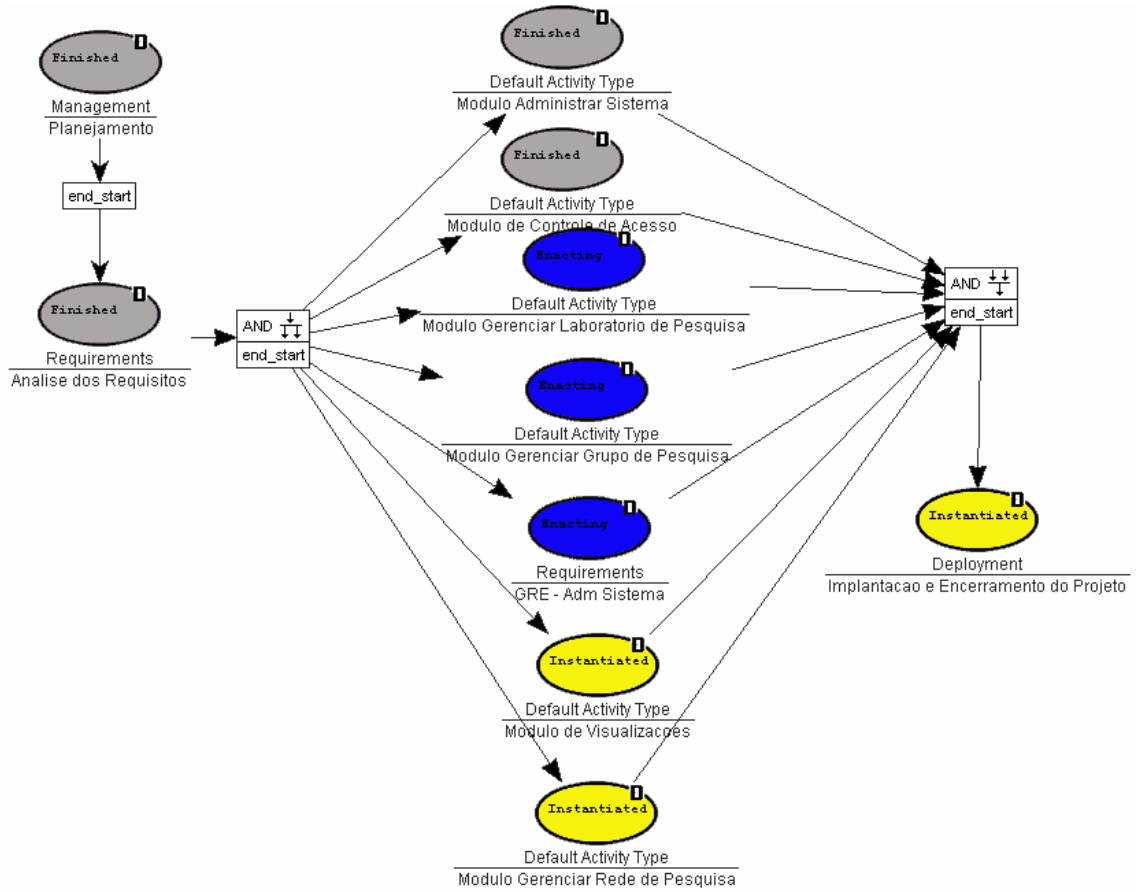

Figura 61 – Processo SIGAP Coleta em andamento [Lemos et al., 2009].

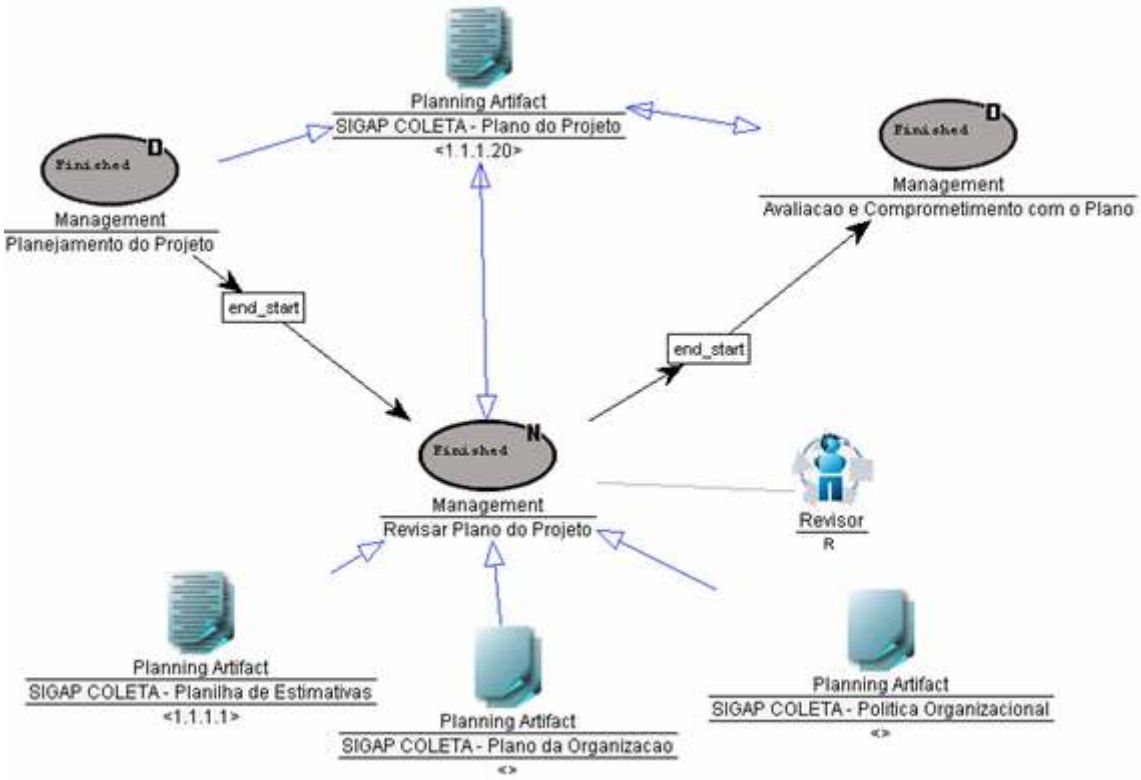

Figura 62 – Subprocesso *Planejamento* do processo SIGAP Coleta.

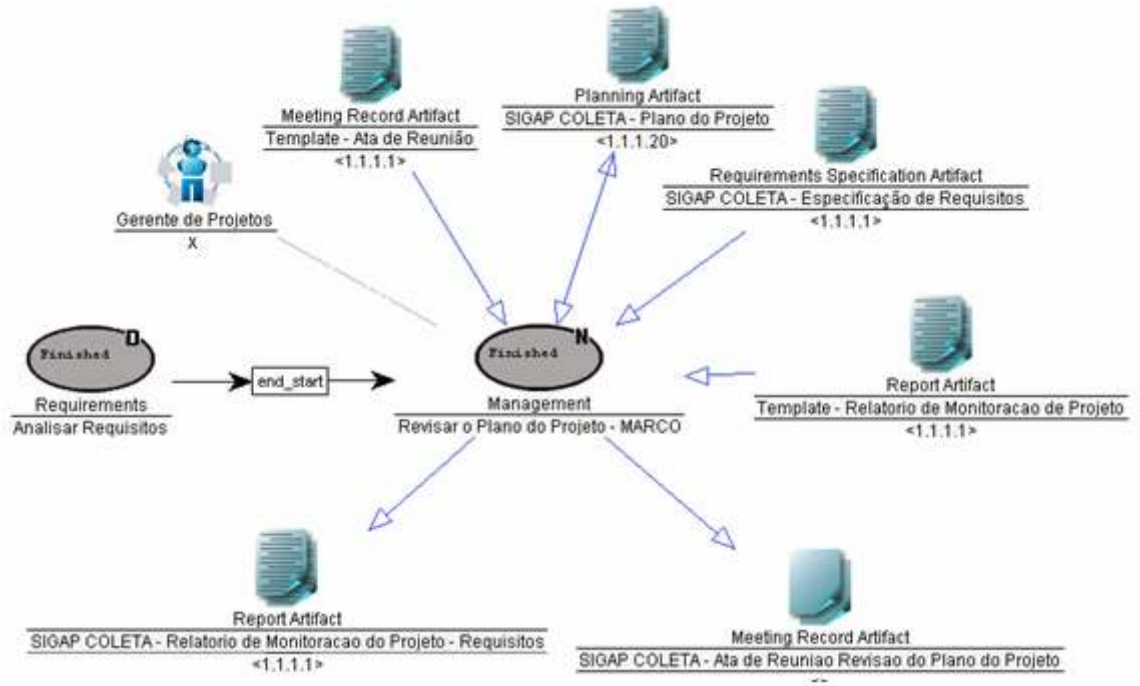

Figura 63 – Subprocesso *Análise dos Requisitos* do processo SIGAP Coleta.

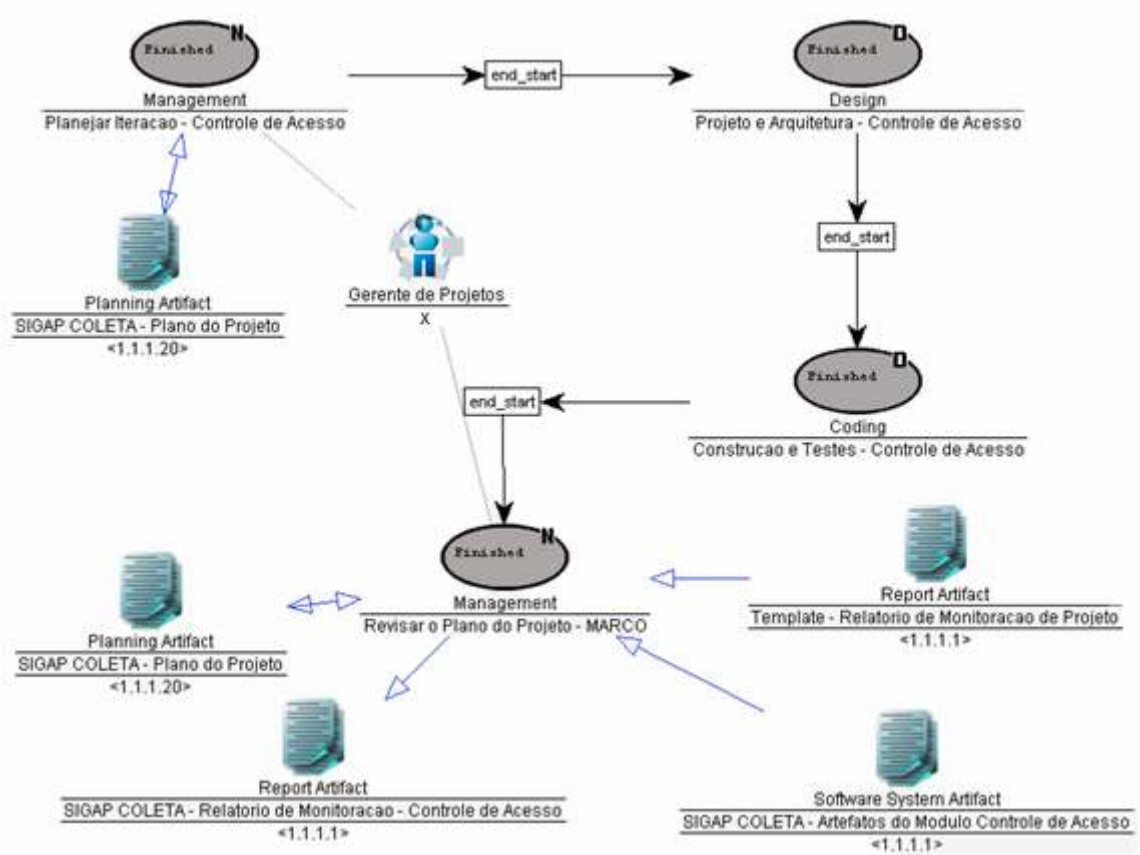

Figura 64 – Subprocesso *Módulo Controle de Acesso* do processo SIGAP Coleta.

