UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE INFORMÁTICA
MESTRADO EM INFORMÁTICA

BRUNO MARQUES SEGRINI

DEFINIÇÃO DE PROCESSOS BASEADA EM COMPONENTES

VITÓRIA
2009

BRUNO MARQUES SEGRINI

DEFINIÇÃO DE PROCESSOS BASEADA EM COMPONENTES

Dissertação submetida ao Programa de Pós-graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do título de Mestre em Informática.

Orientador: Prof. Dr. Ricardo de Almeida Falbo.

VITÓRIA
2009

BRUNO MARQUES SEGRINI

DEFINIÇÃO DE PROCESSOS BASEADA EM COMPONENTES

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovada em 28 de agosto de 2009.

COMISSÃO EXAMINADORA

_____

Prof. Ricardo de Almeida Falbo, D.Sc.
Orientador

_____

Prof. Davidson Cury, D.Sc.
UFES

_____

Prof. Carlos Alberto Marques Pietrobon, D.Sc.
UFOP

Aos meus pais, irmãs e Lívia, por simplesmente existirem em minha vida e me fazerem tão bem;

Ao meu avô, João Segrini, pelas vitórias alcançadas em sua vida;

Ao meu mestre Ricardo Falbo, por ter me ajudado e incentivado a concluir o mestrado, e pelas cobranças nos momentos necessários.

AGRADECIMENTOS

Aos meus pais, irmãs e Lívia, pelo eterno amor, grande compreensão nos momentos de ausência e imensa torcida; essa conquista também é de vocês;

Ao meu grande mestre, Ricardo Falbo, por me guiar em minha jornada acadêmica desde os tempos de iniciação científica e pelo prazer de sua orientação;

Ao meu amigo-irmão Bruno, pelas palavras nos momentos mais difíceis durante essa caminhada e pelos impressos de grande valia para este trabalho;

Aos meus amigos "ODEanos", em especial Aline, Rodrigo Calhau, Bruno Nandolpho, Ana e Alexandre, pela colaboração, apoio e companheirismo;

Aos professores do Departamento de Informática da UFES, pelos ensinamentos;

À Idália Pôrto, Fabrício Miranda e Roberto Heleno, líderes dos projetos em que trabalhei durante a elaboração desta dissertação, pelas liberações para realizar as reuniões de mestrado;

A todos aqueles que, de forma direta ou indireta, colaboraram para a conclusão deste trabalho.

RESUMO

Sabe-se que a qualidade dos produtos de software depende da qualidade dos processos de software utilizados em seu desenvolvimento e manutenção. Entretanto, definir processos de software não é uma tarefa trivial, sendo necessários conhecimento, experiência e abordagens de apoio, visando à redução do esforço necessário para executá-la.

Neste trabalho, foi desenvolvida uma abordagem para Definição de Processos Baseada em Componentes (DPBC), a qual utiliza conceitos e faz uma analogia à abordagem de Desenvolvimento Baseado em Componentes (DBC) para o domínio de processos de software, visando facilitar a tarefa da definição de processos. Nessa abordagem, a definição de processos de software em uma organização é feita por meio da composição de componentes pré-existentes, podendo ser definidos processos padrão organizacionais e processos específicos de projeto.

Este trabalho contempla também a evolução da ferramenta de definição de processos de ODE (Ontology-based software Development Environment), um ambiente de desenvolvimento de software centrado em processos e baseado em ontologias, visando oferecer apoio automatizado à definição de processos, agora com a perspectiva de componentes de processo.

Palavras-chave: Processos de Software, Definição de Processos de Software, Reutilização de Processos de Software, Componentes de Processo.

ABSTRACT

It is known that software product quality depends on the quality of the software processes used in its development and maintenance. However, defining software processes is not a trivial task, being necessary knowledge, experience and support approaches, aiming to reduce the effort required to carry it out.

In this work, a Component-Based Process Definition (CBPD) approach was defined. It uses some concepts and makes analogies to the Component-Based Development (CBD), with the goal of facilitating the processes definition task. In this approach, the software process definition in an organization is performed through the composition of pre-existing process components, being possible to define standard processes and project-specific processes.

This work also includes the evolution of the software process definition tool of ODE (Ontology-based software Development Environment), a Process-Centered Software Engineering Environment based on ontologies. This evolution aims to provide automated support to the process definition task, now considering process component composition.

Keywords: Software Process, Software Process Definition, Software Process Reuse, Process Components.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

Introdução

O objetivo deste capítulo é apresentar a motivação para este trabalho, os objetivos buscados e a maneira como a pesquisa foi conduzida, procurando informar também a estrutura da dissertação.

1.1. Motivação

Atualmente, é cada vez maior a preocupação das organizações de software em desenvolver produtos de qualidade, visando tornarem-se mais competitivas. Muitos esforços têm sido despendidos na definição de processos de software de qualidade, uma vez que, reconhecidamente, a qualidade dos processos de software seguidos influencia diretamente na qualidade dos produtos de software desenvolvidos (ARENT et al., 2000). Entretanto, definir processos de software não é uma tarefa trivial. Há uma grande quantidade de material indicando as melhores práticas a serem seguidas, mas cada organização deve definir seus processos levando em consideração não só essas informações, mas também suas próprias características (BERTOLLO et al., 2006).

Assim, é primordial que existam abordagens que auxiliem a definição de processos, visando diminuir o esforço necessário para executá-la. Neste contexto, a reutilização de processos é uma boa alternativa. A área de reutilização de software já vem sendo estudada há algum tempo, sobretudo com foco em produtos de software, podendo ser uma grande aliada na busca por abordagens para o reúso de processos.

1.2. Objetivos e Contexto do Trabalho

Considerando-se a necessidade de mecanismos que auxiliem a tarefa de definição de processos de software, o objetivo principal deste trabalho é desenvolver uma abordagem baseada em reutilização de componentes de processo para a definição de processos de software, denominada Definição de Processos Baseada em Componentes (DPBC).

Tomando por base a abordagem de Desenvolvimento Baseado em Componentes (DBC) (GIMENES e HUZITA, 2005), são realizadas analogias com o domínio de processos de software, de modo que a definição de processos de software em uma organização seja feita por meio da composição de componentes pré-existentes, podendo ser definidos processos padrão organizacionais e processos específicos de projeto. A abordagem proposta estabelece componentes de processo, seus níveis de granularidade e abstração, interfaces e repositórios, bem como processos para a definição *para* reutilização e *com* reutilização. Além disso, trabalha-se o apoio automatizado à abordagem proposta.

Este trabalho está inserido no contexto do Projeto ODE (*Ontology-based Software Development Environment*) (FALBO et al., 2003), um Ambiente de Desenvolvimento de Software Centrado em Processos (ADSCP) baseado em ontologias, que vem sendo desenvolvido no Núcleo de Estudos em Modelagem Conceitual e Ontologias (NEMO) da Universidade Federal do Espírito Santo (UFES).

Por estar inserido no contexto do Projeto ODE, o apoio automatizado é trabalhado por meio da evolução da ferramenta de definição de processos de ODE para que ela passe a prover apoio automatizado à abordagem de Definição de Processos Baseada em Componentes proposta.

Assim, são objetivos específicos deste trabalho:

- Definir componentes de processo em diferentes níveis de granularidade (processos complexos, processos simples e macroatividades) e níveis de abstração (processos padrão ou de projeto), bem como suas interfaces e repositórios;
- Definir um processo de DPBC para o nível de processos padrão, contemplando a definição *para* reúso – componentes no nível de processo padrão são definidos sempre visando à reutilização, ainda que envolva também a definição com reúso, já que componentes de processo padrão podem ser definidos a partir de outros componentes de processo padrão já existentes;
- Definir um processo de DPBC para o nível de processos de projeto, contemplando a definição *com* reúso – componentes no nível de processo de projeto são definidos sempre por meio da reutilização de componentes de processo padrão ou de outros componentes de processo de projeto;

- Evoluir a ferramenta de definição de processos de ODE para prover apoio automatizado para a abordagem proposta.

1.3. Histórico do Trabalho

Este trabalho teve início com uma revisão bibliográfica sobre processos de software e definição de processos de software, na qual foram avaliados e discutidos artigos científicos, relatórios técnicos, livros e trabalhos acadêmicos. O foco desse estudo foi investigar a área de definição de processos de software para entender os principais conceitos envolvidos e descobrir uma forma de auxiliar essa tarefa.

Percebeu-se com esse estudo inicial que a reutilização de processos se mostrava uma boa alternativa. Então, partiu-se para o estudo específico de trabalhos na literatura que envolvessem esse tema. Ainda, foram estudados temas relativos à reutilização de produtos de software, uma vez que é uma área com trabalhos desenvolvidos há algum tempo, tendo, assim, um bom nível de maturidade, podendo, então, servir de base para proposição de novos mecanismos para reutilização de processos de software.

Em seguida, partiu-se para a definição da abordagem de Definição de Processos Baseada em Componentes (DPBC). A abordagem foi definida buscando-se fazer analogias dos conceitos empregados no Desenvolvimento Baseado em Componentes (DBC) para o domínio de processos de software. Assim, foram definidos os conceitos envolvidos na abordagem e, em seguida, definiu-se um processo para definição de processos de software a partir da reutilização de componentes e considerando os níveis de abstração de processo padrão e processo de projeto.

O passo seguinte foi avaliar a ferramenta de definição de processos de software de ODE, analisando o impacto que a abordagem proposta teria na infraestrutura de processos de software adotada em ODE. Assim, a ferramenta de definição de processos do ambiente (SEGRINI et al., 2006) teve que ser revisada para atender aos novos conceitos da abordagem de DPBC, sendo necessária a evolução da mesma.

Finalizado o desenvolvimento da abordagem de Definição de Processos Baseada em Componentes, foi iniciada a escrita da dissertação. Em paralelo com a elaboração da dissertação, iniciou-se a evolução da ferramenta de definição de processos de ODE.

1.4. Organização do Trabalho

Nesta dissertação há, além deste capítulo que apresenta a *Introdução*, mais quatro capítulos, a saber:

- Capítulo 2 – *Processos de Software e Reutilização*: fornece uma fundamentação teórica sobre processos de software, definição de processos de software e reutilização de processos de software. Fornece, ainda, uma visão geral sobre reutilização de produtos de software e desenvolvimento baseado em componentes, bem como do ambiente ODE.
- Capítulo 3 – *Definição de Processos Baseada em Componentes*: apresenta a abordagem de Definição de Processos Baseada em Componentes (DPBC) proposta, trazendo os conceitos e os processos definidos.
- Capítulo 4 - *Apoio Automatizado à Definição de Processos Baseada em Componentes*: apresenta a nova versão da ferramenta de definição de processos de ODE, gerada a partir da revisão da ferramenta de modo a prover apoio automatizado para a abordagem de DPBC proposta.
- Capítulo 5 – *Considerações Finais*: contém as considerações finais sobre o trabalho aqui desenvolvido, apresentando suas contribuições e propostas para trabalhos futuros.

Capítulo 2

Processo de Software e Reutilização

O objetivo deste capítulo é estabelecer uma fundamentação teórica sobre definição de processos de software e reutilização em um contexto geral para, em seguida, vê-la no contexto da reutilização de processos. Desse modo, a abordagem adotada para este capítulo se dá por: (i) uma visão do conceito de processo de software; (ii) uma discussão sobre a definição de processos; (iii) uma visão geral da reutilização de software; (iv) uma apresentação do desenvolvimento baseado em componentes; (v) uma discussão sobre reutilização de processos de software; e, por fim, (vi) uma apresentação sobre como o tema definição de processos de software é tratado atualmente no ambiente ODE.

2.1. Introdução

Um dos principais motivos para que uma organização de software se preocupe em possuir processos de software bem definidos é o fato da qualidade do produto de software desenvolvido estar diretamente ligada à qualidade do processo de software adotado para desenvolvê-lo. Assim, quanto maior a preocupação com processos de qualidade, maiores tendem a ser os benefícios alcançados pela organização.

Contudo, a definição de processos de software não é uma tarefa trivial. Existem modelos e normas de qualidade, tais como CMMI (SEI, 2006), MPS.BR (SOFTEX, 2009) e ISO/IEC 12207 (ISO/IEC, 2008) voltados exclusivamente para a definição de processos de qualidade, definindo as melhores práticas a serem adotadas. Entretanto, cada organização deve definir seus processos levando em consideração não só essas informações, mas também suas próprias características (BERTOLLO et al., 2006).

Dessa forma, surge a necessidade de se desenvolver mecanismos capazes de auxiliar os engenheiros de processo e os gerentes de projeto na definição dos processos padrão organizacionais e dos processos específicos de projeto. Nesse sentido, a reutilização de processos se mostra uma boa alternativa. A reutilização de produtos de

software vem sendo objeto de estudo há algum tempo, tendo conseguido grande avanços, podendo ser uma grande aliada na tentativa de se desenvolver abordagens eficientes de reúso de processos.

Este capítulo está estruturado da seguinte forma: a Seção 2.2 fornece uma base teórica acerca de processos de software; a Seção 2.3 fornece uma visão tradicional da definição de processos; na Seção 2.4, é feita uma breve apresentação sobre reutilização de software num contexto geral; a Seção 2.5 trata do Desenvolvimento Baseado em Componentes (DBC); a Seção 2.6 fala a respeito da reutilização de processos de software, tratando da componentização de processos e outras abordagens relacionadas ao reúso de processos; a Seção 2.7 faz uma apresentação sobre o mecanismo de definição de processos de software do ambiente ODE, contexto no qual este trabalho está inserido; por fim, a Seção 2.8 apresenta as considerações finais do capítulo.

2.2. Processo de Software

Um processo de software pode ser definido como um conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, implantar e manter um produto de software (FUGGETTA, 2000). Um processo de software bem definido deve indicar as atividades a serem executadas, os recursos (humanos, de hardware e de software) requeridos, os artefatos consumidos e produzidos e os procedimentos a serem adotados (métodos, técnicas, modelos de documentos, entre outros) (FALBO, 1998) (GRUHN, 2002).

As atividades são as tarefas a serem realizadas, sendo que elas podem requerer certos recursos, consumirem e/ou produzirem artefatos e adotarem procedimentos durante suas realizações. Ainda, as atividades podem ser decompostas em outras atividades (ditas subatividades) e, também, podem depender de outras atividades (ditas pré-atividades).

Os artefatos são produtos de software que podem ser consumidos ou produzidos durante a execução de uma atividade (FALBO, 1998). Como exemplo, podem-se citar diagramas de classes e código fonte. Já os procedimentos são condutas bem estabelecidas utilizadas para realização de uma atividade (FALBO, 1998), tais como roteiros de documentos e normas de programação.

A área de processos de software, como uma disciplina autônoma, surgiu em meados da década de 80. Ao longo desses anos, vários esforços vêm sendo realizados,

com destaque para a criação de padrões de qualidade de processo (FUGGETTA, 2000), tais como a norma ISO/IEC 12207 (ISO/IEC, 2008) e o modelo de maturidade CMMI (SEI, 2006).

O conceito de modelo de ciclo de vida está relacionado diretamente com a noção de processo de software, definindo os principais estágios na vida do produto de software e suas relações. Tipicamente um modelo de ciclo de vida envolve as seguintes atividades: especificação e análise dos requisitos, projeto, implementação, verificação e validação, distribuição, operação, manutenção e retirada de uso (FUGGETTA, 2000).

2.3. Definição de Processo

A qualidade de um produto de software depende fortemente da qualidade do processo de software utilizado em seu desenvolvimento (FUGGETTA, 2000). Portanto, torna-se fundamental ter um processo de software para se obter produtos de software de qualidade. Entretanto, se por um lado ter processos de software é fundamental, seguir um processo de software mal definido pode ocasionar danos consideráveis ao andamento do projeto. Logo, é essencial possuir processos de software bem definidos.

Sabe-se que os projetos possuem características distintas e, portanto, para ser efetivo e conduzir a produtos de qualidade, um processo deve ser adequado às características específicas do projeto, tais como o tipo de software a ser desenvolvido, o paradigma, o domínio de aplicação e características da equipe. Por outro lado, mesmo conhecendo as especificidades de cada projeto, é possível definir um conjunto de elementos que devem estar presentes em todos os processos dos projetos de uma organização. Esse conjunto de elementos, na maioria das vezes proveniente da cultura organizacional, é o que se convenciona chamar de processo padrão da organização. Processos padrão contemplam apenas os ativos de processo essenciais (atividades, artefatos, recursos e procedimentos) que devem ser incorporados a quaisquer processos da organização (BERTOLLO, 2006).

Os processos padrão são vistos como elementos primordiais das normas e modelos de maturidade e o uso deles como base para o planejamento dos processos de software específicos de projetos torna mais direta a definição dos planos em conformidade com os padrões de qualidade da organização (BERGER, 2003).

Uma abordagem flexível de definição de processos de software envolve três níveis de abstração (ROCHA et al., 2001): processos padrão, processos especializados e processos de projeto. A Figura 2.1 mostra esquematicamente essa abordagem.

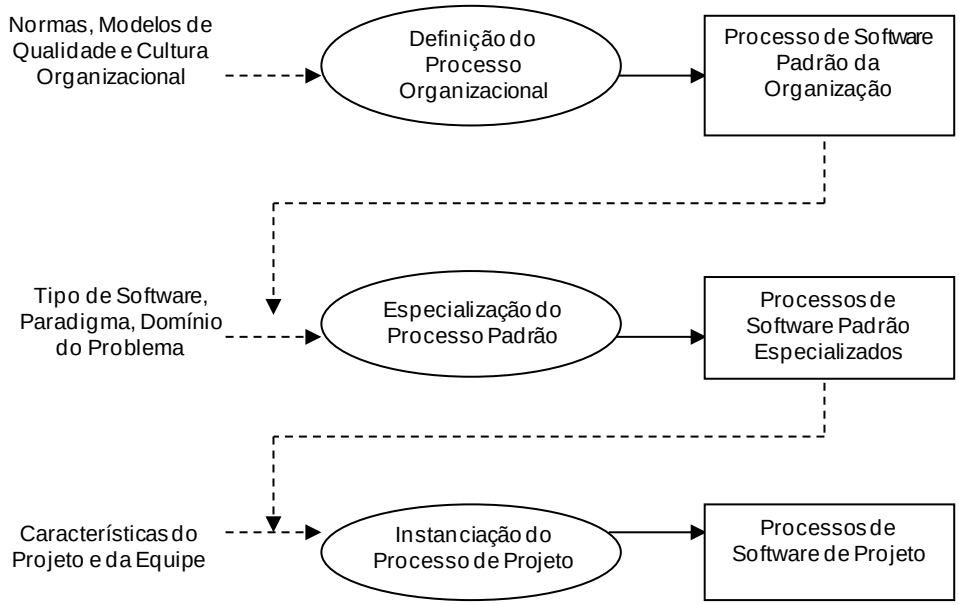

Figura 2.1 – Abordagem em Níveis para a Definição de Processos (BERTOLLO et al., 2006).

No nível mais alto, a organização define seu processo de software padrão, contemplando apenas os ativos de processo essenciais (atividades, artefatos, recursos e procedimentos) que devem ser incorporados a quaisquer processos da organização. O processo padrão da organização pode ser definido considerando padrões e modelos de qualidade, como o CMMI e a ISO/IEC 12207. Assim, os processos padrão estabelecem uma estrutura comum para os processos específicos de projetos, servindo como ponto de partida para suas definições.

Para acomodar todos os possíveis processos de projetos de uma organização, um processo padrão tende a estar em um alto nível de abstração, requerendo muito esforço para que um gerente de projeto o adapte para seu projeto específico. Entretanto, em uma organização pode ocorrer de muitos projetos serem realizados para um mesmo tipo de software (sistemas de informação, aplicações Web etc) ou seguindo um mesmo paradigma (por exemplo, orientado a objetos). Assim, no nível intermediário, o processo padrão da organização pode ser especializado para considerar algumas classes de tipos de software, paradigmas ou domínios de aplicação. Durante a especialização, ativos de processos podem ser adicionados ou modificados de acordo com o contexto da

especialização. Ainda que especializados, esses processos continuam sendo padrões organizacionais, só que agora particularizados para uma classe típica de projetos da organização.

No último nível surgem os processos específicos de projetos. Suas definições se dão a partir da instanciação de processos padrão, especializados ou não, para os contextos dos projetos. Durante a instanciação do processo de projeto, particularidades desse projeto e características da equipe devem ser consideradas. Nesse momento, o modelo de ciclo de vida deve ser definido e novas atividades, assim como artefatos consumidos e produzidos, recursos requeridos e procedimentos, podem ser adicionadas.

Com base no exposto, pode-se perceber que a definição de processos de software não é uma tarefa trivial, haja vista que envolve muitos fatores, tais como necessidades e características da organização e características específicas de projetos. Dessa maneira, são exigidos profissionais experientes e com conhecimento nesses diversos aspectos, que nem sempre estão disponíveis na organização. Neste contexto, a reutilização de processos surge como uma alternativa para diminuir o esforço necessário para definição de novos processos de software (ISODA, 1992).

2.4. Reutilização de Software

A preocupação em tentar reutilizar partes de produtos de software já existentes no desenvolvimento de novos produtos é de longa data. Em 1968, durante a Conferência de Engenharia de Software da OTAN, surgiu a ideia de decompor sistemas de software em unidades reutilizáveis, ditas componentes. MCILROY (1968) argumentou que deveria haver empenho em produzir componentes reutilizáveis com o intuito de facilitar a tarefa dos desenvolvedores de software. Ele propôs a criação de uma indústria de componentes normalizados para que aplicações complexas pudessem ser desenvolvidas a partir de rotinas disponíveis em catálogos.

Segundo FREEMAN (1987), reúso é a utilização de qualquer informação já disponível que um desenvolvedor necessite no processo de criação de software. Já TRACZ (1995) diz que reúso é a utilização de software projetado para ser reutilizado. Há uma discussão importante no contexto da reutilização de software sobre a possibilidade de modificação do software a ser reutilizado. COOPER (1994) define reutilização de software como a capacidade de um componente de software previamente desenvolvido ser reutilizado, com ou sem modificação, em parte ou no todo. Em uma

visão mais geral, a reutilização de software pode ser vista como o processo de criação de software a partir de software já existente, ao invés de construí-lo do zero (KRUEGER, 1992).

O reúso tem impactos positivos tanto na qualidade dos produtos de software, quanto em seus custos e produtividades. A reutilização de software resulta em melhorias de qualidade, produtividade, confiabilidade e interoperabilidade (SAMETINGER, 1997).

No que tange à melhoria de qualidade, a cada reutilização, potenciais erros no componente podem ser identificados e consertados. Com isso, componentes reutilizados tendem a possuir maior qualidade do que aqueles desenvolvidos e utilizados apenas uma única vez. Contudo, para que essa melhoria seja alcançada, é necessário que exista um controle sobre os componentes e que os mesmos sofram manutenções. Ainda, a utilização de um componente diversas vezes aumenta as chances de detecção de erros nesse componente, reforçando a confiança em sua reutilização (SAMETINGER, 1997).

Com relação à produtividade, existe um ganho no sentido de que menos código precisa ser desenvolvido e, consequentemente, despende-se menos esforço em testes também. Uma ressalva é feita para a fase de implantação do reúso, em que pode haver uma perda de produtividade, em virtude da necessidade de se desenvolver os componentes reutilizáveis (SAMETINGER, 1997).

No quesito desempenho, o reúso extensivo pode ser benéfico à otimização. A partir das reutilizações de um componente, melhorias podem ser identificadas e implementadas visando a um melhor desempenho. Por outro lado, as generalizações utilizadas para aumentar as chances de reutilização de um componente podem onerar o seu desempenho (SAMETINGER, 1997).

Por fim, a padronização facilita que sistemas se comportem de maneira mais uniforme, aumentando a interoperabilidade. Esse é o caso quando os mesmos componentes são utilizados. Embora os padrões descritos melhorem a interoperabilidade, diferentes implementações podem surgir de diferentes interpretações de partes desses padrões. A utilização de um mesmo componente em sistemas diferentes leva a um comportamento mais uniforme, aumentando a interoperabilidade (SAMETINGER, 1997).

Além dos benefícios de melhoria de qualidade, SAMETINGER (1997) também aponta alguns benefícios sobre a redução do esforço necessário ao desenvolvimento, a saber:

- Trabalho redundante / tempo de desenvolvimento: ao se desenvolver todos os sistemas do zero, trabalhos redundantes são realizados, como o desenvolvimento de interfaces com usuário e algoritmos básicos. Ao se disponibilizar essas partes como componentes reutilizáveis, evita-se esse trabalho redundante e, por conseguinte, reduz-se o tempo de desenvolvimento;
- Documentação: mesmo sendo considerada importantíssima para manutenção dos sistemas, muitas vezes a documentação é tratada com negligência. A reutilização de componentes de software reduz a documentação a ser escrita, uma vez que apenas a estrutura geral do sistema e os novos componentes desenvolvidos necessitam ser documentados;
- Custos de manutenção: espera-se que poucos defeitos sejam encontrados ao se utilizar componentes reutilizáveis – uma vez que eles já passaram por diversos testes – e, consequentemente, menos esforço de manutenção para o sistema é esperado. Os componentes reutilizáveis são mantidos por um grupo separado;
- Tamanho da equipe: equipes grandes sofrem com o problema da comunicação interna. A produtividade de uma equipe não aumenta proporcionalmente ao tamanho da equipe e, com isso, dobrar o seu tamanho não acarreta necessariamente em dobrar a produtividade. Se boa parte dos componentes de um sistema puder ser reutilizada, ele pode ser desenvolvido por uma equipe menor.

Mesmo com todos os benefícios que a reutilização pode trazer, a adoção dessa prática não é tão simples e imediata. É necessário um grande esforço para mudar a cultura organizacional com o intuito de institucionalizá-la. GRISS (1995) aponta que são necessárias alterações no negócio, nos processos, no gerenciamento e na organização para:

- Financiar o projeto e a construção de famílias de produtos que otimizem pontos importantes do negócio, tais como tempo de entrega e custo de desenvolvimento;
- Criar, manter e gerenciar ativos reutilizáveis e seus respectivos repositórios;
- Separar os desenvolvedores dos ativos reutilizáveis dos utilizadores desses ativos (desenvolvedores de aplicações);
- Introduzir novos papéis de gerenciamento e responsabilidades, e processos de desenvolvimento;

- Criar e disseminar treinamentos, tecnologias e ferramentas.

Ao se pensar em reutilização de componentes de software, logo vem à cabeça a reutilização de código ou componentes já compilados. Quando alguém, por exemplo, se depara com um problema que requer um algoritmo de ordenação, nenhum tempo é despendido para tentar criar uma nova solução; recorre-se rapidamente a um dos vários algoritmos de ordenação já difundidos. Entretanto, os benefícios alcançados com a reutilização podem ser ainda maiores quando aplicados em outras fases do processo de desenvolvimento, reutilizando-se também artefatos de níveis mais altos de abstração (GUIZZARDI, 2000).

Contudo, alcançar a reutilização de informações em etapas iniciais do processo de desenvolvimento, tais como análise e projeto, não é simples, uma vez que essas informações não costumam estar organizadas de forma coerente e de fácil reutilização. Dentre os problemas que afetam essas etapas de desenvolvimento de software e mostram a necessidade de criação de métodos que organizem o conhecimento da área, destacam-se (WERNER e BRAGA, 2006): desconhecimento e/ou conhecimento errôneo do domínio da aplicação; vocabulário conflitante; e incerteza nas decisões devido à falta de conhecimento sobre a área.

Assim, faz-se necessário que o processo de desenvolvimento de software sofra alterações no sentido de incorporar etapas que visem explicitamente à reutilização, a fim de difundi-la entre todas as atividades do desenvolvimento, envolvendo a reutilização de artefatos em diversos níveis de abstração (p. ex., desde modelos de estrutura até código fonte). Em linha com essas necessidades, na seção seguinte é apresentada a abordagem de Desenvolvimento Baseado em Componentes (DBC).

2.5. Desenvolvimento Baseado em Componentes (DBC)

A principal ideia da abordagem de Desenvolvimento Baseado em Componentes (DBC) é construir sistemas a partir de componentes já existentes, o que traz algumas consequências para o processo de desenvolvimento, dentre elas (CRNKOVIC et al., 2006):

- Separação entre o processo de desenvolvimento de sistemas baseados em componentes e o processo de desenvolvimento de componentes: os componentes já devem ter sido desenvolvidos para serem utilizados em outros produtos;

- Surgimento de três importantes etapas: busca, avaliação e adaptação de componentes;
- Diferença de foco das atividades do processo quando comparadas às do processo de desenvolvimento tradicional: no desenvolvimento de sistemas baseados em componentes existe uma grande ênfase na procura por componentes adequados. O desenvolvimento de componentes tem como foco principal o desenvolvimento de componentes reutilizáveis.

Com o intuito de promover a reutilização em todo o processo de desenvolvimento, SAMETINGER (1997) define algumas etapas que devem integrar as atividades do processo de desenvolvimento baseado em componentes:

1. Entendimento do Problema: Estudar o problema e desenvolver uma estrutura de solução baseada em componentes já existentes;
2. Reconfiguração: Adaptar a estrutura de solução para aumentar a possibilidade de se utilizar componentes disponíveis;
3. Busca: Obter, avaliar e instanciar componentes já existentes;
4. Adaptação: Modificar e adaptar os componentes selecionados que não atendam completamente aos requisitos;
5. Integração: Integrar os componentes ao produto final;
6. Avaliação: Avaliar os componentes que precisaram ser desenvolvidos e os que foram modificados quanto à possibilidade de reutilização e inserção no repositório de componentes.

Para que haja reúso efetivo, é necessário que haja um processo que dê ênfase às duas perspectivas do desenvolvimento baseado em componentes: desenvolvimento de componentes e desenvolvimento de sistemas a partir de componentes.

O desenvolvimento de componentes deve ser voltado para reutilização (CRNKOVIC et al., 2006). É desejável que um componente reutilizável atenda a características como: ser geral e autocontido; ter alta coesão e dependência mínima de outros componentes; utilizar modelos de arquiteturas padrão e interfaces padronizadas; haver possibilidade de parametrização; haver independência de hardware, compilador e sistema operacional; e possui documentação adequada (SAMETINGER, 1997).

Entretanto, a criação de componentes reutilizáveis é uma das grandes dificuldades na reutilização de software. Como alternativa para solucionar esse problema, surge a Engenharia de Domínio, auxiliando no desenvolvimento de componentes reutilizáveis em um dado domínio (GIMENES e HUZITA, 2005).

Componentes de software podem ser desde artefatos de código fonte até artefatos mais abstratos, como modelos de estrutura e especificações (KRUEGER, 1992). Assim, para que a reutilização seja efetiva, é necessário considerá-la em todas as fases do desenvolvimento de sistemas.

Um processo de desenvolvimento baseado em componentes (DBC) deve conter caminhos paralelos, que permitam ocorrer, simultaneamente, a engenharia de domínio e o desenvolvimento baseado em componentes (PRESSMAN, 2006). A engenharia de domínio busca estabelecer componentes de software que possam ser reusados ao longo de todas as etapas do DBC. É fundamental que os componentes reutilizáveis sejam consistentes ao longo de todo o processo de desenvolvimento, ou seja, um componente de software reutilizável utilizado durante a especificação do sistema, por exemplo, deve ser compatível com os componentes utilizados durante as etapas de projeto e implementação (GIMENES e HUZITA, 2005). A Figura 2.2 ilustra um modelo de processo que abrange essas duas perspectivas.

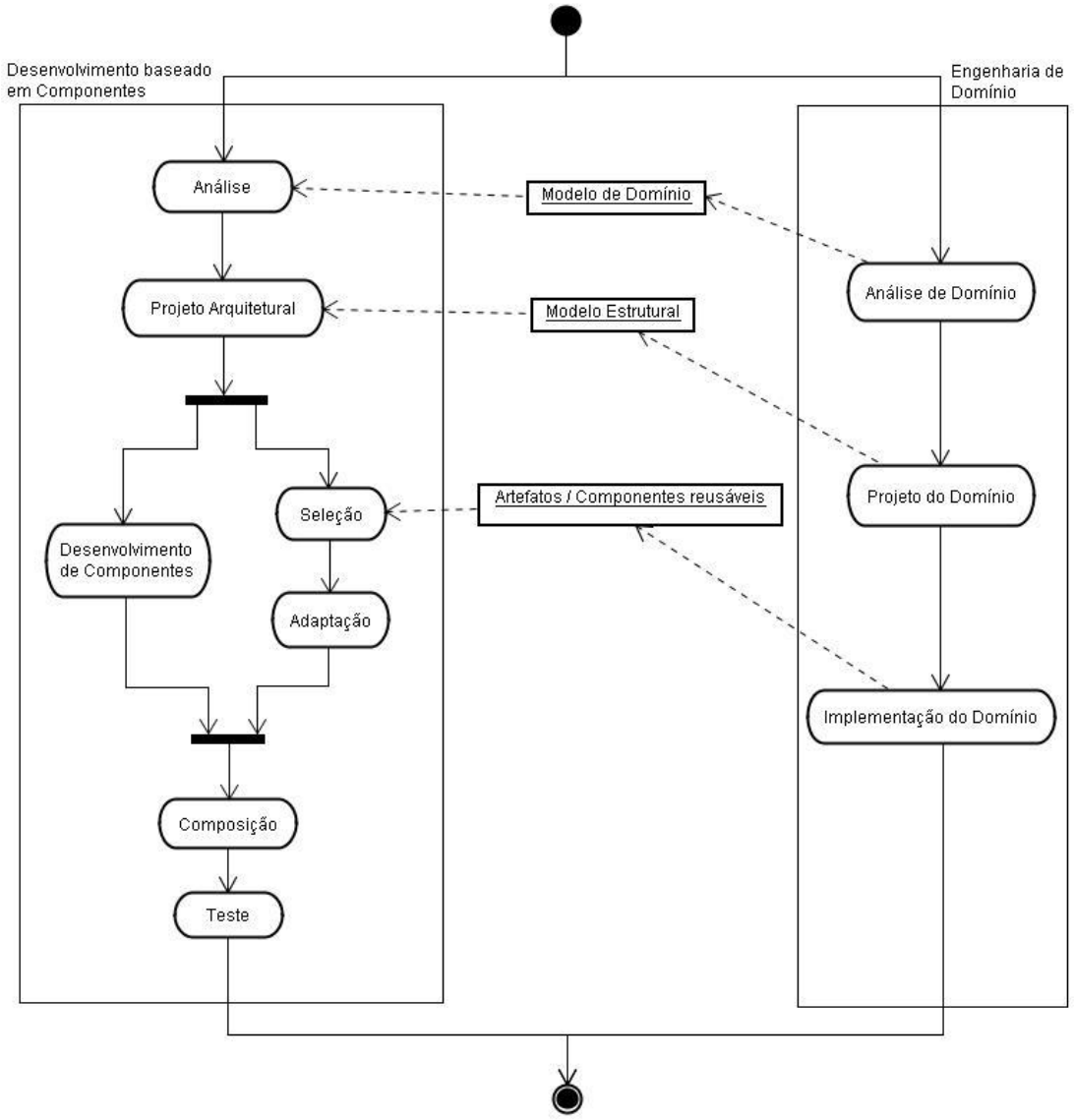

Figura 2.2 – Modelo de Processo de Desenvolvimento Baseado em Componentes (adaptada de PRESSMAN, 2006).

São os seguintes os objetivos das macroatividades propostas para a Engenharia de Domínio (PRESSMAN, 2006) (GIMENES e HUZITA, 2005):

- Análise do Domínio: especificar os requisitos comuns das aplicações de um domínio, buscando identificar as oportunidades de reutilização. Deve-se capturar o contexto e a abrangência do domínio. Os principais conceitos e funcionalidades devem estar explicitados nos artefatos gerados durante esta atividade.
- Projeto do Domínio: identificar e generalizar soluções para os requisitos levantados durante a análise do domínio, especificando uma arquitetura de

software do domínio. Modelos de especificação da estrutura arquitetural que deve ser seguida pelas aplicações do domínio devem ser gerados.

- Implementação do Domínio: implementar os componentes gerados pelo projeto. Tais componentes devem abranger as principais funcionalidades encontradas em aplicações do domínio em questão. Eles devem estar em conformidade com os modelos de abstração gerados durantes as atividades de análise e projeto do domínio.

Já no desenvolvimento de aplicações, as atividades têm os seguintes propósitos (GIMENES e HUZITA, 2005):

- Análise: obter um entendimento sobre o quê será o sistema, especificando-se seus requisitos. Os modelos gerados durante a atividade de análise do domínio devem ser utilizados como base para a análise dos requisitos do sistema.
- Projeto: definir a arquitetura geral do sistema e de seus componentes. A estrutura arquitetural proposta para as aplicações do domínio durante a atividade de projeto do domínio deve servir de entrada para esta atividade.
- Seleção: selecionar componentes que atendam aos requisitos do sistema em desenvolvimento e que sejam compatíveis com a arquitetura de software definida.
- Adaptação: ajustar os componentes selecionados de forma a cobrirem os requisitos do sistema e solucionar possíveis conflitos identificados entre componentes.
- Desenvolvimento de Componentes: desenvolver os componentes necessários para se cobrir os requisitos do sistema que não foram atendidos pelos componentes selecionados para reutilização.
- Composição: combinar os componentes selecionados, adaptados e desenvolvidos para compor o sistema. Deve-se desenvolver uma infraestrutura que integre os componentes, fornecendo um modelo de coordenação.
- Teste: testar os componentes em conjunto e a infraestrutura de integração desenvolvida durante a composição. Espera-se que os componentes utilizados para composição do sistema tenham sido testados durante seus desenvolvimentos. Entretanto, mesmo que os componentes já tenham sido

testados separadamente, é necessário testá-los em conjunto, uma vez que o funcionamento dos componentes isoladamente não garante o funcionamento do sistema.

2.6. Reutilização de Processos de Software

Existem na literatura diversos trabalhos que advogam a semelhança entre processos de software e produtos de software, fortemente inspirados pelo trabalho precursor de OSTERWEIL (1987), sendo possível aplicar, inclusive, técnicas e métodos de desenvolvimento de software na definição de processos de software (GROENEWEGEN et al., 1996).