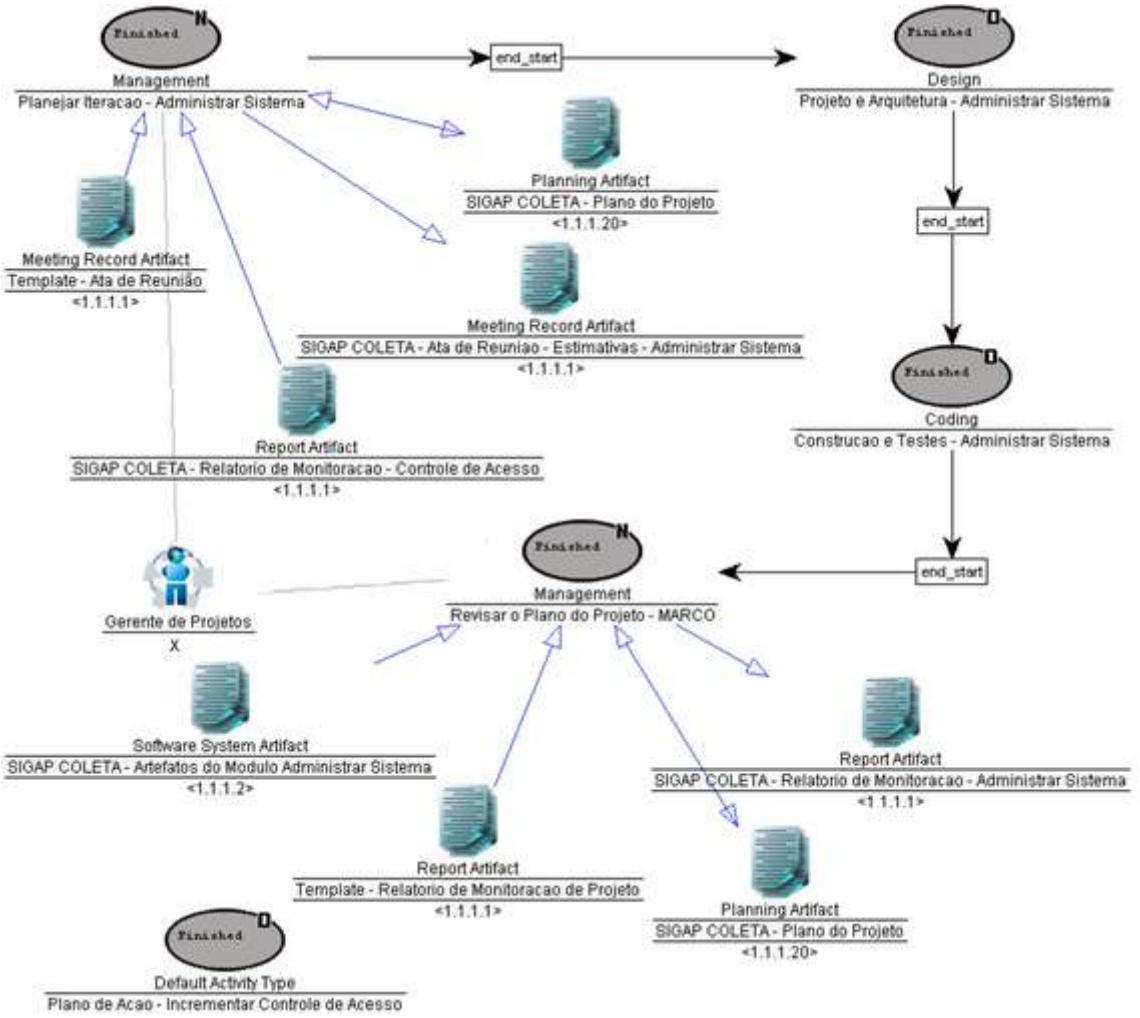

Figura 65 – Subprocesso *Módulo Administrar Sistema* do processo SIGAP Coleta.

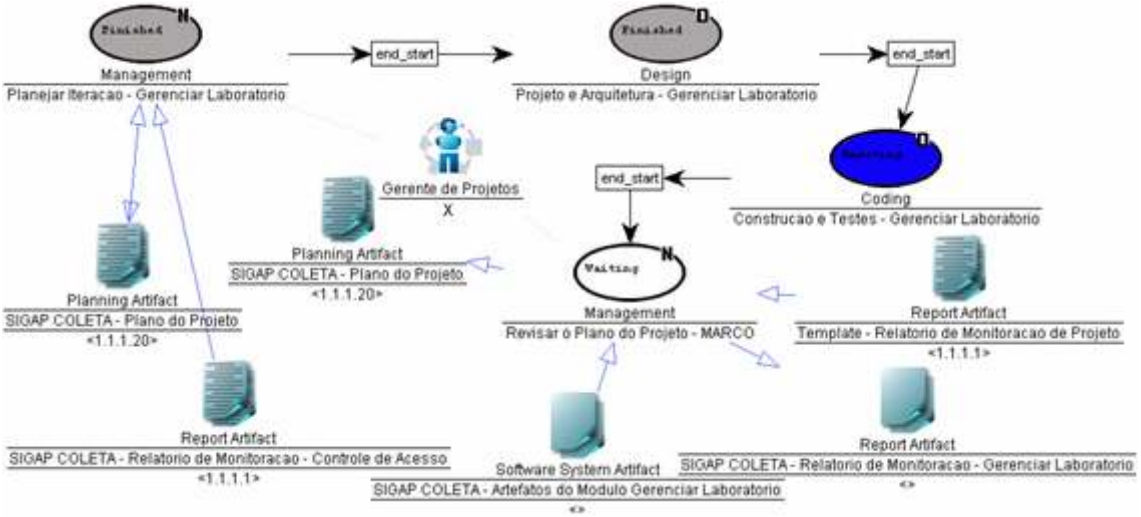

Figura 66 – Subprocesso *Módulo Gerenciar Laboratório de Pesquisa* do processo SIGAP Coleta.

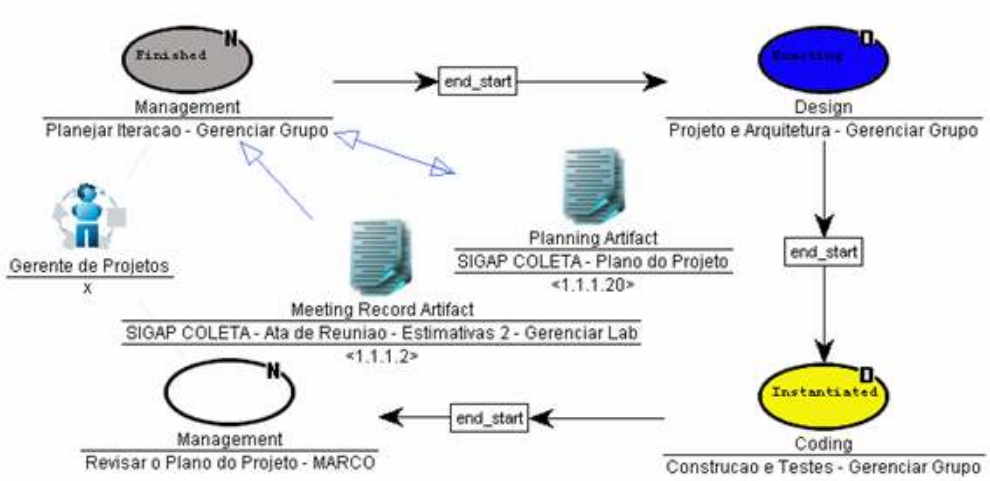

Figura 67 – Subprocesso Módulo Gerenciar Grupo de Pesquisa do processo SIGAP Coleta.

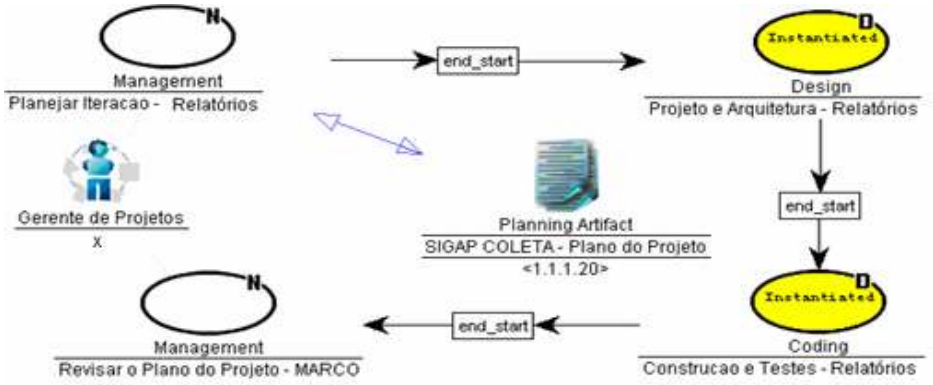

Figura 68 – Subprocesso *Módulo de Visualizações do processo* SIGAP Coleta.

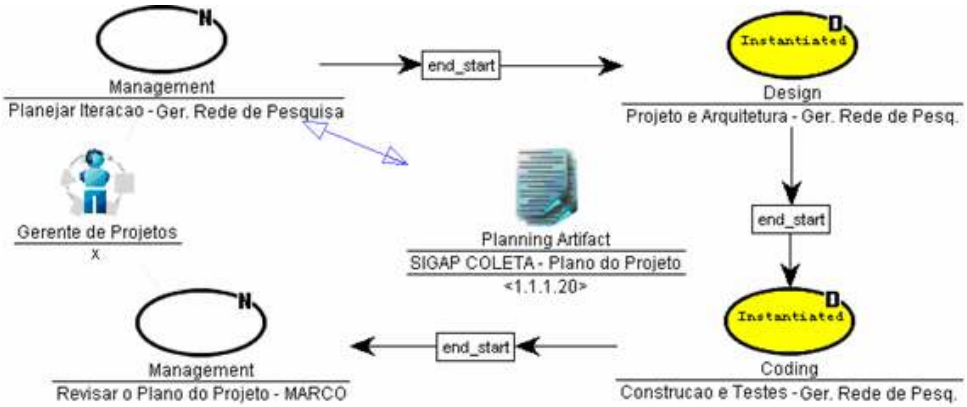

Figura 69 – Subprocesso Módulo de Gerenciar Rede de Pesquisa do processo SIGAP Coleta.

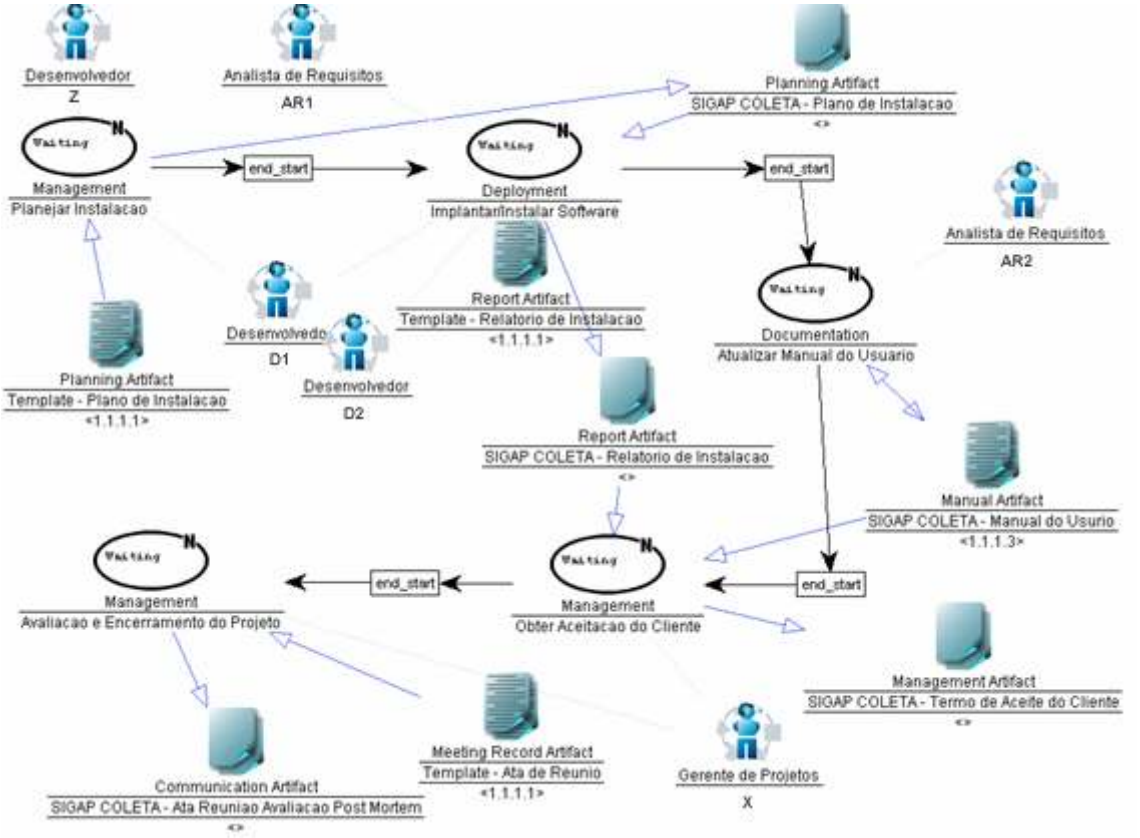

Figura 70 – Subprocesso Implantação e Encerramento do processo SIGAP Coleta.

6.2.2.2 Principais Alterações do Processo

Na simulação deste processo em específico, as adaptações necessárias apresentam como conseqüência alterações no processo de grandes proporções. Na Tabela 18 são listados os principais eventos, adaptados de [Lemos et al., 2009], que desencadeiam estas mudanças.

Tabela 18 – Lista de principais eventos do processo SIGAP Coleta e suas causas e efeitos adaptados de [Lemos et al., 2009]

Evento	Categoria	Causa	Efeito
1	Requisitos	Devido à alta complexidade dos requisitos e do seu pouco entendimento pelo cliente, a especificação de requisitos tornou-se muito demorada.	Resultou em mudanças significativas e contínuas em todos os módulos do sistema. A principal mudança foi a alteração do ciclo de vida do processo instanciado do *template* LABES-UFPA para iterativo incremental, ocasionando a criação de atividades decompostas (subprocessos) para cada módulo: Administrar Sistema, Controle de Acesso, Gerenciar Laboratório de Pesquisa, Gerenciar Grupo de Pesquisa, Gerenciar Rede de Pesquisa e Relatórios (Visualizações), conforme Figura 61.
2	Esforço	Devido à mudança do ciclo de vida para iterativo e incremental, cada módulo precisa realizar seu planejamento.	Criação da atividade "Planejar Iteração" em cada módulo (Figuras 64 a 69).
3	Eficiência	Com a alteração realizada no Evento 1, o detalhamento dos requisitos precisa ocorrer dentro de cada módulo.	Inclusão da atividade "Detalhar Requisitos" no subprocesso Projeto e Arquitetura de cada módulo (Figura 71).
4	Tamanho	De acordo com os insumos das métricas de desvios de prazos, observa-se que as estimativas de codificação são irreais, visto que o insumo utilizado é uma lista de requisitos iniciais que contém apenas requisitos em alto nível.	Necessidade de melhorar a estimativa de codificação, com isto, a atividade "Realizar Estimativas" que antes ficava no Subprocesso Planejamento do Projeto foi modelada para acontecer após o detalhamento dos requisitos, subprocesso Projeto e Arquitetura de cada módulo (Figura 72).
5	Tamanho	Mesmo após a mudança ocorrida no Evento 2, ainda assim houve necessidade de alteração das estimativas após a definição do projeto de arquitetura e a definição do projeto de interface dos módulos "Administrar Sistema" e "Gerenciar Laboratório de Pesquisa".	Criação de uma atividade decomposta denominada Plano de Ação para ajuste de estimativas e prazo do cronograma (Figuras 73 e 74).
6	Requisito	Devido à criação de módulos para as iterações e a inclusão do planejamento de cada iteração (Evento 2), as revisões de planejamento não precisam mais ocorrer em cada subprocesso.	Inclusão da atividade "Revisar Plano de Projeto" ao final de cada módulo (Figura 75).

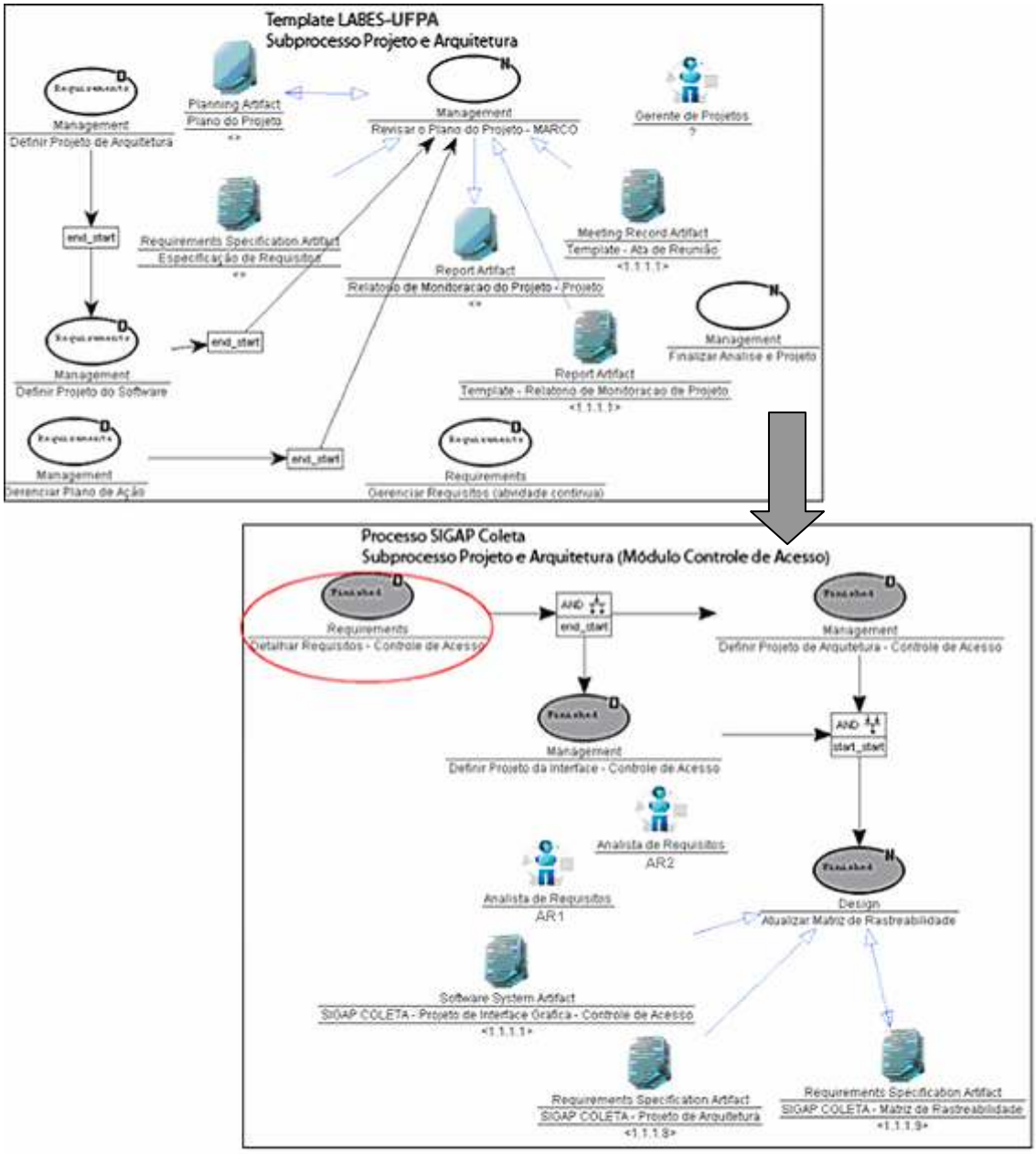

Figura 71 – Inclusão da atividade “Detalhar Requisitos” no subprocesso *Projeto e Arquitetura* de cada módulo.

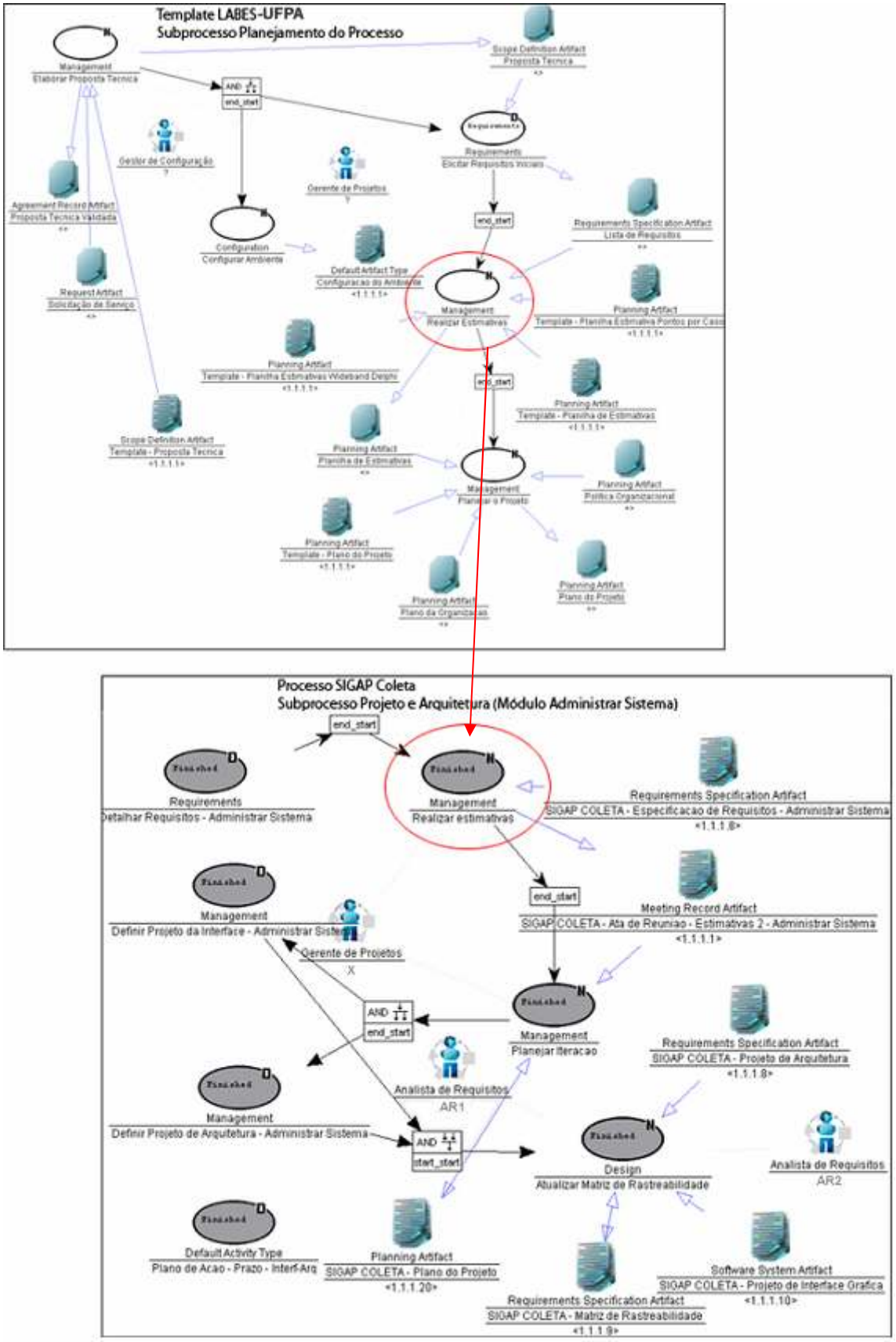

Figura 72 – Inclusão da atividade Realizar Estimativas após o detalhamento dos requisitos.