KELLNER (1996) aponta que o conhecimento existente em reutilização de produtos de software pode ser aplicado na reutilização de processos. Alguns exemplos desse conhecimento são:

- Arquiteturas povoadas com componentes reutilizáveis;
- Gerenciamento de repositórios para armazenar, catalogar, procurar e acessar os ativos reutilizáveis;
- Gerência de configuração de ativos reutilizáveis.

Diversos trabalhos têm sido desenvolvidos na área de reutilização de processos de software, sendo que eles podem ser divididos em dois grandes grupos (JAUFMAN e MÜNCH, 2005): aqueles focados na componentização de processos, pregando a definição de processos a partir da composição de componentes de menor granularidade, e os que seguem a linha de padrões de processo, *templates* de processo e arquiteturas de linhas de processo, em que processos específicos são definidos a partir de processos genéricos.

Segundo JAUFMAN e MÜNCH (2005), a principal deficiência da abordagem de definição de processos baseada em componentes é a falta de apoio à adaptação dos processos, sendo a sua principal vantagem o fato de permitir que apenas partes do processo sejam reutilizadas. Por outro lado, a abordagem de adaptação de processos genéricos tem como principal vantagem o apoio à adaptação dos processos, tornando-os mais consistentes. Entretanto, essas abordagens não costumam tratar a reutilização de fragmentos de processo.

Nas subseções seguintes são apresentadas as idéias principais de reúso de processo segundo essas duas linhas de pesquisa.

2.6.1. Componentização de Processos de Software

Alguns modelos de qualidade de processo, notadamente o CMMI (SEI, 2006) e o MPS.BR (SOFTEX, 2009), apresentam a ideia de que organizações com alto grau de maturidade devem ser capazes de definir seus processos a partir de subprocessos, elementos de processos ou componentes de processo. Os processos devem ser definidos através da seleção dos subprocessos mais adequados para compô-lo, considerando, inclusive, a estabilidade e a capacidade desses subprocessos.

GARY e LINDQUIST (1999) definem um componente de processo como um encapsulamento de informações e comportamentos de processo em um dado nível de granularidade. Ainda segundo eles, um modelo de processo definido com base em componentes é uma coleção de componentes de processo que interagem de alguma forma. Os modelos de processo definidos com base em componentes também são vistos como componentes de processo. FUSARO et al. (1998) reafirmam essa idéia de componentização de processos de software, tratando um processo como a integração de um conjunto de componentes de processo em diferentes níveis de granularidade.

Entretanto, para que se possam definir processos de software a partir de componentes de processo pré-existentes, alguns problemas precisam ser tratados (FUSARO et al, 1998):

- É necessário que existam componentes de processo descritos com o mesmo formalismo utilizado na descrição dos processos;
- Deve ser possível integrar os componentes de processo aos processos;
- Deve existir um mecanismo para seleção e escolha do componente mais adequado, quando existirem mais de um componente com o mesmo propósito.

Ainda nesse sentido, segundo BASILI e ROMBAC (1987), definir processos através da composição de componentes de processo já existentes exige um mecanismo adequado para avaliação do componente quanto à sua aplicação no contexto específico. Deve-se ser capaz de verificar se um componente pré-existente pode ser utilizado para a definição de um determinado processo, atendendo aos requisitos estabelecidos (RU-ZHI et al., 2005). Para que os componentes possam ser avaliados e utilizados na composição de novos processos, é necessário que haja uma forma adequada de representá-los. Além disso, é preciso que eles sejam definidos de forma compatível para tornar possíveis suas conexões (KELLNER, 1996).

A seguir são apresentados alguns trabalhos relatados na literatura que dizem respeito à reutilização de processos com uma visão de componentes de processos de software.

REP – *ChaRacterizing and Exploiting Process Components*

REP (FUSARO et al., 1998) é um *framework* de apoio à compreensão e avaliação de componentes de processo de software. Ele tem por objetivo auxiliar o engenheiro de software a: (i) reconhecer os componentes de processo que satisfazem aos requisitos do processo, (ii) avaliá-los e (iii) selecionar aquele que melhor lhe atende.

Para que os componentes possam ser analisados, é necessário que estejam formalizados. Nesse sentido, REP utiliza a PCM (*Process Conceptual Modeler*), uma ferramenta do ambiente PROMETHEUS (*PROcess Model Evolution Through Experience Unfolded Systematically*) (CIMITILE e VISAGGIO, 1994).

O *framework* se baseia em duas fases, uma de especificação e outra de avaliação. A primeira delas visa determinar se um componente analisado satisfaz aos requisitos. São utilizados cinco grupos de características: (i) Entrada (*Input*) – define as características dos objetos necessários para operação do componente; (ii) Saída (*Output*) – define as características dos objetos produzidos pelo componente; (iii) Ênfase (*Emphasis*) – identifica os objetivos do componente; (iv) Domínio (*Domain*) – define as características que devem ser satisfeitas pelos objetos de entrada para que o componente possa ser executado; e (v) Ambiente (*Environment*) – define o estado inicial do ambiente externo necessário para o início da execução do componente e o estado final produzido pela sua execução.

Durante a fase de avaliação, os componentes de processo que satisfazem aos requisitos são avaliados e se definem os riscos em adotar cada um deles no processo de software. São considerados três conjuntos de categorias: (i) Técnicas – para delinear as características tecnológicas do componente; (ii) Formalismo – para indicar o rigor com o qual o componente é definido; e (iii) Qualidade – para expressar a qualidade do componente de processo e de seus produtos.

OPC – *Open Process Components*

A abordagem OPC (GARY e LINDQUIST, 1999) propõe um *framework* baseado em componentes de processo para apoiar as atividades de construção, execução, análise e evolução de modelos de processo. Essa abordagem não resolve

todos esses problemas em si, mas fornece uma plataforma através da qual questões de apoio à automatização de processos podem ser tratadas de forma mais profunda.

A OPC modulariza os modelos de processo visando definir as fronteiras entre os fragmentos de processo. Ela enumera objetos no domínio de processos, a fim de encapsular informações e comportamentos. As dependências não são eliminadas, mas sim movidas para as fronteiras dos componentes, onde é mais fácil reconhecê-las e tratá-las. A visão da OPC é de um ambiente de componentes de processo distribuído, em que os componentes podem ser localizados e adaptados em decorrência de um conjunto de requisitos de processo.

São três os principais tópicos abordados pela OPC (GARY e LINDQUIST, 1999):

- Componentes de processo: um componente de processo pode ser visto como informações e comportamentos de processo encapsulados em uma determinada granularidade;
- Modelo de processo baseado em componentes: uma coleção de componentes de processo que interagem entre si. Também é um componente de processo;
- Execução de modelos de processo: refere-se à criação e evolução do componente, assim como à interação dele com outros componentes e com o ambiente.

Representação de Componentes de Processo orientada a Reúso

Esta abordagem foi proposta por RU-ZHI et al. (2005) para a descrição e classificação de componentes de processo. Ela inclui três níveis de descrição de informações sobre componentes: (i) descrição das informações gerais do componente de processo (PGD – *Process Component General Information Description*), (ii) descrição da especificação do componente de processo (PSD – *Process Component Specification Description*) e (iii) descrição dos dados do componente de processo (PDD – *Process Component Data Description*). As informações sobre os componentes foram divididas em três níveis, pois diferentes tipos de usuários buscam diferentes tipos de informação.

A descrição das informações gerais do componente de processo apresenta os objetivos do componente e algumas informações específicas, tais como domínios de aplicação e classificação. Já a descrição da especificação do componente de processo descreve as atividades do componente e seus relacionamentos, os artefatos (produtos e insumos), recursos necessários, pré-condições e pós-condições. Por fim, a descrição dos

dados do componente de processo apresenta as informações necessárias para o gerenciamento da execução do componente, tais como esforço necessário, custo de execução e riscos associados.

A abordagem conta também com uma etapa de recuperação de componentes, em que são buscados componentes candidatos ao reúso em um determinado projeto. Para efetuar a busca, deve-se formular a consulta desejada. Quanto maior o nível de detalhamento da consulta, maiores são as chances de se encontrar um componente de processo mais próximo do desejado. Assim, consultas vagas tendem a retornar componentes com um alto grau de abstração.

Uma Abordagem para Definição de Processos Baseada em Reutilização

BARRETO et al. (2008) apresentam uma abordagem para definição de processos baseada em reutilização que tem por objetivo facilitar a definição de processos através de sua composição a partir de componentes de processo reutilizáveis. Os autores consideram que um componente de processo pode ser definido com qualquer nível de detalhamento, isto é, desde uma simples atividade até um processo completo.

A abordagem prega, ainda, a utilização de repositórios para o armazenamento de componentes de processo, linhas de processo, conhecimento relacionado à execução dos processos, medições relacionadas ao uso dos componentes, entre outros. Esses repositórios podem ser utilizados durante a definição de processos para as organizações ou projetos (definição com reúso), em que componentes e outros itens reutilizáveis podem ser buscados e usados para compor os processos.

Depois de certo tempo, o repositório pode estar demasiadamente povoado, dificultando a busca por componentes adequados. A abordagem preconiza, então, a utilização das características de processo para amenizar esse problema e guiar a definição do processo baseando-se nos requisitos existentes. As características restringem o uso dos componentes, estabelecendo um conjunto de componentes que podem ser utilizados em uma determinada situação.

A abordagem contempla também uma forma de componentização de processos legados das organizações como uma possível forma de povoar os repositórios de reutilização (BARRETO et al., 2009). O termo processo legado é utilizado pelos autores para designar quaisquer processos de software existentes na organização que precisem passar por adaptação para se tornarem mais adequados à reutilização de processos. A componentização dos processos legados passa por quatro etapas principais, a saber: (i)

Definir componentes de processos; (ii) Definir características de processos; (iii) Definir uma linha de processos; e (iv) Aprovar inclusão de elementos reutilizáveis na biblioteca de componentes.

2.6.2. Adaptação de Modelos de Processo Genéricos

Os processos genéricos fornecem modelos de solução para problemas comuns em um nível elevado de abstração (WANG e KING, 2000). Utilizar processos genéricos para definição dos processos de projeto é uma alternativa para facilitar essa tarefa. Entretanto, para que isso seja possível, é necessário que haja mecanismos de definição dos processos genéricos e de adaptação dessas soluções para os projetos específicos (REIS, 2002).

Assim, alguns tópicos têm sido alvo de estudos, visando desenvolver mecanismos que facilitem a definição dos processos genéricos e a adaptação desses processos para os projetos, levando em consideração suas especificidades.

HUANG e ZHANG (2003) argumentam que padrões (*patterns*) têm sido amplamente utilizados para representarem soluções de análise e projeto de sistemas e que eles também poderiam ser utilizados para documentar processos de software. Assim, um padrão de processo pode ser visto como um padrão que descreve uma abordagem ou um conjunto de ações comprovadamente bem-sucedido para o desenvolvimento de software (AMBLER, 1998).

Similar aos padrões de processo, os *templates* de processo são soluções para um conjunto de problemas semelhantes, os quais não possuem informações relacionadas a uma organização específica (FRANCH e RIBÓ, 2002). Um *template* de processo pode ser adaptado para os requisitos de um determinado projeto, ou ainda combinado com outros *templates* (COSTA et al., 2007).

JAUFMAN e MÜNCH (2007) também apontam para a necessidade de se adaptar esses processos definidos em um alto grau de abstração (processos padrão, padrões de processo, *templates* etc). Entretanto, ao se adaptar esses processos para contextos específicos, variações do processo começam a surgir. Embora cada uma dessas variações corresponda a um processo distinto, esses processos podem ser considerados como variações uns dos outros, ao invés de tratá-los como processos completamente diferentes (SIMIDCHIEVA et al., 2007). Assim, outro foco de estudo

de reutilização de processos são as linhas de processos e as arquiteturas de linhas de processos.

Uma linha de processos é um conjunto de processos de um domínio específico ou para um propósito particular, que possuem características comuns (WASHIZAKI, 2006). Ainda segundo WASHIZAKI (2006), uma arquitetura de linha de processos é uma estrutura de processos que reflete os aspectos comuns e as variabilidades de uma linha de processos.

A ideia é que processos sejam definidos para um contexto específico a partir das arquiteturas de linhas de processos (JAUFMAN e MÜNCH, 2005). Uma arquitetura possui um processo central, que é comum a todos os processos que têm origem nela, mas também possui pontos de variação. Os pontos de variação são atividades que podem ser alteradas de acordo com as características do projeto. As variantes de processo são as atividades candidatas a serem aplicadas nos pontos de variação. Os pontos de variação e as variantes de processo representam, juntos, a variabilidade da arquitetura da linha de processos.

A seguir são apresentados alguns trabalhos relatados na literatura que dizem respeito à reutilização de processos com essa visão.

HPP – *Hierachical Process Pattern*

O framework HPP (HUANG e ZHANG, 2003) fornece meios para a modelagem de processos de software baseada em padrões de processo. Esse *framework* contempla três tipos de padrões:

- Padrão de ciclo de vida: mostra o modelo de ciclo de vida no qual se deve mapear as atividade para produzir o modelo do processo (p. ex., cascata e incremental);
- Padrão de atividade: mostra o esqueleto do trabalho a ser executado, incluindo seus artefatos de entrada e saída;
- Padrão de *workflow*: mostra uma possível ordem de execução de um conjunto de atividade ordenadas.

Para auxiliar o entendimento e a escolha dos padrões, deve haver um método eficiente para descrevê-los. Assim, os autores definiram um conjunto de notações para representá-los, sendo este uma adaptação da UML (*Unified Modeling Language*).

WebAPSEE-Reuse

A ferramenta WebAPSEE-Reuse (COSTA et al., 2007) foi construída visando à reutilização de processos de software. Para tal, ela utiliza o conceito de *templates* de processo para representar modelos de processo abstratos e utiliza um mecanismo para modelagem de *templates*, uma máquina de estados para os *templates* e um conjunto de funcionalidade agregadas a esses modelos.

WebAPSEE-Reuse possui uma funcionalidade que permite reutilizar os *templates* definidos por meio de instanciações para um contexto específico. As instâncias dos *templates* podem, ainda, serem adaptadas para melhor atenderem às necessidades dos projetos. Ainda na linha da reutilização de processos, WebAPSEE-Reuse possui outra funcionalidade que permite utilizar um *template* para compor um fragmento de um modelo de processo de projeto em definição. Por fim, essa ferramenta possui uma funcionalidade para generalização de um modelo de processo definido para um projeto, dando origem a um novo *template*.

EPAc – *Emergent Process Acquisition*

EPAc (JAUFMAN e MÜNCH, 2005) é um método para definição de processos específicos de projeto, que utiliza uma linha de processos específica de domínio para definir esses processos a partir da adaptação de processos genéricos. Além disso, o método trata do refinamento do processo adaptado conforme sua execução vai ocorrendo.

O método consiste em duas etapas principais. Primeiramente, uma linha de processos específica de domínio é utilizada para a definição do processo de projeto. O propósito da linha de processos específica de domínio é prover conhecimento sobre o domínio necessário para a definição do processo de software. Em seguida, o processo adaptado é refinado conforme sua execução, considerando os dados gerados por ela.

Assim, existem dois momentos distintos: a definição das linhas de processo da organização e a definição dos processos de projeto. Uma linha de processo específica de domínio é criada a partir da seleção das variantes de processo no repositório de processos mais adequadas para o domínio em questão, conforme descrito em (JAUFMAN, 2005). Já para a definição do processo de um projeto, é escolhida a variante de uma determinada linha de processo mais adequada para o projeto. Essa variante é adaptada, originando o processo específico do projeto.

Técnica "*Bottom-up*" de Construção de Arquiteturas de Linhas de Processo

WASHIZAKI (2006) propõe uma técnica para construção de arquiteturas de linhas de processos de software utilizando conhecimento existente em definição de processos de um domínio.

Antes de se construir de fato uma arquitetura de linha de processos (PLA – *Process Line Architecture*), devem-se selecionar processos existentes para o domínio em questão. Esses processos, que devem conter partes em comum, dão origem à linha de processo.

Depois, os pontos comuns da linha de processo originada são definidos como o processo central da PLA em construção, incluindo os pontos de variação. A variabilidade é definida como o conjunto de variantes para cada ponto de variação existente no processo central da PLA.

Uma extensão do SPEM (*Software Process Engineering Metamodel*) (OMG, 2007) é utilizada para descrição das PLAs. Essa extensão foi desenvolvida para melhor expressar os pontos comuns e a variabilidade de um processo.

2.7. Definição de Processos de Software em ODE

O ambiente ODE (*Ontology-based software Development Environment*) (FALBO et al., 2003) é um ambiente de desenvolvimento de software (ADS) centrado em processos (ARBAOUI et al., 2002) (GRUHN, 2002) e baseado em ontologias (FALBO et al., 2004a). A premissa em que ODE se baseia é a de que, se as ferramentas são construídas baseadas em ontologias, a sua integração pode ser facilitada, pois os conceitos envolvidos estão bem definidos nas ontologias (FALBO et al., 2004a).

Por ser um ambiente de desenvolvimento de software centrado em processos (ADSCP), a principal ontologia de ODE é uma ontologia de processos de software. Essa ontologia foi originalmente desenvolvida em (FALBO, 1998), tendo sido alvo de evoluções recentes em (BERTOLLO, 2006) e (GUIZZARDI et al., 2008). Contudo, outras ontologias estão envolvidas no contexto do ambiente.

O ambiente é constituído de várias ferramentas, tais como ferramenta de apoio à definição de processos (BERTOLLO et al., 2006) (SEGRINI et al., 2006), de acompanhamento de projetos (DAL MORO, 2005), de gerência de requisitos (MARTINS et al., 2006), de gerência de riscos (FALBO et al., 2004b) e de gerência de conhecimento (NATALI et al., 2003).

A ferramenta de definição de processos vem evoluindo ao longo do tempo. Em sua primeira versão, a ferramenta permitia apenas a definição do processo de desenvolvimento de um projeto específico, tomando por base um conhecimento previamente cadastrado no ambiente. Esse conhecimento era organizado segundo a ontologia de processo de software definida em (FALBO, 1998). Em uma segunda versão (BERTOLLO e FALBO, 2003), essa abordagem foi estendida para permitir a definição de processos padrão e especializados, e a instanciação desses para um projeto específico. Porém, apenas o processo de desenvolvimento continuava sendo contemplado.

Em sua versão mais recente (SEGRINI et al., 2006), durante a definição de um processo padrão, o engenheiro de processo informa os subprocessos que compõem o processo. Dos subprocessos selecionados, um, e somente um, deve ser um processo de engenharia (nome adotado para referenciar os subprocessos de desenvolvimento e manutenção). Para cada subprocesso selecionado, o engenheiro de processos informa as formas de interação entre esse processo e o processo de engenharia e as atividades que o compõem. Para cada atividade, podem-se definir subatividades, pré-atividades, artefatos produzidos (produtos) e requeridos (insumos), recursos necessários (recursos humanos, de hardware e de software) e procedimentos (roteiros, métodos, técnicas, normas, checklists etc) a serem adotados. Por fim, definem-se os modelos de ciclo de vida que podem ser utilizados para o processo padrão.

Um processo especializado é definido a partir de um processo padrão. Todos os modelos de ciclo de vida, subprocessos, atividades, interação entre as atividades, artefatos, recursos e procedimentos definidos para o processo padrão são automaticamente definidos para o processo especializado. Depois, é permitido alterá-lo.

Por fim, um processo de um projeto pode ser definido a partir de um processo padrão ou especializado. Nesse caso, são criados os subprocessos, as atividades, os artefatos, os recursos e os procedimentos para o processo do projeto, de acordo com o processo padrão e o modelo de ciclo de vida escolhidos. Para cada atividade, podem ser definidos subatividades, artefatos, recursos e procedimentos necessários para a sua execução. Para cada artefato produzido, pode-se realizar o planejamento da documentação.

Nessa versão da ferramente já é possível reutilizar subprocessos no nível de abstração de processo padrão. Entretanto, ainda não é adotada a visão de componentes de processo, não sendo possível definir um componente a partir da composição de

componentes de menor granularidade. A reutilização de subprocessos é possível apenas durante a definição de um processo padrão, quando se permite reutilizar subprocessos padrão previamente definidos. Ou seja, dado um tipo de subprocesso previamente selecionado, é possível selecionar um subprocesso padrão do mesmo tipo, definido no contexto de outro processo padrão, para compor o processo padrão em definição.

2.8. Considerações Finais do Capítulo

Neste capítulo foram apresentados os conceitos relativos ao domínio de processos de software, abordando a definição desses processos, inclusive via reutilização de processos, tema central desta dissertação.

Além disso, foi discutido o tema reutilização de software, focando no desenvolvimento baseado em componentes, uma vez que essa abordagem serviu de base para a definição de abordagem de reutilização de processos apresentada no próximo capítulo.

Por fim, fez-se um apanhado geral sobre o tratamento de processos de software no ambiente ODE. Esse tema foi tratado, uma vez que as ideias propostas no Capítulo 3 desta dissertação foram materializadas no ambiente ODE, por meio da evolução da ferramenta de definição de processos do ambiente. Os aspectos relacionados a essa evolução são discutidos no Capítulo 4.

Capítulo 3

Definição de Processos Baseada em Componentes

O objetivo deste capítulo é discutir as definições sobre componentes de processo de software, trançando um paralelo com o Desenvolvimento Baseado em Componentes (DBC), e apresentar um processo de Definição de Processos Baseada em Componentes (DPBC). Desse modo, a abordagem adotada para este capítulo se dá por: (i) uma discussão sobre os conceitos de componentes de processo, bem como seus níveis de granularidade e abstração, suas interfaces e seus repositórios; (ii) a apresentação do processo de DPBC definido.

3.1. Introdução

Reconhecidamente a qualidade dos processos de software seguidos influencia diretamente na qualidade dos produtos de software desenvolvidos (ARENT et al., 2000). Entretanto, definir processos de software não é uma tarefa trivial. Há uma grande quantidade de material indicando as melhores práticas a serem seguidas, mas cada organização deve definir seus processos levando em consideração não só essas informações, mas também suas próprias características (BERTOLLO et al., 2006).

Conforme discutido no Capítulo 2, uma abordagem utilizada frequentemente para definição de processos é a definição de processos em níveis (ROCHA et al., 2001), em que são definidos processos padrão organizacionais, os quais são instanciados para projetos, originando os processos de projeto. Destaca-se que os processos padrão podem ser gerais ou especializados, sendo que um processo padrão especializado constitui a especialização de um processo padrão geral (ou especializado) para uma determinada classe de projetos. Embora essa abordagem se baseie na reutilização de processos, ela não explora amplamente as possibilidades de reúso de processos.

Uma alternativa para tentar amenizar as dificuldades na definição de processos é trazer para a o domínio de processos de software as ideias do Desenvolvimento de software Baseado em Componentes (DBC) (GIMENES e HUZITA, 2005). Como

apresentado no capítulo anterior, o DBC busca institucionalizar o reúso de artefatos de software, indo desde código executável até modelos de análise e projeto, advogando que novos artefatos sejam gerados pela composição de outros artefatos já existentes.

Em uma abordagem intitulada Definição de Processos Baseada em Componentes (DPBC), procura-se igualmente institucionalizar o reúso de processos ou de partes deles, tratados como componentes de processo. Esses componentes de processo podem ser definidos no nível de abstração de processos padrão e em variados níveis de granularidade, podendo ser utilizados na definição de outros componentes de processo (de nível de granularidade mais grosso) ou na definição de processos de projeto.

Assim, é importante estabelecer o conceito de componentes para o domínio de processos de software, bem como discutir sobre as interfaces desse tipo de componente. Como processos de software existem em diferentes níveis de abstração (padrão, especializado e de projeto), é necessário também que se discuta como a reutilização dos componentes ocorre nesses níveis de abstração. Por fim, é essencial definir um processo para suportar a abordagem de DPBC. Assim como no DBC, o novo processo para DPBC deve se preocupar com as questões de definição **com** reúso e definição **para** reúso.

Este capítulo está estruturado da seguinte forma: a Seção 3.2 discute o conceito de componentes de processo, bem como seus níveis de granularidade e de abstração, suas interfaces e repositórios necessários; a Seção 3.3 apresenta o processo definido para a abordagem de DPBC; a Seção 3.4 apresenta uma comparação entre a abordagem DPBC e alguns trabalhos correlatos na literatura; por fim, a Seção 3.5 apresenta as considerações finais do capítulo.

3.2. Componentes de Processo

Conforme discutido no Capítulo 2, componentes de software são unidades de software independentes que oferecem serviços por meio de interfaces bem definidas. Esses componentes permitem o desenvolvimento de aplicações a partir de partes já existentes, numa perspectiva de desenvolvimento denominada Desenvolvimento de software Baseado em Componentes (DBC) (GIMENES e HUZITA, 2005). Uma abordagem para definição de processos baseada em componentes também parte da premissa que existem componentes de processo que podem ser reutilizados na definição dos processos de software.

Segundo a ontologia de processos proposta em (BERTOLLO, 2006), um processo de software pode ser composto por outros processos ou por atividades. Os processos que compõem outro processo são chamados subprocessos. Por outro lado, as atividades diretas de um processo, ou seja, aquelas atividades que compõem diretamente o processo e que não compõem outras atividades, são denominadas macroatividades. A Figura 3.1 mostra um fragmento da ontologia de processo de software referente a essas definições.

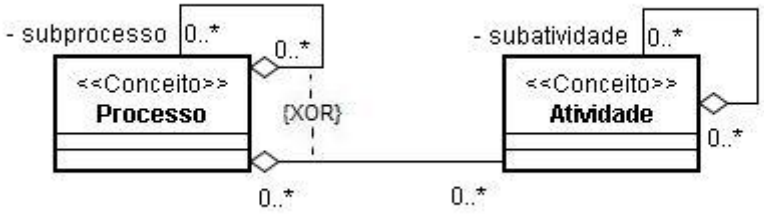

Figura 3.1 – Fragmento da Ontologia de Processo de Software.

Ainda segundo BERTOLLO (2006), uma atividade é definida por diversos ativos: pré-atividades, subatividades, artefatos requeridos (insumos) e produzidos (produtos), recursos necessários (recursos humanos, de hardware e de software) e procedimentos a serem adotados (roteiros, métodos, técnicas etc). A Figura 3.2 ilustra a estrutura proposta para processos e atividades.

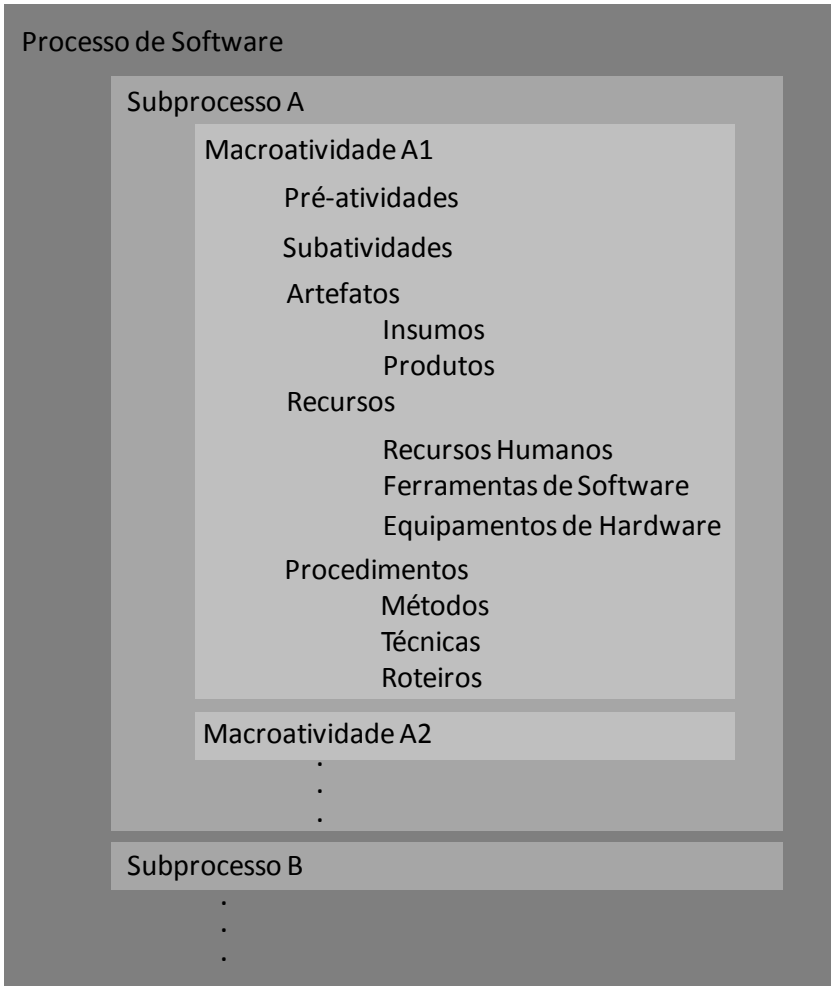

Figura 3.2 – Estrutura de Processos e Atividades.

Voltando à Figura 3.1, esta mostra apenas o conceito de processo, ainda que, de acordo com sua composição, processos possam ser de dois tipos: processos compostos por outros processos e processos compostos diretamente por atividades. Analogamente, é mostrado também apenas o conceito de atividade, ainda que as atividades possam ser de dois tipos: atividades que compõem outras atividade (subatividades) e atividades que compõem diretamente um processo, não sendo parte de nenhuma atividade (macroatividades). Este trabalho está fortemente relacionado a esses tipos de processos e atividades e, portanto, fez-se uma releitura do modelo anterior, explicitando essas distinções e tornando o modelo mais adequado para o propósito deste trabalho. Essa releitura é mostrada na Figura 3.3. Nesse modelo, os processos compostos diretamente por outros processos são denominados *processos complexos* e os processos que são compostos diretamente por atividades são denominados *processos simples*. Por outro lado, as atividades que compõem diretamente um processo são denominadas

macroatividades e as atividades que compõem outras atividades são denominadas *subatividades*.

Essas relações de composição dão origem ao que chamamos coletivamente de subelementos, a saber: (i) ao se estabelecer que um processo complexo é composto por subprocessos simples, esses são denominados subelementos do processo complexo; (ii) as macroatividades que compõem um processo simples, por sua vez, são ditas subelementos de processos simples; finalmente, (iii) as subatividade diretas de uma macroatividade são ditas subelementos dessa macroatividade.

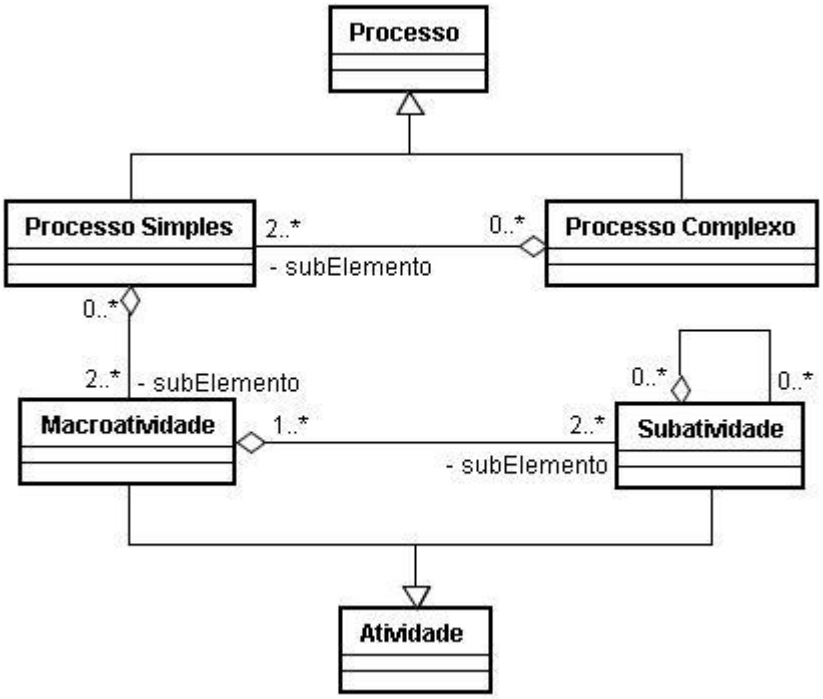

Figura 3.3 – Modelo de Processos.

O modelo da Figura 3.3 leva à ideia de que componentes de processo podem ser definidos em diferentes níveis de granularidade (processos complexos, processos simples e macroatividades). Além disso, para alinhamento com a abordagem de definição de processos em níveis, componentes de processo devem ser definidos considerando-se os níveis de abstração de processo (processos padrão, processos padrão especializados e processos de projeto). Ainda, de maneira análoga ao DBC, componentes de processo devem possuir interfaces que os descrevam, possibilitando a reutilização. Por fim, componentes devem ser armazenados em repositórios, de onde possam ser buscados visando à reutilização. A seguir, esses temas são discutidos para o universo dos componentes de processo de software.

3.2.1. Níveis de Granularidade de Componentes de Processo

Para uma abordagem de DPBC, é importante que sejam definidos quais dos elementos da estrutura de processos constituem componentes de processo, ainda que em diferentes níveis de granularidade. No contexto deste trabalho, são ditos componentes de processo as macroatividades, os processos simples e os processos complexos. Dessa visão surge a relação de subcomponentes, que é definida entre dois componentes de processo em níveis de granularidade adjacentes, sendo o componente de granularidade mais fina (p.ex., macroatividade) parte do componente de granularidade mais grossa (p.ex., subprocesso). Uma vez que o nível de macroatividades é o nível de granularidade de componente de processo mais fino, não há subcomponentes nesse nível, ainda que haja subelementos (subatividades). Isso decorre do fato de, no contexto deste trabalho, dentre as atividades, apenas as macroatividades serem consideradas componentes de processo. Como exemplo, suponha um processo simples de gerência de projetos composto pelas macroatividades de planejamento, acompanhamento e encerramento do projeto. Nesse caso, essas macroatividades (nível de granularidade de macroatividade) são ditas subcomponentes do componente processo de gerência de projetos (nível de granularidade de processo simples). A Figura 3.4 ilustra os níveis de granularidade de componentes de processo considerados neste trabalho.

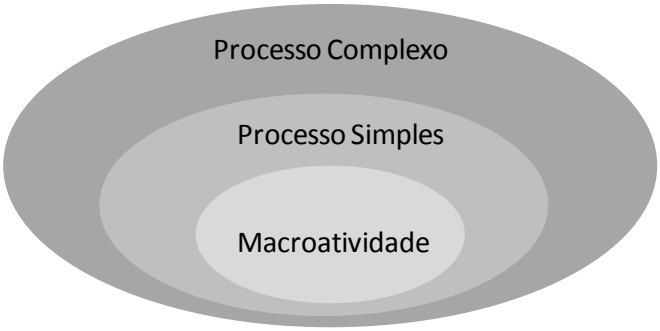

Figura 3.4 – Níveis de Granularidade dos Componentes de Processos de Software.

Vale reforçar que, apesar de subatividades (atividades que não compõem diretamente um processo) também serem importantes elementos de um processo, para a abordagem de definição de processos baseada em componentes aqui proposta, elas não foram considerados como componentes de processo. Assim, apenas processos complexos, processos simples e macroatividades podem ser recuperados, adaptados e utilizados para compor um processo de software. Contudo, quando uma macroatividade é reutilizada, suas subatividades também o são, assim como as demais definições

relativas a elas: artefatos requeridos (insumos), artefatos produzidos (produtos), recursos necessários e procedimentos adotados. Essa decisão foi tomada, uma vez que as subatividades estão em um nível de granularidade muito fino. Com isso, seria difícil conseguir definir componentes a partir de subatividades num nível de abstração que permitisse serem reutilizados na composição de processos distintos. Esse problema desencadeia outro: o repositório de componentes ficaria demasiadamente povoado em virtude da definição de componentes de processo em um nível tão fino de granularidade, o que tornaria a busca por componentes para reutilização menos eficiente, conforme apontado por SEACORD (1999).

A Figura 3.5 mostra um diagrama de classes relacionando as definições de componentes de processo com os elementos do modelo de processo da Figura 3.3. É importante ressaltar que as relações de subcomponentes definidas são especializações das relações de subelementos descritas no modelo da Figura 3.3, com exceção da relação de subelemento entre subatividade e macroatividade que não possui especialização para os componentes de processo, uma vez que neste trabalho subatividades não são consideradas componentes de processo. Além disso, as relações de subcomponentes também são especializações da relação todo-parte entre *Componente Composto* e *Componente*, onde um componente composto é composto por dois ou mais componentes. Desse modo, os subcomponentes de um componente de processo complexo são componentes de processos simples; os subcomponentes de um componente de processo simples, por sua vez, são componentes de macroatividade; e finalmente os componentes de macroatividade não possuem subcomponentes, ainda que macroatividades tenham como subelementos as subatividades que o compõem.

O modelo da Figura 3.5 deve ser interpretado da seguinte forma: Componentes de software, em geral (de produto como no DBC ou de processo como no DPBC), definem uma interface e podem ser atômicos ou compostos. Os componentes compostos são compostos de dois ou mais outros componentes, enquanto um componente atômico não é composto de outros componentes, ainda que possa ser composto de diversos elementos (p.ex., classes em um componente de produto no desenvolvimento orientado a objetos).