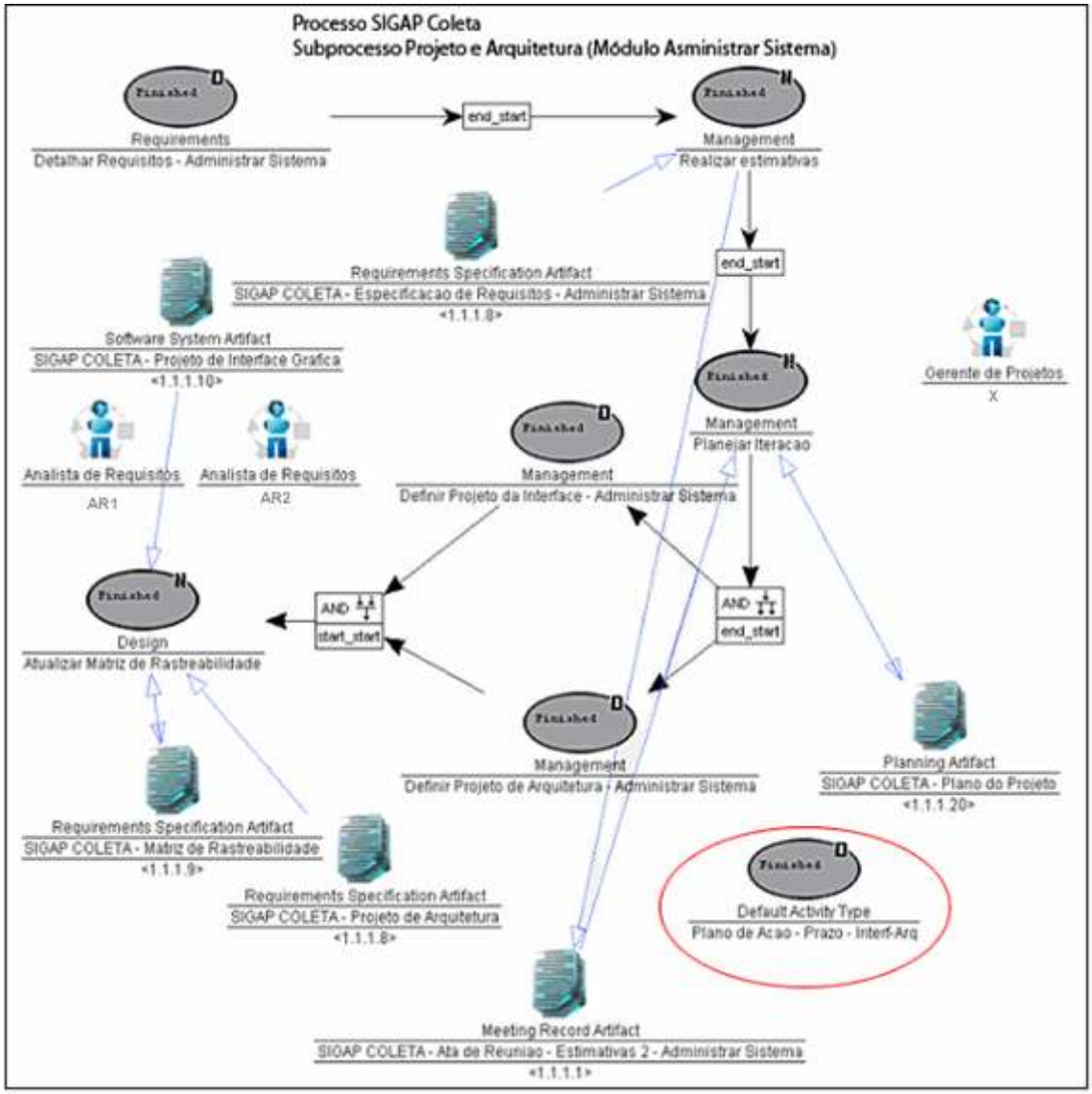

Figura 73 – Inclusão da atividade decomposta para ajuste de prazo no subprocesso Projeto e Arquitetura do módulo Administrar Sistema.

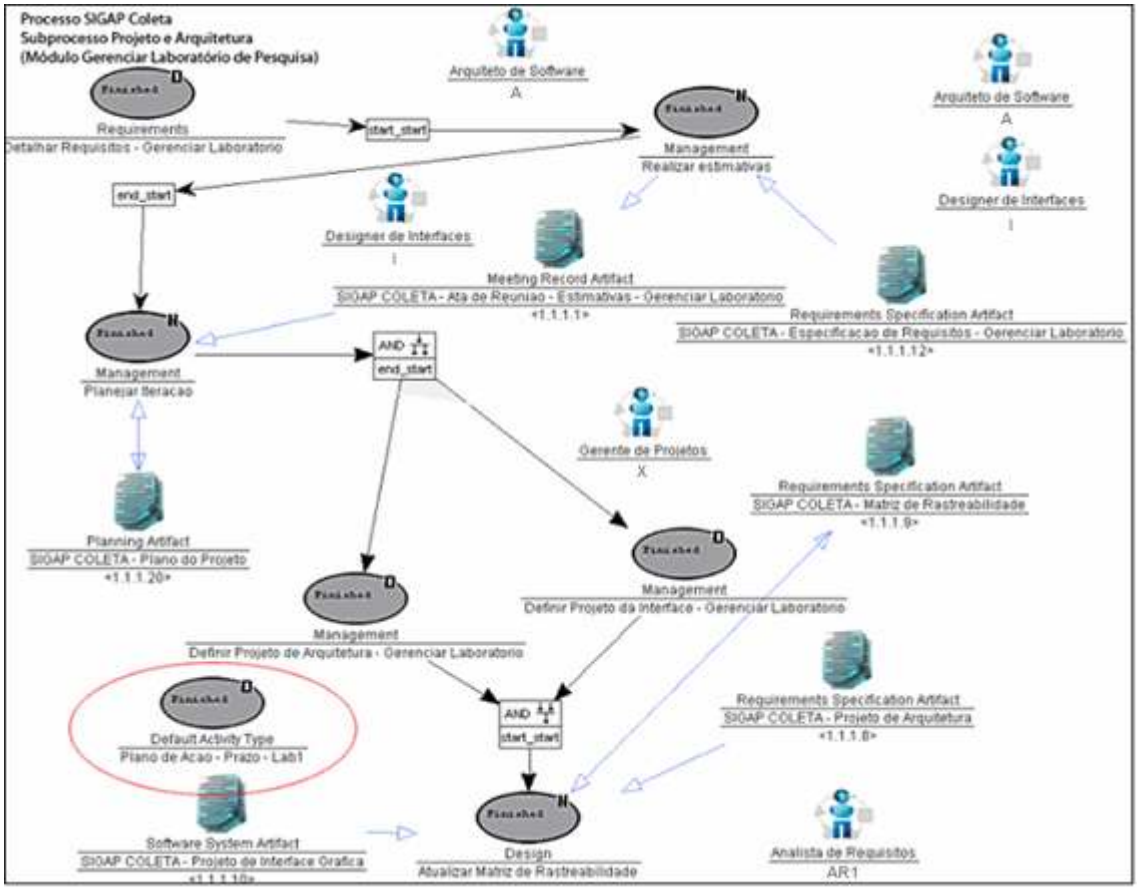

Figura 74 – Inclusão da atividade decomposta para ajuste de prazo no subprocesso Projeto e Arquitetura do módulo Gerenciar Laboratório de Pesquisa.

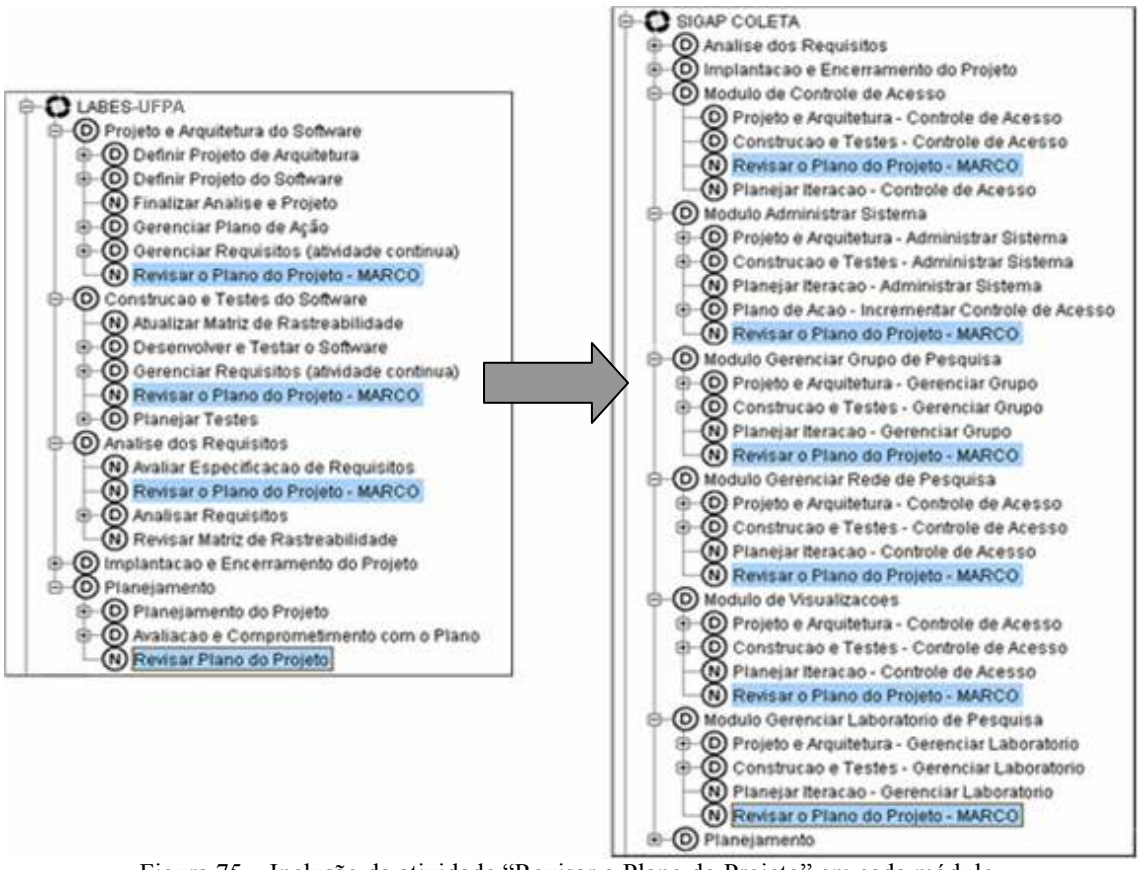

Figura 75 – Inclusão da atividade “Revisar o Plano do Projeto” em cada módulo.

6.2.2.3 Avaliação e Retenção de Processo Encerrado

Este processo, em especial, ainda encontra-se em andamento (não concluído), portanto sua avaliação será realizada dentro da modelagem inicialmente definida para a sua execução.

As mudanças são evidentes neste processo, o que se percebe é que temos um novo processo frente ao *template* reutilizado. É fundamental avaliá-lo para que as alterações realizadas sirvam de aprendizado para a organização de software.

De acordo com a Seção 4.4, para realizar a avaliação, o usuário precisa fornecer a representação real de contextos do processo para subsidiar a comparação das similaridades globais (preliminar e real). Estas informações são apresentadas na Tabela 19.

Tabela 19 – Representação real de contextos do processo SIGAP Coleta

Contexto	j	Atributo	Valor	Peso
Projeto	1	Modelo de Ciclo de Vida	Iterativo Incremental	Médio
	2	Complexidade	Alta	Médio
	3	Tamanho do Projeto	Grande	Médio
	4	Tamanho da Equipe	7 pessoas	Baixo
	5	Prazo	18 meses	Alto
	6	Conhecimento em Engenharia de Software	Alto	Alto
	7	Paradigma de Desenvolvimento	OO	Alto
Processo	8	Modelo de Desenvolvimento	-	Baixo
	9	Modelo de Maturidade	MPS.BR	Alto
	10	Nível de Maturidade	G	Alto
	11	Complexidade	Média	Médio
	12	Processo	-	Baixo
	13	Experiência no uso de processos	Baixa	Alto

Após o fornecimento da representação real dos contextos do processo SIGAP Coleta (Tabela 19) e a realização do cálculo de similaridade global real deste processo com o *template* escolhido para o reuso (Tabela 20) é possível calcular a comparação das similaridades globais (*CSG*), de acordo com a Seção 4.4.1. Este cálculo é apresentado na fórmula (14).

Tabela 20 – Representação real de contextos do processo SIGAP Coleta

Contexto	j	Atributo	sim_j	sim_j x *peso*
Projeto	1	Modelo de Ciclo de Vida	0	0
	2	Complexidade	0,67	33,33
	3	Tamanho do Projeto	0,67	33,33
	4	Tamanho da Equipe	0,82	8,2
	5	Prazo	0,58	58
	6	Conhecimento em Engenharia de Software	1	100
	7	Paradigma de Desenvolvimento	1	100
Processo	8	Modelo de Desenvolvimento	0	0
	9	Modelo de Maturidade	1	100
	10	Nível de Maturidade	1	100
	11	Complexidade	0,67	33,33
	12	Processo	0	0
	13	Experiência no uso de processos	1	100
***SIM*(LABES-UFPA, SIGDCT)**				666,19

$$CSG = \left(100 \times \frac{666,19}{682,86}\right) - 100 = 97,55 - 100 = -2,45 \quad (14)$$

Neste processo, conforme apresentado no resultado da fórmula (14), houve mudança, mesmo sendo pouca, nos requisitos de processo preliminar e real, neste cenário, o *template* LABES-UFPA, sugerido inicialmente como o mais similar, poderia ser substituído por um outro *template* mais apropriado às suas características.

O Nível de Reuso (*NR*) do *template* LABES-UFPA no processo SIGAP Coleta é obtido pela fórmula abaixo:

$$NR = \frac{95}{36 \times 10} = 0,26388888 \quad (15)$$

O resultado desta métrica (Figura 76) representa pouca reutilização do *template*, isto significa que a adaptação realizada no processo influenciou na reutilização do *template*.

Considerando o sucesso satisfatório do processo após a sua adaptação, o seu Nível de Sucesso resultou na nota apresentada na Figura 76 e a opção por generalizar o processo para a criação de um novo *template*. Esta opção é importante, pois permite que o processo se transforme em um *template*. Assim como são atribuídas a representação de contextos real e o nível de sucesso a este novo *template*, denominado LABES_Incremental. A Figura 77 ilustra o macro-processo deste novo *template,* evidenciando a inclusão do subprocesso *Módulo*.

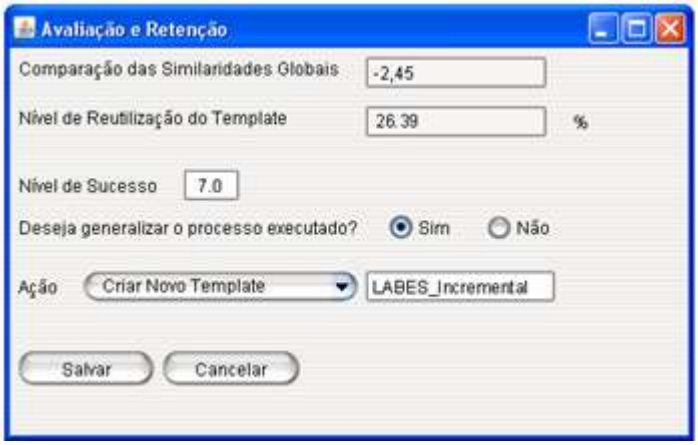

Figura 76 – Tela de avaliação e retenção do processo SIGAP Coleta.

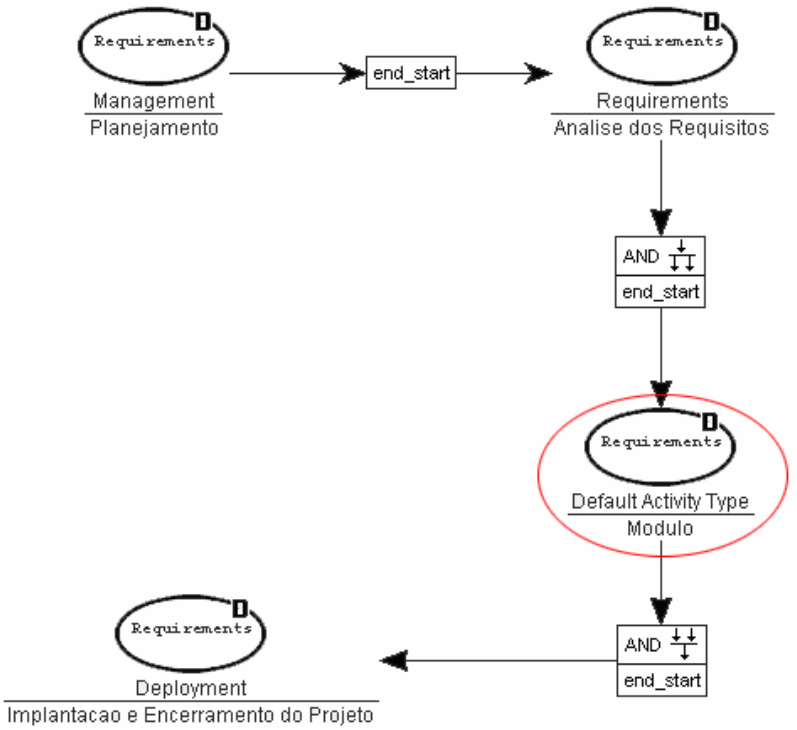

Figura 77 – *Template* LABES_Incremental.

6.2.2.4 Análise dos Resultados

Nesta etapa da simulação, optou-se por reutilizar o mesmo processo padrão, o *template* LABES-UFPA, e adaptou-o sempre que houve necessidade. A cada ciclo é possível avaliar o processo executado e assim, o projeto se beneficia das melhorias ainda durante sua execução.

As alterações no processo foram tão evidentes que a generalização deste processo foi imprescindível para a construção do conhecimento organizacional para este cenário de projeto. Houve muita mudança de um processo para o outro, por isto, já que o processo executado foi considerado satisfatório para o cenário, é fundamental criar um novo *template* a partir do processo SIGAP Coleta.

Com isto um novo *template* foi gerado e chamado LABES_Incremental (Figura 76). Este *template* é adequado para o desenvolvimento de sistemas grandes e modulares, que utilizam o ciclo de vida iterativo e incremental.

Com a generalização do processo, através da criação de um novo *template*, a organização pode aplicá-lo e aprimorá-lo em novos projetos com as mesmas características.

Da mesma forma, nas próximas buscas será possível identificar este novo *template* no *ranking* como sugestão para reutilização (Figura 78), dependendo das características definidas. Além da representação de contextos real, a nota atribuída ao processo é atribuída à matriz de notas do *template*.

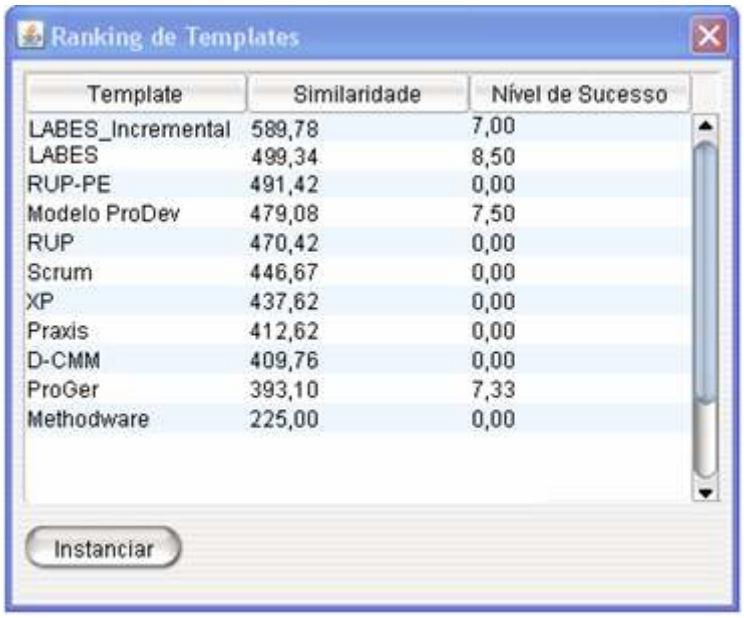

Figura 78 – *Ranking* de *templates* listando o novo *template* LABES_Incremental.

6.3 Considerações Finais

Este capítulo demonstrou a simulação de dois processos de um projeto. Os dois processos: SIGDCT e SIGAP Coleta possuem características específicas e bastante divergentes entre si. Enquanto o SIGDCT apenas seguiu o fluxo para o desenvolvimento de um protótipo, o SIGAP Coleta serviu para nortear o desenvolvimento de um projeto grande que enfrentou diversos situações características de um projeto de software durante sua execução e por isto realizou diversas mudanças no processo instanciado a partir do *template* LABES-UFPA.

Ambos os processos foram importantes para demonstrar que em diversas situações a abordagem possibilita que a organização de software identifique quais

templates são mais adequados às suas características, além de apresentar *feedback* sobre cada reutilização.

Quando um novo *template* é criado ou uma nova versão de *template* é gerada, com representação de contextos e nível de sucesso definidos, o conhecimento sobre este novo *template* vai estar disponível para outros usuários em projetos futuros. Isto contribui para a adoção de *templates* com históricos de sucesso nos projetos da organização de software.

Mesmo que um *template* seja totalmente aderente às características do projeto, o incremento da nota atribuída para um *template* permite expor seu desempenho nos projetos anteriores e, desta forma, gerar aprendizado organizacional e favorecer seu aprimoramento.

A execução de ambas as simulações permitiu testar a abordagem desta proposta, visualizar o seu funcionamento, suas vantagens e também melhorias a serem submetidas em trabalhos futuros, a serem abordados no Capítulo 7.

Por fim, a abordagem desta proposta provê o necessário para a construção do aprendizado organizacional através do gerenciamento dos ativos de processo do repositório. Desde a representação de contextos até a decisão sobre a retenção do processo através das métricas de avaliação.

7 CONCLUSÃO

De posse da necessidade de adoção de processos no desenvolvimento de software e da dificuldade em aplicá-los para a obtenção da qualidade do produto, é que o reuso de processos de software aliado ao RBC, tem se mostrado uma ferramenta útil para a construção do aprendizado organizacional.

O aprendizado organizacional é fundamental para evitar retrabalho e melhorar a qualidade do processo com o uso de melhores práticas e experiências passadas, remetendo a construção de uma base de conhecimento que registre seus acertos e erros.

Este trabalho aborda o reuso de processos de software utilizando RBC para realizar a gestão de ativos de processos e promover, gradativamente, a melhoria de processos de software para atingir a adoção de processos que atendam às necessidades da organização de software.

Esta abordagem oferece um mecanismo de representação atributo-valor dos casos no repositório. Estes casos são classificados de acordo com atributos relevantes para permitir uma recuperação de processos normalizada e eficiente.

Para garantir o aprendizado, este estudo propõe a avaliação do processo através de três métricas: comparação das similaridades globais, a qual avalia a variação da representação de contextos preliminar e real; nível de reuso, que apresenta o grau de reutilização do *template* no processo; e também uma avaliação subjetiva que a organização pode inferir sobre a satisfação do processo através do estabelecimento do seu nível de sucesso, esta avaliação é uma decisão da organização e considera suas características e necessidades.

Após a avaliação, a organização pode decidir sobre o destino do novo caso. É altamente sugerido realimentar o repositório com o novo caso e suas representações atributo-valor para prover reuso em projetos futuros similares.

Esta abordagem foi implementada no ambiente WebAPSEE-Pro, o que permitiu sua avaliação através de mecanismos de simulação. Com isso, foi possível perceber que a abordagem se aplica a qualquer projeto e possibilita identificar quais processos são mais aderentes às características do projeto ou da organização. Além da disponibilização do conhecimento relativo ao processo executado em projetos futuros, possibilitando a sugestão de modelos de processo mais similares e bem-sucedidos na organização.

Uma das principais contribuições da abordagem proposta é o suporte à gestão dinâmica de ativos de processos de software, favorecendo a melhoria contínua do processo através do *feedback* permanente para o repositório. Este *feedback* garante a incorporação de experiências de sucesso e falha que podem ser consideradas na tomada de decisão em projetos similares.

7.1 Trabalhos Futuros

Considerando que este trabalho corresponde a um passo inicial em direção à construção do conhecimento organizacional através do reuso de processos de software, diversas perspectivas sobre a sua evolução foram identificadas:

- Durante a execução do processo, permitir a instanciação de outros *templates* através da recuperação para a composição de subprocessos e
- Da mesma forma, registrar a reutilização dos diversos *templates* para realizar a avaliação de acordo com os diversos *templates* instanciados;
- Na avaliação do processo, permitir o estabelecimento de pesos nos subprocessos ou atividades que priorizem passos fundamentais do processo. Com isto será possível calcular a métrica de nível de reuso ponderado de acordo com a definição do usuário;
- Favorecer a realimentação do repositório a cada ciclo, possibilitando realizar busca, avaliação e retenção ao final de uma determinada iteração.
- Possibilitar avaliar o nível de reuso durante a execução para oferecer ao usuário o percentual de reutilização até o momento da execução, assim como o percentual de quanto ainda falta para reutilizar todo o *template*.
- Aumentar o número de indicadores para dar apoio à avaliação do nível de sucesso do *template* instanciado;
- Desenvolver um mecanismo genérico que possibilite a avaliação de vários *templates* no processo, bem como comparar um processo instanciado com um *template* diferente do foi instanciado;

Com este trabalho espera-se contribuir, de maneira automatizada, para o auto-conhecimento organizacional e proporcionar aumento da qualidade dos processos no desenvolvimento de software, visto que a capacidade de aprendizado dos sistemas de RBC favorece para a adoção de soluções melhores e mais eficientes.

REFERÊNCIAS BIBLIOGRÁFICAS

Aamodt, A., Plaza, E. **Case-based reasoning: Foundational issues, methodological variations, and system approaches**. Institut d'Investigació en Intel.ligència Artificial (IIIA). Artificial Intelligence Communications (AICom), IOS Press, Vol. 7: 1, p. 39-59. Catalonia, Spain, 1994.

Barbosa, I. M. **Análise de Características de Projetos de Software para a Definição de Processo de Software**. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Centro de Informática, Universidade Federal de Pernambuco. Recife, 2005.

Bareiss, R. **Exemplar-Based Knowledge Acquisition: A unified Approach to Concept Representation, Classification and Learning**. Academic Press, 1989. ISBN: 978-0120782604.

Beck, K. **Extreme Programming Explained: Embrace Change**. Pearson, 2004. ISBN: 978-0321278654.

Berger, P. M. **Instanciação de Processos de Software em Ambientes Configurados na Estação TABA**. Dissertação (Mestrado em Ciência da Computação) - COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2003.

Bertollo, G., Falbo, R. A**. Apoio Automatizado à Definição de Processos de Software em Níveis**. II Simpósio Brasileiro de Qualidade de Software. Fortaleza, Ceará, 2003.