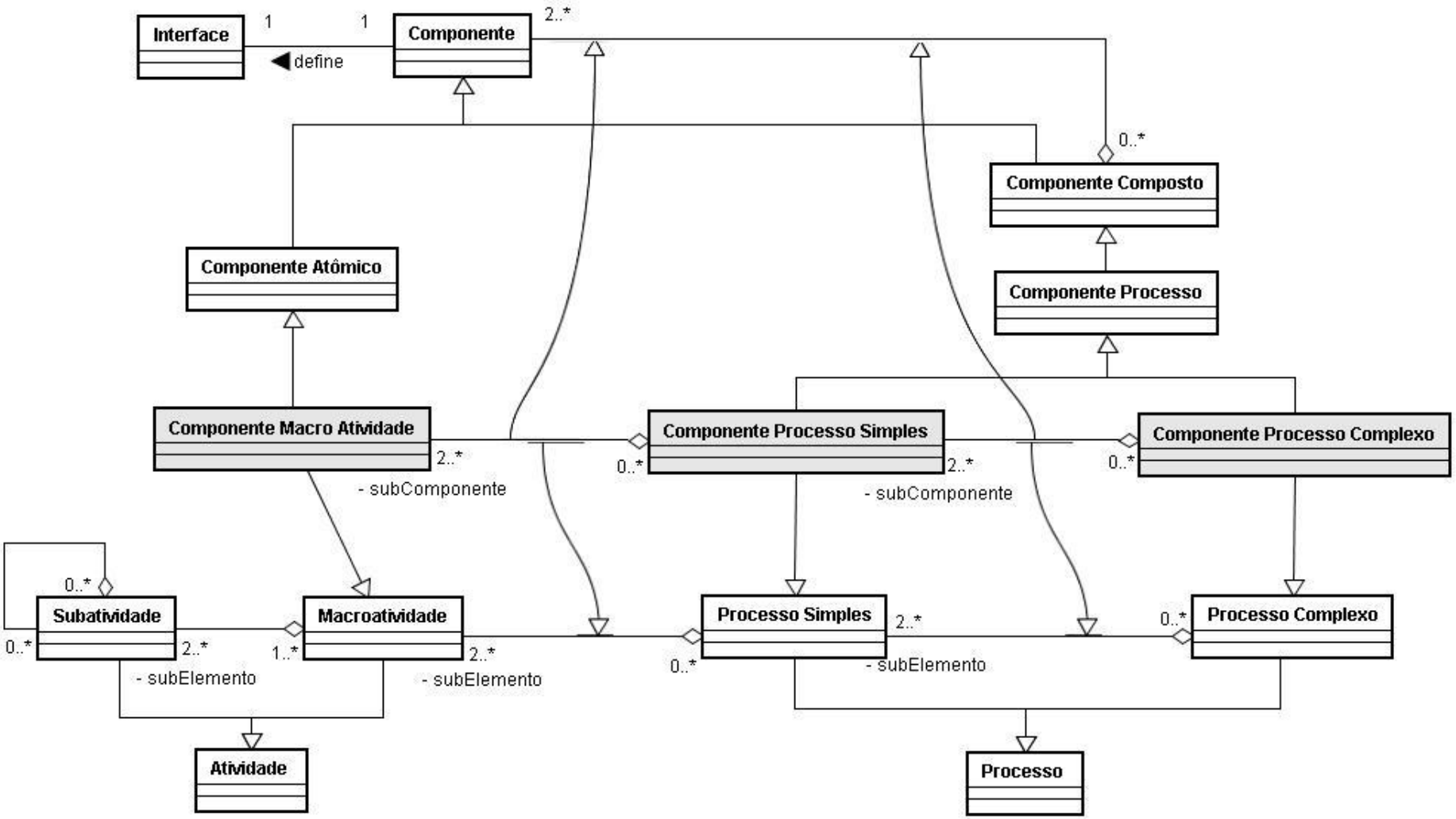

Figura 3.5 – Níveis de Granularidade dos Componentes de Processos de Software.

Processos (simples e complexos) e macroatividades podem ser tratados como componentes, ainda que nem todos eles o sejam. Assim, ser um componente é um papel que um processo ou macroatividade pode desempenhar e, para tal, precisa ter uma interface definida.

Um componente de macroatividade é um componente atômico derivado a partir de uma macroatividade. Por ser um componente atômico, ele não possui subcomponentes. Por outro lado, por ser uma macroatividade, deve possuir subatividades como subelementos.

Um componente de processo simples é um componente composto de componentes de macroatividade, ditos seus subcomponentes. De fato, a relação de subcomponentes é derivada neste caso da relação de subelementos que há entre o processo simples que deriva o componente e suas macroatividades.

Por fim, um componente de processo complexo é um componente composto que tem como subcomponentes componentes de processo simples. Analogamente aos componentes de processo simples, a relação de subcomponentes é derivada da relação de subelementos que há entre o processo complexo que o deriva e os correspondentes processos simples.

3.2.2. Níveis de Abstração de Processos e Componentes de Processo

Conforme discutido na Seção 2.3 do Capítulo 2, uma abordagem flexível para definição de processos envolve três níveis de abstração de processo: processos padrão, processos especializados e processos de projeto. Destaca-se que, embora um processo especializado seja definido a partir de um processo padrão geral (ou especializado), ele ainda constitui um padrão organizacional, podendo ser aplicado nos diversos projetos de uma organização que se enquadrem na classe de projetos para a qual ele foi especializado.

Dessa maneira, pode-se enxergar essa divisão de níveis de abstração de outra forma, em que processos padrão gerais e especializados constituem um único nível de abstração, enquanto os processos de projeto constituem outro nível de abstração. Portanto, essa forma de segmentação dos níveis de abstração de processos contempla apenas dois níveis, a saber: o *nível de abstração de processo de padrão*, englobando processos padrão gerais e especializados, que são definidos com o intuito de serem utilizados na definição de processos de projeto; e o *nível de abstração de processo de*

projeto, que engloba os processos definidos especificamente para os projetos de uma organização. Se comparados aos níveis de abstração de processo tratados na Figura 2.1 da Seção 2.3 do Capítulo 2, pode-se observar que os níveis de processo padrão geral e de processo padrão especializados foram fundidos, como mostra a Figura 3.6.

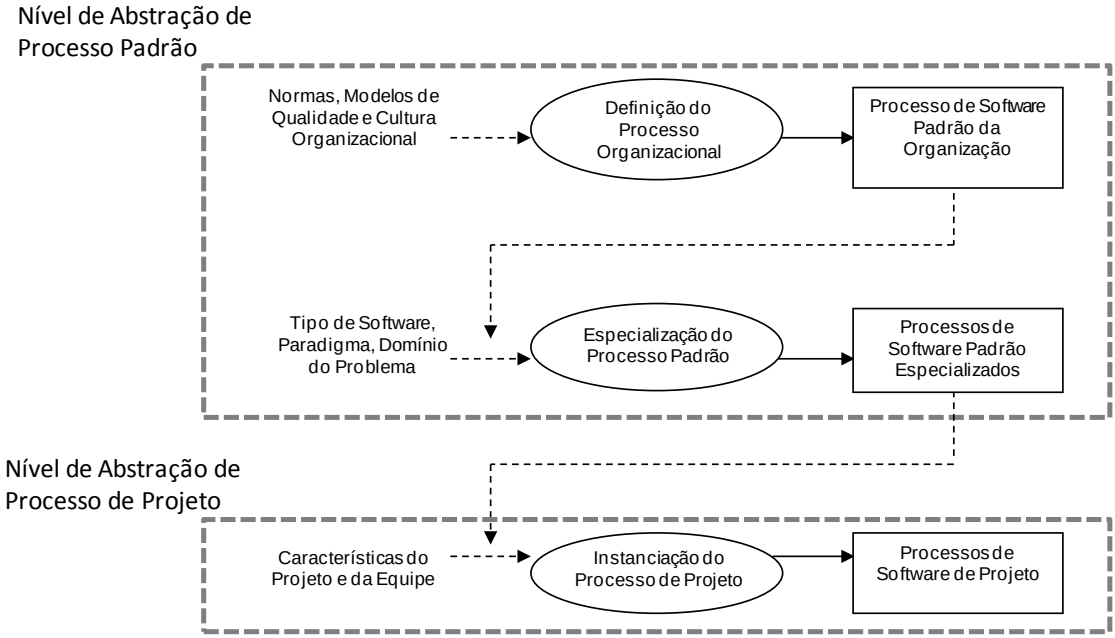

Figura 3.6 – Níveis de Abstração de Processos na Abordagem de DPBC.

Independentemente do nível de abstração em que um processo se encontra (padrão ou de projeto), sua estrutura segue o modelo apresentado na Figura 3.2. Da mesma forma, as definições de componentes e subcomponentes, discutidas na subseção anterior, são válidas em ambos os níveis.

O esquema apresentado na Figura 3.6 deixa clara a intenção de se reutilizar processos. Entretanto, ao se instanciar um processo padrão geral ou especializado para um projeto específico, a reutilização ocorre apenas verticalmente, isto é, um processo do nível de abstração mais alto (processo padrão) é usado como base para a definição de um processo do nível de abstração mais baixo (processo de projeto). Neste trabalho busca-se promover o reúso de processos também de maneira horizontal, ou seja, em um mesmo nível de abstração, por meio da definição de processos a partir de componentes de processo já existentes, na abordagem denominada Definição de Processos Baseada em Componentes (DPBC).

A ideia da DPBC é análoga à do Desenvolvimento de software Baseado em Componentes (DBC). Se por um lado o DBC visa à construção de software a partir da reutilização de componentes de produto já existentes, a DPBC advoga a definição de

processos a partir de componentes de processos previamente definidos. Assim, é necessário que componentes de processos sejam definidos visando à reutilização.

O conceito de componente de processo é aplicável em ambos os níveis de abstração de processo (padrão e de projeto), apesar de em cada um dos dois níveis os componentes serem definidos com propósitos diferentes. Os componentes no nível de processo padrão são sempre definidos visando à reutilização (definição **para** reúso), embora suas definições possam ocorrer através da reutilização de outros componentes no mesmo nível de abstração (definição **com** reúso). Para simplificar a referência aos componentes no nível de processo padrão, eles são denominados CompPP neste texto. Por outro lado, os componentes no nível de processo de projeto são definidos a partir da reutilização de outros componentes (definição **com** reúso), podendo esses componentes serem de ambos os níveis de abstração de processo (padrão e de projeto).

Ainda que os componentes no nível de processo de projeto sejam definidos especificamente para um projeto e, por isso, não sejam definidos visando ao reúso, eles não podem ser descartados quando à reutilização. Esses componentes são importantes ativos de conhecimento de uma organização, pois são a memória de projetos anteriores, trazendo informações relevantes sobre experiências, bem ou mal sucedidas (HENNINGER, 1999) (MAURER e DELLEN, 1999).

Dessa forma, a reutilização de componentes de nível de processos de projeto também é considerada, possibilitando reutilizar componentes de processos definidos para processos de projeto anteriores, ainda que esse (definição **para** reúso) não seja o foco dos componentes de processo de projeto. Assim, durante a definição de um processo de projeto, pode-se optar pela reutilização tanto CompPPs quanto de componentes de processos de projetos anteriores. Note que não existe definição explícita de componentes de nível de processo de projeto para reutilização. Os componentes reutilizados desse nível de abstração são sempre definidos em processos de projetos anteriores, com possibilidade de serem reutilizados em novos processos projetos.

Usualmente, na abordagem de definição de processos em níveis, os processos padrão são definidos por completo, ou seja, processos simples e macroatividades são apenas parte da estrutura do processo complexo. Essa visão possibilita o reúso apenas no nível de granularidade de processo complexo. A ideia aqui é introduzir uma visão mais ampla, onde componentes de processo dos diversos níveis de granularidade podem, e devem, ser definidos isoladamente. Isso permite institucionalizar o reúso de

processos não somente entre os níveis de abstração de processos, dito reúso vertical, como também o reúso de componentes de processo definidos em um mesmo nível de abstração, dito reúso horizontal.

Assim, resumindo, na abordagem aqui proposta, a reutilização pode acontecer das seguintes formas:

- Vertical: componentes reutilizáveis de processo definidos no nível de processo padrão (CompPPs) são utilizados na definição de processos de projeto. Entretanto, além de reutilizar um processo padrão por completo, conforme já proposto pela abordagem de definição de processos em níveis, é possível reutilizar componentes de processo padrão em níveis de granularidade mais fina. Por exemplo, suponha que o processo padrão escolhido para dar origem ao processo de um projeto não contenha um processo de medição. Entretanto, identifica-se a necessidade de um processo desse tipo para o projeto em questão. Com a abordagem de DPBC, é possível instanciar um componente reutilizável de processo simples do tipo medição, definido no nível de processo padrão, para o processo de projeto em questão.
- Horizontal: componentes de granularidade mais fina, definidos em um determinado nível de abstração de processos (padrão ou de projeto), são utilizados na definição de outros componentes de maior granularidade no mesmo nível de abstração. Por exemplo, suponha que se tenha definido em algum momento um componente de processo simples de gerência de configuração no nível de abstração de processo padrão. Depois, durante a definição de um CompPP de processo complexo, identifica-se a necessidade de um processo simples desse tipo. Com a abordagem de DPBC, é possível reutilizar o CompPP de processo simples de gerência de configuração definido previamente para compor o novo CompPP de processo complexo em definição.

3.2.3. Interface de Componentes de Processo

Da mesma forma que quaisquer componentes de software, os componentes de processos de software devem possuir uma interface e, apesar de serem domínios distintos, pode-se traçar um paralelo entre esses dois tipos de interface.

Em componentes de produto de software, uma interface define as funcionalidades providas pelo componente e diz como um componente pode ser reutilizado e conectado a outros componentes (SAMETINGER, 1997). Para os componentes de processos de software, uma interface deve definir as características pertinentes ao componente, tais como domínio de aplicação e paradigma de desenvolvimento, que especifiquem as condições de aplicação do componente. Cada organização deve definir as características a serem utilizadas na caracterização de seus componentes de processo.

Ainda, se no domínio de componentes de produto de software uma interface define as funcionalidades providas pelos componentes que a implementam, no domínio de componentes de processos de software a interface indica os subelementos que devem compor o componente de processo, havendo, para cada um deles, uma indicação se o mesmo é obrigatório ou não. Como a ideia é não entrar nos detalhes de um componente, as interfaces não contemplam a definição dos subelementos. Portanto, ao se definir, por exemplo, a interface de um componente de processo complexo, deve-se definir os processos simples que o compõem. Ainda, ao se definir um componente de processo simples, deve-se definir as macroatividades que o compõem. Logo, interfaces de componentes de processo de software definem as características de aplicação do componente e os seus subelementos, ou seja, definem tanto a estrutura do componente quanto o seu contexto de aplicação. A Figura 3.7 ilustra a estrutura das interfaces para os tipos de componentes considerados neste trabalho.

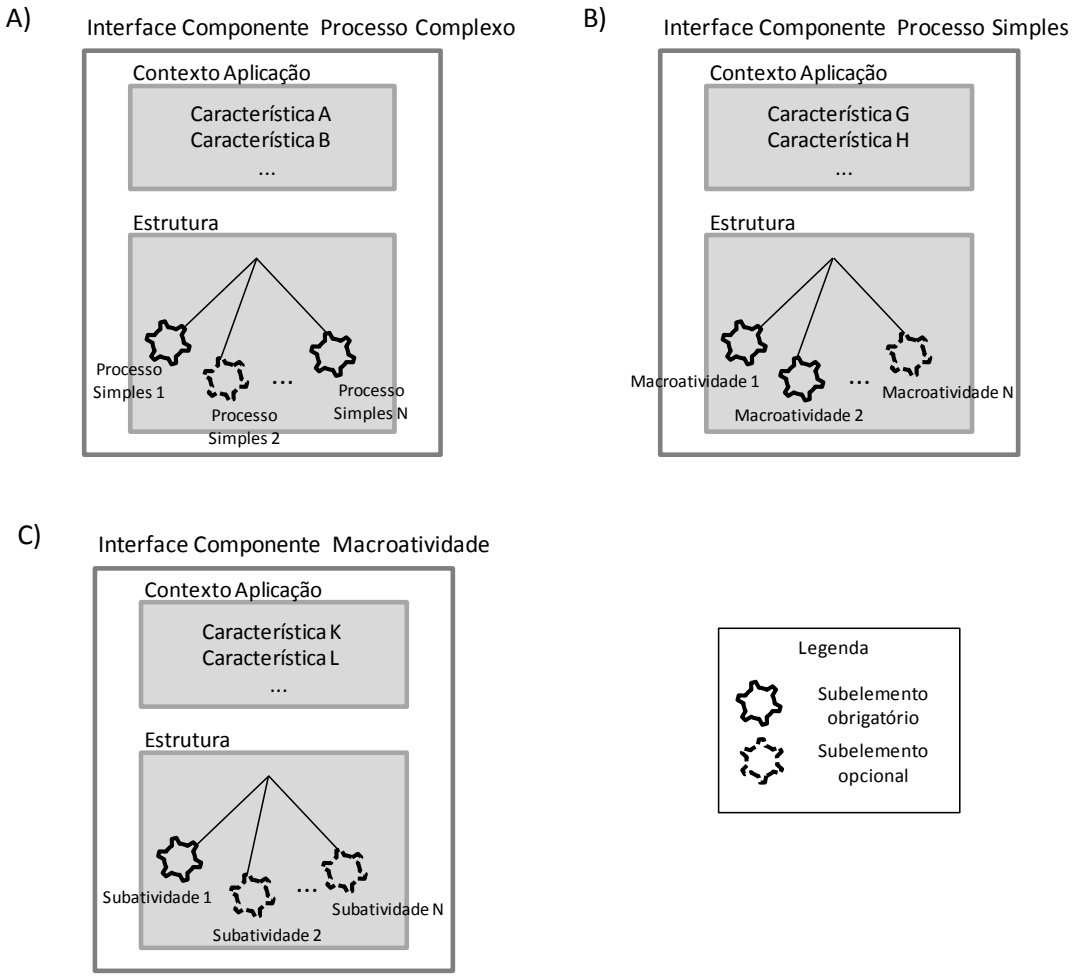

Figura 3.7 – Interfaces de Componentes.

O estabelecimento da interface de um componente ocorre de duas maneiras distintas, dependendo do seu nível de abstração (processo padrão ou de projeto). No nível de abstração de processos padrão, a interface de um componente é definida explicitamente. Assim, durante a definição de um componente nesse nível de abstração, uma das etapas existentes trata da definição da interface do componente, devendo ser definidos seu contexto de aplicação e sua estrutura.

Por outro lado, no nível de abstração de processos de projeto, a interface não é definida explicitamente durante a definição do componente, mas sim extraída dos processos no momento de uma busca. Como um componente nesse nível de abstração é definido sempre no contexto de um projeto, não sendo definido visando à reutilização, o seu contexto de aplicação é dado por um subconjunto de características extraído da caracterização do projeto. Para que as características do componente de processo de projeto sejam extraídas da caracterização do projeto para o qual ele foi definido, a

caracterização dos projetos tem que considerar as características que a organização definiu para caracterizar seus componentes de processo. Já a estrutura do componente é extraída a partir do nível de granularidade desejado em um dado contexto. Suponha que, durante a definição de um processo de projeto, deseje-se incorporar um processo de medição. Serão, então, considerados componentes de processo de projeto aqueles processos simples de medição, definidos em projetos anteriores, sendo sua estrutura dada pelas macroatividades do correspondente processo de medição (de nível de abstração de processo de projeto).

3.2.4. Repositórios de Componentes de Processo

Os repositórios de componentes de software podem ser vistos como bases de componentes que os armazenam, centralizando-os, a fim de facilitar a reutilização. Eles são a principal ligação entre o desenvolvimento de componentes e o desenvolvimento com componentes (SAMETINGER, 1997). As principais funcionalidades de um repositório são a pesquisa e a recuperação dos componentes (HOLANDA et al., 2001). Na definição de processos baseada em componentes, a existência de repositórios também é essencial.

Este trabalho propõe a utilização de dois repositórios: um para componentes de processo do nível de abstração de processo padrão (CompPP) e outro para componentes de processo do nível de abstração de processos de projeto. A Figura 3.8 mostra os repositórios de componente e suas respectivas estruturas.

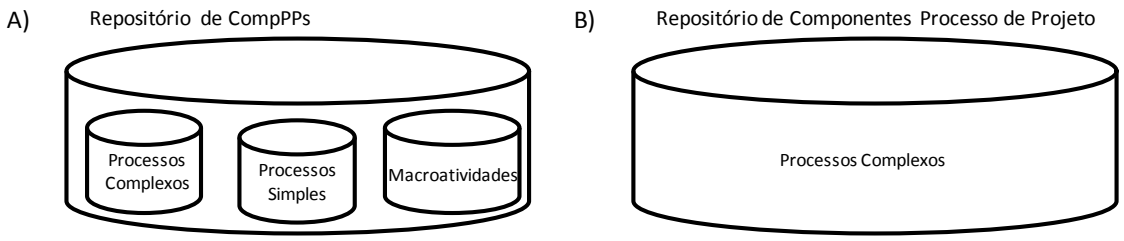

Figura 3.8 – Repositórios da DPBC.

O primeiro deles, denominado repositório de componentes de processo padrão, ou repositório de ComPPs, abriga os componentes de processo definidos no nível de abstração de processo padrão. Conforme discutido anteriormente, os componentes desse nível de abstração são definidos visando à reutilização e, portanto, são armazenados separadamente, já no momento de sua definição. Como no contexto deste trabalho

componentes são definidos em três níveis de granularidade (processo complexo, processo simples e macroatividade), esse repositório está dividido em três partes, sendo que cada uma delas abriga apenas componentes do nível de granularidade correspondente.

Já o segundo repositório, denominado repositório de componentes de processo de projeto, abriga componentes de processo definidos no nível de abstração de processos de projeto. Como os processos de projeto são sempre definidos no contexto de um projeto, não sendo definidos visando à reutilização, esse repositório abriga os processos de projeto como um todo, ou seja, o seu nível de granularidade é o de processos complexos, embora seja possível recuperar componentes de nível de granularidade mais fino. Ainda, também em decorrência de não serem definidos visando à reutilização, não existe definição explícita de interface para os componentes de processo de projeto. Assim, para que as características contidas em suas interfaces sejam extraídas, conforme discutido na subseção anterior, é necessário saber o contexto em que eles foram definidos, o que leva ao fato de não poderem ser armazenados separadamente, ainda que seja possível a busca por componentes de diferentes níveis de granularidade.

3.3. Definição de Processos Baseada em Componentes

De forma análoga ao desenvolvimento baseado em componentes (DBC), em que deve haver desenvolvimento de componentes e desenvolvimento com componentes, a Definição de Processos Baseada em Componentes (DPBC) se baseia na definição de componentes de processos reutilizáveis e na definição de processos com componentes. Ainda que os componentes no nível de abstração de processo padrão possam ser definidos através da composição de outros componentes do mesmo nível, constituindo uma forma de definição com reúso (horizontal), a definição desses componentes pode ser encarada como definição para reúso, uma vez que seu propósito é a reutilização. Por outro lado, a definição de processos no nível de abstração de processos de projeto é encarada como definição com componentes. Embora componentes de processo de projeto possam ser reutilizados para a definição de outros componentes de processo de projetos, esse não é o foco principal da definição desses componentes. Os componentes de processo de projeto são sempre definidos para atenderem objetivos específicos de um projeto. Assim, apesar de poderem ser reutilizados, componentes de processo de projeto

não são definidos visando à reutilização. Eles são sempre definidos para um projeto específico, com possibilidade de serem reutilizados posteriormente. Por outro lado, a definição de componentes de processo de projeto envolve a reutilização (vertical e horizontal) de componentes de processo previamente definidos de ambos os níveis de abstração de processo (padrão e de projeto).

Um processo de DPBC deve contemplar as duas formas de reutilização de componentes de processo: vertical e horizontal. Neste trabalho foram definidos dois processos distintos, um para a definição de componentes no nível de abstração de processo padrão e outro para a definição de processos de projeto (nível de abstração de processo de projeto), devido ao fato de algumas de suas etapas serem bastante específicas. Durante a definição desses processos, o foco principal foi definir as atividades a serem realizadas para propiciar uma reutilização de processos mais efetiva e sistemática em uma organização.

3.3.1. Processo de DPBC para o Nível de Abstração de Processo Padrão

A Figura 3.9 mostra um diagrama de atividades apresentando as principais atividades do processo de DPBC proposto para o nível de processos padrão. Esse processo trata apenas da modalidade de reutilização horizontal, uma vez que componentes no nível de abstração de processo padrão (CompPPs) são utilizados na definição de novos componentes do mesmo nível. Nesse momento a definição é realizada envolvendo apenas CompPPs, independentemente de serem especializações de outros componentes do mesmo tipo.

Destaca-se que, no nível de abstração de processo padrão, todos os componentes são definidos visando à reutilização. Assim, todos os componentes definidos possuem uma interface explicitamente declarada e se encontram no repositório de CompPPs, podendo ser recuperados, adaptados e reutilizados a qualquer momento.

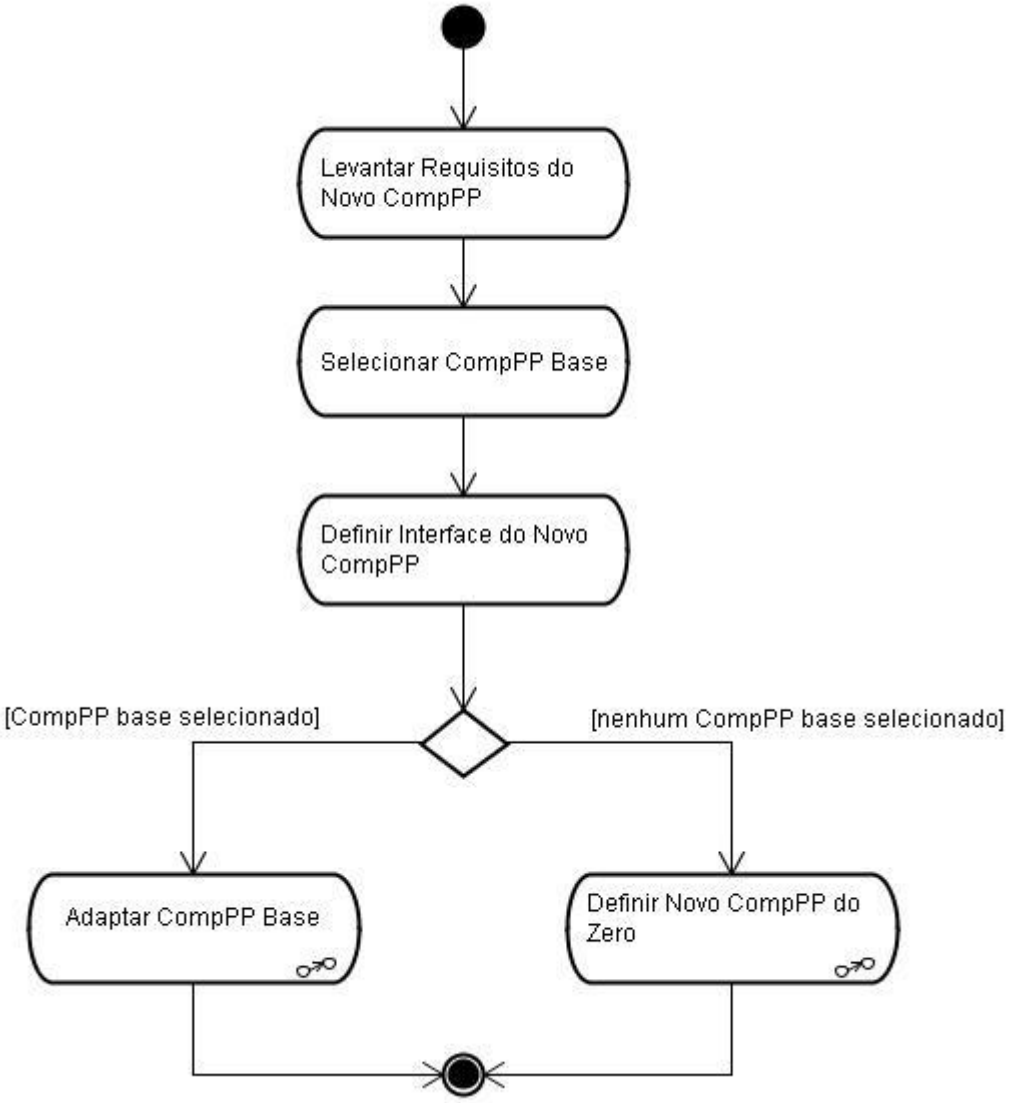

Figura 3.9 – Processo de DPBC para o Nível de Processos Padrão.

As subseções a seguir descrevem as atividades desse processo. Para ajudar no entendimento do processo, um exemplo ilustrativo passando pelas várias atividades descritas é apresentado. Nesse exemplo, uma organização possui o processo padrão mostrado na Figura 3.10A, denominado Processo Padrão Geral. Devido à grande quantidade de projetos desenvolvidos segundo o paradigma orientado a objetos (OO), deseja-se especializar esse processo padrão para essa classe de projetos. Assim, o novo processo padrão (especializado), denominado Processo Padrão OO, tem como principal característica o fato de ser especializado para o paradigma da orientação a objetos. Além disso, o novo processo especializado deve ser capaz de manter a integridade dos produtos de trabalho e demanda um Processo de Gerência de Configuração de Software. A organização já possui um componente de nível de processo simples para a gerência de

configuração, mostrado na Figura 3.10B, mas ele não se encontra na composição do processo padrão base que está sendo especializado. Assim, esse processo simples deve ser adicionado ao novo processo especializado. Ainda, deseja-se que o novo processo especializado seja capaz de assegurar que os produtos de trabalho estejam em conformidade com os planos definidos e, portanto, requer um Processo de Garantia da Qualidade. Por fim, o novo processo padrão especializado não requer um Processo de Treinamento e, portanto, o processo de treinamento que compõe o processo padrão base que está sendo especializado não deve fazer parte do novo processo especializado.

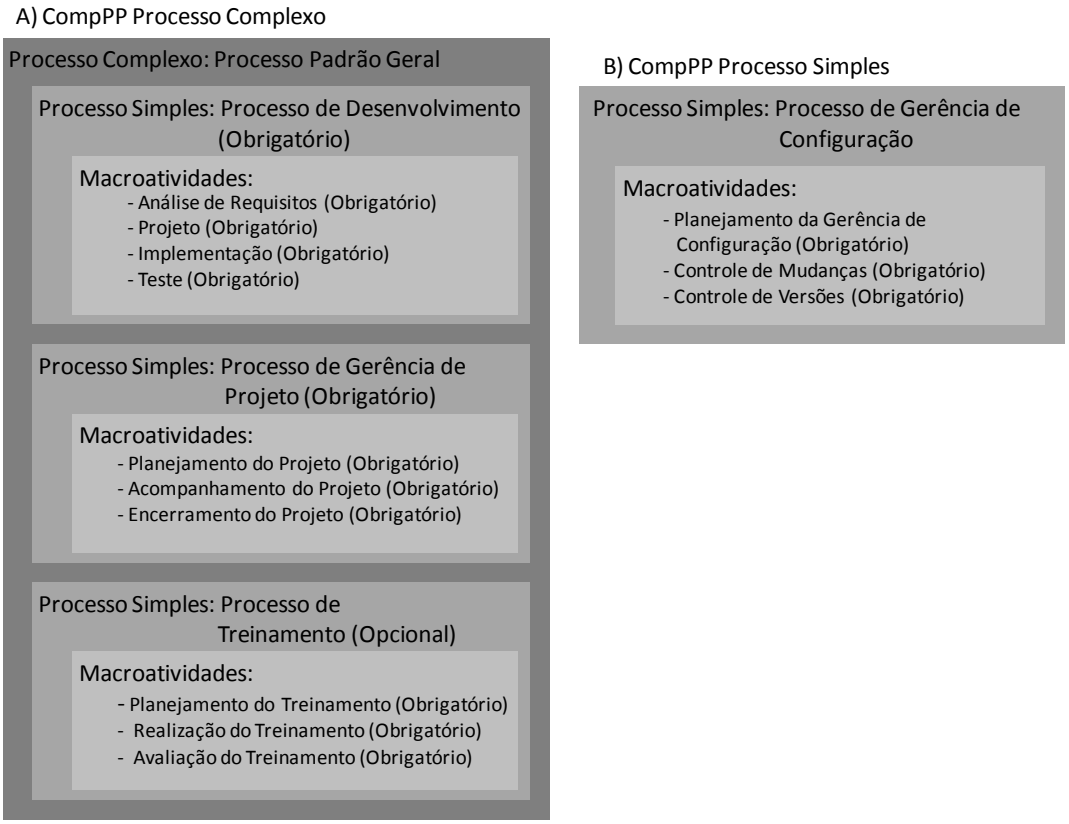

Figura 3.10 – CompPPs para Exemplo de Execução do Processo de DPBC para o Nível de Processos Padrão.

3.3.1.1. Levantar Requisitos do Novo CompPP

A definição de um CompPP inicia-se pelo levantamento de seus requisitos, quando são definidos os seus requisitos e objetivos a serem alcançados. Ainda, são definidas as características do componente, devendo ser consideradas as características utilizadas pela organização na caracterização de seus componentes.

A definição dos objetivos e requisitos do componente serve de base para a definição de sua interface, enquanto as características definem o seu contexto de aplicação, ainda que possam influenciar também a sua estrutura ou a definição dos ativos (recursos, artefatos e procedimentos) de suas atividades.

No exemplo ilustrativo, tomando por base a descrição feita anteriormente, têm-se os requisitos mostrados na Tabela 3.1.

Tabela 3.1 – Exemplo Ilustrativo: Requisitos do Processo Padrão Especializado para o Paradigma Orientado a Objetos.

Nome: Processo Padrão Especializado OO
Descrição: CompPP de nível de processo complexo voltado para o desenvolvimento OO.
Objetivo: Guiar o desenvolvimento de produtos de software segundo o paradigma da orientação a objetos, sendo capaz de manter a integridade dos produtos de trabalho e assegurar que eles estejam em conformidade com os planos definidos.
Características: • Paradigma: Orientado a Objetos.
Requisitos: 1. O processo deve ser uma especialização do processo padrão já existente, Processo Padrão Geral, para a classe de projetos de desenvolvimento orientado a objetos. 2. Durante a especialização, o processo de treinamento que compõe o Processo Padrão Geral deve ser desconsiderado. 3. Deve-se ser capaz de manter a integridade dos produtos de trabalho por meio de um Processo de Gerência de Configuração de Software. 4. Deve-se ser capaz de assegurar que os produtos de trabalho estejam em conformidade com os planos definidos e, portanto, requer um Processo de Garantia da Qualidade.

3.3.1.2. Selecionar CompPP Base

Um CompPP pode ser definido por meio da adaptação de outro CompPP já existente. Para que isso ocorra, é necessário selecionar o CompPP a partir do qual o novo CompPP será definido, dito CompPP base. A adaptação pode ocorrer em quaisquer dos níveis de granularidade considerados neste trabalho, a saber: processo complexo, processo simples e macroatividade.

A seleção do CompPP base pode ocorrer de duas maneiras. Na primeira delas, utilizada quando já se tem conhecimento de qual será o CompPP base, seleciona-se diretamente o CompPP desejado. A segunda é utilizada quando não se tem conhecimento prévio de qual CompPP será adaptado. Então, tomando por base os

requisitos do processo, primeiramente são indicadas as características desejáveis para o componente e os subelementos que devem aparecer em sua composição. Em seguida, são buscados os CompPP que possuam as características desejáveis e os subelementos indicados anteriormente. Por fim, deve-se decidir sobre a utilização ou não de algum deles como CompPP base.

Não é obrigatória a seleção de um CompPP base. Assim, caso nenhum CompPP seja encontrado ou se decida pela não utilização de nenhum daqueles encontrados, o novo CompPP pode ser definido a partir do zero.

No exemplo ilustrativo, devido à grande quantidade de projetos desenvolvidos segundo o paradigma orientado a objetos, deseja-se especializar o processo padrão existente para essa classe de projetos. Assim, já se tem conhecimento de qual CompPP será utilizado como base e, por isso, seleciona-se diretamente o Processo Padrão Geral.

3.3.1.3. Definir Interface do Novo CompPP

Esta atividade tem como objetivo definir a interface do novo CompPP, sendo decomposta em duas subatividades:

- Definir Estrutura do Novo CompPP: definem-se os subelementos necessários para se atender aos requisitos do novo CompPP, indicando se eles são obrigatórios ou não.
- Definir Contexto de Aplicação do Novo CompPP: a caracterização do componente realizada durante o levantamento de requisitos é transcrita para sua interface, utilizando-se as características definidas pela organização para caracterização de seus componentes reutilizáveis.

Dando continuidade ao exemplo ilustrativo, estabelece-se que o novo processo padrão especializado deve possuir processos simples de Desenvolvimento e Gerência de Projetos, oriundos do processo padrão base, e processos de Gerência de Configuração e Garantia da Qualidade, devido aos requisitos levantados. Ainda, define-se que os processos simples de Desenvolvimento, Gerência de Projetos e Gerência de Configuração são obrigatórios e que o processo simples de Garantia da Qualidade não é. Assim, a estrutura do novo processo padrão especializado é composta por quatro processos simples: desenvolvimento, gerência de projeto, gerência de configuração e garantia da qualidade. Portanto, são esses quatro subprocessos que aparecem na interface do novo CompPP. Por fim, em sua caracterização consta que o paradigma ao qual o novo CompPP se aplica é a orientação a objetos. Como nenhuma outra

característica foi definida durante o levantamento de seus requisitos, o novo CompPP (processo padrão especializado) possui como característica de contexto de aplicação apenas a exigência de que o paradigma de desenvolvimento seja o orientado a objetos. A Figura 3.11 mostra a interface do novo CompPP.

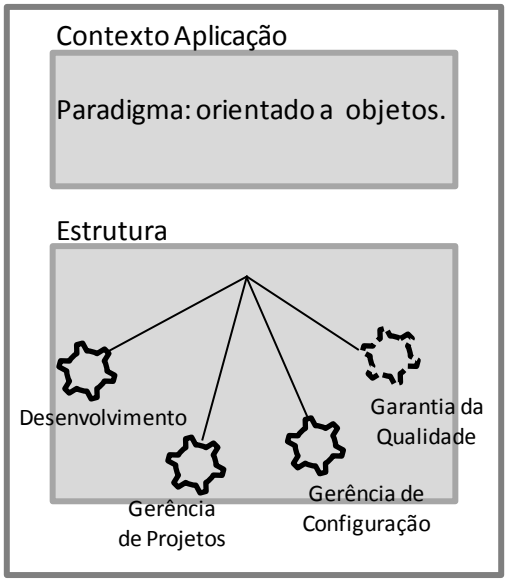

Figura 3.11 – Interface do Novo CompPP.