Borges, L. M. S., Falbo, R. A. **Gerência de Conhecimento sobre Processos de Software**. VIII Workshop de Qualidade de Software. XV Simpósio Brasileiro de Engenharia de Software, Rio de Janeiro, 2001.

Booch, G., Martin, R. C., NewKirk, J. **Object Oriented Analysis and Design with Applications**. 2. ed. Addison Wesley Longman, Inc. 1998. ISBN: 978-0201895513

Brasil, M., Cortés, M. **Definição de Processo de Software através da maximização da similaridade de atributos de casos similares**. XIII Simpósio de Informática da Pontifícia Universidade Católica do Rio Grande do Sul, Uruguaiana. Revista HÍFEN ISSN 1983-6511. 2008.

Broomé, M., Runeson, P. **Technical Requirements for the Implementation of an Experience Base**. Lecture Notes In Computer Science; Vol. 1756. Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering, Learning Software Organizations, Methodology and Applications. p. 87 - 102. Springer-Verlag. London, UK. 1999. ISBN:3-540-41430-4

Chrissis, M. B., Konrad, M., Shrum, S. **CMMI guidelines for process integration and product improvement**. Boston, MA. Addison-Wesley. 2006. ISBN: 978-0321279675.

Costa, A. J. S., Sales, E. O. **Uma Proposta para Reutilização de Processos de Software para o Ambiente WebAPSEE**. Trabalho de Conclusão de Curso

(Graduação em Ciência da Computação). Universidade Federal do Pará. Belém, 2007.

Cunha, M. B. F. L. **Análise das Características Organizacionais para a Definição de Processo de Software**. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação). Centro de Informática, Universidade Federal de Pernambuco, Recife, 2005.

Falbo, R. A. **Integração de conhecimento em um ambiente de desenvolvimento de software**. Tese (Doutorado em Ciência da Computação) – COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1998.

França, B. B. N. **Proposta de um Modelo de Simulação de Processo de Software para o Ambiente WebAPSEE**. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Federal do Pará, Belém, 2007.

Franch, X.; Ribó, J. **Searching for Expressiveness, Modularity, Flexibility and Standardisation in Software Process Modeling**. Simpósio Brasileiro de Engenharia de Software, SBES. João Pessoa, Brasil. 2002.

Georgakopoulos, D., Hornik, M., Sheth, A. **An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure**. Disponível em: <http://www.workflowpatterns.com/documentation/documents/workflow95.pdf>. Acesso em: 17 Ago. 2009. Distributed and Parallel Databases, p. 119-153. Kluwer Academic Publishers, Boston. Manufactured in The Netherlands. 1995.

Jacobson, I., Booch, G., Rumbaugh, J. **The unified software development process**. Addison-Wesley. ISBN 0-201-57169-2. 1999.

Jørgensen, H., Carlsen, S. **Writings in Process Knowledge Management: Management of Knowledge Captured by Process Models**. Technical Report, No. STF40 A00011. SINTEF Telecom and Informatics. Oslo, 2001.

Humphrey, W. S. **Managing the Software Process**. Addison Wesley. New York. 1989. ISBN: 978-0201180954

Humphrey, W. S. **Introduction to the Personal Software Process**. Addison Wesley. 1997. ISBN:978-0201548099.

Humphrey, W. S. **TSP: Coaching Development Teams**. Addison Wesley Professional. Massachusetts. 2006. ISBN: 978-0201731132.

Kahn, C. E. J., Anderson, G. M. **Case-Based Reasoning and Imaging Procedure Selection**. Investigative Radiology: A Journal of Clinical and Laboratory Reasearch. v. 29. p. 643–647. June, 1994.

Kniberg, H. **Scrum e XP direto das trincheiras**. Editora C4Media Inc. 2007. ISBN: 978-1-4303-2264-1.

Kolodner, J. **Case-Based Reasoning**. Publisher Morgan Kaufmann. 1993. ISBN: 978-1558602373

Koton, P. **Reasoning about Evidence in Causal Explanation**. In Proceedings Case-Based Reasoning Workshop. Morgan Kaufmann Publishers, pages 260–170. 1988.

Kruchten P., Kroll P. **The Rational Unified Process Made Easy**. Addison-Wesley. 2003. ISBN: 978-0321166098.

LABES-UFPA. **Doc. de Referência do WebAPSEE**. Disponível em: <http://www.labes.ufpa.br>. Acesso em 03 Mai. 2009.

Lemos, A. M., Sales, E.O., Nascimento, L.M. A., Reis, C. A. L. **Uso de Práticas de Gerenciamento de Projetos no Desenvolvimento de um Sistema de Apoio a Redes de Pesquisa no Estado do Pará**. II Workshop de Gerenciamento de Projetos de Software. Simpósio Brasileiro de Qualidade de Software, 2009. Relato de Experiência. Ouro Preto, Minas Gerais, 2009.

Lenz, M., Bartsch-Spoerl, B., Burkhard, H.-D. **Case-Based Reasoning Technology: From Foundations to Applications**. Springer. October, 1998. ISBN: 978-3540645726.

Lima Reis, C. A. **Um Gerenciador de Processos de Software para o Ambiente PROSOFT**. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal do Rio Grande do Sul. Porto Alegre, 1998.

Lima Reis, C. A., Reis, R. Q., Nunes, D. J. **Gerenciamento do Processo de Desenvolvimento Cooperativo de Software no Prosoft**. XII Simpósio Brasileiro de Engenharia de Software (SBES'98). p. 221 - 236. Maringá, Paraná. 1998.

Lima Reis, C. A. Uma Abordagem Flexível para Execução de Processos de Software Evolutivos. Tese (Doutorado em Ciência da Computação) - Universidade Federal do Rio Grande do Sul. Porto Alegre, 2003.

Lins, R. G., Chwif, L. **Utilização da Ferramenta de Simulação em Projetos de Desenvolvimento de Software**. Revista da Pós-Graduação. v. 3. p. Centro Universitário FIEO. Osasco, São Paulo. 2008.

Machado, L. F., Oliveira, K. M., Rocha, A. R. C. **Modelo para definição de processos de software baseado na ISO/IEC 12207, em modelos de maturidade e características do projeto**. III Workshop Ibero-Americano de Engenharia de Requisitos e Ambientes de Software (IDEAS'00). Cancun, México. 2000.

Mcfeeley, B. **IDEALSM: A User's Guide for Software Process Improvement. Software Engineering Institute Handbook**. Carnegie Mellon University. CMU/SEI-96-HB-001. 1996.

Melo, C. O. **Reutilização de Software: Classificação e Seleção de Artefatos Reutilizáveis**. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2004.

Mikos, W. L. **Modelo Baseado em Agentes em Apoio à Solução de Problemas de Não-Conformidades em Ambientes de Manufatura com Recursos Distribuídos**.

Tese (Doutorado em Engenharia Mecânica) - Universidade Federal de Santa Catarina. Florianópolis. 2008.

OMG, Object Management Group. **Unified Modeling Language**. UML Resource Page. Disponível em: <http://www.uml.org/>. Acessado em: 01 Ago. 2009.

Oliveira, E. H. R. **Reuso em um Ambiente de Implementação de Processo de Software**. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade Federal de Pernambuco. 2005.

Oliveira, S. R. B., Vasconcelos, A. M. L. **A Continuous Improvement Model in ImPProS**. In 30th Annual International Computer Software and Applications Conference. Proceedings on COMPSAC Fast Abstract Session. Chicago, EUA. 2006.

Oliveira, S. R. B. **ProDefiner: Uma Abordagem Progressiva para a Definição de Processos de Software no Contexto de um Ambiente Centrado no Processo**. Tese (Doutorado em Ciência da Computação). Centro de Informática – Universidade Federal de Pernambuco. 2007.

Orci, T., Laryd, A. **Dynamic CMM for Small Organizations**. Umea University, Department of Computer Science. Umea, Sweeden. 2000.

Pal, S., Shiu, S. **Foundations of Soft Case-Based Reasoning**. Wiley series in intelligent systems. 2004. ISBN: 0-471-08635-5.

Pantleon, T., Göker, M. H., Roth-Berghofer, T., Bergmann, R., Traphöner, R., Wess, S., Wilke, W. **The Development of HOMER: A Case-Based CAD/CAM Help-Desk Support Tool**. Lecture Notes in Computer Science, v. 1488. p. 346-357. Advances in Case-Based Reasoning, 4th European Workshop. Dublin, Ireland. Springer. 1998, ISBN 3-540-64990-5.

Paula Filho, W. P. **Engenharia de Software: Fundamentos, Métodos e Padrões**. Editora LTC. 2. ed. 2003. ISBN: 8521613393.

Paulk, M. C., Weber, C. V., Curtis, B., Chrissis, M. B. **The Capability Maturity Model: Guidelines for Improving the Software Process**. Carnegie Mellon University, Software Engineering Institute, Addison-Wesley Longman Inc. ISBN: 978-0201546644. 1994.

Pfleeger, S. L. **Software Engineering: Theory and Practice**. 2. ed., Prentice Hall. 2001. ISBN: 978-0136061694.

PMI, Project Management Institute. **A Guide to the Project Management Body of Knowledge: PMBOK Guide**. Project Management Institute, 3. ed., 2008. ISBN: 978-1933890517.

Pollice, G., Augustine, L., Lowe, C., Madhur, J. **Software development for small teams - a rup centric approach**. Addison-Wesley. 2004. ISBN: 0-321-19950-2.

Poltrock, S., Grudin, J. **CSCW, groupware and workflow: experiences, state of art, and future trends.** Conference on Human Factors in Computing Systems. p. 119 – 120. Los Angeles, California. 1999. ISBN:1-58113-028-7.

Pressman, R. **Software Engineering**. 5. ed. McGraw-Hill. 2005. ISBN: 85-346-0237-9.

Reis, R.Q. **Uma Proposta de Suporte ao Desenvolvimento Cooperativo de Software no Ambiente PROSOFT**. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre. 1998.

Reis, R. Q., Reis, C.A.L., Nunes, D.J. **Automated Support for Software Process Reuse: Requirements and Early Experiences with the APSEE model**. In 7th International Workshop on Groupware. IEEE Computer Society Press. Darmstadt, Germany. 2001.

Reis, R. Q. **APSEE-Reuse: Um Meta-Modelo para Apoiar a Reutilização de Processos de Software**. PhD Thesis – Universidade Federal do Rio Grande do Sul, Porto Alegre. 2002.

Ricci, F., Arslan, B., Mirzadeh, N., Venturini, A. **Detailed Descriptions of CBR Methodologies**. Information Society Technologies. 2002. Disponível em: http://dietorecs.itc.it/ PubDeliverables/D4.1-V1.pdf. Acessado em: 13 Abr. 2009.

RMC. **Rational Method Composer**. Disponivel em: <http://www-01.ibm.com/software/awdtools/rmc/>. 2006. Acesso em: 19 Mai. 2009.

Rouiller, A. C. **Gerenciamento de Projetos de Software para Empresas de Pequeno Porte**. Tese (Doutorado em Ciência da Computação) - Universidade Federal de Pernambuco. 2003.

Royce, W. **Software Project Management – A Unified Framework**. Addison-Wesley. 1998. ISBN: 978-1933890937.

Sales, E.; Freitas, S.; Reis, R. Q. **Uma Ferramenta para Recuperação de Modelos de Processo de Software Reutilizáveis**. 19º. Simpósio Brasileiro de Engenharia de Software – Sessão de Ferramentas. Florianópolis-SC. Outubro, 2006.

Freitas Júnior, S. F. F., Reis, R. Q. **Um Modelo para Adaptação de Processos de Software**. 2007.

Santos, V., Cortes, M., Setúbal, C. **Estabelecendo a Cultura de Utilização de Processos de Software em uma Micro-Empresa**. III Workshop Um Olhar Sociotécnico sobre a Engenharia de Software (WOSES). VI Simpósio Brasileiro de Qualidade de Software. Porto de Galinhas-PE. 2007.

Santos, V. A., Cortés, M. I. **Software Process Reuse Using Case-Based Reasoning**. International Conference on Agents and Artificial Intelligence (ICAART´2009). Porto, Portugal. 2009.

Santos, V. A., Cortés, M. I., Brasil, M. **Reuse and Adaptation of Software Process Using Similarity Measurement**. In the Proceedings of International Conference on

Evaluation of Novel Approaches to Software Engineering (ENASE´2009). Milão, Itália. 2009a.

Santos V., Cortés M., Brasil M. **Dynamic Management of the Organizational Knowledge Using Case-Based Reasoning**. Accepted for publication in the Lecture Notes in Communications in Computer and Information Science. Springer Paper Book. Springer-Verlag. 2009b.