3.3.1.4. Adaptar CompPP Base

Esta atividade é executada quando um CompPP base para adaptação é selecionado durante a atividade Selecionar CompPP Base. Nesse caso, deve-se adaptar o CompPP base conforme necessário. Essa adaptação ocorre de maneiras distintas, dependendo do nível de granularidade do CompPP em adaptação (processo complexo, processo simples e macroatividade). Vale ressaltar que a adaptação de um CompPP de um determinado nível de granularidade envolve também a adaptação de CompPPs de níveis de granularidade mais finos. A Figura 3.12 mostra o primeiro nível de decomposição desta atividade.

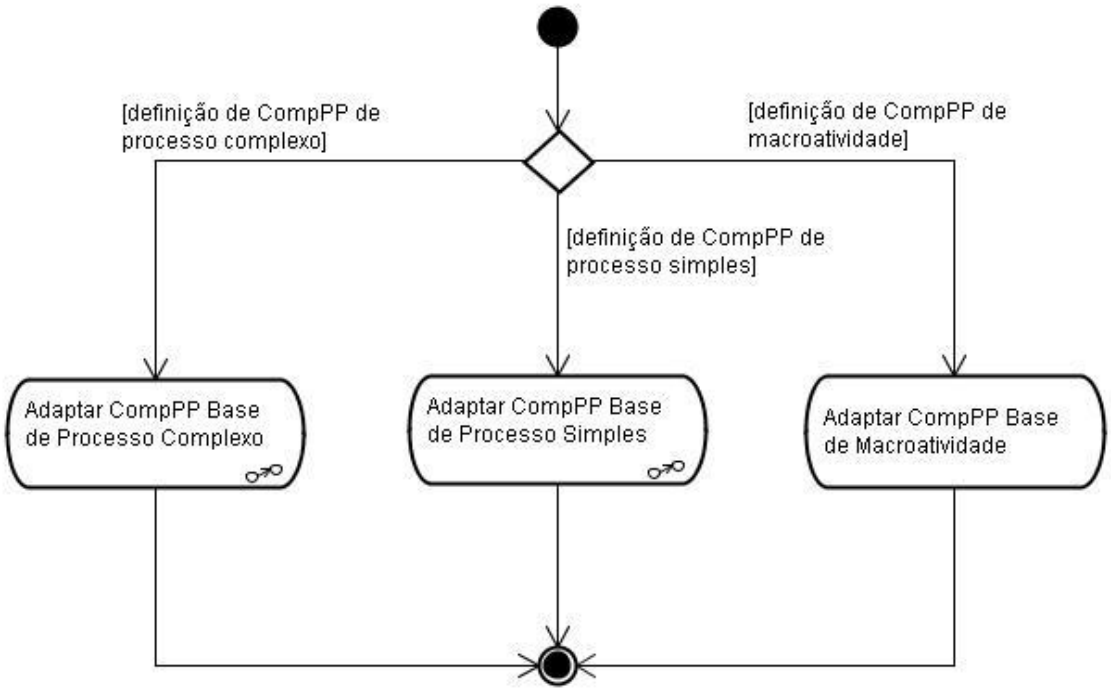

Figura 3.12 – Decomposição da Atividade Adaptar CompPP Base.

Assim, a atividade Adaptar CompPP Base é decomposta em outras três, uma para cada nível de granularidade de CompPP. O caminho seguido depende do nível de granularidade do CompPP em definição. A Figura 3.13 mostra a decomposição da atividade Adaptar CompPP Base de Processo Complexo.

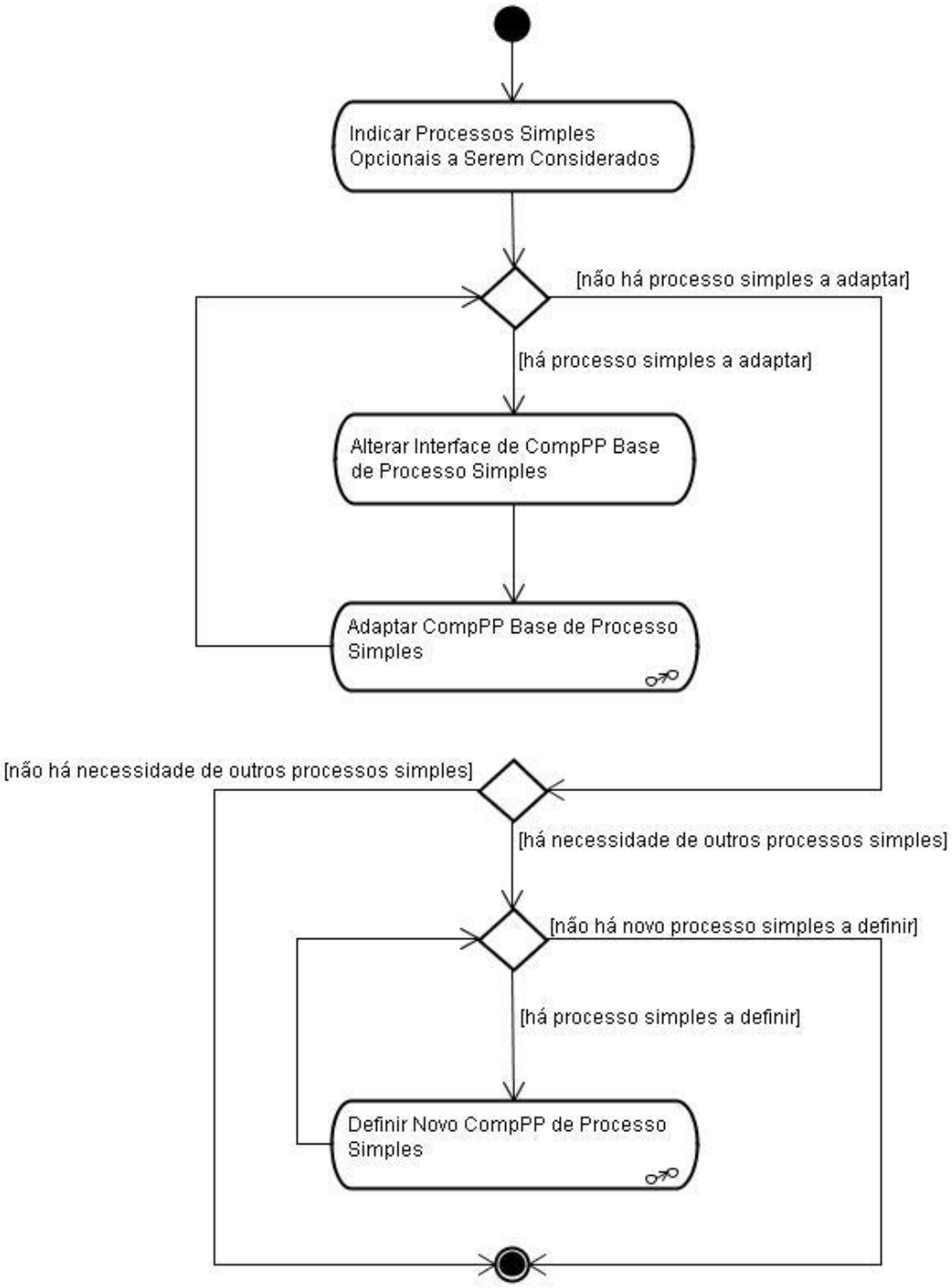

Figura 3.13 – Decomposição da Atividade Adaptar CompPP Base de Processo Complexo.

A adaptação de um CompPP de processo complexo inicia-se pela indicação de quais processos simples (subelementos) não obrigatórios do processo complexo base serão reutilizados. Logo, nem sempre todos os processos simples oriundos do processo

complexo base são reutilizados, sendo facultativa apenas a reutilização daqueles não obrigatórios.

Em seguida, para cada um dos processos simples reutilizados indiretamente, ou seja, via processo complexo base, que necessite de adaptação, deve-se alterar a sua interface respeitando a obrigatoriedade de suas macroatividades e, em seguida, adaptá-lo segundo a atividade Adaptar CompPP Base de Processo Simples, mostrada na Figura 3.14.

Caso não haja necessidade de definição de novos processos simples, chega-se ao fim da atividade Adaptar CompPP Base de Processo Complexo. Do contrário, devem-se definir os demais processos simples que comporão o processo complexo em definição, mas que não foram reutilizados indiretamente via o CompPP base. Assim, deve-se definir cada um deles realizando a atividade Definir Novo CompPP de Processo Simples, descrita posteriormente na subseção 3.3.1.5.

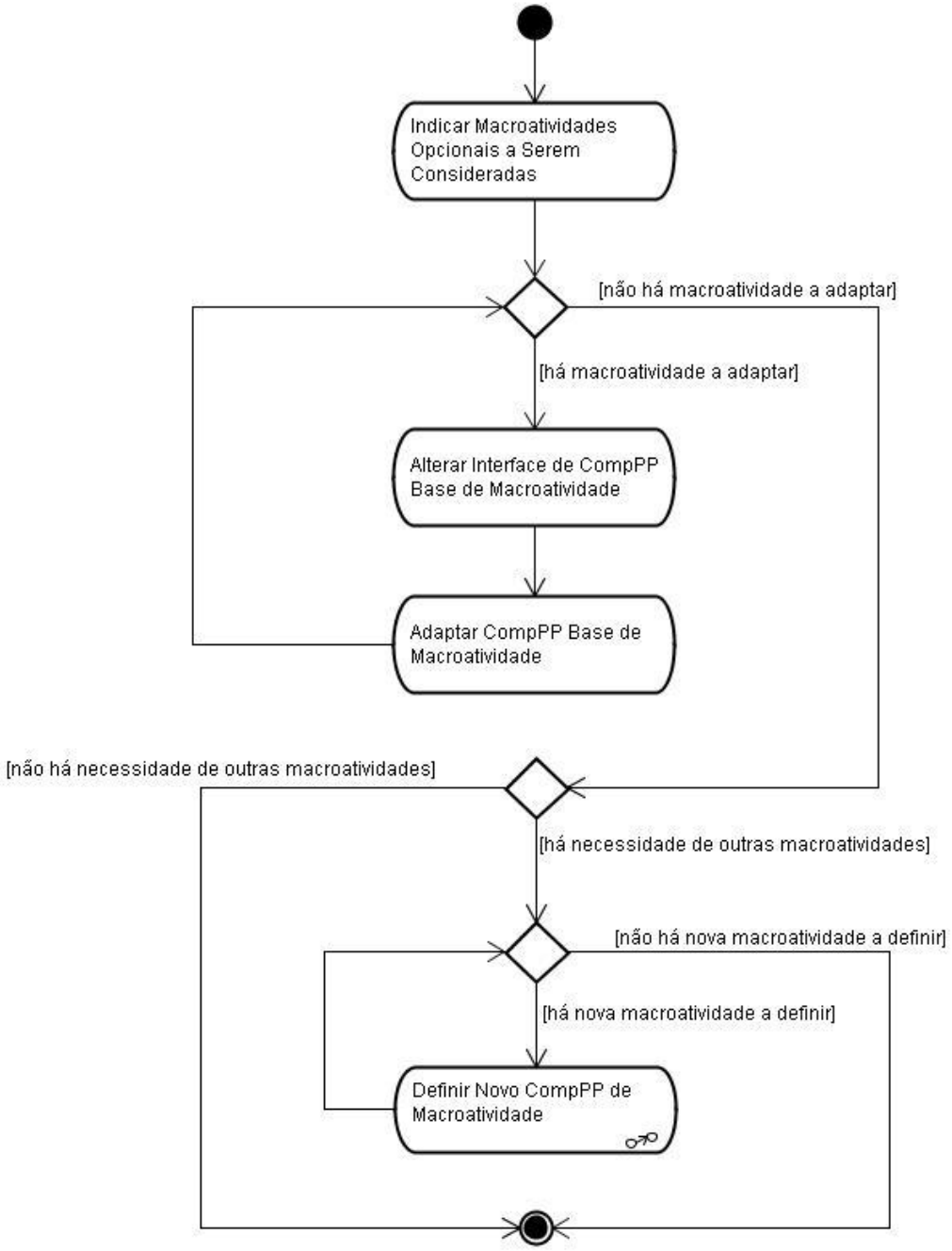

Figura 3.14 – Decomposição da Atividade Adaptar CompPP Base de Processo Simples.

Analogamente à adaptação de um CompPP de processo complexo, a adaptação de um CompPP de processo simples inicia-se pela indicação de quais macroatividades (subelementos) não obrigatórios do processo simples base serão reutilizadas. Logo, nem sempre todas as macroatividades oriundas do processo simples base são reutilizadas, sendo facultativa apenas a reutilização daquelas não obrigatórias.

Em seguida, para cada uma das macroatividades reutilizadas indiretamente, ou seja, via processo simples base, que necessite de adaptação, deve-se alterar a sua

interface respeitando a obrigatoriedade de suas subatividades. Em seguida, a macroatividade deve ser adaptada segundo a atividade Adaptar CompPP Base de Macroatividade, descrita posteriormente nesta subseção.

Caso não haja necessidade de definição de novas macroatividades, chega-se ao fim da atividade Adaptar CompPP Base de Processo Simples. Do contrário, devem-se definir as macroatividades que comporão o processo simples em definição, mas que não foram reutilizadas indiretamente via componente base. Assim, deve-se definir cada uma delas realizando a atividade Definir Novo CompPP de Macroatividade, descrita posteriormente na subseção 3.3.1.5.

A atividade Adaptar CompPP Base de Macroatividade permite a adaptação de todos os ativos de uma macroatividade, a saber: subatividades (seus subelementos), artefatos requeridos (insumos) e produzidos (produtos), recursos necessários (recursos humanos, de hardware e de software), procedimentos a serem adotados (roteiros, métodos, técnicas etc) e pré-atividades. Destaca-se que as alterações referentes às subatividades (subelementos) da macroatividade em adaptação devem ser realizadas respeitando-se a obrigatoriedade das mesmas. Desse modo, não podem ser excluídas subatividades obrigatórias, isto é, as subatividades (subelementos) sinalizadas como obrigatórios na interface do CompPP macroatividade em adaptação.

No exemplo ilustrativo, o processo padrão geral da organização é adaptado segundo os requisitos levantados para o novo processo especializado. Inicialmente, é indicado que o processo de treinamento não será reutilizado. Além disso, não há necessidade de adaptação do Processo de Gerência de Projeto.

O processo simples de desenvolvimento, por sua vez, precisa ter sua interface alterada para contemplar em seu contexto de aplicação a característica de paradigma de desenvolvimento orientado a objetos. Durante a adaptação desse processo para o paradigma orientado a objetos, opta-se pela reutilização de todas as suas macroatividades (análise de requisitos, projeto, implementação e teste). É necessário, ainda, que cada uma delas seja adaptada para o paradigma orientado a objetos. Contudo, para não alongar demasiadamente o exemplo, é discutida apenas a adaptação da atividade Análise de Requisitos.

A interface da atividade Análise de Requisitos é alterada para contemplar em seu contexto de aplicação a característica de paradigma orientado a objetos. Em seguida as suas subatividades são adaptadas. Para a subatividade Levantamento de Requisitos definem-se: como produto, Modelo de Casos de Uso; como método e técnicas

utilizadas, Modelagem de Casos de Uso e Entrevistas. Já para a subatividade Modelagem Estrutural, definem-se: como produto, Diagrama de Classes; e como método, Análise Orientada e Objetos, que também é definido para a subatividade Modelagem Comportamental, além do produto Diagramas de Estados e de Sequência. A subatividade Elaboração do Documento de Especificação de Requisitos não sofre nenhuma alteração. As Figuras 3.15 e 3.16 mostram, respectivamente, o CompPP Análise de Requisitos antes e depois da adaptação. As adaptações aparecem em destaque na Figura 3.16.

Por fim, os processos de Gerência de Configuração e Garantia da Qualidade são definidos segundo a atividade Definir Novo CompPP de Processo Simples, descrita a seguir.

Figura 3.15 – CompPP Análise de Requisitos antes da Adaptação.

Figura 3.16 – CompPP Análise de Requisitos após Adaptação.

3.3.1.5. Definir Novo CompPP do Zero

Esta atividade é executada quando um CompPP base para adaptação não é selecionado durante a atividade Selecionar CompPP Base. Nesses casos, deve-se definir o novo CompPP do zero. Essa definição ocorre de maneiras distintas, dependendo do nível de granularidade do CompPP em definição (processo complexo, processo simples e macroatividade). Entretanto, a definição de um novo CompPP de um determinado nível de granularidade envolve também a definição de CompPPs de níveis de granularidade mais finos. A Figura 3.17 mostra o primeiro nível de decomposição desta atividade.

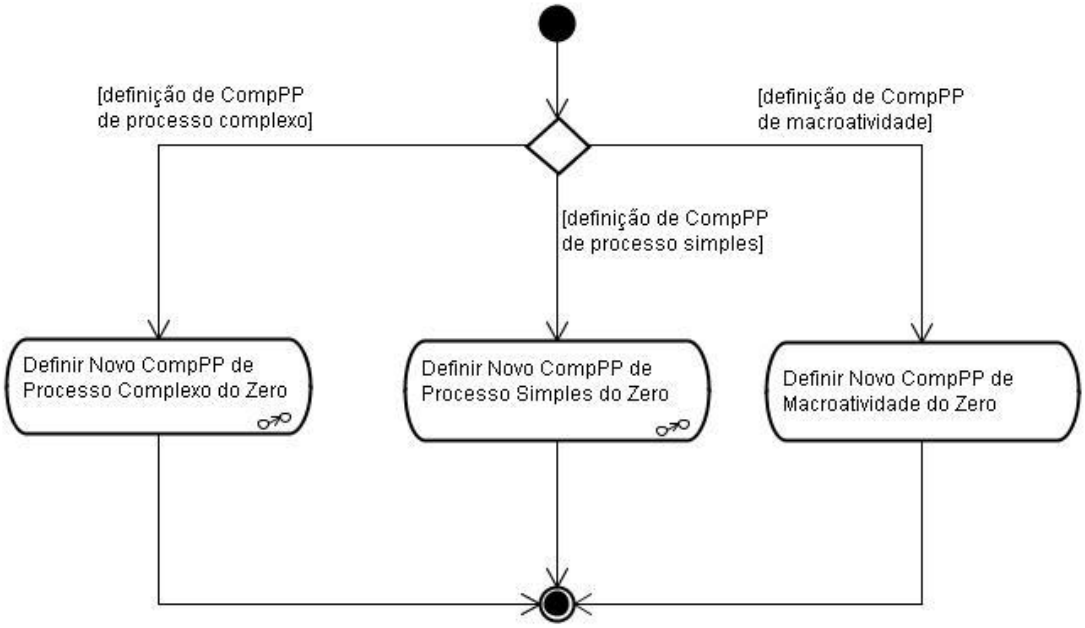

Figura 3.17 – Decomposição da Atividade Definir Novo CompPP a partir do Zero.

A atividade Definir Novo CompPP do Zero é decomposta em outras três, uma para cada nível de granularidade de CompPP. O caminho seguido depende do nível de granularidade do CompPP em definição. A Figura 3.18 mostra a decomposição da atividade Definir Novo CompPP de Processo Complexo do Zero.

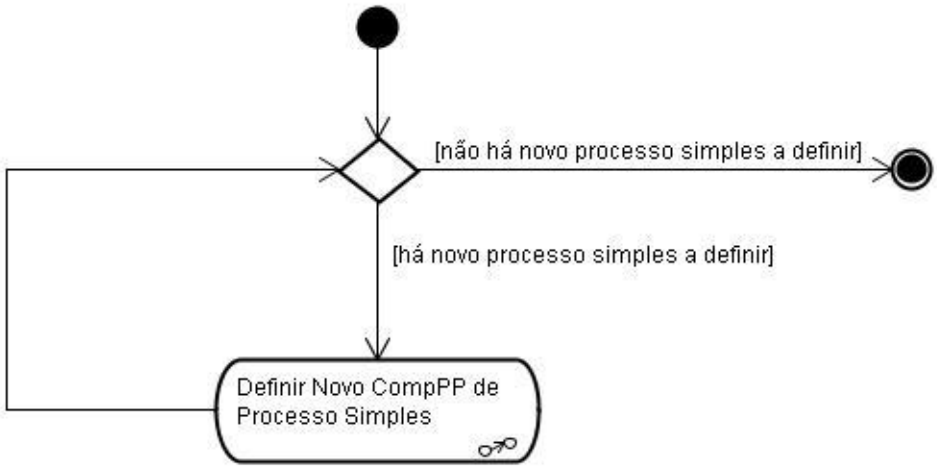

Figura 3.18 – Decomposição da Atividade Definir Novo CompPP de Processo Complexo a partir do Zero.

A definição de um novo CompPP de processo complexo do zero consiste da definição de todos os processos simples que o compõem. Assim, deve-se definir cada um deles realizando a atividade Definir Novo CompPP de Processo Simples, mostrada na Figura 3.19.

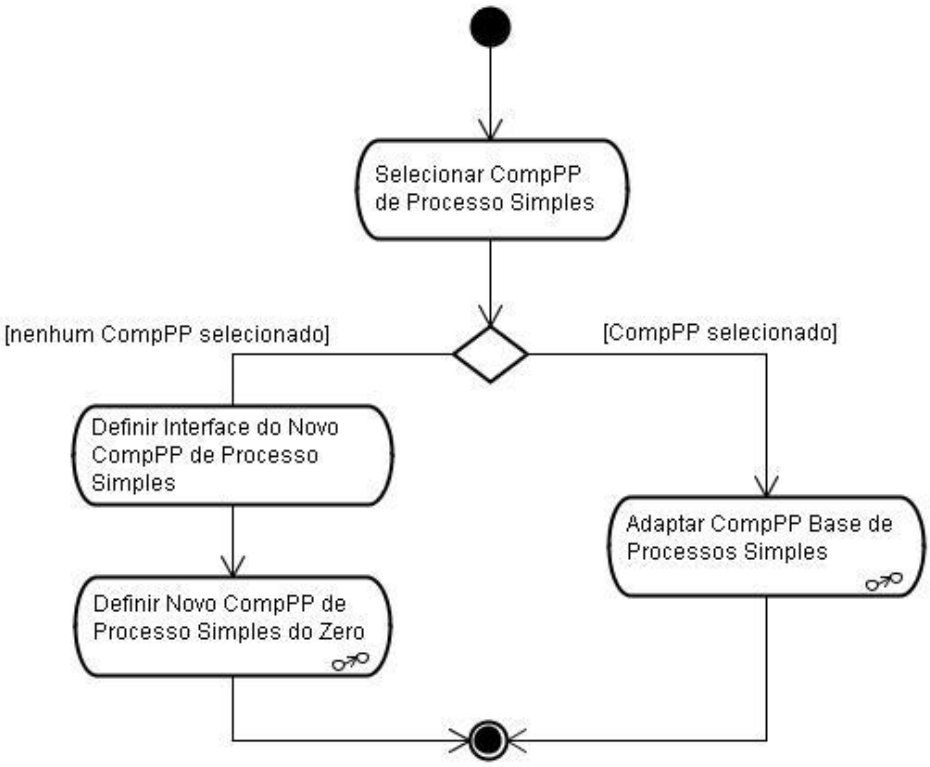

Figura 3.19 – Decomposição da Atividade Definir Novo CompPP de Processo Simples.

A definição de um novo CompPP de processo simples inicia-se pela seleção de um CompPP de processo simples base. Caso um CompPP base seja selecionado, ele é adaptado segundo a atividade Adaptar CompPP Base de Processo Simples descrita na subseção 3.3.1.4. Do contrário, deve-se definir sua interface e, em seguida, defini-lo a partir do zero, realizando a atividade Definir Novo CompPP de Processo Simples do Zero mostrada na Figura 3.20.

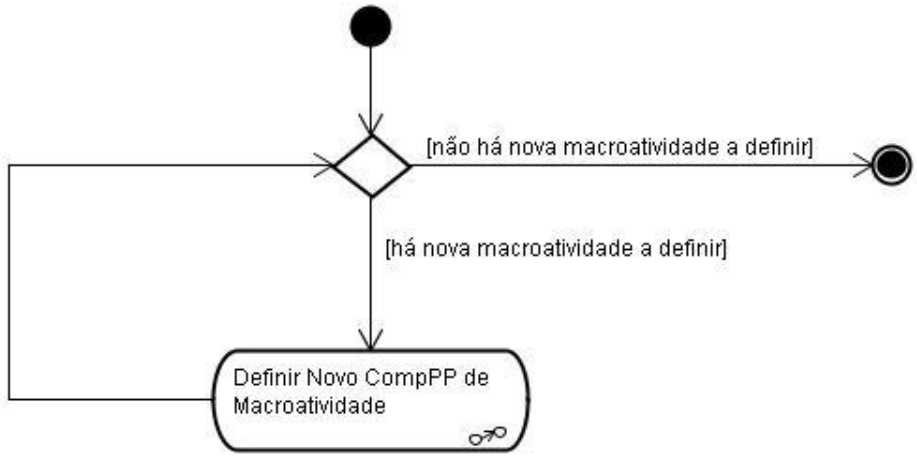

Figura 3.20 – Decomposição da Atividade Definir Novo CompPP de Processo Simples a partir do Zero.

A definição de um novo CompPP de processo simples a partir do zero passa pela definição de todas as macroatividades que o compõem. Assim, deve-se definir cada uma delas realizando a atividade Definir Novo CompPP de Macroatividade mostrada na Figura 3.21.

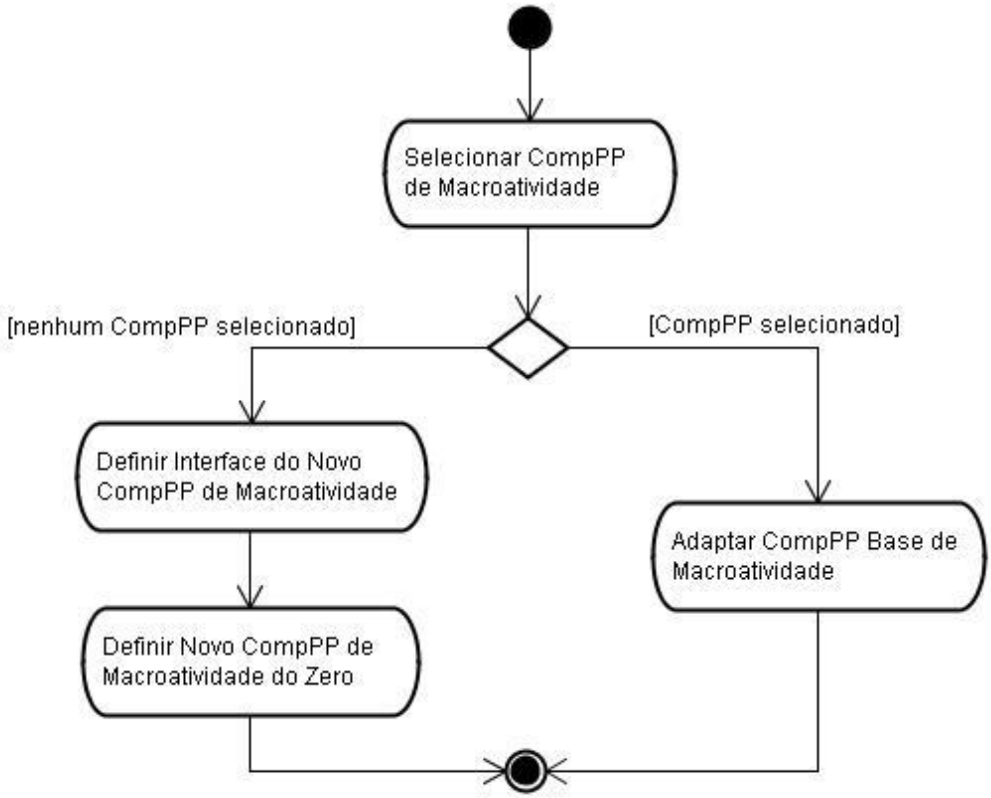

Figura 3.21 – Decomposição da Atividade Definir Novo CompPP de Macroatividade.

A definição de um novo CompPP de macroatividade inicia-se pela seleção de um CompPP de macroatividade base. Caso um CompPP base seja selecionado, ele é adaptado segundo a atividade Adaptar CompPP Base de Macroatividade descrita na subseção 3.3.1.4 Do contrário, deve-se definir sua interface e, em seguida, definir a macroatividade a partir do zero.

A definição de um CompPP de macroatividade a partir do zero envolve a definição de todos os seus ativos, a saber: subatividades (seus subelementos), artefatos requeridos (insumos) e produzidos (produtos), recursos necessários (recursos humanos, de hardware e de software), procedimentos a serem adotados (roteiros, métodos, técnicas etc) e pré-atividades.

No exemplo ilustrativo, o novo CompPP (processo especializado) é uma adaptação (especialização) do CompPP base (processo padrão geral). Entretanto, ele

deve ser composto por processos simples de Garantia da Qualidade e Gerência de Configuração, os quais não foram reutilizados indiretamente via o CompPP Base (Processo Padrão Geral) e, portanto, os mesmos devem ser definidos durante esta atividade.

Primeiramente é definido o processo simples de Gerência de Configuração. O primeiro passo é selecionar um CompPP de processo simples base, tendo sido selecionado o Processo de Gerência de Configuração já existente mostrado na Figura 3.10B. O passo seguinte seria adaptá-lo, entretanto nenhuma adaptação foi julgada necessária.

Já na definição do processo simples de Garantia da Qualidade, nenhum processo simples foi selecionado como CompPP Base, pois não existem processos simples desse tipo definidos na organização. Logo, ele deve ser definido a partir do zero. Durante a definição de sua interface, é indicado que ele deve ser composto pelas macroatividades Planejamento da Garantia da Qualidade, Realização de Auditoria, Registro de Não Conformidade e Controle de Não Conformidade, sendo todas elas obrigatórias. Não existe nenhuma característica específica de contexto de aplicação.

Depois, cada uma de suas macroatividades é definida. Como não existem CompPPs de macroatividade desses tipos já definidas, todas elas são definidas a partir do zero. Para não alongar demasiadamente o exemplo, apenas a definição da macroatividade Planejamento da Garantia da Qualidade é aqui discutida. Essa atividade não é decomposta em subatividades e nem possui características específicas de contexto de aplicação. Como recursos necessários, têm-se dois recursos humanos: Gerente de Projeto e Revisor. Como artefato produto, tem-se o Plano de Avaliação para Garantia da Qualidade, como mostra a Figura 3.22.

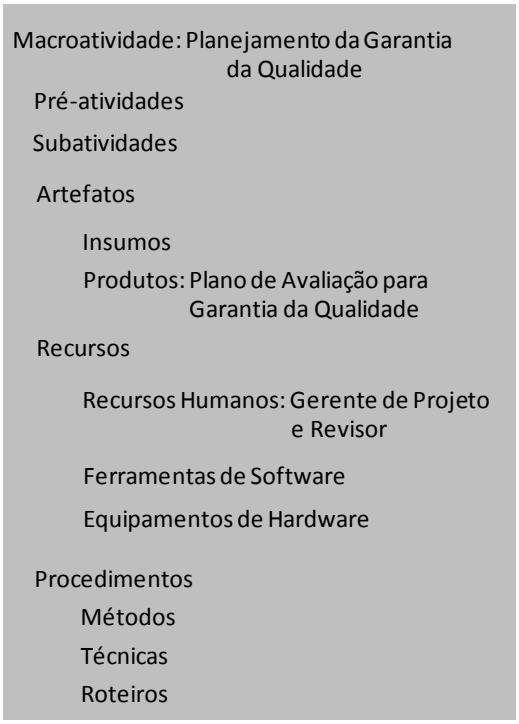

Figura 3.22 – Macroatividade Planejamento da Garantia da Qualidade.

O Processo Padrão Especializado OO, resultado da especialização do Processo Padrão Geral, é apresentado na Figura 3.23.

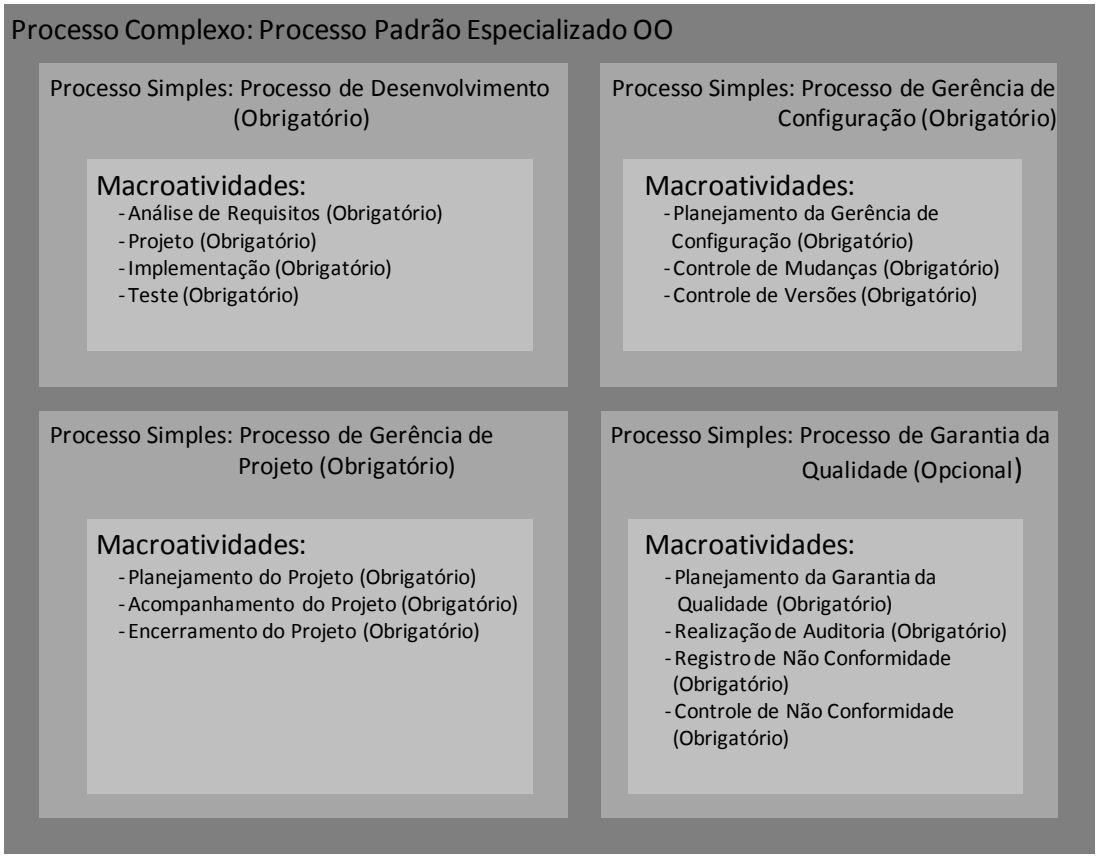

Figura 3.23 – Processo Padrão Especializado OO Resultante.

3.3.2. Processo de DPBC para o Nível de Abstração de Processo de Projeto

A Figura 3.24 mostra um diagrama de atividades apresentando as principais atividades do processo de DPBC para o nível de processos de projeto. Para esse nível de abstração é possível definir apenas processos como um todo (componentes de nível de granularidade de processo complexo), uma vez que a definição aqui sempre ocorre no contexto de um projeto. Esse processo trata das duas modalidades de reutilização (horizontal e vertical), já que tanto CompPPs quanto componentes de nível de abstração de processo de projeto podem ser utilizados na definição de novos processos de projeto, ou seja, a definição de processos de projeto é realizada envolvendo componentes dos dois níveis de abstração de processo.

Destaca-se que, nesse nível de abstração, os componentes não são definidos visando à reutilização e, por isso, não possuem interfaces definidas explicitamente. As suas interfaces, contudo, podem ser extraídas, sendo seus contextos de aplicação dados pelas caracterizações de seus respectivos projetos e suas estruturas dadas por seus respectivos subelementos. Os componentes desse nível também podem ser recuperados, adaptados e reutilizados. Entretanto, diferentemente dos CompPPs, que são definidos visando à reutilização, os componentes de processo de projeto são sempre definidos especificamente para um projeto, podendo ser reutilizados posteriormente.

As subseções a seguir descrevem as atividades do processo de DPBC para o nível de processos de projeto. Para ajudar no entendimento do processo proposto, um exemplo ilustrativo é usado para mostrar como as atividades descritas podem ser realizadas. Nesse exemplo, são utilizados os CompPPs mostrados nas Figuras 3.10 e 3.20 e o processo de projeto mostrado na Figura 3.25, denominado Processo de Projeto XPTO, utilizado no desenvolvimento de um sistema governamental (domínio da aplicação) em plataforma Web (tipo de software), segundo o paradigma orientado a objetos (paradigma utilizado). O contexto do exemplo é a definição do processo de um novo projeto, em que são reutilizados componentes de ambos os níveis de abstração de processo. O novo projeto desenvolverá um sistema de informação em plataforma Web, seguindo o paradigma orientado a objetos, em que os requisitos possuem alto grau de estabilidade. Ainda, é necessário que o processo do projeto considere parcialmente o nível F do MPS.BR, à exceção dos processos de Medição e Aquisição. O cliente

necessita que seja disponibilizada rapidamente uma versão do sistema, sendo necessário configurar o ambiente e converter algumas bases de dados de sistemas legados para que ele entre em produção.

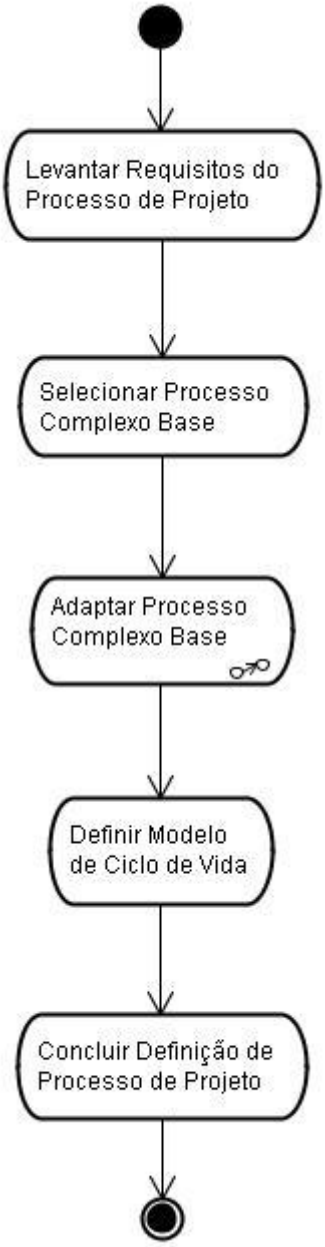

Figura 3.24 – Processo de DPBC para o Nível de Processos de Projeto.

Figura 3.25 –Exemplo de Execução do Processo de DPBC para o Nível de Processo de Projeto: Processo de Projeto Anterior.

3.3.2.1. Levantar Requisitos do Novo Processo de Projeto

Esta atividade visa ao levantamento dos requisitos do processo de projeto em definição. De maneira análoga à definição de CompPP, devem-se definir os objetivos e requisitos do novo processo de projeto. Destaca-se que as características do projeto podem derivar requisitos para o processo.

Antes de se iniciar de fato a definição do processo de um projeto, é necessário efetuar a caracterização desse projeto. Essa caracterização é necessária, uma vez que é a partir das características do projeto que será possível recuperar componentes para apoiar a definição de seu processo.

No exemplo ilustrativo, a organização está iniciando um novo projeto, o qual desenvolverá um novo sistema de informação. Assim, duas características do projeto já estão estabelecidas: o tipo de software (sistema de informação) e a natureza do projeto (novo desenvolvimento). Ainda, sabe-se que os requisitos possuem um alto grau de estabilidade e o cliente necessita que seja disponibilizada rapidamente uma versão do sistema. Por fim, o projeto será desenvolvido utilizando-se o paradigma orientado a objetos, em plataforma Web.

Além disso, conforme citado anteriormente, como parte do levantamento de requisitos, constatou-se que o projeto deveria ser conduzido considerando parcialmente

o nível F do MPS.BR, excluindo-se os processos de Medição e Aquisição. Assim, o processo do novo projeto deve possuir processos de Gerência de Configuração e Garantia da Qualidade, em adição aos processos que compõem o Processo Padrão Geral da organização (Figura 3.10A). Por fim, sabe-se no decorrer do projeto será necessário ministrar um treinamento aos desenvolvedores.

3.3.2.2. Selecionar Processo Complexo Base

Um processo de projeto deve ser definido através da adaptação de um componente de processo complexo já existente, dito processo base, sendo que esse componente pode ser de qualquer um dos níveis de abstração de processos (padrão e de projeto).

Assim como a seleção do CompPP base, a seleção do processo complexo base pode ocorrer de duas maneiras. Na primeira delas, utilizada quando já se tem conhecimento de qual será o processo base, seleciona-se diretamente o processo complexo desejado. A segunda é utilizada quando não se tem conhecimento prévio de qual processo complexo será adaptado. Então, tomando por base os requisitos do processo, inicia-se pela indicação das características desejáveis para ele e dos processos simples que devem aparecer em sua composição. Note que o processo complexo base não necessariamente deve possuir todos os processos simples necessários ao novo processo de projeto.

Em seguida, são buscados os componentes de processo complexo (padrão e de projeto) que possuam as características desejáveis e os processos simples indicados anteriormente. Por fim, deve-se decidir sobre a utilização de algum deles como processo base.

No exemplo ilustrativo, o gerente de projeto indicou como característica desejável para o componente de processo complexo base o paradigma orientado a objetos. Indicou também que devem aparecer em sua composição processos simples de desenvolvimento, gerência de projetos, gerência de configuração, garantia da qualidade e treinamento. Com base nesses requisitos, ele decidiu que o novo processo de projeto será definido a partir do processo padrão especializado OO da organização (Figura 3.23).

3.3.2.3. Adaptar Processo Complexo Base

Esta atividade tem por objetivo a adaptação do processo base a partir do qual o novo processo de projeto está sendo definido. Destaca-se que a definição de um processo de projeto sempre ocorre a partir de um processo base, podendo o processo base pertencer a qualquer um dos dois níveis de abstração de processos (padrão e de projeto). Esta atividade é decomposta no diagrama de atividades da Figura 3.26.

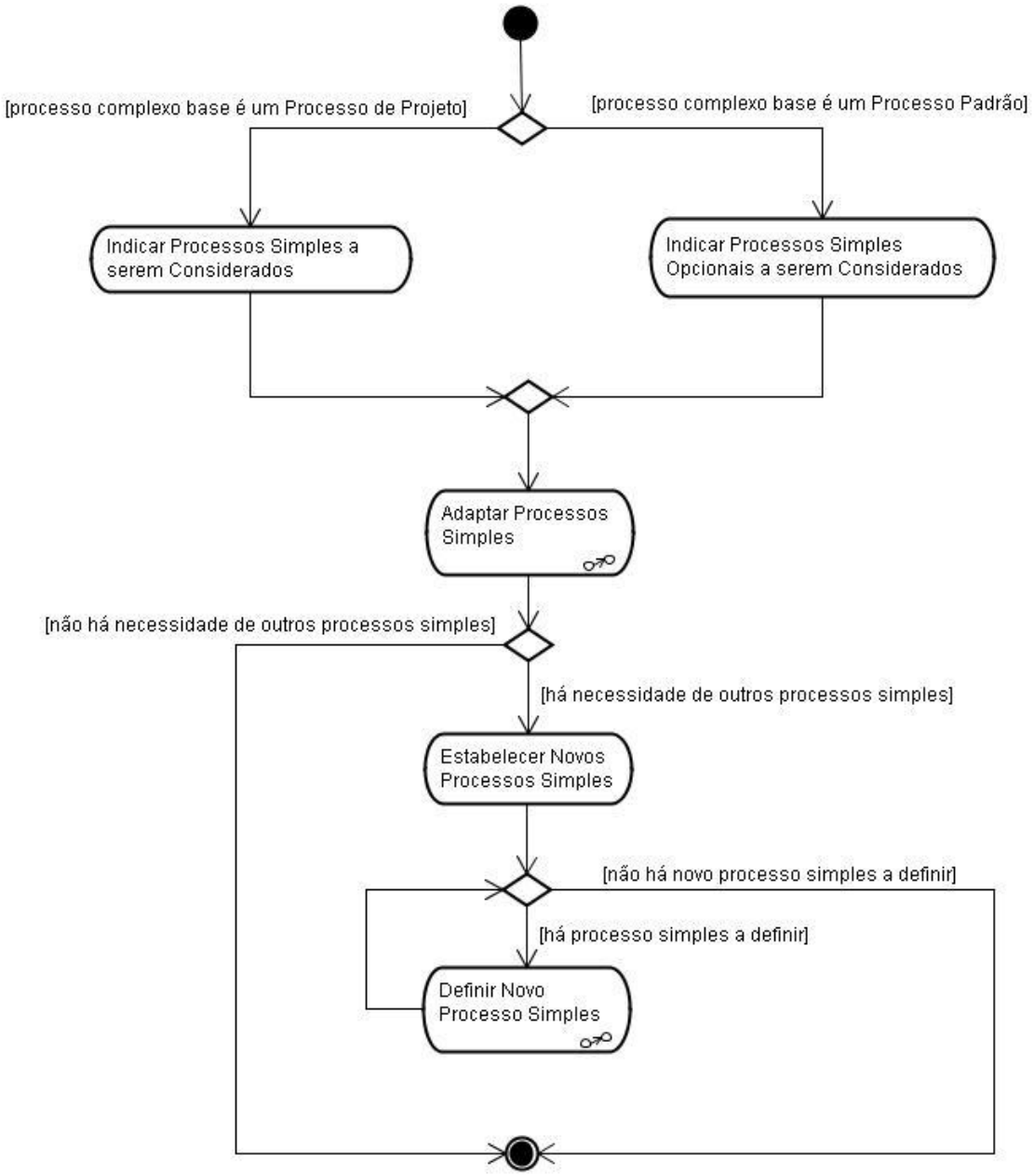

Figura 3.26 – Decomposição da Atividade Adaptar Processo Complexo Base.

A adaptação do processo base inicia-se pela indicação de seus processos simples que serão reutilizados. Nos casos em que o processo base pertence ao nível de abstração

de processos padrão, deve-se respeitar a obrigatoriedade de seus subelementos. Por outro lado, caso o processo base pertença ao nível de abstração de processos de projeto, a indicação de reutilização de seus subelementos não está sujeita a nenhuma restrição.

A seguir, cada um dos processos simples que estão sendo reutilizados indiretamente, ou seja, via processo base, pode ser adaptado. A atividade de adaptação de processo simples é mostrada na Figura 3.27.

Após a adaptação dos processos simples reutilizados indiretamente, caso não haja necessidade de definição de novos processos simples, encerra-se a execução desta atividade. Do contrário, são estabelecidos os demais processos simples que deverão compor o processo de projeto em definição, mas que não foram reutilizados via processo base, para posterior definição. A atividade de adaptação do processo complexo base chega ao fim quando todos os novos processos simples necessários forem definidos. A atividade de definição de novo processo simples é discutida mais à frente nesta subseção.

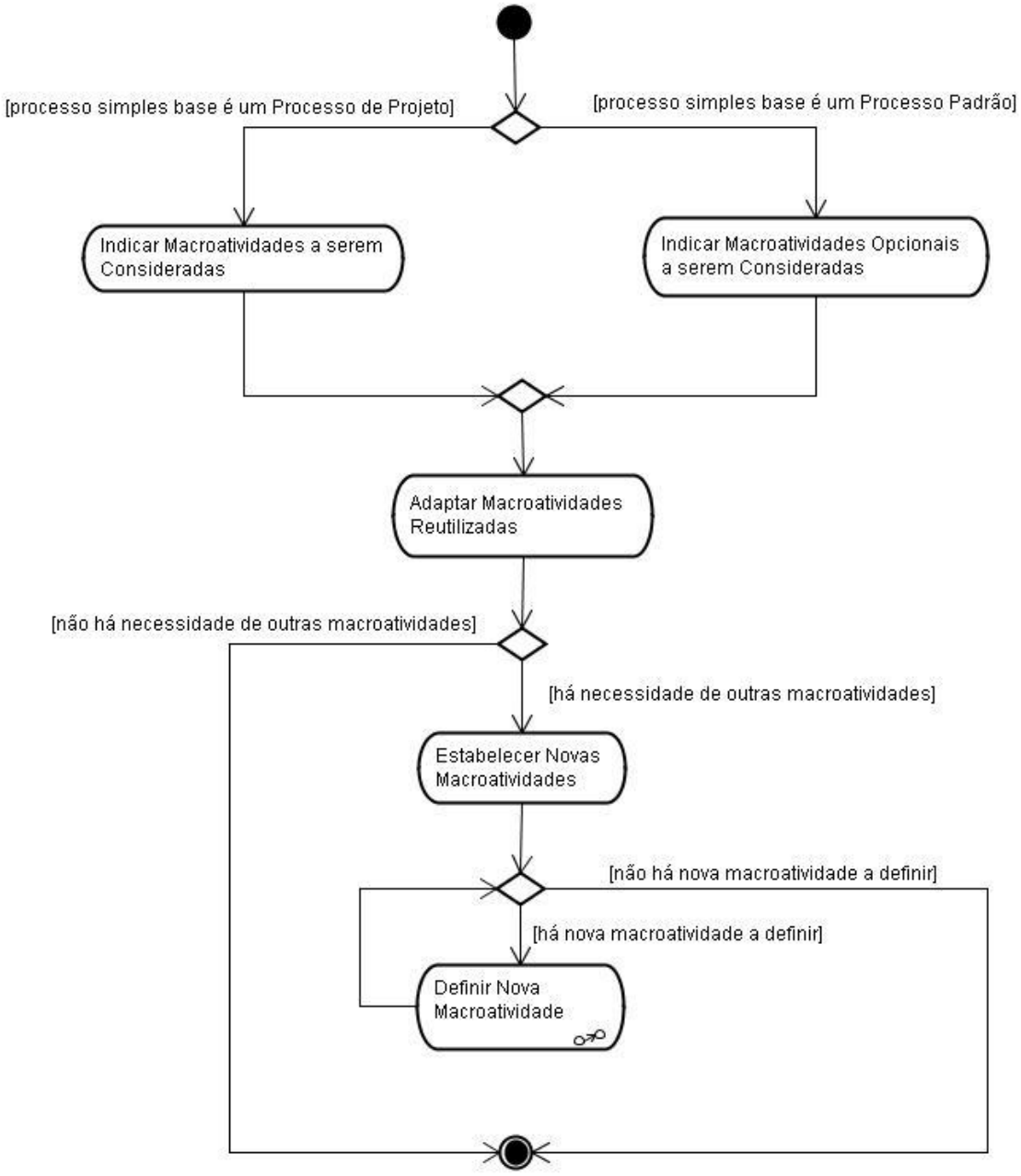

Figura 3.27 – Decomposição da Atividade Adaptar Processo Simples.

A adaptação de um processo simples inicia-se pela indicação das suas macroatividades que serão reutilizadas. Caso o processo simples seja do nível de abstração de processos padrão, deve-se respeitar a obrigatoriedade de suas macroatividades. Por outro lado, caso o processo simples seja do nível de abstração de processos de projeto, não há restrições quanto à reutilização de suas macroatividades.

Depois, cada uma das macroatividades que estão sendo reutilizadas indiretamente pode ser adaptada. A atividade de adaptação de macroatividades reutilizadas é descrita mais à frente nesta mesma seção.

Após a adaptação das macroatividades reutilizadas indiretamente, caso não haja necessidade de definição de novas macroatividades, encerra-se a adaptação do processo simples. Do contrário, são estabelecidas as demais macroatividades que deverão compor o processo simples em adaptação, mas que não foram reutilizadas via processo simples base, para posterior definição. A atividade de adaptação do processo simples base chega ao fim quando todas as novas macroatividades necessárias forem definidas. A atividade de definição de nova macroatividade é discutida posteriormente nesta mesma subseção.

A Figura 3.28 mostra a decomposição da atividade Definir Novo Processo Simples.

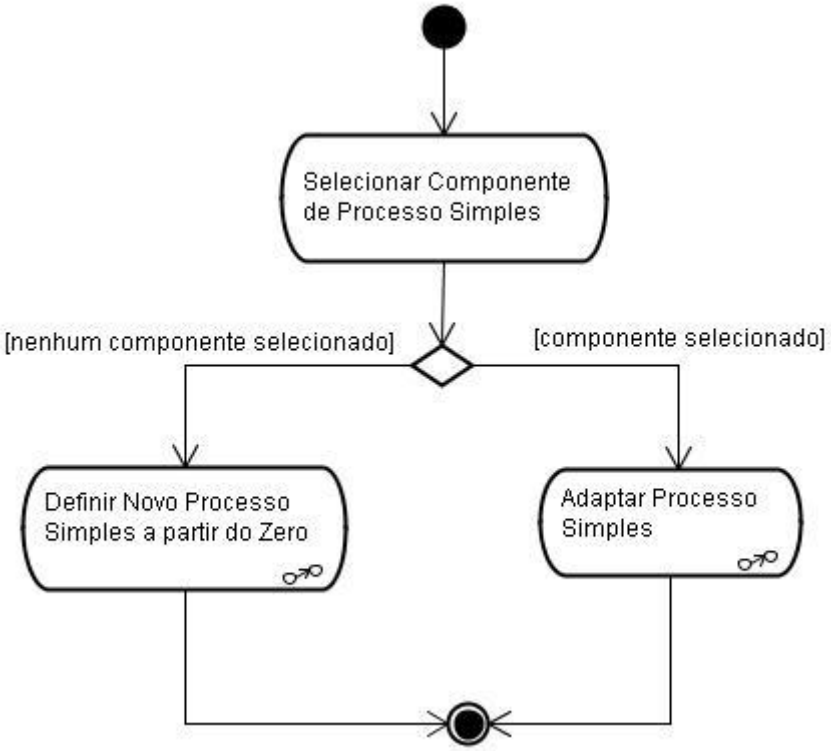

Figura 3.28 – Decomposição da Atividade Definir Novo Processo Simples.

A definição de um novo processo simples inicia-se pela seleção de um componente de processo simples base. Caso seja selecionado um componente de processo simples base, ele é adaptado conforme a atividade Adaptar Processo Simples, descrita anteriormente. Do contrário, deve-se definir um novo processo simples a partir do zero. A Figura 3.29 mostra a decomposição da atividade Definir Novo Processo Simples a partir do Zero.

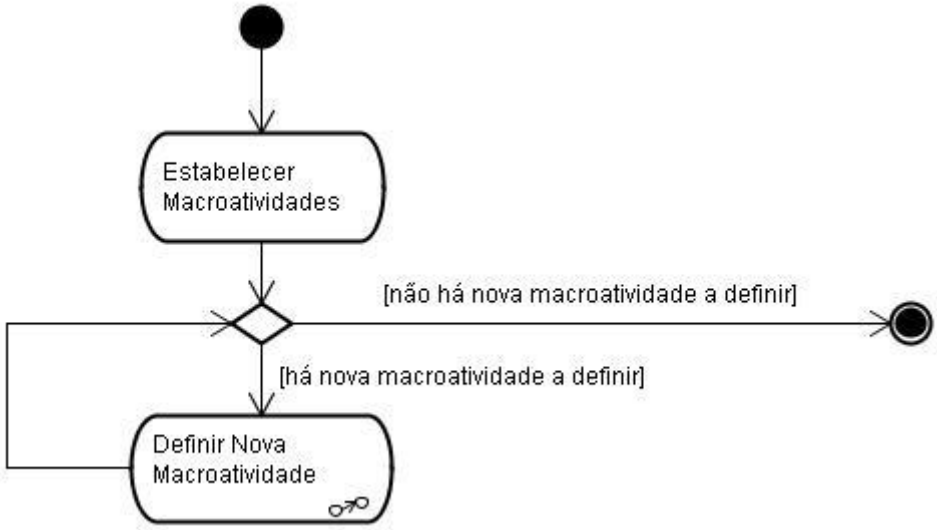

Figura 3.29 – Decomposição da Atividade Definir Novo Processo Simples a partir do Zero.

Ao se definir um novo processo simples a partir do zero, deve-se primeiramente estabelecer as macroatividades que o comporão. Depois, cada uma dessas macroatividades deve ser definida. A atividade Definir Novo Processo Simples a partir do Zero encerra-se quando todas elas tiverem sido definidas. A Figura 3.30 mostra a decomposição da atividade Definir Nova Macroatividade.

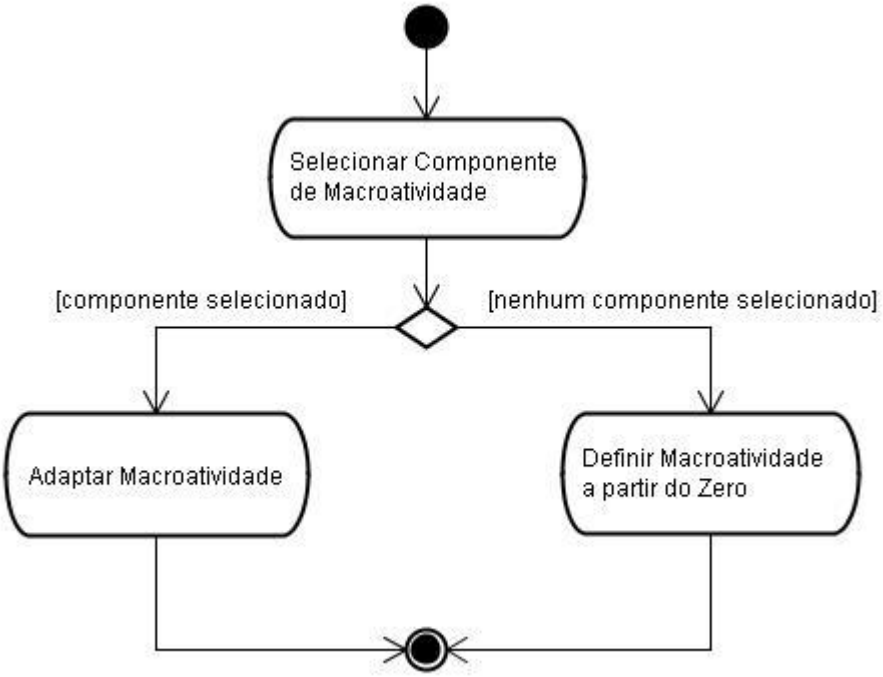

Figura 3.30 – Decomposição da Atividade Definir Nova Macroatividade.

A definição de uma nova macroatividade inicia-se pela seleção de um componente de macroatividade base. Caso seja selecionado um componente de macroatividade base, ele pode ser adaptado. Do contrário, deve-se definir uma nova macroatividade a partir do zero.

A adaptação de uma macroatividade pode envolver a redefinição de todos os seus ativos, a saber: subatividades (seus subelementos), artefatos requeridos (insumos) e produzidos (produtos), recursos necessários (recursos humanos, de hardware e de software), procedimentos a serem adotados (roteiros, métodos, técnicas etc) e pré-atividades. Destaca-se que as alterações referentes às subatividades (subelementos) da macroatividade em adaptação oriundas do nível de abstração de processos padrão devem ser realizadas respeitando-se suas obrigatoriedades. De maneira análoga, a definição de uma nova macroatividade também envolve a definição de todos os seus ativos.

No exemplo ilustrativo, o novo processo de projeto é definido a partir do CompPP Processo Padrão Especializado OO, que já foi selecionado como processo base durante a atividade Selecionar Processo Complexo Base. Inicialmente, são indicados quais dos processos simples que compõem o processo complexo base serão reutilizados. Neste caso, todos os processos simples do processo complexo base serão reutilizados (Desenvolvimento, Gerência de Projeto, Gerência de Configuração e Garantia da Qualidade).

Em seguida, esses processos simples reutilizados indiretamente, ou seja, via processo complexo base, são adaptados. No exemplo, apenas o Processo de Desenvolvimento precisa ser adaptado. Opta-se pela reutilização de todas as suas macroatividades (Análise de Requisitos, Projeto, Implementação e Teste). Contudo, é necessário, ainda, que cada uma delas seja adaptada considerando-se o desenvolvimento em plataforma Web. Para não alongar demasiadamente o exemplo, é discutida apenas a adaptação da atividade Implementação. São adicionadas a essa macroatividade as normas Programação em JSP e Programação em Java. Ainda, define-se a utilização de da IDE NetBeans como ferramenta de software. A Figura 3.31 mostra a macroatividade Implementação antes e depois da adaptação, com destaque para as alterações realizadas.

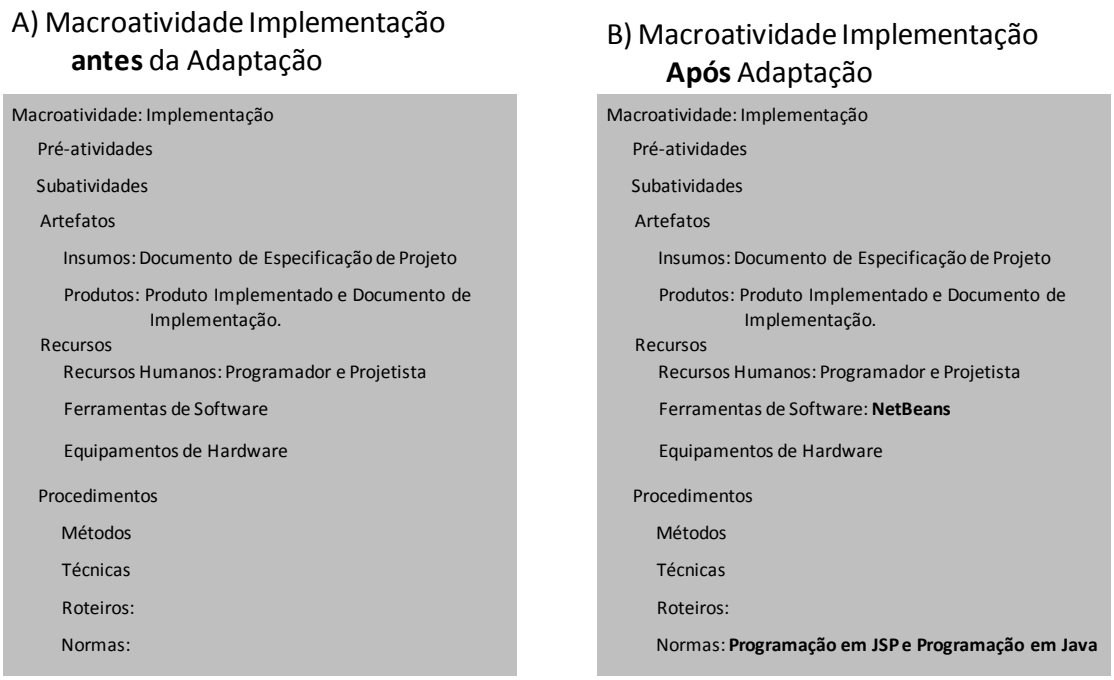

Figura 3.31 – Adaptação da Macroatividade Implementação.

Continuando com a adaptação do Processo de Desenvolvimento, é estabelecido que uma macroatividade de implantação deve ser acrescentada ao Processo de Desenvolvimento. Então, a macroatividade de implantação do Processo de Projeto XPTO é selecionada e reutilizada sem necessidade de adaptação.

Na sequência, é estabelecido que um processo de treinamento deve ser definido para o novo processo de projeto, conforme os requisitos levantados durante a atividade Levantar Requisitos do Novo Processo de Projeto e, em seguida, parte-se para sua definição. O CompPP Processo de Treinamento, que aparece na composição do CompPP Processo Padrão Geral (Figura 3.10), é selecionado e reutilizado sem necessidade de adaptação. A Figura 3.32 mostra o novo processo de projeto resultante da adaptação do processo complexo base.

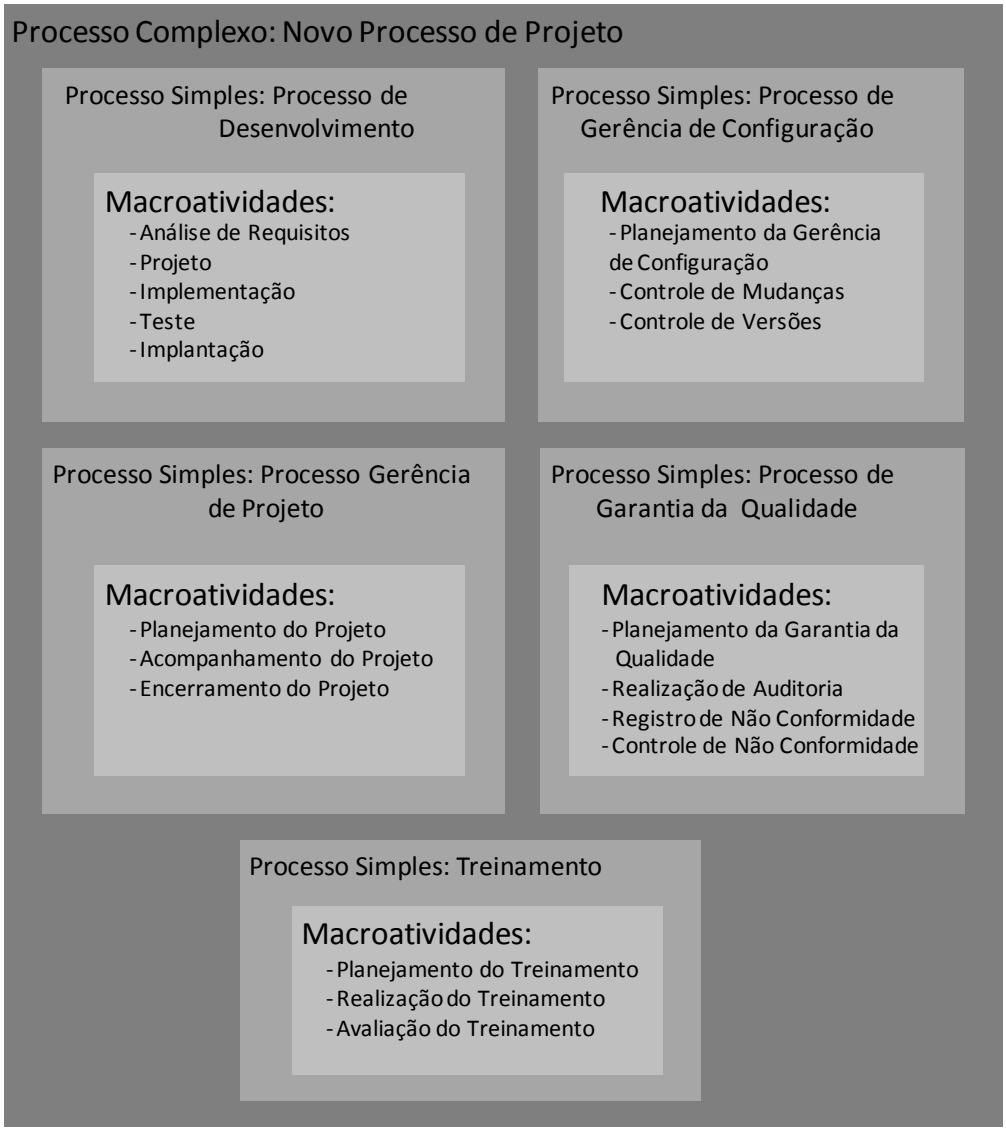

Figura 3.32 – Processo de Projeto Parcial.

3.3.2.4. Definir Modelo de Ciclo de Vida

Durante esta atividade é definido o modelo de ciclo de vida a ser adotado (p. ex., sequencial, evolutivo, incremental etc.). Nos casos de modelos iterativos, para cada combinação iterativa de atividades, deve-se indicar o número de iterações a serem realizadas.

No exemplo ilustrativo, decidiu-se utilizar o modelo incremental, haja vista que, conforme apontado na caracterização do projeto, os seus requisitos são estáveis e há necessidade de entregar rapidamente uma versão do sistema. As atividades do processo simples de desenvolvimento foram divididas em duas combinações, sendo a primeira delas uma combinação sequencial contemplando apenas a macroatividade de Análise de Requisitos e a segunda uma combinação iterativa contemplando as demais macroatividades (Projeto, Implementação, Teste e Implantação). Ainda, foi indicado que seriam utilizadas duas iterações. A Figura 3.33 mostra a estrutura das combinações de atividades definida.

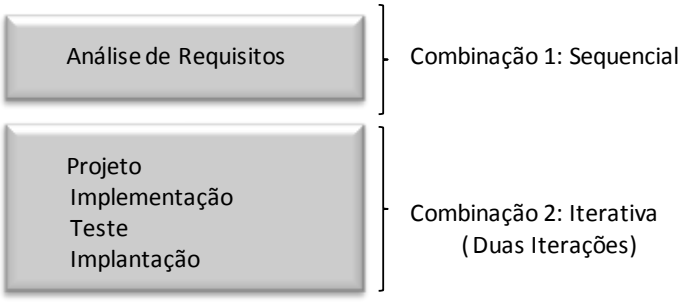

Figura 3.33 – Combinações de Atividades do Modelo de Ciclo de Vida Definido.

3.3.2.5. Concluir Definição de Processo de Projeto

Após a escolha do modelo de ciclo de vida a ser adotado, pode ser necessário realizar alguns ajustes de dependência (pré-atividades) entre as atividades oriundas das iterações dos modelos de ciclo de vida iterativos. Esses ajustes são realizados durante esta atividade.

No exemplo ilustrativo, a segunda iteração depende da primeira e, dessa forma, são realizados os seguintes ajustes nas pré-atividades: Projeto 02 tem como pré-atividade Projeto 01; Teste 02 tem como pré-atividade Teste 01; Implementação 02 tem como pré-atividade Implementação 01; e, por fim, Implantação 02 tem como pré-atividade Implantação 01. A Figura 3.34 mostra o processo de projeto resultante, inclusive com a repetição das atividades devido às iterações definidas pelo modelo de ciclo de vida adotado.

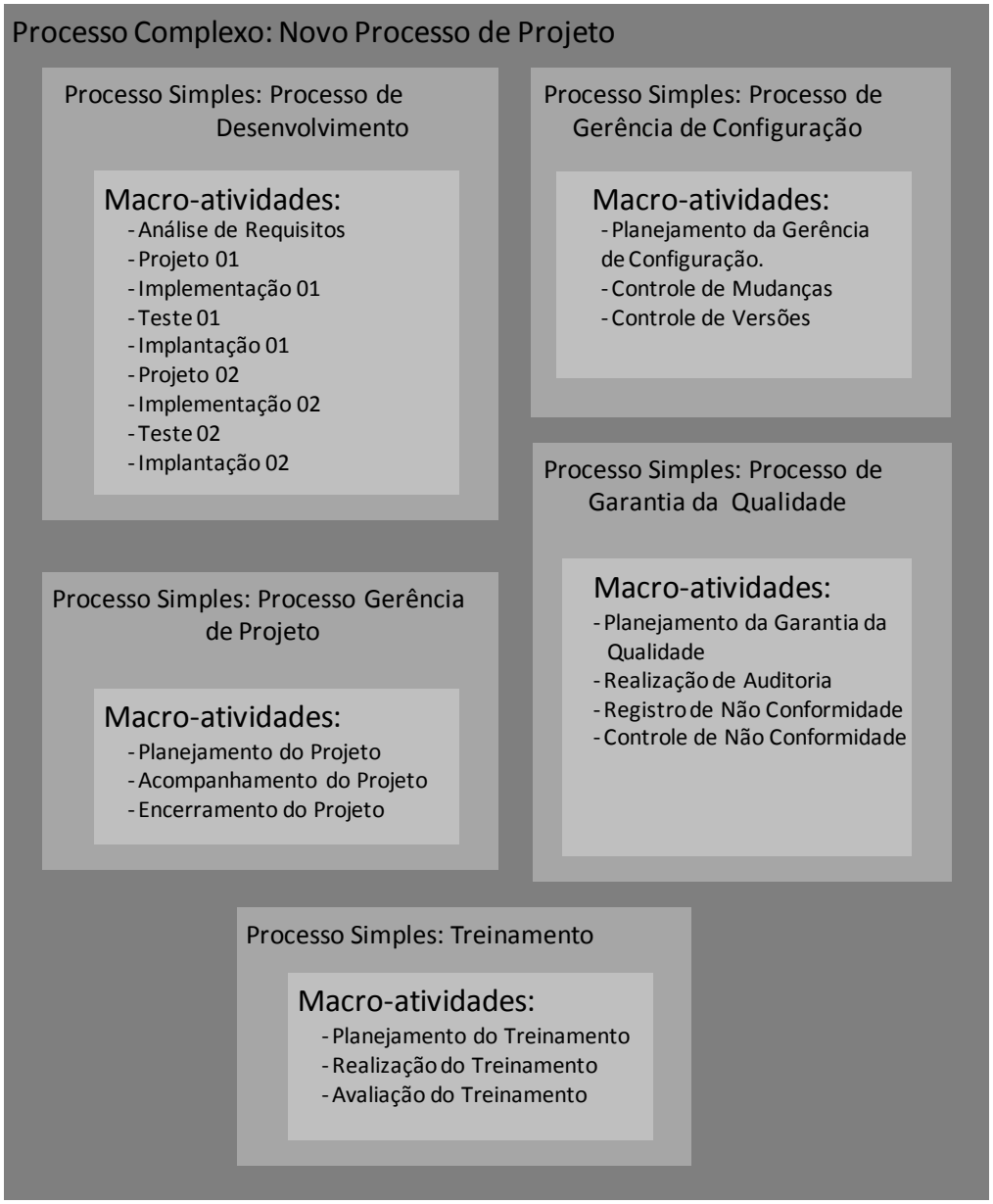

Figura 3.34 – Processo de Projeto Resultante.

3.4. Comparação com Trabalhos Correlatos na Literatura

Segundo JAUFMAN e MÜNCH (2005), os trabalhos desenvolvidos na área de reutilização de processos de software podem ser divididos em dois grandes grupos: aqueles focados na componentização de processos, pregando a definição de processos a partir da (re)utilização de componentes de menor granularidade, e os que seguem a linha de padrões de processo, *templates* de processo e arquiteturas de linhas de processo, em que processos específicos são definidos a partir de processos genéricos. A abordagem de Definição de Processos Baseada em Componentes (DPBC) aqui proposta está, portanto, inserida no contexto do primeiro grupo de trabalhos.

GARY e LINDQUIST (1999) definem um componente de processo como um encapsulamento de informações e comportamentos de processo em um dado nível de granularidade. Ainda segundo eles, um modelo de processo definido com base em componentes é uma coleção de componentes de processo que interagem de alguma forma. Os modelos de processo definidos com base em componentes também são vistos como componentes de processo. FUSARO et al. (1998) reafirmam essa idéia de componentização de processos de software, tratando um processo como a integração de um conjunto de componentes de processo em diferentes níveis de granularidade. BARRETO et al. (2008) consideram que um componente de processo pode ser definido com qualquer nível de detalhamento, isto é, desde uma simples atividade até um processo completo.

A DPBC trata componentes de processo utilizando a visão de que um processo de software pode ser composto por outros processos ou por atividades, conforme apontado pela ontologia de processos apresentada em (BERTOLLO, 2006). Assim, a DPBC trata componentes de processo em três níveis de granularidade, a saber: processo complexo, processo simples e macroatividade. Apesar de subatividades também serem importantes elementos de um processo, para a DPBC, elas não foram consideradas como componentes de processo. Contudo, quando uma macroatividade é reutilizada, suas subatividades também o são, assim como as demais definições relativas a elas. Essa decisão foi tomada, uma vez que as subatividades estão em um nível de granularidade muito fino. Com isso, seria difícil conseguir definir componentes a partir de subatividades num nível de abstração que permitisse serem reutilizados na composição de processos distintos. Esse problema desencadeia outro: o repositório de componentes ficaria demasiadamente povoado, o que tornaria a busca por componentes para reutilização menos eficiente.

Segundo BASILI e ROMBAC (1987), definir processos através da composição de componentes de processo já existentes exige um mecanismo adequado para avaliação do componente quanto à sua aplicação no contexto específico. Deve-se ser capaz de verificar se um componente pré-existente pode ser utilizado para a definição de um determinado processo, atendendo aos requisitos estabelecidos (RU-ZHI et al., 2005).