Salviano, C. F. **Uma Proposta Orientada a Perfis de Capacidade de Processo para Evolução da Melhoria de Processo de Software**. Exame de qualificação. Programa de Doutorado, Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas. Campinas, SP. 2006.

Schwaber, K. **Agile Project Management with Scrum**. Microsoft Press. 2005. ISBN 0-7356-1993-X.

Shannon, R. E. **Systems simulation, The Art and Science**. Prentice-Hall. 1975. ISBN: 978-0138818395.

Softex. **Guia Geral MR-MPS (Versão 1.2)**. 2007. Disponível em: <http://www.softex.br/mpsbr/_guias/guias/MPS.BR_Guia_Geral_V1.2.pdf>. Acesso em: 17 Fev. 2009.

Sommerville, I. **Engenharia de Software**. 6. ed. Prentice-Hall. 2003. ISBN: 978-0130639189.

The International Organization for Standardization and the International Electrotechnical Commission. **NBR ISO/IEC 12207:2008**. Tecnologia de Informação: Processos de Ciclo de Vida de Software. Genebra. 2008.

The International Organization for Standardization and the International Electrotechnical Commission. **ISO/IEC 15504 Information Technology Process Assessment Part 5: An exemplar Process Assessment Model**. Genebra. 2006.

Vasconcelos, A., Rouiller, A., Oliveira, S. **Uma Proposta de um Ambiente de Implementação de Processo de Software**. INFOCOMP Journal of Computer Science 4(1):pp. 70-77. 2005.

Vasconcelos, A., Rouiller, A., Oliveira, S. **Modelo Comportamental de um Ambiente de Implementação de Processo de Software**. INFOCOMP Journal of Computer Science V5(1): pp.76-86 (pdf). 2006. Disponível em: <http://www.dcc.ufla.br/infocomp/artigos/v5.1/art10.pdf>. Acesso em: 07 Jun. 2009.

Xavier, C. M. S, Vivacqua, F. R., Macedo, O. S., Xavier, L. F. S. **Metodologia de Gerenciamento de Projetos – Methodware**. Abordagem prática de como iniciar, planejar, executar, controlar e fechar projetos. 3. ed. Editora Brasport. 2005. ISBN: 85-7452-222-8.

Wangenheim, C. G. V., Wangenheim A. V. **Raciocínio Baseado em Casos**. Curitiba: Editora Manole, 2003. v. 1. 296 p. 2003. ISBN: 85-204-1459-1.

Wangenheim, C. G., Althoff, K. D., Barcia, R. M. **Intelligent Retrieval of Software Engineering Experienceware**. In the Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering (SEKE'99). Kaiserslautern, Germany. Junho, 1999.

Watson, Ian. **Applyig Knowledge Management: Techniques for building Organizational Memories**. San Francisco, CA: Morgan Kaufmann Publishers Inc., p. 252. 2003.

Wu, Z., Palmer, M. S. **Verb semantics and lexical selection**. In: ACL - 32nd Annual Meeting of the Association for Computacional Linguistic, p. 133 - 138. 1994.

APÊNDICE A

Este apêndice apresenta as representações de contextos, conforme Seção 4.2, dos modelos de processo integrantes do repositório deste trabalho. Trata-se dos modelos inicialmente definidos para compor o repositório e não estão incluídos os modelos ProDev e LABES-UFPA, visto que estão descritos respectivamente nos Capítulos 5 e 6.

A.1 *Dynamic* CMM (D-CMM)

O D-CMM [Orci e Laryd, 2000] é uma adaptação do CMMI-SW Nível 2 [Chrissis et al., 2006] desenvolvida para organizações pequenas e em crescimento que enfrentam dificuldades em aplicar esse modelo de qualidade devido ao elevado número de papéis, responsabilidades e tarefas.

Para se adequar às pequenas organizações, este modelo de processo foi objeto de modificações omitindo alguns papéis, atribuindo as tarefas para outros papéis e simplificando a documentação, mas sem afetar a sua intenção original.

De forma geral, para cada KPA[3] é necessário definir uma política e procedimentos documentados. A Tabela 21 apresenta uma visão geral do modelo.

[3] *Key Process Area* - Área de Processo Chave do modelo CMMI.

Tabela 21 - Visão geral do modelo D-CMM

KPA	Macro-Atividades	Artefatos
Gerência de Requisitos	• Análise dos requisitos • Controle e documentação dos requisitos • Controle das mudanças de requisitos • Revisão dos requisitos	Documento de Requisitos, Plano de Projeto (seção Gerência de Requisitos), *Checklist* de Revisão dos Requisitos e Documento de Mudança de Requisitos.
Planejamento de Projeto	• Realização de estimativas e planejamentos • Gerência do projeto • Revisão do planejamento	Declaração de trabalho, Estimativas de recursos, tempo e custo e Plano de projeto;
Monitoração e Controle	• Revisão do planejamento	Estimativas de recursos, tempo e custo e Plano de projeto.
Gerência de Aquisições	• Definição e planejamento das aquisições • Definição de atividades do fornecedor • Garantia de qualidade • Gerência de configuração • Controle do produto	Contrato.
Garantia da Qualidade	• Planejamento de qualidade • Revisão de qualidade • Relato de não-conformidades	Plano de Garantia da Qualidade.
Gerência de Configuração	• Planejamento da configuração • Gerência das atividades de configuração • Relato de problemas e mudanças • Gerência de mudanças • Controle de versão • Gerência de configuração • Revisão de *baseline*	*Baselines*, Relatórios de configuração e Relatório de problemas.

Para o modelo D-CMM, os autores caracterizaram seu contexto de uso de acordo com a coluna *Valor*, da Tabela 22. Com base nesta classificação são realizados os cálculos de similaridade local e global de acordo com o processo de recuperação da Seção 4.3.

Tabela 22 - Representação de contexto do *template* D-CMM

Contexto	j	Atributo	Valor
Projeto	1	Modelo de Ciclo de Vida	Iterativo
	2	Complexidade	Média
	3	Tamanho do Projeto	Médio
	4	Tamanho da Equipe	10
	5	Prazo	7
	6	Conhecimento em Engenharia de Software	Alto
	7	Paradigma de Desenvolvimento	OO
Processo	8	Modelo de Desenvolvimento	-
	9	Modelo de Maturidade	CMMI-SW
	10	Nível de Maturidade	2
	11	Complexidade	Média
	12	Processo	Gerência de Projetos
	13	Experiência no uso de processos	Média

A.2 *eXtreme Programming* (XP)

A metodologia XP é uma abordagem disciplinada que tem como objetivo entregar funcionalidades de forma rápida e eficiente ao cliente [Beck, 2004]. Além

disto, esta metodologia foi criada considerando que mudanças são inevitáveis, e devem ser incorporadas constantemente.

XP é considerada uma metodologia ágil e se baseia em valores, tais como a comunicação, a simplicidade, o *feedback* e a coragem; princípios tais como *feedback* rápido, assumir simplicidade, mudança incremental, abraçar mudanças e trabalho de qualidade. Além dos valores e princípios, a metodologia propõe a adoção de doze práticas, que são o jogo do planejamento, *releases* curtos, metáfora, projeto simples, desenvolvimento orientado a testes, refatoração, programação em pares, propriedade coletiva do código, integração contínua do código, semana de quarenta horas, cliente presente e uso de padrões de codificação [Beck, 2004]. A Tabela 23 apresenta uma visão geral deste modelo.

Tabela 23 - Visão geral do modelo XP

Fase	Macro-Atividades	Artefatos
Exploração	• Verificação da viabilidade do projeto	Atas de reunião
Planejamento do Release	• Definição e priorização dos requisitos • Estabelecimento do cronograma	Estórias e Atas de reunião.
Iteração do *Release*	• Escrita dos casos de testes • Projeto e refatoração • Codificação • Integração • Realização dos testes • Gerenciamento de configuração	Plano de testes de aceitação, Casos de testes, Diagramas de classes, Modelos de dados, Manual do usuário, Tutorial, Código-fonte, Quadro de acompanhamento diário e Atas de reunião.
Produção	• Simulação de ambiente • Testes de aceitação adicionais	Atas de reunião e Casos de testes.
Manutenção	• Identificação de defeitos • Correção de defeitos • Realização dos testes	Casos de testes, Plano de testes de aceitação e Atas de reunião.
Finalização	• Retrospectiva	Quadro de acompanhamento diário e Atas de reunião.

Mesmo com todos os benefícios declarados, XP possui alguns fatores que podem restringir a sua utilização, tais como, pouca familiaridade com as práticas de XP e a filosofia, equipes grandes, espaço físico inapropriado, clientes desconfiados e tecnologia de apoio inadequado para a gestão de mudanças. A Tabela 24 apresenta a representação de contextos deste modelo.

Tabela 24 - Representação de contexto do *template* XP

Contexto	j	Atributo	Valor
Projeto	1	Modelo de Ciclo de Vida	Iterativo\Espiral
	2	Complexidade	Média
	3	Tamanho do Projeto	Pequeno
	4	Tamanho da Equipe	3 pessoas
	5	Prazo	3 meses
	6	Conhecimento em Engenharia de Software	Alto
	7	Paradigma de Desenvolvimento	OO
Processo	8	Modelo de Desenvolvimento	Ágil
	9	Modelo de Maturidade	-
	10	Nível de Maturidade	-
	11	Complexidade	Baixa
	12	Processo	-
	13	Experiência no uso de processos	Baixa

A.3 Praxis

Modelo desenvolvido por [Filho, 2003] para apoiar projetos de médio porte, combina elementos do PSP (*Personal Software Process*) [Humphrey, 1997], do TSP (*Team Software Process*) [Humphrey, 2006] e do Processo Unificado [Jacobson et al., 1999]. A sigla Praxis significa **PR**ocesso para **A**plicativos e**X**tensíveis **I**nterativo**S**, refletindo uma ênfase no desenvolvimento de aplicativos gráficos interativos, baseados na tecnologia orientada a objetos.

As práticas gerenciais são inspiradas nas práticas chaves dos níveis 2 e 3 do CMMI-SW. O material que consta em [Filho, 2003] inclui modelos de documentos, que facilitam a preparação dos documentos requeridos, e roteiros de revisão, que facilitam a verificação destes. Além disso, a UML é a notação de modelagem deste modelo.

As disciplinas do Processo Unificado são mapeadas como fluxos no modelo Praxis, conforme descrito na Tabela 25.

Tabela 25 – Detalhamento das fases do Praxis.

Fase	Macro-Atividades	Artefatos
Concepção	• Ativação	Proposta de Especificação de Software
Elaboração	• Levantamento dos Requisitos • Análise dos Requisitos	Modelo de Análise do Software, Especificação dos Requisitos do Software, Cadastro de Requisitos do Software, Modelo de Análise do Software, Especificação dos Requisitos do Software, Cadastro de Requisitos do Software, Memória de Cálculo do Projeto do Software, Modelo de Planejamento do Projeto do Software, Plano de Desenvolvimento do Software, Resultados Plano da Qualidade do Software.
Construção	• Desenho Inicial • Liberação • Testes Alfa	Insumos, Modelo de Desenho do Software, Descrição do Desenho do Software, Descrição dos Testes do Software, Bateria de Testes de Regressão do Software, Insumos, Relatórios dos Testes do Software, Códigos Fontes do Software, Códigos Executáveis do Software, Insumos, Relatórios dos Testes do Software, Manual do Usuário do Software.
Transição	• Testes Beta • Operação Piloto	Insumos, Relatórios dos Testes do Software, Insumos, Relatório Final de Projeto de Software.

A representação de contextos deste modelo é apresentada na Tabela 26.

Tabela 26 - Representação de contextos do *template* Praxis

Contexto	j	Atributo	Valor
Projeto	1	Modelo de Ciclo de Vida	Iterativo\Evolutivo
	2	Complexidade	Média
	3	Tamanho do Projeto	Médio
	4	Tamanho da Equipe	3 pessoas
	5	Prazo	3 meses
	6	Conhecimento em Engenharia de Software	Médio
	7	Paradigma de Desenvolvimento	OO
Processo	8	Modelo de Desenvolvimento	Híbrido
	9	Modelo de Maturidade	-
	10	Nível de Maturidade	-
	11	Complexidade	Baixa
	12	Processo	-
	13	Experiência no uso de processos	Baixa

A.4 ProGer

Este modelo tem como objetivo principal contribuir para que as organizações de pequeno porte possam estabelecer melhorias no seu processo de software. Estas melhorias são realizadas através do uso sistemático de um processo de gerenciamento de projetos de software de alto nível, cujo modelo considera seis elementos essenciais: a política organizacional, os padrões, o processo de gerenciamento, os procedimentos, os treinamentos, e as ferramentas [Rouiller, 2003]. A Tabela 27 descreve, em alto-nível, o modelo ProGer.