Nesse sentido, a interface de componentes de processo descrita pela DPBC auxilia a tarefa de busca e seleção de componentes. As interfaces de componentes de processo de software definem as características de aplicação do componente e os seus subelementos, ou seja, definem tanto a estrutura do componente quanto o seu contexto

de aplicação. Essas informações são utilizadas na busca de componentes, tornando possível saber se um determinado componente de processo (p. ex., macroatividade de planejamento de projeto) pode ser utilizado na definição de outro componente de processo de maior granularidade (p. ex., processo simples de gerência de projeto). Neste aspecto, a DPBC encontra-se bastante alinhada com o trabalho proposto por BARRETO et al. (2008), que preconiza a utilização de características do processo para tratar o problema da busca por componentes adequados. Já FUSARO et al. (1998) utilizam cinco grupos de características de componentes de processo para que eles possam ser compreendidos e avaliados: (i) Entrada, que define as características dos objetos necessários para operação do componente; (ii) Saída, que define as características dos objetos produzidos pelo componente; (iii) Ênfase, que se refere aos objetivos do componente; (iv) Domínio, que define as características que devem ser satisfeitas pelos objetos de entrada para que o componente possa ser executado; e (v) Ambiente, que define o estado inicial do ambiente externo necessário para o início da execução do componente e o estado final produzido pela sua execução. Conforme visto anteriormente, diferentemente do trabalho de FUSARO et al. (1998), a DPBC trabalha com apenas dois tipos de informação sobre os componentes de processo: sua estrutura básica e seu contexto de aplicação.

Por outro lado, o trabalho desenvolvido por RU-ZHI et al. (2005) apresenta três níveis de descrição de informações sobre componentes de processo: (i) descrição das informações gerais do componente de processo, (ii) descrição da especificação do componente de processo e (iii) descrição dos dados do componente de processo. As informações sobre os componentes foram divididas em três níveis, pois diferentes tipos de usuários buscam diferentes tipos de informação. Além disso, a reutilização de processos não inclui apenas a reutilização de componentes de processo, mas também a reutilização de informações relacionadas à utilização dos mesmos, incluindo conhecimento e experiências adquiridas (melhores práticas, lições aprendidas etc). Esse tipo de informação acerca dos componentes não é explicitamente tratada pela DPBC, mas pode vir a ser explorado futuramente.

As abordagens de reutilização de processo que trabalham com a componentização dos processos necessitam de repositórios de componentes, a partir dos quais componentes possam ser buscados e selecionados para serem utilizados na definição de componentes de maior granularidade. A abordagem proposta por BARRETO et al. (2008) prega a utilização de repositórios para o armazenamento de

componentes de processo, linhas de processo, conhecimento relacionado à execução dos processos, medições relacionadas ao uso dos componentes, entre outros. Esses repositórios podem ser utilizados durante a definição de processos para as organizações ou projetos (definição com reúso), em que componentes e outros itens reutilizáveis podem ser buscados e usados para compor os processos.

Em linha com essa necessidade, a DPBC propõe a utilização de dois repositórios: um para componentes de processo do nível de abstração de processo padrão (CompPP) e outro para componentes de processo do nível de abstração de processos de projeto. O primeiro deles está dividido em três partes, sendo que cada uma delas abriga apenas componentes do nível de granularidade correspondente (processo complexo, processo simples e macroatividade). Por outro lado, o repositório de componentes de processo de projeto abriga apenas processos de projeto como um todo, uma vez que os processos de projeto são sempre definidos no contexto de um projeto, não sendo definidos visando à reutilização, embora seja possível recuperar componentes de nível de granularidade mais fino. Ao contrário de BARRETO et al. (2008), os repositórios definidos na DBPC não contemplam informações sobre a estabilidade, capacidade e desempenho histórico dos componentes de processo.

BARRETO et al. (2008) fazem algumas considerações acerca da definição com reúso e para reúso, ponderando, por exemplo, que durante a definição com reúso é importante que alguns aspectos do componente a ser selecionado devem ser considerados, além de apontarem algumas formas de definição para reúso. Entretanto, nenhum dos trabalhos estudados apresenta um processo para definição de processos com e para reúso, conforme definido pela DPBC. A DPBC traz dois processos, sendo um deles para definição de componentes no nível de processos padrão e outro para definição no nível de processos de projeto. Ainda que os componentes no nível de abstração de processo padrão possam ser definidos pela composição de outros componentes do mesmo nível, constituindo uma forma de definição com reúso, a definição desses componentes pode ser encarada como definição para reúso, uma vez que seu propósito é a reutilização. Por outro lado, a definição de processos no nível de abstração de processos de projeto é encarada como definição com componentes. Embora processos de projeto possam ser reutilizados para definição de processos de outros projetos, eles são sempre definidos no contexto específico de um projeto, não caracterizando uma definição para reúso.

Por fim, outro aspecto tratado pela DPBC considerado bastante importante, mas que não é tratado nos trabalhos estudados, é definição de componentes de processo em diferentes níveis de abstração. Isso torna a abordagem mais flexível e em linha com a abordagem de definição de processos em níveis (ROCHA et al., 2001), uma abordagem bem difundida e bastante aceita por definir processos de software em diferentes níveis de abstração, buscando facilitar a tarefa de definição de processos em uma organização.

3.5. Considerações Finais do Capítulo

Neste capítulo foram apresentados os conceitos envolvidos na abordagem de definição de processos baseada em componentes, a saber: componentes de processo – considerações sobre quais elementos da estrutura de um processo são vistos como componentes de processo; níveis de abstração de processos e componentes de processos – importante estratégia para alinhamento da DPBC com a definição de processos em níveis; interface de componentes de processo; e repositório de componentes de processo. Por fim, foi apresentado um processo de DPBC, cobrindo os níveis de abstração de processos padrão e de projeto.

A partir dessas definições, adaptou-se a ferramenta de definição de processos de ODE para que ela contemplasse a DPBC, o que é apresentado a seguir, no Capítulo 4 desta dissertação.

Capítulo 4

Apoio Automatizado à Definição de Processos Baseada em Componentes

O objetivo deste capítulo é apresentar a evolução da ferramenta de definição de processos de ODE (*Ontology-based software Development Environment*) para prover apoio automatizado à abordagem de Definição de Processos Baseada em Componentes.

4.1. Introdução

A definição de processos de software é uma tarefa não trivial, o que leva à necessidade da existência de uma ferramenta que ofereça apoio automatizado a essa atividade. Assim, uma vez definida a abordagem de Definição de Processos Baseada em Componentes (DPBC), surge a necessidade de se desenvolver uma ferramenta que apoie essa abordagem.

O ambiente ODE (*Ontology-based software Development Environment*) (FALBO et al., 2003), um ambiente de desenvolvimento de software centrado em processos e baseado em ontologias, desenvolvido no Núcleo de Estudos em Modelagem Conceitual e Ontologias (NEMO) da Universidade Federal do Espírito Santo (UFES), já possui uma ferramenta de definição de processos. Entretanto, essa ferramenta não está aderente à abordagem de DPBC proposta. Assim, torna-se essencial a evolução dessa ferramenta. À nova versão dessa ferramenta foi dado o nome ProcODE-Com.

Este capítulo apresenta a evolução da ferramenta de definição de processos de ODE (ProcODE-Com) e está estruturado da seguinte forma: a Seção 4.2 apresenta a antiga versão da ferramenta, descrevendo as abordagens por ela apoiadas; na Seção 4.3 é apresentada a evolução da ferramenta de definição de processos de ODE, denominada ProcODE-Com; a Seção 4.4 ilustra o uso de ProcODE-Com apoiando a definição de processos em duas situações: a definição de componentes de processos (definição *para*

reúso) e a definição de processos de projeto utilizando componentes (definição *com* reúso); e, por fim, a Seção 4.5 apresenta as considerações finais do capítulo.

4.2. A Versão Anterior da Ferramenta de Definição de Processos de ODE

A ferramenta de definição de processos de ODE vem evoluindo ao longo do tempo. Em seu primeiro estágio, a ferramenta permitia apenas a definição do processo de desenvolvimento de um projeto específico, tomando por base um conhecimento previamente cadastrado no ambiente. Esse conhecimento era organizado segundo a versão da ontologia de processos de software definida em (FALBO, 1998). Em um segundo estágio (BERTOLLO e FALBO, 2003), essa abordagem foi estendida para permitir a definição de processos padrão e especializados e a instanciação destes para um projeto específico. Porém, apenas o processo de desenvolvimento continuava sendo contemplado.

BERTOLLO (2006) trabalhou em uma nova versão da ontologia de processos na qual ODE se baseia. Essa nova versão da ontologia fez com que a ferramenta fosse evoluída novamente (SEGRINI et al., 2006), sendo as características mais marcantes dessa versão a interação entre processos e a composição de processos. Essa versão da ferramenta já oferecia apoio à definição de processos em níveis e, com isso, nada mais natural que começar sua apresentação pela definição dos processos padrão (especializados ou não). A Figura 4.1 mostra o diagrama de casos de uso para o nível de processos padrão.

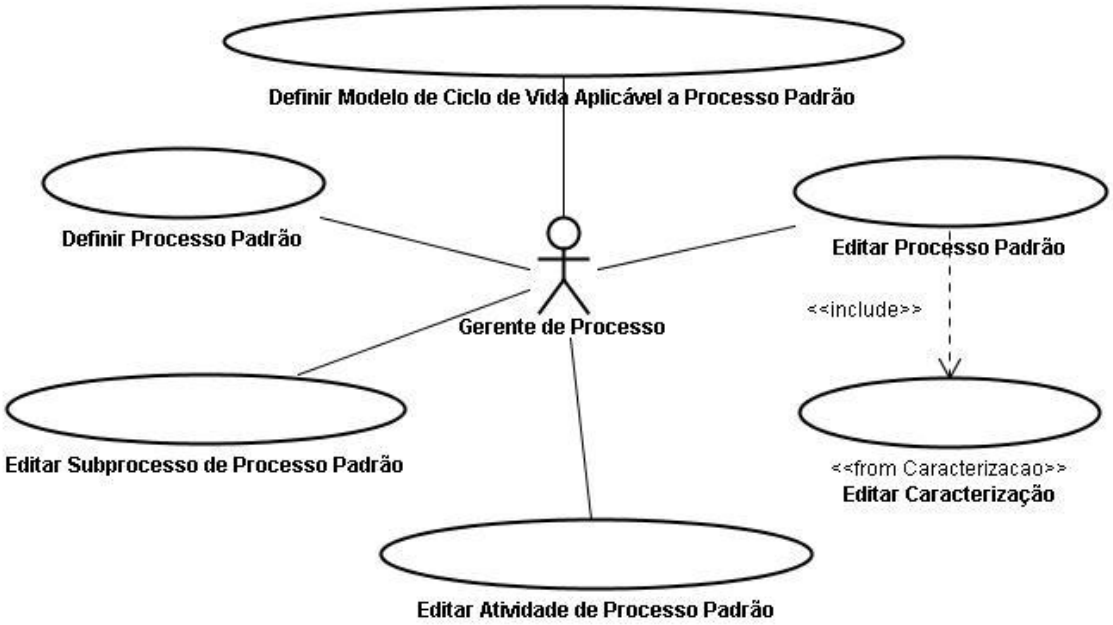

Figura 4.1 – Diagrama de Casos de Uso Nível Processo Padrão da Versão Anterior da Ferramenta.

A definição de um processo padrão para a organização começava pela sua criação no repositório da organização. Em seguida, definiam-se os tipos de processo que iriam compor esse processo padrão (desenvolvimento/manutenção, garantia da qualidade, gerência de projetos etc). Dentre esses subprocessos, apenas um deles poderia ser um processo de engenharia, neste contexto, uma denominação mais genérica para processos de desenvolvimento ou de manutenção. Além disso, o tipo de interação entre os subprocessos devia ser definido.

Com os subprocessos selecionados, era possível começar a definição de cada um deles. Neste momento, permitia-se reutilizar subprocessos padrão previamente definidos. Ou seja, dado um tipo de subprocesso previamente selecionado, era possível selecionar um subprocesso padrão do mesmo tipo para compor o processo em definição. Seja o caso em que se esteja definindo um novo processo padrão composto por subprocessos de manutenção, gerência de configuração, garantia da qualidade, gerência de requisitos e gerência de projetos. Neste momento, era possível selecionar, por exemplo, um subprocesso padrão de gerência de configuração para ser reutilizado no processo em definição. Ressalta-se que, nessa versão da ferramenta, não havia uma forma definida para buscar os subprocessos a serem reutilizados. Assim, a reutilização acontecia da maneira pontual, sempre considerando um subprocesso definido especificamente para outro processo padrão.

Caso não se desejasse reutilizar subprocessos, para cada subprocesso, deveriam ser indicadas as atividades que o iriam compor. No detalhamento de uma atividade, eram informados: pré-atividades, subatividades, artefatos requeridos (insumos) e produzidos (produtos), recursos necessários (recursos humanos, de hardware e de software) e procedimentos a serem adotados (roteiros, métodos, técnicas etc).

Ao término da criação do processo padrão, definiam-se os modelos de ciclo de vida que ele comportaria. Para tal, criavam-se combinações (iterativas ou sequenciais) de macroatividades do processo de engenharia.

A especialização de um processo padrão iniciava-se com a escolha do processo padrão a ser especializado. Quando isso era feito, criava-se automaticamente um novo processo com os mesmos elementos do processo padrão escolhido.

A definição de seus subprocessos e atividades dava-se de maneira análoga à apresentada anteriormente. Um ponto importante a frisar é que não era permitido excluir atividades definidas como obrigatórias no processo padrão. Também não era possível excluir subprocessos definidos no processo padrão, mesmo que estes não possuíssem atividades obrigatórias. Estas restrições deviam-se ao fato de um processo especializado ser também um processo padrão, mas que refinava uma abordagem estabelecida em outro processo padrão, sendo necessário, portanto, que o primeiro estivesse em conformidade com o último.

Como, durante a especialização de processos, geralmente, o subprocesso de engenharia é alterado, não era possível que o processo especializado utilizasse diretamente os modelos de ciclo de vida definidos para o processo padrão que o originou. Assim, os modelos de ciclo de vida tinham de ser definidos novamente, tomando por base as macroatividades do novo subprocesso de engenharia.

Um processo especializado podia ser caracterizado, considerando-se as características segundo as quais ele tinha sido especializado. Apesar de ser possível realizar a caracterização dos processos padrão especializados, nenhuma funcionalidade da ferramenta utilizava essa caracterização.

Em seguida, segundo a abordagem de definição de processos em níveis, era possível realizar a definição de processos de projeto. A Figura 4.2 mostra o diagrama de casos de uso para o nível de processos de projeto.

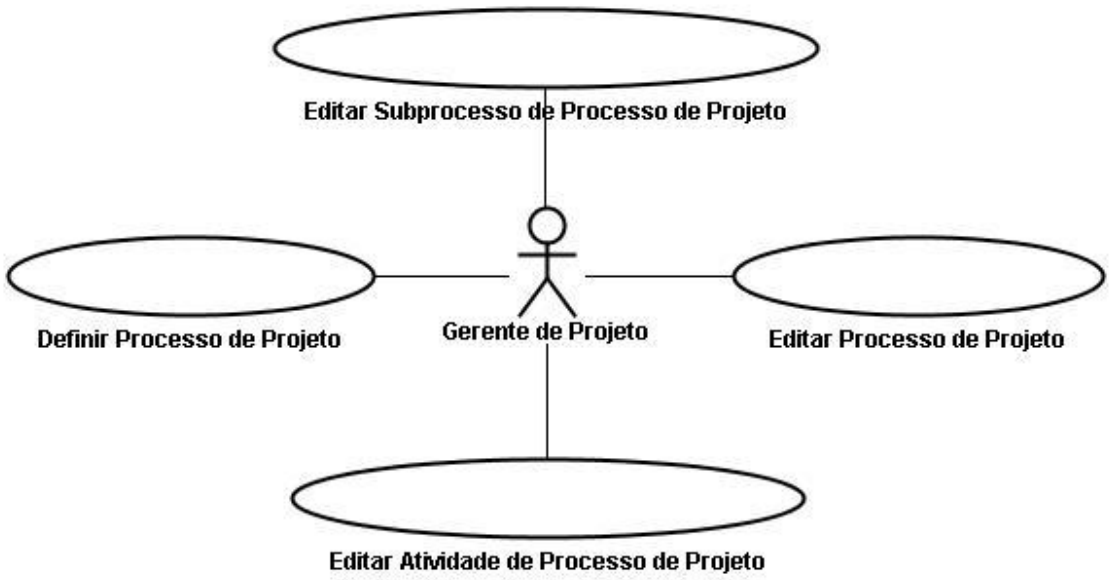

Figura 4.2 – Diagrama de Casos de Uso Nível Processo de Projeto da Versão Anterior da Ferramenta.

A definição de um processo para um projeto específico era feita a partir de um processo padrão ou especializado. Uma vez escolhido o processo padrão (especializado ou não) que serviria de base para o processo do projeto, era necessário escolher o modelo de ciclo de vida a ser adotado, informando, para cada combinação iterativa, o número de iterações a serem realizadas. Neste ponto, instanciava-se um processo com todos os subprocessos do processo de origem e suas atividades. No subprocesso de engenharia, as atividades eram instanciadas considerando-se a quantidade de iterações de cada combinação. Ressalta-se que essa versão da ferramenta não oferecia apoio à escolha do processo padrão (especializado ou não) mais adequado para servir de base para o processo de projeto em definição. A escolha do processo padrão base era uma decisão do gerente de projeto, não apoiada pela ferramenta.

As formas de se definir os subprocessos e as atividades do processo instanciado eram análogas às de se definir processos padrão. Contudo, era possível excluir subprocessos e atividades definidos no processo padrão de origem, mesmo que as atividades tivessem sido definidas como obrigatórias. Porém, essas decisões eram registradas, para que fosse possível saber em que aspectos o processo de projeto não estava em conformidade com o processo padrão usado como base e as justificativas para tal.

4.3. Evolução da Ferramenta de Definição de Processos de ODE

Na versão anterior da ferramenta de definição de processos de ODE, um processo de software era, obrigatoriamente, definido integralmente, não sendo permitido definir partes de processo para uso posterior, nem tão pouco definir novos processos a partir de componentes já existentes. A evolução da ferramenta, denominada ProcODE-Com, visa apoiar a Definição de Processos Baseada em Componentes (DPBC).

Agora os processos de software são definidos por meio da composição de componentes de processo previamente definidos, sendo que um componente de processo tem uma interface, explicitando a sua estrutura básica (seus subelementos) e o seu contexto de aplicação (suas características).

Assim como advoga a DPBC, em ProcODE-Com, os componentes de processo são considerados em três níveis de granularidade, a saber: processo complexo, processo simples e macroatividade. Ainda, mantendo compatibilidade com as versões anteriores da ferramenta, em que os processos eram definidos em dois níveis de abstração, processo padrão e de projeto, os componentes de processo também são definidos nesses mesmos níveis de abstração.

No nível de abstração de processo padrão, podem ser definidos componentes de processo nos três níveis de granularidade, ditos CompPPs. Esses componentes podem ser (re)utilizados na definição de outros componentes de processo padrão. Por exemplo, processos padrão simples (subprocessos) de gerência de configuração e garantia da qualidade podem ser definidos isoladamente e depois, durante a definição de um processo padrão complexo, esses processos simples podem ser reutilizados e acrescentados à sua composição. A definição de um componente de processo padrão envolve também a definição de sua interface.

Vale destacar que nesta nova versão da ferramenta não é mais obrigatório que um processo padrão possua um subprocesso (processo simples) de engenharia. Agora, um processo padrão pode possuir no máximo um subprocesso de engenharia, podendo não possuir nenhum. Nos casos em que o processo padrão possui um subprocesso de engenharia, é possível definir interações entre o subprocesso de engenharia e os demais subprocessos. Assim, para cada interação que se deseja definir, o engenheiro de processo informa o tipo de interação desejada.

Para cada atividade (incluindo os componentes de macroatividade), assim como na versão anterior da ferramenta, podem-se definir subatividades, pré-atividades,

artefatos produzidos (produtos) e requeridos (insumos), recursos necessários (recursos humanos, de hardware e de software) e procedimentos (roteiros, métodos, técnicas etc) a serem adotados. Por fim, definem-se os modelos de ciclo de vida que podem ser utilizados para o processo padrão. Com relação à definição dos modelos de ciclo de vida (MCV), a única diferença é que em ProcODE-Com, um MCV é definido para um processo simples (subprocesso) de engenharia, e não mais para um processo padrão (processo complexo).

A definição de um componente deve, preferencialmente, ocorrer a partir da adaptação de outro já existente. Contudo, quando não é encontrado nenhum componente base para adaptação com as características desejadas, o novo componente é definido a partir do zero. Quando um componente base é selecionado para adaptação, todos os subprocessos, atividades, interação entre as atividades, artefatos, recursos e procedimentos definidos para o processo padrão são automaticamente definidos para o novo componente.

Já no nível de abstração de processo de projeto, a definição sempre ocorre para um processo completo (processo complexo), não sendo possível definir componentes de menor granularidade isoladamente. Entretanto, a definição do processo de projeto também se dá pela composição de componentes de menor granularidade (processos simples e macroatividades), podendo ser selecionados componentes de ambos os níveis de abstração de processo (padrão ou de projeto). Nesse nível de abstração não há definição explícita da interface do componente. A interface de um componente de processo de projeto é extraída a partir das características do projeto corrente (contexto de aplicação) e dos seus subelementos definidos (estrutura).

Também no nível de abstração de processo de projeto, não é mais exigido que o processo complexo (processo de projeto) possua um processo simples (subprocesso) de engenharia. Assim, é possível que ODE seja utilizado para apoiar apenas um determinado processo da organização, tal como um processo de Gerência de Projetos.

4.3.1. Modelo de Casos de Uso

A Figura 4.3 apresenta o diagrama de casos de uso da ferramenta ProcODE-Com referente ao nível de processo padrão. Nesse diagrama está presente o ator Engenheiro de Processo, que representa o papel desempenhado pelos responsáveis pela área de processos de uma organização.

Os casos de uso *Editar Caracterização* e *Buscar Itens de Software Similares* são funcionalidades da infraestrutura de caracterização de itens de software (CARVALHO, 2006). O primeiro permite caracterizar um determinado item de software a partir de características definidas para o tipo do item em questão. Especificamente no contexto de ProcODE-Com, ele é utilizado para a definição do contexto de aplicação dos componentes de processo. Já o segundo é utilizado para encontrar itens de software similares a um determinado item informado. No contexto de ProcODE-Com, ele é utilizado na busca de componentes base. A seguir, os casos de uso específicos da ferramenta ProcODE-Com do nível de processo padrão são brevemente descritos.

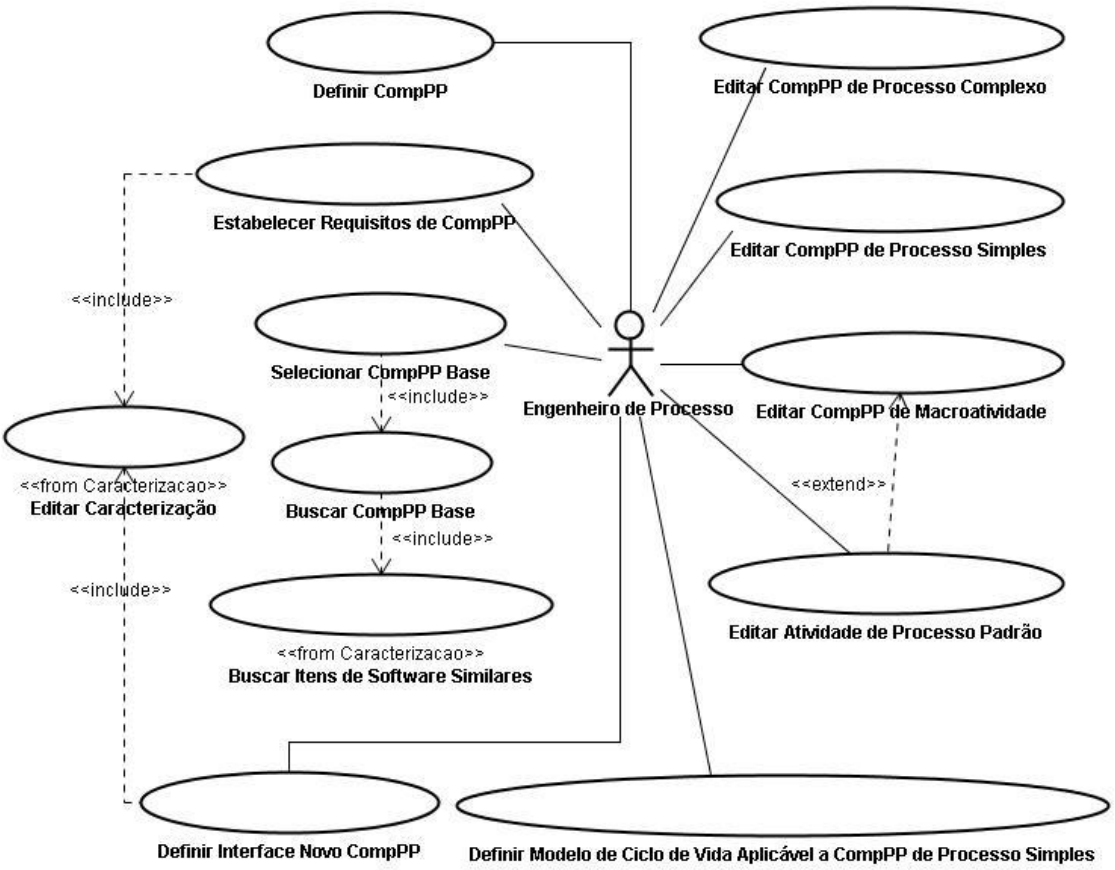

Figura 4.3 – Diagrama de Casos de ProcODE-Com: Nível Processo Padrão.

- **Definir CompPP:** Este caso de uso é responsável pela criação dos CompPPs, independentemente do nível de granularidade (processo complexo, processo simples ou macroatividade).
- **Estabelecer Requisitos de CompPP:** Este caso de uso é responsável por permitir o registro dos requisitos do CompPP em definição.

- **Selecionar CompPP Base:** Este caso de uso é responsável pela seleção do componente base que será adaptado durante a definição do novo CompPP. Não é obrigatória a seleção de um componente base. A execução deste caso de uso sempre ocorre no contexto de um nível de granularidade de componente (processo complexo, processo simples ou macroatividade).
- **Buscar CompPP Base:** Este caso de uso é responsável pela busca de um CompPP a servir como base para a definição de um novo CompPP ou de uma parte de um processo de projeto.
- **Definir Interface Novo CompPP:** Este caso de uso é responsável pela definição da interface do CompPP. Assim, são definidos tanto a sua estrutura quanto o seu contexto de aplicação.
- **Editar CompPP de Processo Complexo:** Este caso de uso é responsável pela edição de um CompPP de processo complexo em definição. É por meio desse caso de uso que se indicam os processos simples opcionais a serem reutilizados, definem-se seus CompPPs de processo simples, excluem-se CompPPs de processo simples da composição do CompPP de processo complexo e definem-se as interações entre os processos simples.
- **Editar CompPP de Processo Simples:** Este caso de uso é responsável pela edição de um CompPP de processo simples em definição. É por meio dele que se indicam as macroatividades opcionais a serem reutilizadas, definem-se os CompPPs de macroatividade que comporão o CompPP de processo simples, definem-se as dependências entre macroatividades no contexto do processo simples e excluem-se os CompPPs de macroatividade da composição de um CompPP de processo simples.
- **Editar CompPP de Macroatividade:** Este caso de uso é responsável pela edição de um CompPP de macroatividade, quando se definem as subatividades de um CompPP de macroatividade.
- **Editar Atividade de Processo Padrão:** Este caso de uso é responsável pela edição de uma atividade de processo padrão. É por meio dele que são editados os artefatos (produzidos e consumidos), os procedimentos e os recursos de uma atividade. Ainda, para as atividades que não definem um CompPP de macroatividade, esse caso de uso também é responsável pela definição de suas pré-atividades e subatividades.

- **Definir Modelo de Ciclo de Vida Aplicável a CompPP de Processo Simples:** Este caso de uso é responsável pelas definições dos modelos de ciclo de vida aplicáveis a um CompPP de processo simples de engenharia.

A Figura 4.4 apresenta o diagrama de casos de uso da ferramenta ProcODE-Com referente ao nível de processo de projeto. Nesse diagrama está presente o ator Gerente de Projeto, que representa o papel desempenhado pelos responsáveis pelos projetos de uma organização. Novamente, aparece o caso de uso *Buscar Itens de Software Similares*, funcionalidade da infraestrutura caracterização de itens de software, utilizado na busca de componentes base. A seguir, os casos de uso específicos da ferramenta ProcODE-Com do nível de processo de projeto são brevemente descritos.

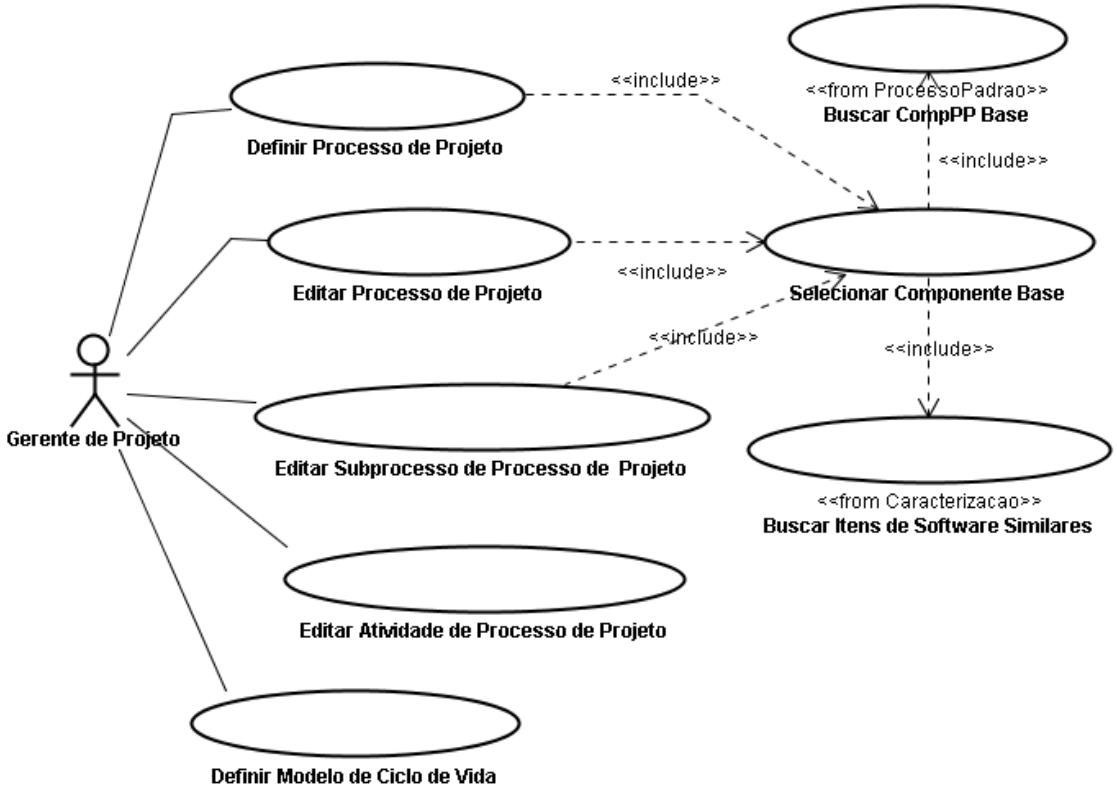

Figura 4.4 – Diagrama de Casos de ProcODE-Com: Nível Processo de Projeto.

- **Definir Processo de Projeto:** Este caso de uso é responsável por iniciar a definição de um processo de projeto. Deve-se, obrigatoriamente, selecionar um processo complexo base, que pode ser tanto um CompPP de processo complexo (processo padrão) ou um processo complexo de projeto (processo de um projeto anterior).

- **Editar Processo de Projeto:** Este caso de uso é responsável pela edição de um processo (complexo) de projeto. Por meio dele indicam-se os subprocessos do processo de projeto a serem considerados (quando o processo de projeto está sendo definido a partir de outro processo de projeto) ou os subprocessos opcionais do processo padrão a serem considerados (quando o processo de projeto está sendo definido a partir de um processo padrão) e indicam-se quais outros subprocessos devem ser considerados.
- **Editar Subprocesso de Processo de Projeto:** Este caso de uso é responsável pela edição de um subprocesso (processo simples) de um processo de projeto. É por meio dele que se indicam as macroatividades do processo simples de projeto a serem consideradas (quando o processo simples está sendo definido a partir de outro processo simples de projeto) ou as macroatividades opcionais do processo padrão simples a serem consideradas (quando o processo simples está sendo definido a partir de um CompPP de processo simples) e indicam-se quais outras macroatividades devem ser consideradas.
- **Selecionar Componente Base:** Este caso de uso é responsável pela seleção do componente base que será adaptado para a definição de um novo elemento do processo de projeto, não sendo obrigatória a seleção de um componente base. A execução deste caso de uso sempre ocorre no contexto de um nível de granularidade (processo complexo, processo simples ou macroatividade). Desse modo, durante a definição de uma macroatividade, por exemplo, este caso de uso pode ser executado para a seleção de macroatividades.
- **Editar Atividade de Processo de Projeto:** Este caso de uso é responsável pela edição das atividades de um subprocesso de processo de projeto. É por meio desse caso de uso que se definem os artefatos (produzidos e consumidos), os recursos necessários, os procedimentos adotados, as pré-atividades e as subatividades de uma atividade do processo de projeto.
- **Definir Modelo de Ciclo de Vida:** Este caso de uso é responsável pela definição do modelo de ciclo de vida a ser adotado no processo de projeto em definição.

4.3.2. Modelo Conceitual

Uma vez definidos os casos de uso da ferramenta, deve-se elaborar o modelo conceitual para tratá-los. A Figura 4.5 apresenta o diagrama de pacotes envolvendo os pacotes que tratam das funcionalidades relacionadas à ferramenta ProcODE-Com.

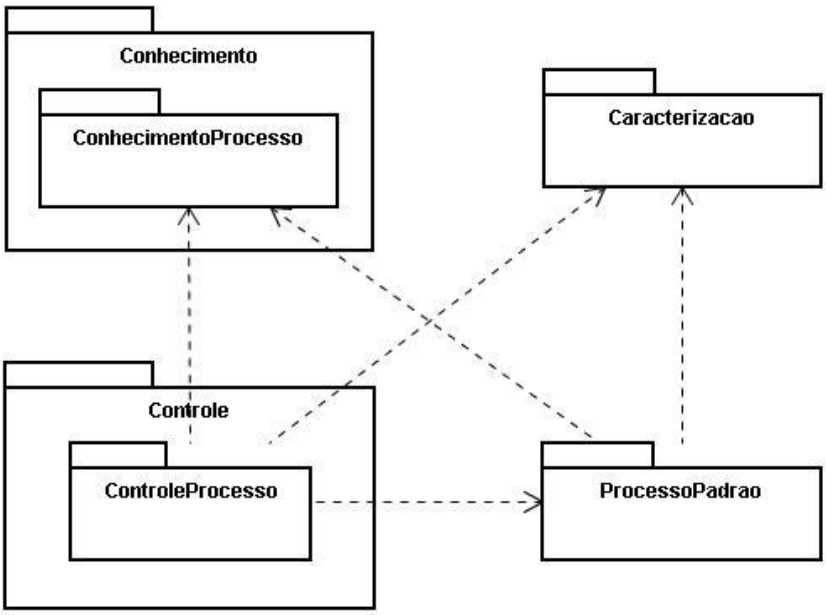

Figura 4.5 – Diagrama de Pacotes de ProcODE-Com.

O pacote *Controle* define as classes responsáveis por implementar as aplicações no contexto do ambiente, representando o núcleo do ambiente, e são de sua responsabilidade as principais funcionalidades referentes ao gerenciamento do sistema. Dentro desse pacote, existe o subpacote *ControleProcesso*, no qual se encontram as classes utilizadas por ProcODE-Com para a definição de processos de projeto.

O pacote *ProcessoPadrao* trata da definição de processos padrão e especializados de uma organização. Na definição de processos em ambos os níveis de abstração (padrão e de projeto), funcionalidades da infraestrutura de caracterização de itens de software de ODE (pacote *Caracterizacao*) (CARVALHO, 2006) são utilizadas para a descrição da interface dos componentes de processo, bem como para realizar a busca desses componentes.

Por fim, o pacote *Conhecimento* abriga as classes que descrevem conhecimento sobre um domínio. Tais classes representam os conceitos das ontologias utilizadas em ODE e são utilizadas para falar a respeito do universo de discurso de outros pacotes, dentre eles o pacote *Controle*. Dentro deste pacote, existe o subpacote *ConhecimentoProcesso*, no qual se encontram as classes que representam os conceitos

da ontologia de processos de software, que é a principal ontologia de ODE. Os pacotes *ControleProcesso* e *ProcessoPadrao* estão fortemente acoplados ao pacote *ConhecimentoProcesso*.

4.3.2.1. Pacote *ConhecimentoProcesso*

A Figura 4.6 apresenta as classes do pacote *ConhecimentoProcesso* que são utilizadas pela ferramenta juntamente com seus relacionamentos.

No pacote *ConhecimentoProcesso* todas as classes herdam da classe *Conhecimento*, que tem como atributos um nome e uma descrição. Por convenção, os nomes das classes do pacote *Conhecimento* começam com o prefixo *K*.

Tipos de processos (*KProcesso*) são classificados em categorias de processo (*KCategoriaProcesso*) e interagem entre si, sendo que essa interação pode ser de vários tipos (*KTipoInteracao*). Além disso, tipos de processos podem ser de engenharia (tipicamente os processos de desenvolvimento e manutenção) ou não (todos os demais processos) e são compostos por tipos de atividades (*KAtividade*).

Com relação à interação entre os tipos de processos, somente podem ser definidos tipos de interação (*KTipoInteracao*) para processos que não sejam de engenharia. Esses tipos indicam as possíveis formas de interação entre esses tipos de processo com processos de engenharia.

Já os modelos de ciclo de vida só podem ser definidos para processos que sejam de engenharia. Eles são compostos por combinações (*KCombinacao*), que podem ser iterativas ou sequenciais (atributo tipo em *KCombinacao*), e estas devem ter uma ordem estabelecida (*KOrdemCombinacao*).

Uma atividade pode ser decomposta em subatividades. Para cada tipo de atividade (*KAtividade*), é possível definir procedimentos a serem utilizados (*KProcedimento*), tipos de recursos necessários (*KRecurso*) e os tipos de artefatos a serem produzidos e consumidos (*KArtefato*). Ainda é possível definir uma ordem de precedência entre as atividades.

Os tipos de recursos necessários para uma atividade podem ser humanos (*KRecursoHumano*), de hardware (*KRecursoHardware*) e de software (*KRecursoSoftware*), sendo que os tipos de recursos de software podem ser sistemas de apoio (*KSistemaApoio*), ferramentas de software (*KFerramentaSoftware*) ou linguagens de programação (*KLinguagemProgramacao*).

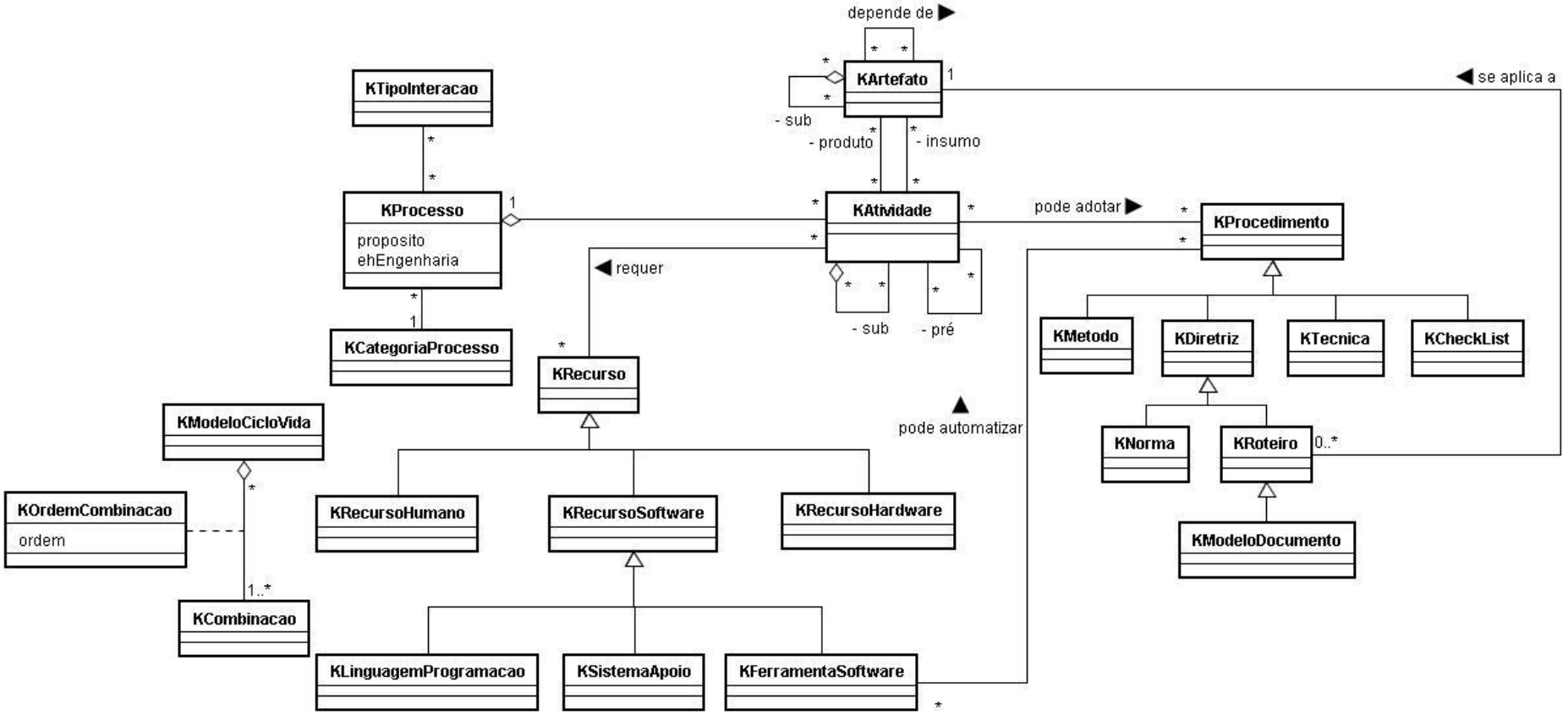

Figura 4.6 – Diagrama de Classes parcial do Pacote *ConhecimentoProcesso*.

4.3.2.2. Pacote *ProcessoPadrao*

A Figura 4.7 apresenta o diagrama de classes do pacote *ProcessoPadrao*. Conforme dito anteriormente, esse pacote abriga as classes envolvidas na definição dos processos padrão organizacionais.

No nível de abstração de processo padrão, os componentes de processos são denominados CompPPs (*CompPP*). Um processo padrão definido para uma organização é dito um processo padrão complexo (*CompPPProcessoComplexo*) e é composto por subprocessos, ditos processos padrão simples (*CompPPProcessoSimples*). Estes, por sua vez, têm um tipo (*KProcesso*) e são compostos por macroatividades (*CompPPMacroatividade*). Um CompPP de macroatividade sempre é composto de uma atividade (*AtividadeProcessoPadrao*), sendo que essa atividade pode ser decomposta em subatividades.

Os CompPPs possuem uma interface (*InterfaceCompPP*), que, de maneira geral, é constituída pelas características do CompPP (*Caracterizacao*) e seus subelementos, com uma indicação se os mesmos são obrigatórios ou não. As interfaces são especializadas para cada tipo de CompPP. Uma interface de um componente de processo complexo (*InterfaceCompPPProcessoComplexo*) aponta os tipos dos processos simples (*KProcesso*) que compõem esse componente de processo complexo. Já as interfaces de componentes de processo simples e de macroatividade (*InterfaceCompPPProcessoSimples* e *InterfaceCompPPMacroatividade*) indicam os tipos das atividades (*KAtividade*) (macroatividades no caso dos componentes de processo simples e subatividades no caso dos componentes de macroatividade) que compõem os componentes aos quais estão associadas.

Dependendo do tipo de um processo padrão simples, ele pode ser de engenharia (tipicamente os processos de desenvolvimento e manutenção) ou não (todos os demais processos). Em um processo padrão complexo pode haver no máximo um único processo padrão simples que seja de engenharia. Para um processo padrão simples de engenharia, podem-se definir modelos de ciclo de vida (*ModeloCicloVidaProcessoPadrao*) que podem ser aplicados quando de sua instanciação. Um modelo de ciclo de vida definido para um processo padrão tem um tipo (*KModeloCicloVida*) e é composto por combinações de atividades (*CombinacaoPP*), que também possuem um tipo (*KCombinacao*). As atividades dentro de uma combinação devem obedecer a uma ordem de precedência (*OrdemAtividade*).

A interação entre processos padrão simples (*InteracaoSubprocessos*) sempre é definida no contexto de um processo padrão complexo. Devem-se indicar os processos padrão simples origem e destino, sendo que um deles deve, obrigatoriamente, ser o processo padrão simples de engenharia que compõe o processo padrão geral. Para dois processos padrão simples, somente podem ser definidas interações cujo tipo (*KTipoInteracao*) esteja definido para o tipo (*KProcesso*) do processo padrão simples que não seja de engenharia.

Para cada atividade do processo padrão devem ser definidos um tipo (*KAtividade*) e seus ativos, que são os tipos de artefatos produzidos e consumidos (*KArtefato*), os procedimento a serem adotados (KProcedimento), os tipos de recursos requeridos (*KRecurso*) e a ordem de precedência entre as atividades. Destaca-se que para os CompPPs de macroatividades, as relações de precedência são definidas sempre no contexto de um CompPP de processo simples. Assim, no contexto de um determinado CompPP de processo simples, um CompPP de macroatividade qualquer pode ser precedido ou preceder outros CompPPs de macroatividade que também compõem o CompPP de processo simples. Essas relações de precedência entre CompPPs de macroatividade no contexto de um CompPP de processo simples são definidas e registradas por meio de conectores de CompPP de macroatividade (*ConectorCompPPMacroAtividade*).

Ainda é possível definir um CompPP com base em outro já existente de mesma granularidade. Nesses casos, seleciona-se o CompPP base e, em seguida, faz-se as alterações desejadas em um cópia desse CompPP base, originando o novo CompPP.

Por fim, só é possível reutilizar (verticalmente e horizontalmente) um CompPP, caso ele tenha tido sua definição finalizada.

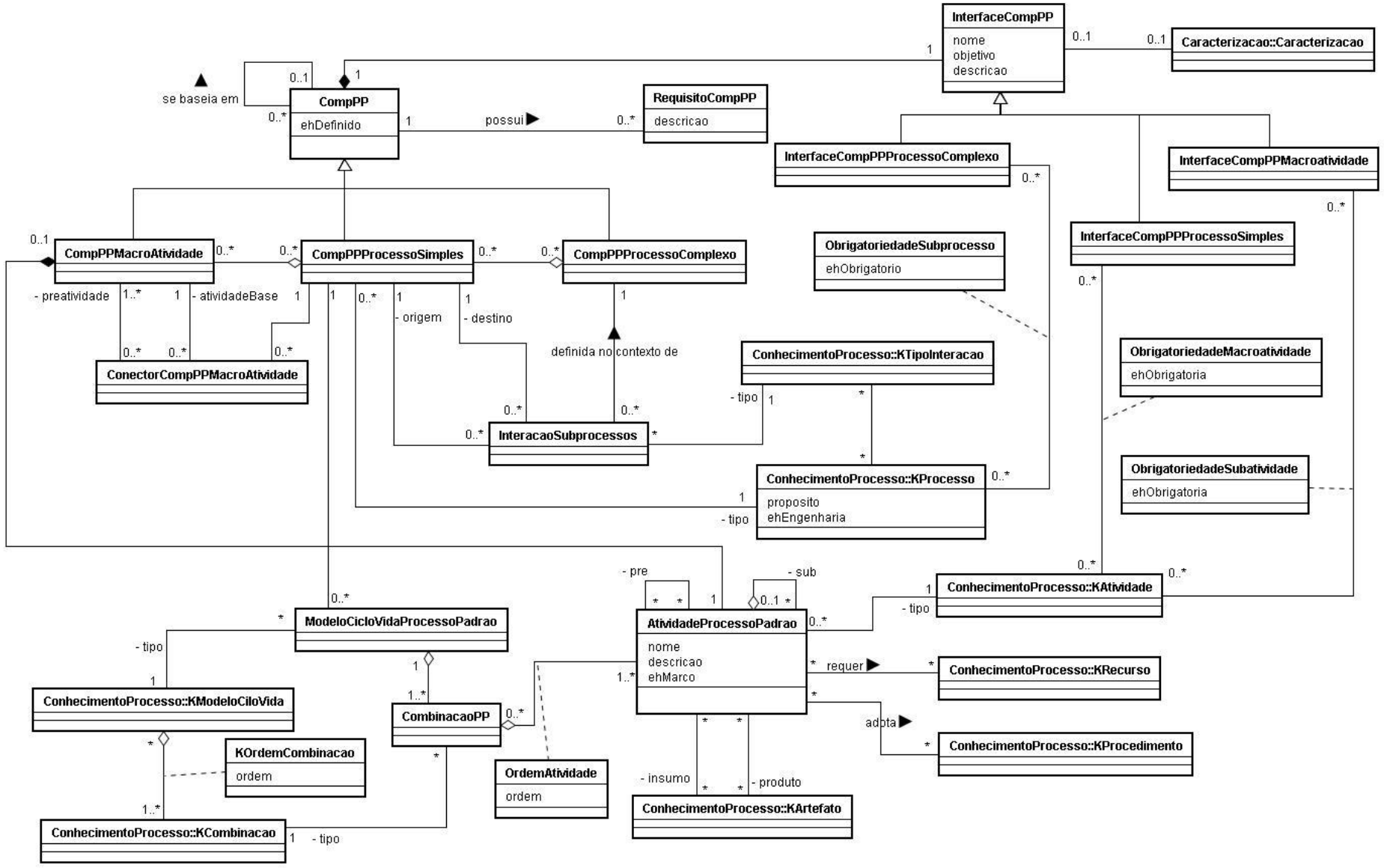

Figura 4.7 – Diagrama de Classes do pacote *ProcessoPadrao*.

4.3.2.3. Pacote *ControleProcesso*

A Figura 4.8 apresenta um diagrama de classes parcial do pacote *ControleProcesso*, envolvendo as classes relevantes para a definição de processos de projeto em ProcODE-Com.

Um processo de projeto é sempre definido para um projeto, a partir de um processo padrão ou de um processo de projeto anterior. Há um processo de projeto geral (*ProcessoProjetoGeral*) para o projeto, sendo este composto por processos de projeto específicos (*ProcessoProjetoEspecifico*).

Os processos de projeto específicos possuem um tipo (*KProcesso*) e são compostos por atividades, sendo que uma atividade pode ser decomposta em outras atividades (subatividades). Um processo específico pode ser definido de três maneiras: (i) com base em um CompPP de processo simples; (ii) com base em um processo específico de um projeto anterior; e (iii) a partir do zero. Ao se definir um processo específico com base em um CompPP de processo simples, deve-se obedecer a obrigatoriedade de suas macroatividade, devendo, necessariamente, reutilizar aquelas macroatividade cujos tipos (*KAtividade*) constem como obrigatórios na interface do CompPP de processo simples (*InterfaceCompPPProcessoSimples*).

Uma atividade tem um tipo (*KAtividade*) e pode ou não ser originada de uma atividade do processo padrão (*AtividadeProcessoPadrao*) ou de outra atividade de processo de projeto anterior. De maneira similar à dos processos específicos, ao se definir uma macroatividade com base em um CompPP de macroatividade, deve-se obedecer a obrigatoriedade de suas subatividades, devendo, necessariamente, reutilizar aquelas subatividades cujos tipos (*KAtividade*) constem como obrigatórios na interface do CompPP de macroatividade (*InterfaceCompPPMacroatividade*).

O ciclo de vida do processo de projeto de engenharia (*CicloVida*) é definido após a definição do processo de projeto. Neste momento, escolhe-se o modelo de ciclo de vida a ser adotado (*ModeloCicloVidaProcessoPadrao*) e define-se o número de iterações para cada combinação (*CombinacaoPP*) iterativa que o compõe. Um ciclo de vida é composto por iterações (*Iteracao*) e estas são compostas por combinações de atividades (*CombinacaoAtividade*). Uma combinação de atividades é a representação de uma combinação de atividades do processo padrão.

Para uma atividade, é possível definir seus ativos. Definem-se os procedimentos adotados na atividade (*KProcedimento*), os tipos dos recursos necessários para a sua

realização (*KRecurso*), os tipos de artefatos que são produtos da e insumos para a atividade (*KArtefato*) e a ordem de precedência entre as atividades.

Para apoiar o processo de documentação, é possível realizar um planejamento da documentação durante a definição do processo de projeto. Esse planejamento refere-se à definição do contexto no qual um artefato será produzido por uma atividade (*ContextoDefinicao*). Essa contextualização consiste em definir o tipo de ferramenta a ser utilizada na produção do artefato (*KFerramentaSoftware*), se ele será produzido por uma ferramenta externa a ODE ou não (artefato externo ou não), se o artefato deverá ser colocado sob gerência de configuração ou não (é gerenciado ou não), de quais outros tipos de artefato ele depende para ser produzido e, se existir, o roteiro segundo o qual ele será produzido.

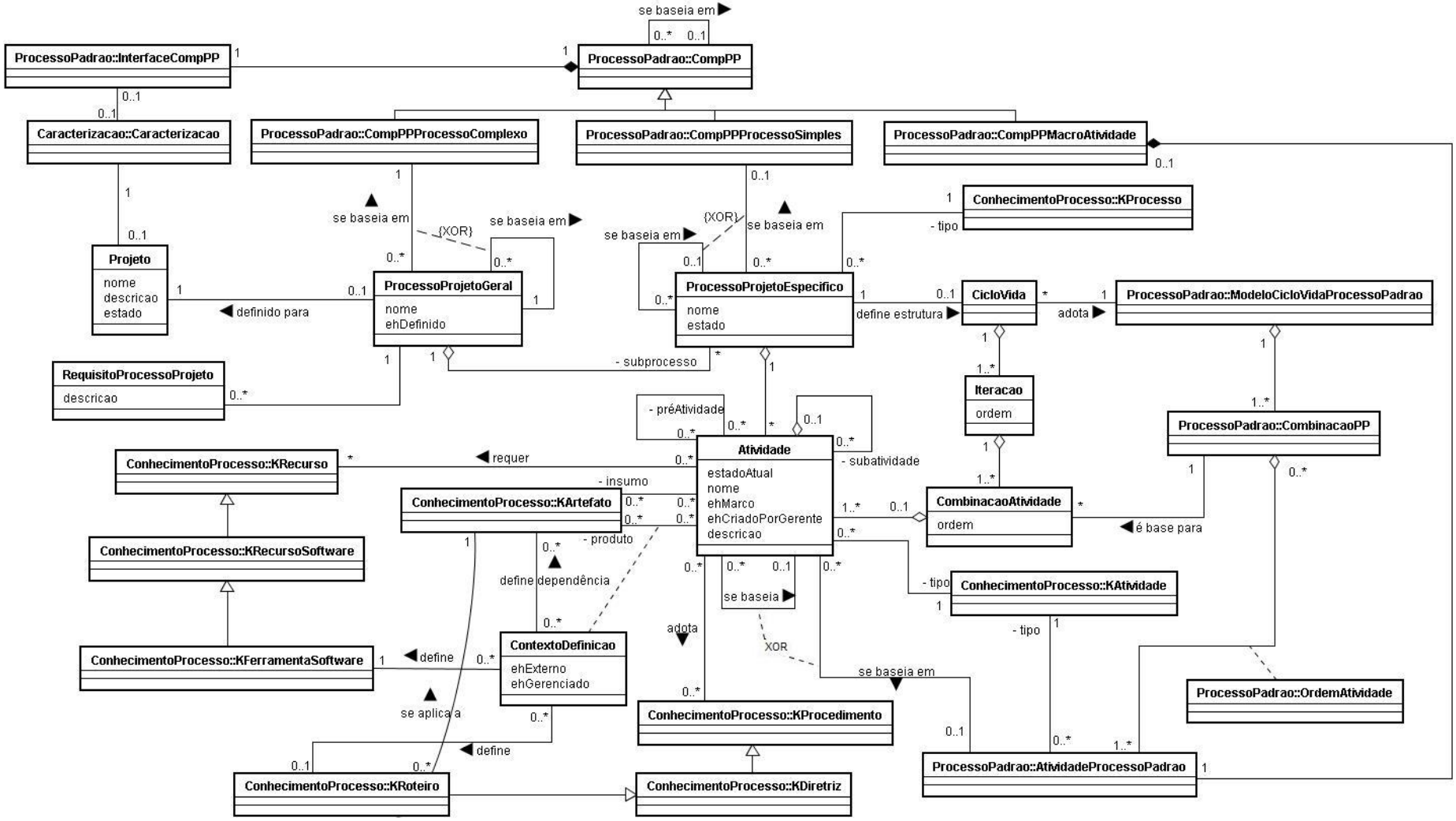

Figura 4.8 – Diagrama de Classes parcial do pacote *ControleProcesso*.

4.4. Apresentação de ProcODE-Com

Nesta seção são apresentadas as principais funcionalidades da ferramenta ProcODE-Com, por meio da apresentação de algumas de suas interfaces. Assim, é ilustra a utilização da ferramenta apoiando a definição de componentes de processos (definição *para* reúso) e a definição de processos de projeto utilizando componentes (definição *com* reúso).

4.4.1. Definição de Componentes de Processo

Nesta subseção são apresentadas algumas funcionalidades de ProcODE-Com utilizadas para a definição de componentes de processo. Para isso, o exemplo ilustrativo seguido na Seção 3.3.1 e em suas subseções foi utilizado.

A definição de um CompPP inicia-se pelo levantamento de seus requisitos, quando são definidos, além dos requisitos propriamente ditos, os objetivos e as características do componente, devendo ser consideradas as características utilizadas pela organização na caracterização de seus componentes. A Figura 4.9 ilustra o apoio ao levantamento dos requisitos do Processo Padrão Especializado OO definido durante o exemplo da Seção 3.3.1, apresentado na Tabela 3.1.

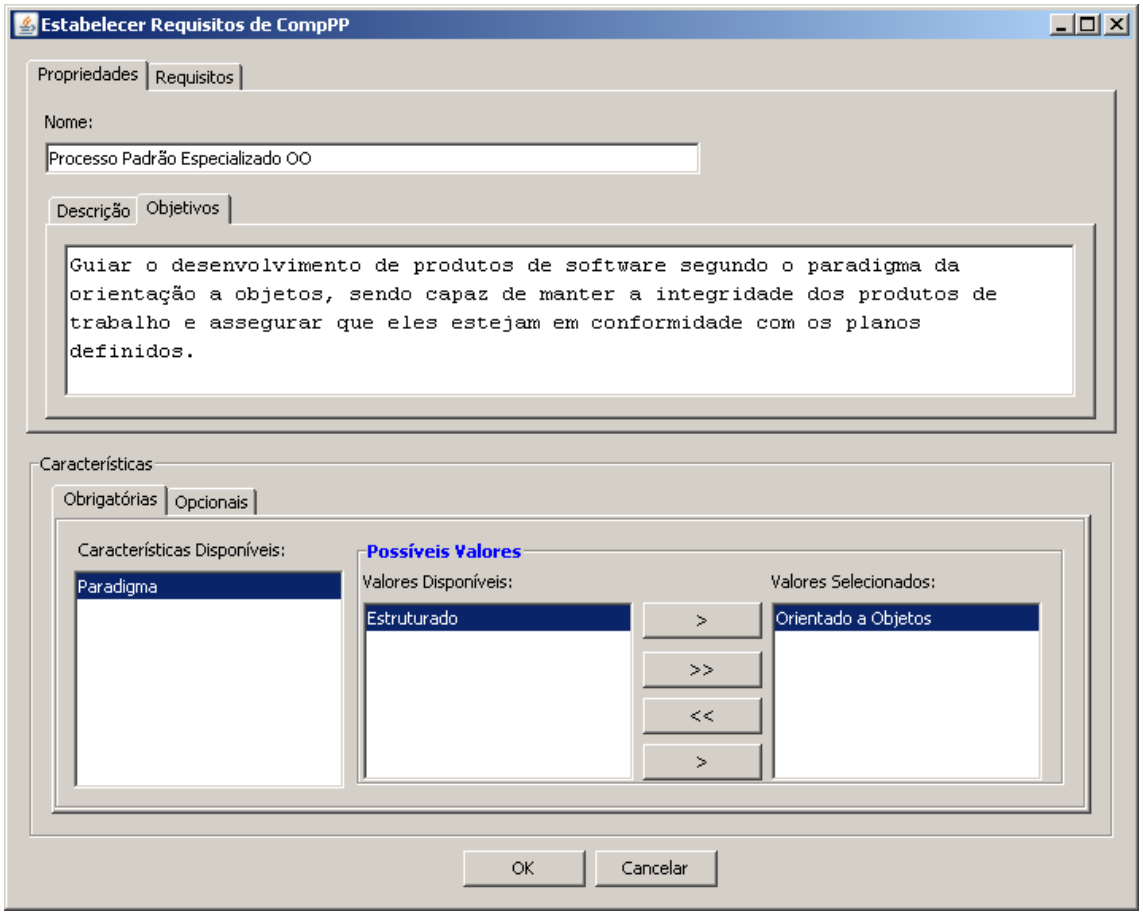

Figura 4.9 – Levantamento dos Requisitos de Novo CompPP.

Uma vez definidos os requisitos do novo CompPP, parte-se para a seleção do CompPP base. A Figura 4.10 ilustra a seleção do CompPP Processo Padrão Geral, que servirá de base para a definição do novo CompPP Processo Padrão Especializado OO. Note que no exemplo apresentado, o CompPP base foi selecionado diretamente.

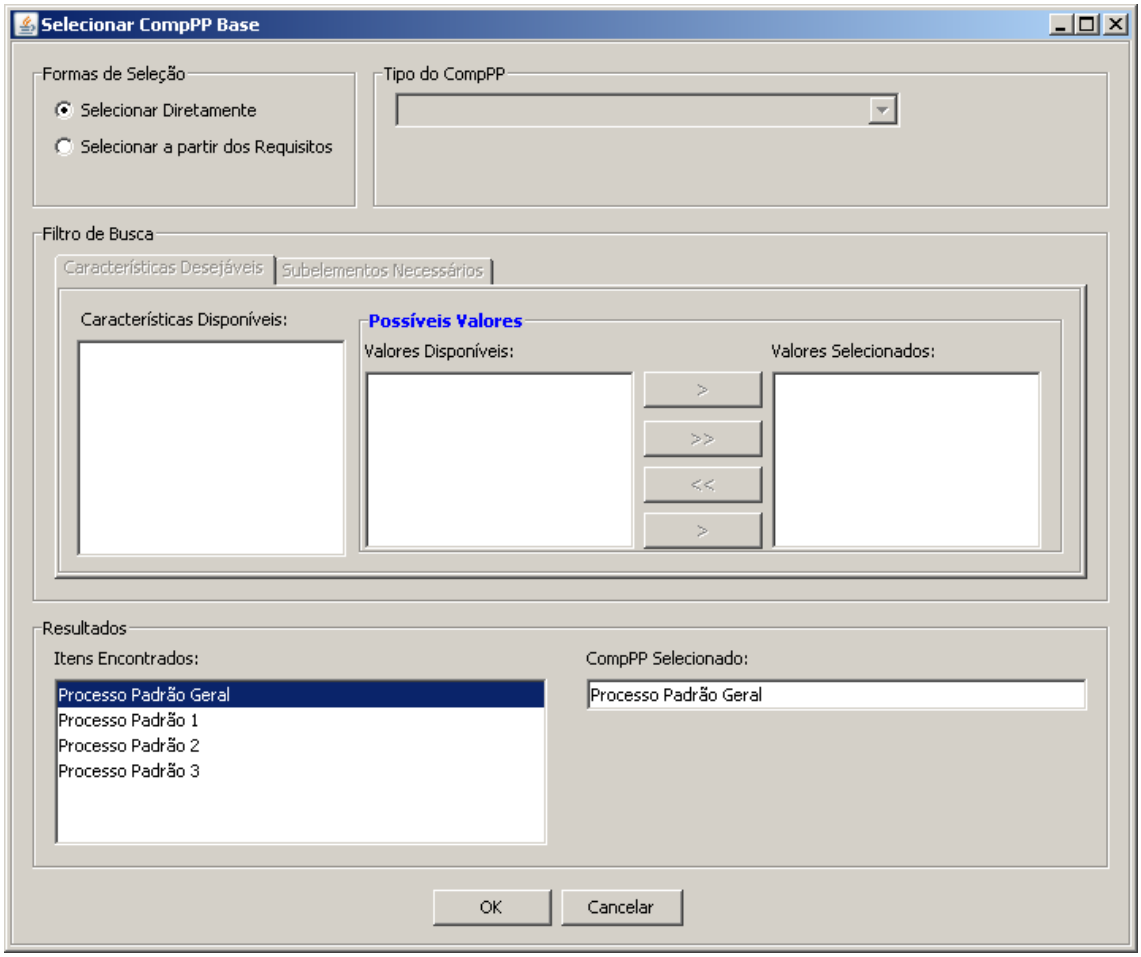

Figura 4.10 – Seleção de CompPP Base de Processo Complexo.

Após a seleção do CompPP base, parte-se para a definição da interface do novo CompPP. A definição da interface de um CompPP envolve a definição de sua estrutura, inclusive com a indicação de quais subelementos são obrigatórios, e a definição do seu contexto de aplicação. A Figura 4.11 ilustra a definição da interface do CompPP de processo complexo Processo Padrão Especializado OO.

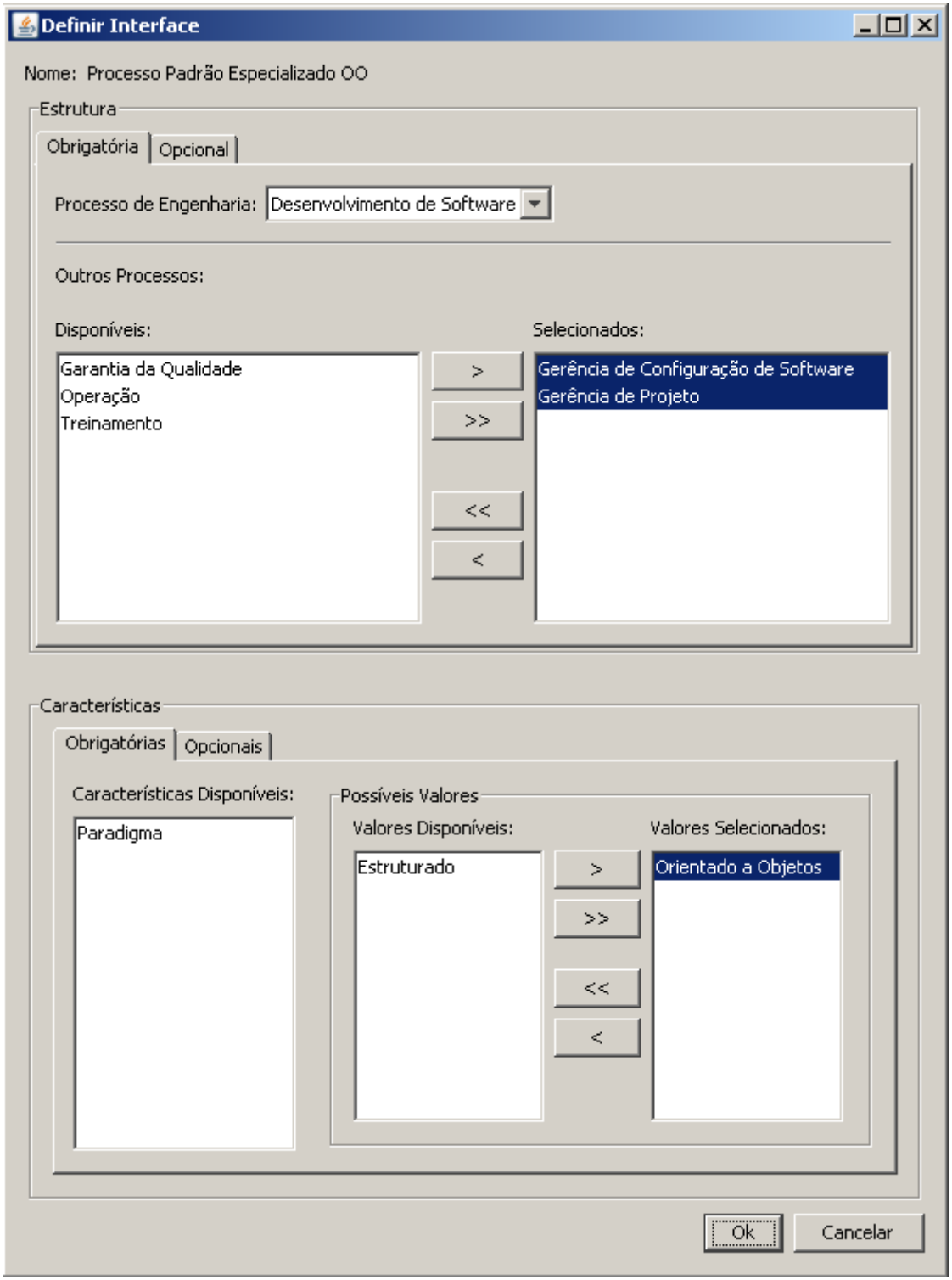

Figura 4.11 – Definição da Interface do Novo CompPP.

Feita a definição da interface do CompPP, passa-se para a adaptação do CompPP base. A adaptação ocorre em todos os níveis de granularidade do componente, chegando até o nível de subatividades (subelementos que compõem um componente de macroatividade). A Figura 4.12 ilustra o momento em que o engenheiro de processo

opta pela reutilização de todas as macroatividades que compõem o processo simples de Desenvolvimento. Já a Figura 4.13 mostra a alteração da interface da macroatividade de Análise de Requisitos para que ela passe a contemplar em seu contexto de aplicação a característica de paradigma orientado a objetos. Por fim, a Figura 4.14 exibe o momento em que a subatividade Levantamento de Requisitos é alterada para contemplar o produto Modelo de Casos de Uso.

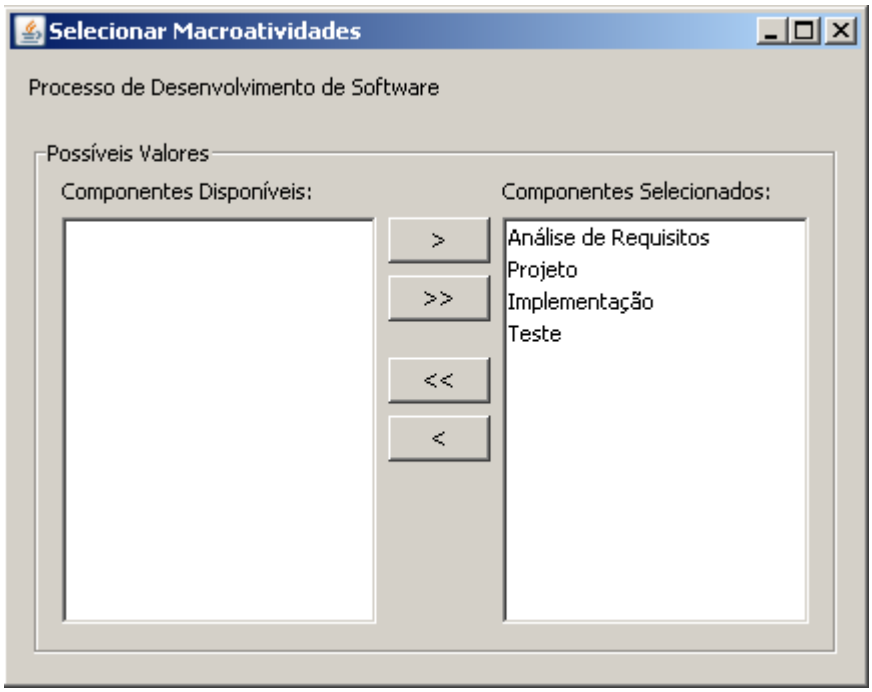

Figura 4.12 – Indicação das Macroatividades Opcionais a Serem Reutilizadas.

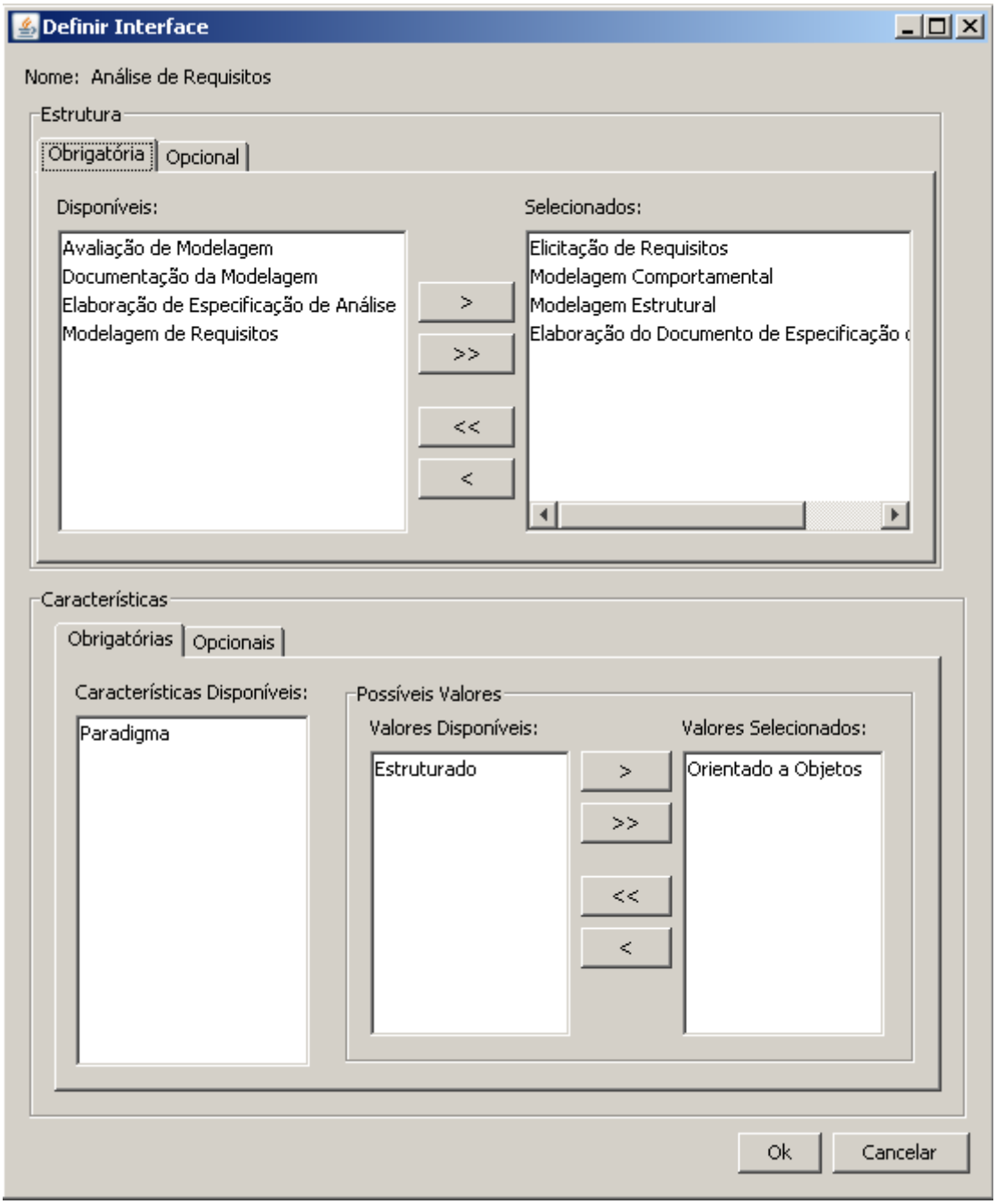

Figura 4.13 – Alteração da Interface de um CompPP.