Tabela 27 - Visão geral do modelo ProGer

Fase	Macro-Atividades	Artefatos
Prospecção	• Identificação de necessidades/demandas • Planejamento da Fase • Construção do protótipo • Apresentação do protótipo	Documentos de requisitos e atas de reunião.
Proposta	• Solicitação de serviço • Planejamento da fase • Delimitação do escopo • Elaboração de Propostas	Documentos de requisitos, propostas comerciais, propostas técnicas e atas de reunião.
Execução	• Abertura do projeto • Elaboração do Plano de projeto • Realização do plano • Entrega • Obtenção do aceite do cliente	Documentos de requisitos, relatório de teste, relatórios de aceite, planos de projeto, ordens de serviço e relatórios de visitas técnicas.
Garantia	• Revisão de problemas • Correções e melhorias	Relatórios de teste, atas de reuniões, ordens de serviço e relatórios de visita técnica.
Encerramento	• Aceitação Final • Análise de problemas e melhorias • Aprovação de melhorias • Arquivamento do projeto	Atas de reunião.

De acordo com a autora, o ProGer, foi aplicado em 81 projetos de uma organização de software de pequeno porte, viabilizando um melhor desempenho através da utilização sistemática do modelo. Na coluna *Valor* da Tabela 28 é apresentada a representação de contextos deste modelo.

Tabela 28 - Representação de contexto do *template* ProGer

Contexto	j	Atributo	Valor
Projeto	1	Modelo de Ciclo de Vida	Fases do PMBOK
	2	Complexidade	Média
	3	Tamanho do Projeto	Médio
	4	Tamanho da Equipe	4
	5	Prazo	7
	6	Conhecimento em Engenharia de Software	Médio
	7	Paradigma de Desenvolvimento	OO
Processo	8	Modelo de Desenvolvimento	Híbrido
	9	Modelo de Maturidade	-
	10	Nível de Maturidade	-
	11	Complexidade	Baixa
	12	Processo	Gerência de Projetos
	13	Experiência no uso de processos	Baixa

A.5 *Methodware*

Esta é uma metodologia de gerenciamento de projetos [Xavier et al., 2005], tendo como base os processos de gerenciamento propostos pelo PMBOK. Ela pode ser adaptada para o tipo de projeto ou empresa onde será aplicada. Os processos desta metodologia são apresentados em alto nível na Tabela 29.

Tabela 29 - Visão geral da metodologia *Methodware*

Fase	Macro-Atividades	Artefatos
Iniciação	• Selecionar o projeto • Autorizar o início do projeto	Proposta do projeto, Termo de abertura do projeto.
Planejamento	• Identificar os *stakeholders* • Mobilizar a equipe de planejamento • Elaborar a declaração do escopo • Elaborar a estrutura analítica do projeto • Elaborar o cronograma • Calcular os custos • Planejar as comunicações • Planejar as aquisições • Planejar as respostas aos riscos • Planejar a qualidade • Planejar os recursos humanos • Consolidar o plano do projeto	Plano de gerenciamento do projeto, Declaração de escopo, Estrutura analítica do projeto (EAP), Dicionário da EAP, Cronograma, Orçamento, Plano de gerenciamento das comunicações, Ata de reunião, Relação das partes interessadas, Mapa das comunicações, Plano de gerenciamento de aquisições, Declaração de trabalho, Fórmula para avaliação de fornecedores, Quadro resumo da proposta, Plano de resposta aos riscos, Plano de gerenciamento da qualidade, Lista de verificação da qualidade, Plano de gerenciamento de pessoal, Autorização do trabalho, Matriz de atribuição de responsabilidades, Relação da equipe do projeto, Plano integrado de mudanças.
Execução	• Gerenciar a execução do plano do projeto	Plano de gerenciamento do projeto, Pedido de proposta, Plano de gerenciamento de pessoal, Avaliação de desempenho individual, Solicitação de mudanças.
Monitoramento e Controle	• Monitorar e controlar o trabalho do projeto • Controlar as mudanças no plano do projeto	Relatório de desempenho, Solicitação de mudanças, Plano de gerenciamento do projeto, Plano de resposta aos riscos, Aceite de produtos e serviços, Plano de gerenciamento das comunicações.
Encerramento	• Registrar as lições aprendidas • Encerrar o projeto	Relatório final do projeto.

A coluna *Valor* da Tabela 30 apresenta a representação de contextos deste modelo.

Tabela 30 - Representação de contexto do *template Methodware*

Contexto	j	Atributo	Valor
Projeto	1	Modelo de Ciclo de Vida	Fase do PMBOK
	2	Complexidade	Média
	3	Tamanho do Projeto	Médio
	4	Tamanho da Equipe	4 pessoas
	5	Prazo	6 meses
	6	Conhecimento em Engenharia de Software	-
	7	Paradigma de Desenvolvimento	-
Processo	8	Modelo de Desenvolvimento	Híbrido
	9	Modelo de Maturidade	-
	10	Nível de Maturidade	-
	11	Complexidade	Baixa
	12	Processo	Gerência de Projetos
	13	Experiência no uso de processos	Baixa

A.6 *Rational Unified Process* (RUP)

Este modelo possui o objetivo de aumentar a produtividade no desenvolvimento de software através da aplicação de técnicas a serem seguidas pelos

membros da equipe e utilização da abordagem de orientação a objetos com a notação UML (*Unified Modeling Language*) [OMG, 2009] para ilustrar os processos em ação.

É um processo considerado pesado e preferencialmente aplicável a grandes equipes de desenvolvimento e a grandes projetos, porém o fato de ser amplamente customizável torna possível que seja adaptado para projetos de qualquer escala. Para a gerência do projeto, o RUP provê uma solução disciplinada de como assinalar tarefas e responsabilidades dentro de uma organização de desenvolvimento de software. A Tabela 31 apresenta uma visão geral deste modelo.

Tabela 31 – Visão geral do RUP

Fase	Macro-Atividades	Artefatos
Iniciação	• Solicitação de serviço • Formulação do escopo • Planejamento e preparação do caso de negócio • Sintetização de possível arquitetura • Preparação do ambiente • Gerenciar configuração e mudança	Documento de Visão, Casos de Uso preliminares, Lista de Riscos e Plano de Projeto.
Elaboração	• Planejamento do projeto • Consolidar a *baseline* da arquitetura • Análise e projeto • Planejar os testes • Preparar ambiente para a iteração • Gerenciar configuração e mudança	Plano de Projeto, Casos de Uso, Arquitetura, Plano de testes, Casos de testes de unidade, Casos de testes de aceitação, Protótipo de interface e Modelo de dados.
Construção	• Implementação • Testes Alfa • Integração das iterações • Garantia da Qualidade • Avaliação dos *releases* • Gerenciar configuração e mudança • Gerenciamento do projeto	Plano de projeto, Casos de uso, Caso de testes de unidade, Caso de testes de aceitação, Arquitetura, Código-fonte e Modelo de dados.
Transição	• Implantação • Treinamento dos usuários • Testes Beta • Correções e melhorias • Aceitação final • Gerenciar configuração e mudança • Gerenciamento do projeto	Código-fonte, Produto final, Documentação do usuário e Revisão de Post-Mortem.

A representação de contextos deste modelo é apresentada na Tabela 32.

Tabela 32 - Representação de contexto do *template* RUP

Contexto	j	Atributo	Valor
Projeto	1	Modelo de Ciclo de Vida	Iterativo\Incremental
	2	Complexidade	Alta
	3	Tamanho do Projeto	Grande
	4	Tamanho da Equipe	12 pessoas
	5	Prazo	24 meses
	6	Conhecimento em Engenharia de Software	Alto
	7	Paradigma de Desenvolvimento	OO
Processo	8	Modelo de Desenvolvimento	Pesado\RUP
	9	Modelo de Maturidade	-
	10	Nível de Maturidade	-
	11	Complexidade	Alta
	12	Processo	-
	13	Experiência no uso de processos	Alta

A.7 RUP para Pequenas Equipes (RUP-PE)

Esta abordagem [Pollice et al., 2004], direcionada para pequenas equipes, é centralizada no RUP (Rational Unified Process) [Kruchten e Kroll, 2003], mas incorpora práticas de outras metodologias, tais como PSP, XP e outras metodologias ágeis. Na Tabela 33 é apresentada uma descrição sucinta das fases, macro-atividades e artefatos deste modelo de processo.

Tabela 33 - Abordagem do RUP para pequenas equipes

Fase	Macro-Atividades	Artefatos
Concepção	• Definição dos objetivos e não objetivos do projeto • Elicitação e priorização de requisitos preliminares • Estabelecimento de um cronograma em alto-nível • Identificação e mitigação dos riscos • Configuração do ambiente de desenvolvimento • Avaliação da iteração	Documento de Visão, Casos de Uso preliminares, Lista de Riscos e Plano de Projeto.
Elaboração	• Produção de uma arquitetura • Levantamento detalhado dos requisitos • Criação de testes e planos de testes • Definição dos detalhes da implementação • Gerenciamento do Escopo • Avaliação da iteração	Plano de Projeto, Casos de Uso, Arquitetura, Plano de testes, Casos de testes de unidade, Casos de testes de aceitação, Protótipo de interface e Modelo de dados.
Construção	• Controle de versão, defeitos e requisitos • Planejamento da implementação • Implementação	Plano de projeto, Casos de uso, Caso de testes de unidade, Caso de testes de aceitação, Arquitetura, Código-fonte e Modelo de dados.
Transição	• Aplicação de testes beta • Treinamento de usuários • Ajustes de código e correção de defeitos • Aceitação final • Post-Mortem	Código-fonte, Produto final, Documentação do usuário e Revisão de Post-Mortem.

Além da definição do processo, a abordagem privilegia a comunicação efetiva entre os integrantes da equipe e o cliente, através de técnicas tais como dinâmicas de formação de equipe, assim como também a definição de estratégias para

lidar em situações diversas seja com clientes ou membros da equipe, promovendo a utilização de ferramentas de colaboração.

Na busca pela melhoria contínua, os autores enfatizam a prática da auto-reflexão através da atividade de Post-Mortem que consiste em fazer uma analise retrospectiva em relação à qualidade do trabalho feito, analisando os acertos e erros na busca pela identificação de suas causas.

Na Tabela 34 é apresentada a representação de contextos deste modelo, de acordo com [Pollice et al., 2004].

Tabela 34 - Representação de contexto do *template* RUP-PE

Contexto	j	Atributo	Valor
Projeto	1	Modelo de Ciclo de Vida	Iterativo\ Incremental
	2	Complexidade	Média
	3	Tamanho do Projeto	Médio
	4	Tamanho da Equipe	5
	5	Prazo	5
	6	Conhecimento em Engenharia de Software	Alto
	7	Paradigma de Desenvolvimento	OO
Processo	8	Modelo de Desenvolvimento	Híbrido
	9	Modelo de Maturidade	-
	10	Nível de Maturidade	-
	11	Complexidade	Média
	12	Processo	Gerência de Projetos
	13	Experiência no uso de processos	Baixa

A.8 Scrum

Metodologia ágil de gerenciamento de projetos de software customizável de acordo com realidade do projeto [Schwaber, 2005]. A Tabela 35 apresenta uma visão geral desta metodologia [Kniberg, 2007].

Tabela 35 – Visão geral da metodologia Scrum

Fase	Macro-Atividades	Artefatos
Visão	• *Product backlog* inicial • Comprometimento • Planejamento de releases • Montagem do time • Planejamento de sprints	*Product backlog* e S*print backlog*.
Planejamento do Sprint	• Atualização do *backlog* de produto • Planejamento do *sprint*	*Product backlog* e S*print backlog*.
Sprint	• *Daily Scrum* • Incremento do produto • Revisão *sprint* • Retrospectiva *sprint*	*Backlog* de *sprint*, Gráfico *burndown*, Plano de riscos, Impedimentos.
Release	• Entrega	*Release* do produto

A representação de contextos deste *template* no repositório desta proposta é apresentada na Tabela 36.

Tabela 36 - Representação de contextos do *template* Scrum

Contexto	j	Atributo	Valor
Projeto	1	Modelo de Ciclo de Vida	Iterativo\Espiral
	2	Complexidade	Média
	3	Tamanho do Projeto	Médio
	4	Tamanho da Equipe	4 pessoas
	5	Prazo	3 meses
	6	Conhecimento em Engenharia de Software	Alto
	7	Paradigma de Desenvolvimento	OO
Processo	8	Modelo de Desenvolvimento	Ágil
	9	Modelo de Maturidade	-
	10	Nível de Maturidade	-
	11	Complexidade	Baixa
	12	Processo	Gerência de Projetos
	13	Experiência no uso de processos	Baixa