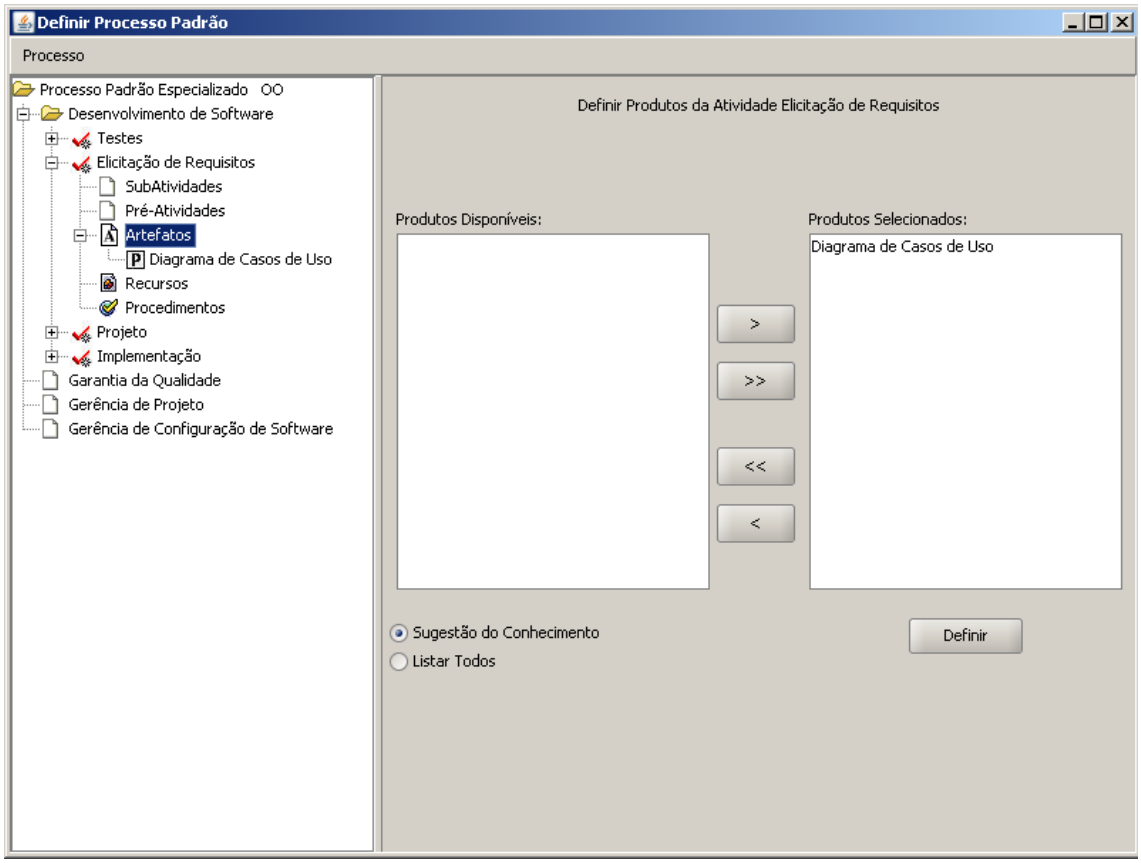

Figura 4.14 – Adaptação de atividade: Inclusão de Novo Artefato Produto.

Além de adaptar o CompPP base, novas definições podem ser necessárias. No exemplo seguido, foi necessária a definição de um CompPP de processo simples Garantia da Qualidade, uma vez que não existia nenhum CompPP desse tipo no repositório de componentes, o que impossibilitou a reutilização. A Figura 4.15 ilustra o momento em que é definido quais as macroatividades que compõem o processo simples de Garantia da Qualidade. Essas macroatividades serão definidas a partir do zero.

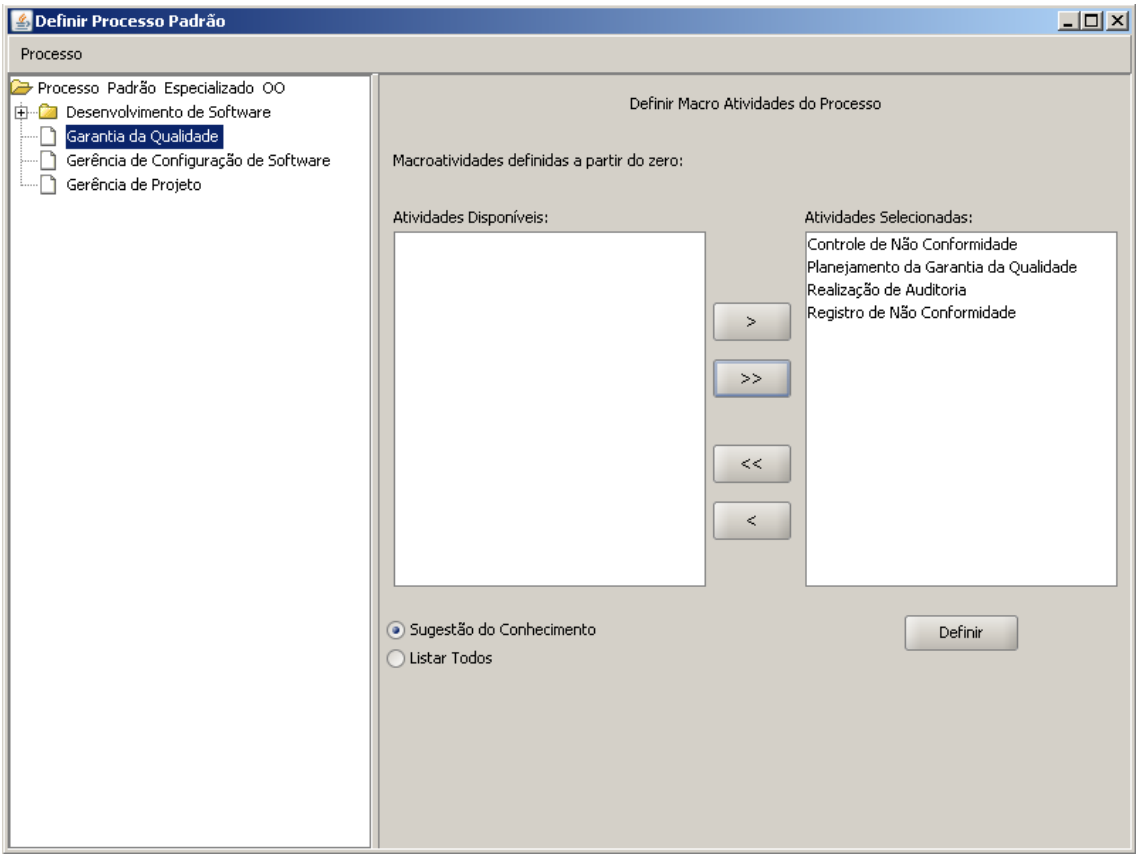

Figura 4.15 – Definição de Novo CompPP: Definição das Macroatividades a Serem Definidas a partir do Zero.

4.4.2. Definição de Processos de Projeto

Nesta subseção são apresentadas algumas funcionalidades de ProcODE-Com utilizadas para a definição de processos de projeto. Para isso, o exemplo ilustrativo seguido na Seção 3.3.2 e em suas subseções foi utilizado.

Assim como a definição de um CompPP, a definição de um processo de projeto inicia-se pela definição de seus requisitos. Nesse momento são definidos os objetivos e requisitos do novo processo de projeto. A Figura 4.16 ilustra a definição dos requisitos do Novo Processo Projeto considerado no exemplo da Seção 3.3.2.

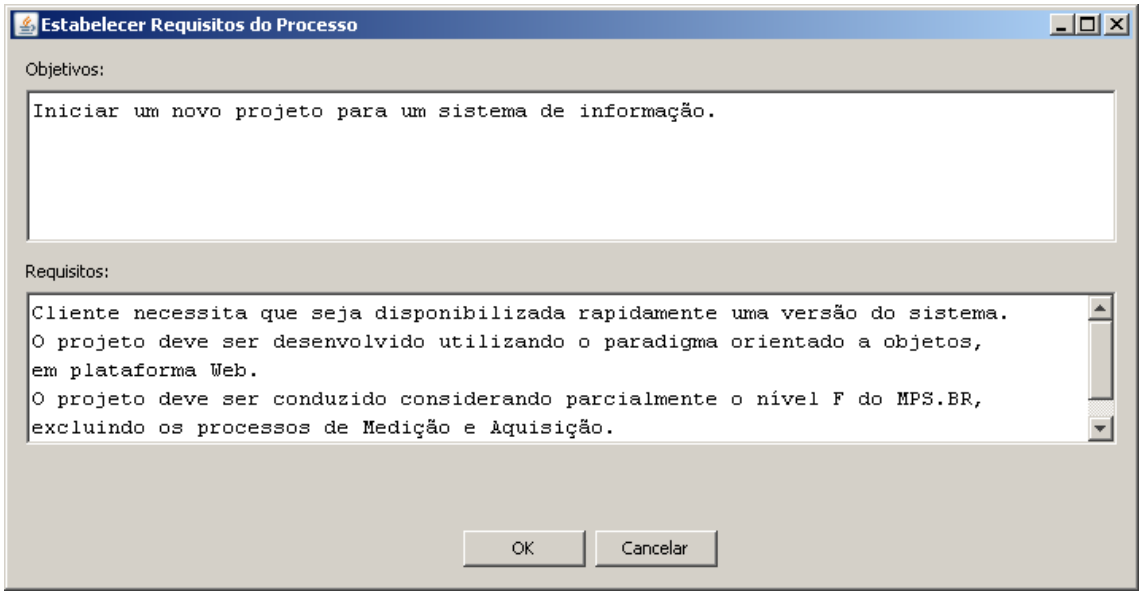

Figura 4.16 – Definição dos Requisitos do Novo Processo de Projeto.

Após a definição dos requisitos do novo processo complexo, deve-se selecionar o processo complexo base. No exemplo ilustrativo, o gerente de projeto indicou como característica desejável para o componente de processo complexo base o paradigma orientado a objetos. Indicou também que devem aparecer em sua composição processos simples de desenvolvimento, gerência de projetos, gerência de configuração, garantia da qualidade e treinamento. A busca retorna apenas o CompPP de processo complexo Processo Padrão Especializado OO, que é selecionado pelo gerente de projeto. A Figura 4.17 ilustra os passos descritos acima.

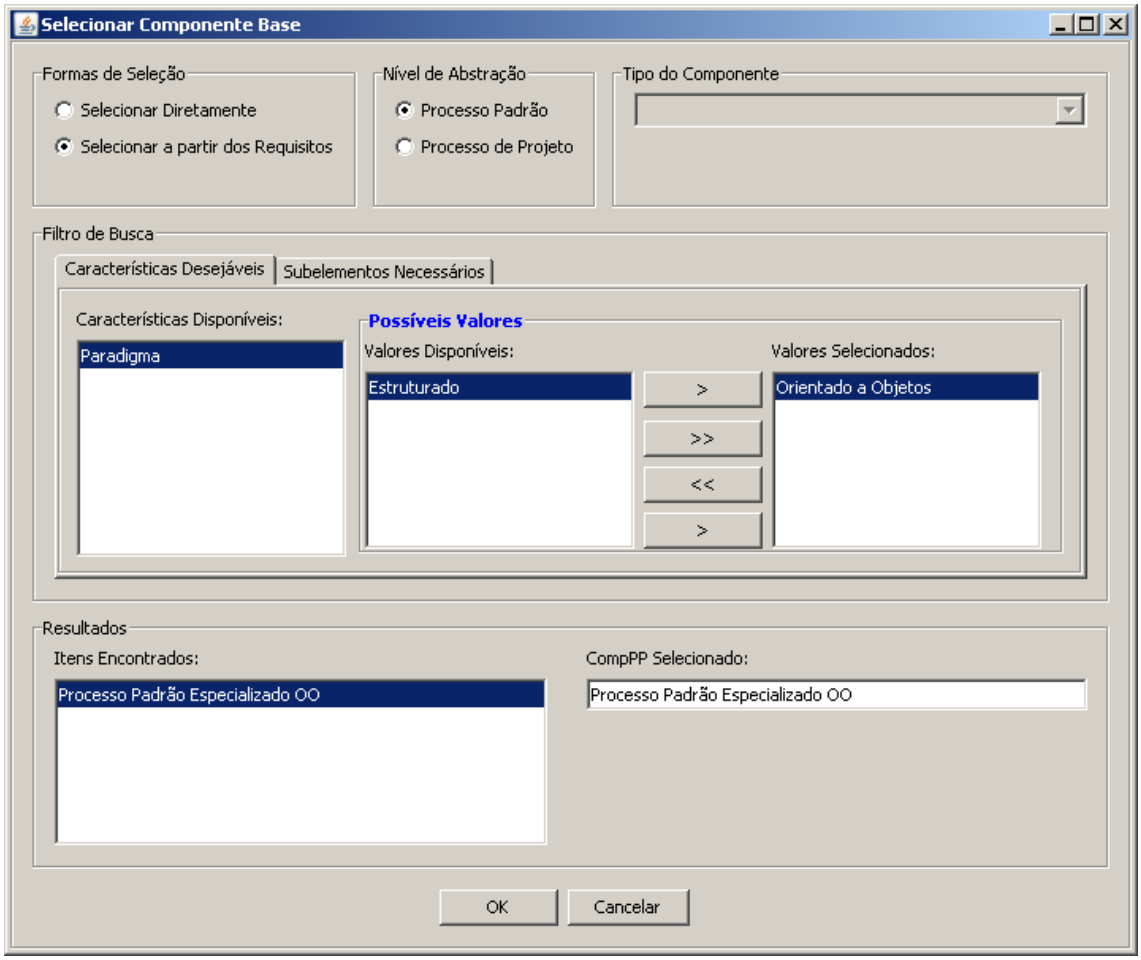

Figura 4.17 – Seleção de Componente Base de Processo Complexo.

Como no nível de abstração de processo de projeto não há definição explícita da interface dos componentes, o próximo passo é a adaptação do processo complexo base. A adaptação de um processo complexo base inclui, também, a adaptação dos componentes de menor granularidade que o compõem, podendo envolver, ainda, a definição de novos componentes a partir do zero. A Figura 4.18 mostra a seleção da macroatividade Implantação para compor o processo simples Desenvolvimento. Já a Figura 4.19 ilustra a adaptação da macroatividade Implementação, que envolve a definição das normas de Programação em JSP e Programação em Java como procedimentos a serem seguidos nessa atividade.

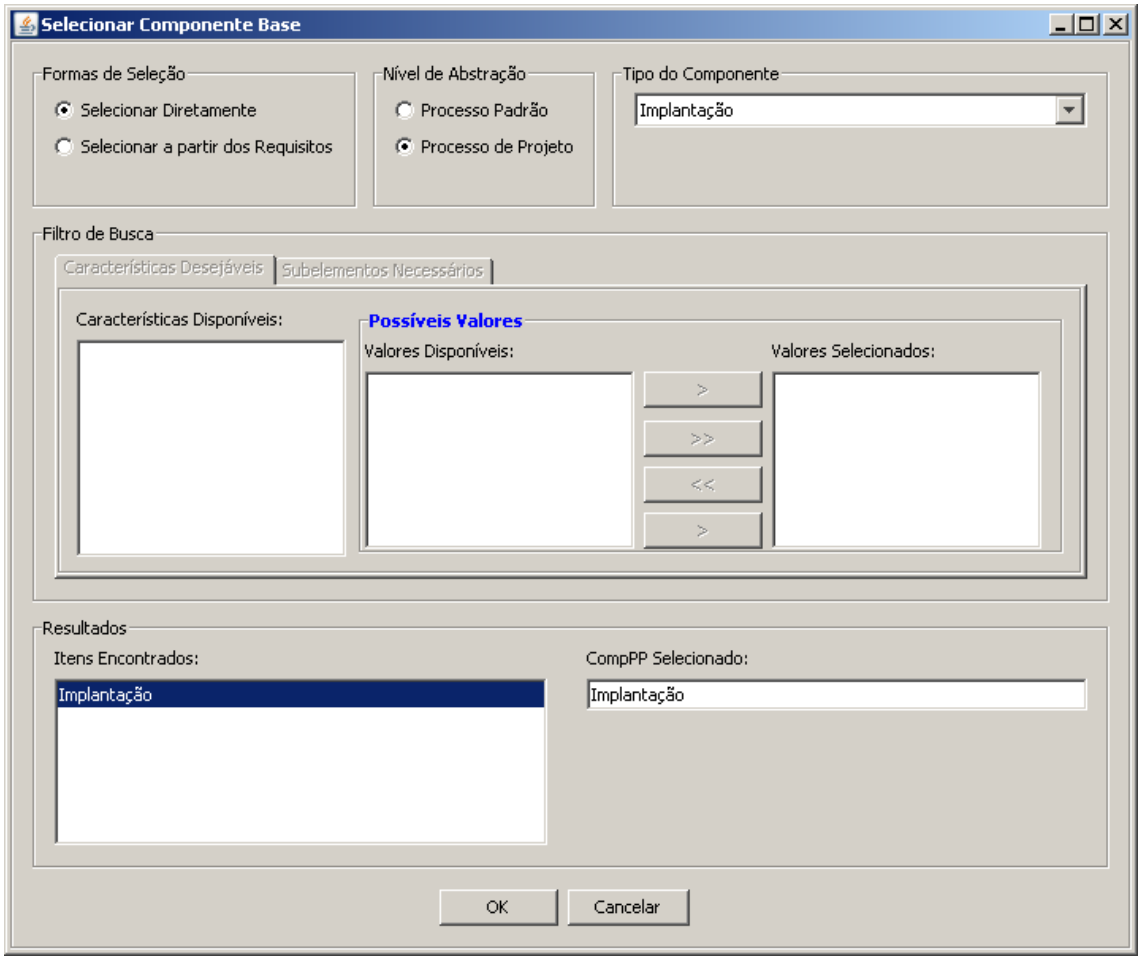

Figura 4.18 – Seleção de Componente Base de Macroatividade.

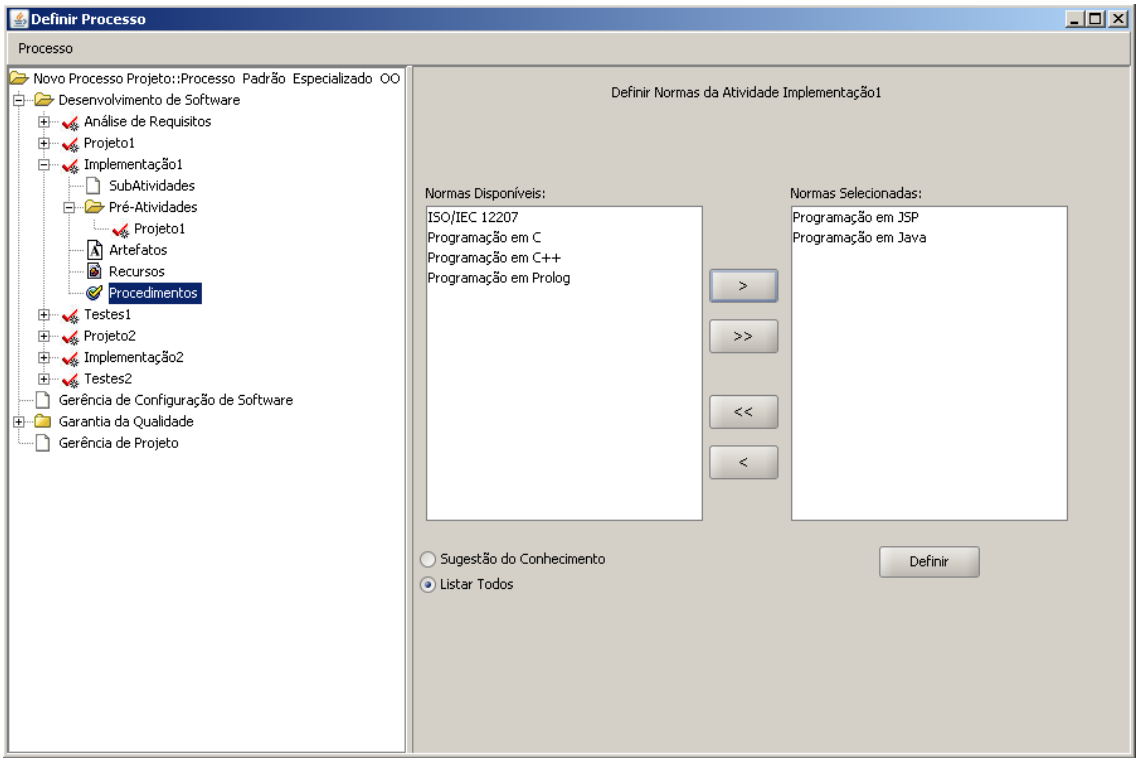

Figura 4.19 – Adaptação de atividade: Inclusão de Novos Procedimentos.

Em seguida, caso o processo de projeto em definição possua um processo simples de engenharia em sua composição, deve-se definir um modelo de ciclo de vida a ser adotado (p. ex., sequencial, evolutivo, incremental etc.). Nos casos de modelos iterativos, para cada combinação iterativa de atividades, deve-se indicar o número de iterações a serem realizadas. No exemplo ilustrativo, decidiu-se utilizar o modelo incremental, com as atividades do processo simples de desenvolvimento divididas em duas combinações, sendo a primeira delas uma combinação sequencial contemplando apenas a macroatividade de Análise de Requisitos e a segunda uma combinação iterativa contemplando as demais macroatividades (Projeto, Implementação, Teste e Implantação). Ainda, foi indicado que seriam utilizadas duas iterações. A Figura 4.20 mostra a definição do modelo de ciclo de vida.

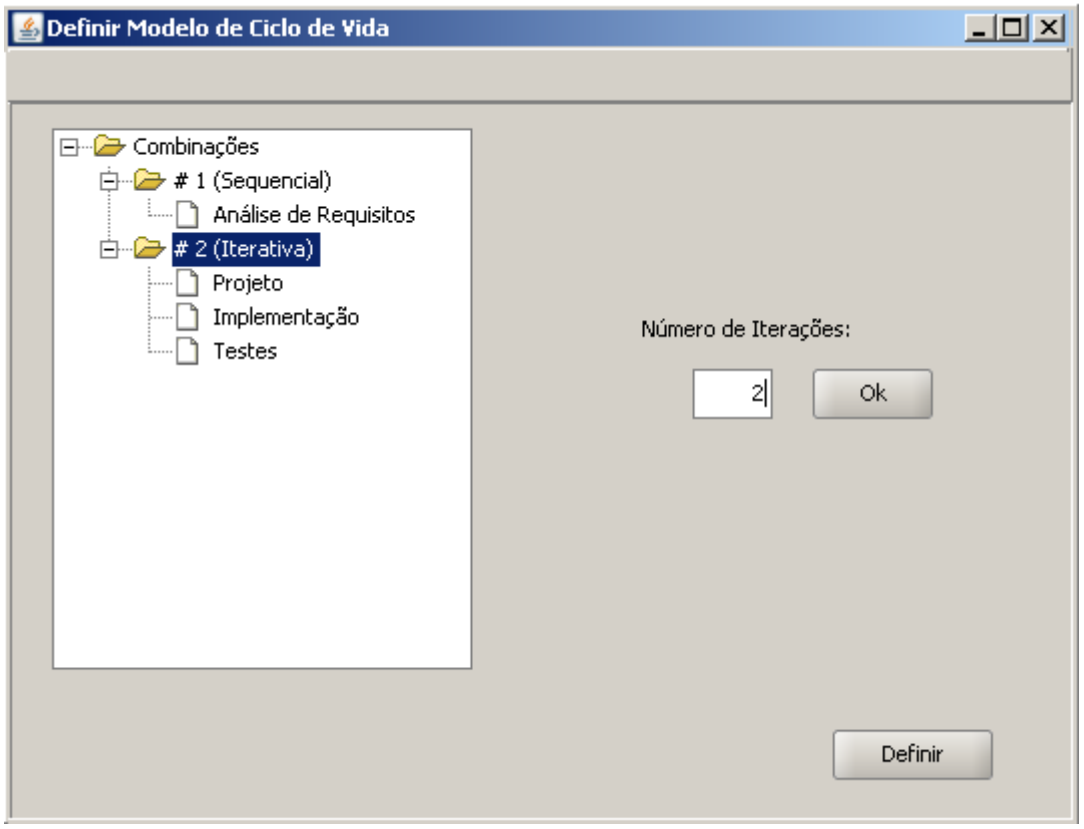

Figura 4.20 – Definição do Modelo de Ciclo de Vida Adotado.

4.5. Considerações Finais do Capítulo

Este capítulo apresentou a evolução da ferramenta de definição de processos de ODE, ProcODE-Com, que visa apoiar a abordagem de Definição de Processos Baseada em Componentes (DPBC) proposta.

Capítulo 5

Considerações Finais

O objetivo deste capítulo é tecer comentários finais acerca do trabalho realizado, com enfoque nas contribuições e perspectivas futuras de pesquisas, levantadas a partir dos resultados e ideias que surgiram ao longo da realização deste trabalho.

5.1. Conclusões

A percepção por parte das organizações produtoras de software de que a qualidade do produto desenvolvido está diretamente ligada à qualidade do processo adotado faz com que elas se preocupem com a definição de seus processos de software. Essa percepção confirma a importância da existência de abordagens que auxiliem a definição de processos.

Contudo, definir processos de software não é uma tarefa trivial. Embora exista farto material indicando as melhores práticas a serem seguidas, processos de software têm de ser definidos caso a caso, considerando características específicas do projeto em questão, tais como domínio de aplicação, equipe de desenvolvimento, tecnologia a ser usada e restrições de custos e prazos. Neste contexto, a reutilização de processos é uma boa alternativa para apoiar essa tarefa.

Considerando-se que a área de reutilização de produtos de software vêm sendo objeto de estudo já há algum tempo, ideias dessa área podem servir de base para proposição de abordagens voltadas para a reutilização de processos. No contexto deste trabalho, foi proposta uma abordagem de Definição de Processos Baseada em Componentes (DPBC) que procura incorporar algumas das ideias do Desenvolvimento Baseado em Componentes (DBC), permitindo a definição de processos a partir de componentes de processo previamente definidos.

Para apoiar a abordagem proposta, a ferramenta de definição de processos de software de ODE (*Ontology-based Software Development Environment*) (FALBO et al., 2003) foi evoluída.

Desse modo, as principais contribuições deste trabalho foram:

- Definição da abordagem de Definição de Processos Baseada em Componentes (DPBC), visando apoiar a definição de processos fomentando a reutilização de componentes de processo de software. Foram definidos componentes de processo em diferentes níveis de granularidade (processos complexos, processos simples e macroatividades) e níveis de abstração (processos padrão ou de projeto), bem como suas interfaces e repositórios. Ainda, definiram-se processos de DPBC para os níveis de processos padrão e processos de projeto, focando no desenvolvimento *para* e *com* reúso;
- Evolução da ferramenta de definição de processos de ODE, fazendo com que ela passe a prover apoio automatizado para a abordagem proposta.

Em suma, este trabalho buscou contribuir com a evolução da área de processos de software em duas vertentes: (i) conceitualmente, através da proposição de uma abordagem que visa reduzir o esforço necessário para a definição de processos de software; e (ii) no que se refere ao apoio automatizado à definição de processos, propondo a evolução da ferramenta de definição de processos de ODE para prover apoio automatizado à definição de processos de software com reutilização de componentes de processo pré-existentes, tanto no nível de processos padrão quanto no nível de processos de projeto.

5.2. Perspectivas Futuras

Um ponto negativo que pode ser citado em relação à abordagem de DPBC proposta é a falta de sua utilização na prática. A utilização da abordagem em uma organização desenvolvedora de software poderia auxiliar na avaliação da abordagem quanto às suas reais eficiência e eficácia. Ela poderia ser primeiramente utilizada em projetos piloto para ser avaliada e, caso necessário, sofrer os devidos ajustes. Assim, um importante trabalho futuro é a avaliação em situações reais da abordagem proposta.

Além disso, por ser uma área em expansão e de grande importância para a Engenharia de Software, muito pode ser feito no contexto da definição de processos de software para dar continuidade ao trabalho realizado.

Embora a definição da abordagem de Definição de Processos Baseada em Componentes (DPBC) tenha sido a principal contribuição deste trabalho, foram

identificados alguns aspectos que podem ser desenvolvidos para dar continuidade ao trabalho realizado.

Um importante assunto a ser tratado na DPBC é a gerência de configuração para componentes de processo. MURTA (2006) definiu uma abordagem para gerência de configuração (GCS) no contexto do Desenvolvimento Baseado em Componentes (DBC), denominada Odyssey-SCM. Uma alternativa ao problema da gerência de configuração de componentes de processo seria realizar analogias dessa abordagem para o domínio de componentes de processo, assim como foi feito com DBC para definição da abordagem de DPBC, permitindo tratar o importante problema da gerência de configuração dos componentes de processo.

Modelos de maturidade como o CMMI-DEV e o MPS.BR advogam que, em níveis mais altos de maturidade, a seleção de componentes mais adequados para a composição de um processo deve considerar sua estabilidade, capacidade e desempenho histórico, uma vez que selecionar os componentes considerando essas informações tende a produzir processos melhores. Dessa forma, um interessante trabalho futuro é estudar como medir componentes de processo, possibilitando verificar o seu comportamento, e considerar esses dados, obtidos a partir de medições, na definição de processos.

A reutilização de processos não inclui apenas a reutilização de componentes de processo, mas também a reutilização de informações relacionadas à utilização dos mesmos, incluindo conhecimento e experiências adquiridas (melhores práticas, lições aprendidas etc). Assim, um possível trabalho a ser desenvolvido diz respeito à descrição de dados dos componentes de processo relacionados à suas execuções em projeto, apresentando informações importantes sobre o controle e execução do componente, tais como esforço necessário, custo de execução, riscos associados e quantidade de membros na equipe. Desse modo, o planejamento da execução dos componentes poderia ser realizado considerando-se essas informações de componentes similares executados anteriormente.

Arquiteturas e linhas de processo buscam auxiliar a adaptação de processos gerais para originarem os processos específicos de projeto. Nesse sentido, um trabalho a ser desenvolvido é possibilitar a definição de componentes de processo padrão especializados para um determinado domínio a partir de componentes de processos de projetos desse mesmo domínio. Considerando-se componentes de processos de projeto de um determinado domínio, um componente padrão especializado poderia ser definido

para esse domínio. O componente padrão especializado poderia ter, ainda, pontos de variação e variantes definidos. Depois, esse componente de processo padrão especializado seria utilizado para definição de componentes de processo de projeto por meio de sua adaptação, o que envolveria a escolha das variantes a serem utilizadas e dos subelementos opcionais a serem considerados, aproximando a abordagem de DPBC com as abordagens de linhas de processos.

No que se refere à ferramenta de definição de processos de ODE, é importante também a sua utilização na prática para avaliação de pontos fortes e fracos e identificação de possíveis pontos de melhoria.

Acredita-se, também, que a utilização da tecnologia de agentes pode ser bastante útil neste contexto. Nesse sentido, um potencial trabalho futuro seria a construção de um sistema multiagente que monitorasse a definição de processos, procurando apresentar proativamente componentes de menor grau de granularidade com potencial para serem utilizados na composição do componente em definição.

No contexto mais geral da definição de processos na ferramenta, é interessante que seja desenvolvido um trabalho visando a um apoio mais efetivo da gerência de conhecimento na definição de processos de software. O objetivo seria permitir criar, capturar, acessar, disseminar, usar, preservar e evoluir o conhecimento organizacional acerca de componentes de processo de software.

Por fim, outro trabalho interessante a ser desenvolvido é a definição de um mecanismo que apoie o engenheiro de processo e o gerente de projeto a visualizarem as adaptações feitas em um componente de processo que esteja sendo reutilizado quando reutilizado em contextos similares. Com o auxílio da tecnologia de agentes, poderiam ser sugeridas adaptações para a definição de novos componentes a partir de outros já existentes. Ainda com esse mesmo mecanismo, o engenheiro de processo poderia visualizar as adaptações realizadas em um determinado componente de processo quando reutilizado, sendo possível diagnosticar necessidades de evolução do componente, criação de um novo componente ou especialização do componente já existente.

Enfim, como pode ser observado pelas reflexões acima, ainda que um considerável trabalho tenha sido realizado ao longo desta pesquisa, muitos outros podem ser desenvolvidos a partir das ideias apresentadas neste trabalho, ficando, assim, como perspectivas para novos estudos e pesquisas futuras.

Referências Bibliográficas

AMBLER, S. W. (1998) “Process Patterns: Building Large-Scale Systems Using Object Technology”. Cambridge University Press, New York.

ARBAOUI, S., DERNIAME, J., OQUENDO, F. (2002) “A Comparative Review of Process – Centered Software Engineering Environments”. In: Kluwer Academic Publishers. Annals of Software Engineering 14. p. 311-340.

ARENT, J., NORBJERG, J., PEDERSEN, M. H. (2000) “Creating Organizational Knowledge in Software Process Improvement”. In Workshop on Learning Software Organizations, LSO'2000, Oulu, Finland. Proceedings...2000.

BARRETO, A., MURTA, L., ROCHA, A. R. (2008) “Software Process Definition: a Reuse-based Approach”. In XXXIV Conferencia Latinoamericana de Informática, CLEI108, Santa Fe, Argentina.

BARRETO, A., MURTA, L., ROCHA, A. R. (2009) “Componentizando Processos de Software Legados visando a Reutilização de Processos”. In VIII Simpósio Brasileiro de Qualidade de Software, SBQS’09, Ouro Preto, Brasil.

BASILI, V. R., ROMBACH, H. D. (1987) “Tailoring the Software Process to Project Goals and Environment”. In: Proc. of the 9ª International Conference on Software Engineering (ICSE’87), Monterey, CA. pp. 345-357.

BERGER, P. (2003) “Instanciação de Processos de Software em Ambientes Configurados na Estação TABA”. Dissertação de Mestrado, COPPE/UFRJ, Rio de Janeiro.

BERTOLLO, G. (2006) “Definição de Processos em um Ambiente de Desenvolvimento de Software”. Dissertação de Mestrado, Mestrado em Informática, UFES.

BERTOLLO, G., FALBO, R. A. (2003) “Apoio Automatizado à Definição de Processos em Níveis”. In: II Simpósio Brasileiro de Qualidade de Software, SBQS’2003, Fortaleza, Brasil. Anais...2003. p. 77-91.

BERTOLLO, G., SEGRINI, B., FALBO, R. A. (2006) “Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias”. In: V Simpósio Brasileiro de Qualidade de Software, SBQS’06, Vila Velha, Brasil. Anais...2006. p. 72-86.

CARVALHO, V. A. (2006) “Gerência de Conhecimento e Decisão em Grupo: Um Estudo de Caso na Gerência de Projetos”. Dissertação Mestrado, Mestrado em Informática, UFES.

CIMITILE, A., VISAGGIO, G. (1994) “A Formalism for Structured Planning of a Software Project”. In: Int. Journal of Software Engineering and Knowledge Engineering, vol. 4, no. 2, , World Scientific Publishing. pp.277-300.

COOPER, J. (1994) “Reuse – the business implications”. In: Marciniak, p. 1071-1077.

COSTA, A., SALES, E., REIS, C. A. L., REIS, R. Q. (2007) "Apoio a Reutilização de Processos de Software através de Templates e Versões". In: VI Simpósio Brasileiro de Qualidade de Software, Porto de Galinhas, Brasil, Junho. pp. 47-61.

CRNKOVIC, I., LARSSON, S., CHAUDRON, M. (2006) “Component-Based Development Process and Component Lifecycle”, In: International Conference on Software Engineering Advances, ICSEA’2006, Tahiti, French Polynesia. Proceedings… 2006.

DAL MORO, R. (2005) “CONTROLPRO: Uma Ferramenta de Apoio ao Acompanhamento de Projetos de Software em ODE”. Monografia (Graduação em Ciência da Computação) - Universidade Federal do Espírito Santo.

FALBO, R. A. (1998) “Integração de Conhecimento em um Ambiente de Desenvolvimento de Software”. Tese (Doutorado em Engenharia de Sistemas e Computação) - COPPE, Universidade Federal do Rio de Janeiro.

FALBO, R. A., NATALI, A. C. C., MIAN, P. G., BERTOLLO, G., RUY, F. B. (2003) “ODE: Ontology-based software Development Environment”, In: Memórias de IX Congreso Argentino de Ciencias de la Computación, p. 1124-1135, La Plata, Argentina.

FALBO, R. A., RUY, F. B., PEZZIN, J., DAL MORO, R. (2004a) “Ontologias e Ambientes de Desenvolvimento de Software Semânticos”. In: 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering, JIISIC’2004, Madrid, Spain. Vol. I. p. 277-292.

FALBO, R. A., RUY, F. B., BERTOLLO, G., TOGNERI, D. F. (2004b) “Learning How to Manage Risks Using Organizational Knowledge”. In: 6th International Workshop on Advances in Learning Software Organizations, LSO’2004, Banff, Canada. Proceedings…2004. p. 7-18.

FRANCH, X., RIBÓ, J. (2002) “Supporting Process Reuse in PROMENADE”. Research Report, No. LSI-02-14-R. Barcelona: Departament de Llenguatges i Sistemes Informàtics, Universitat Politécnica de Catalunya.

FREEMAN, F. (1987) “Reusable software engineering concepts and research directions”. In Tutorial: Software Reusability.

FUGGETTA, A., (2000) “Software Process: A Roadmap”. In: Proceedings of The Future of Software Engineering, ICSE’2000, Limerick, Ireland.

FUSARO, P., VISAGGIO, G., TORTORELLA, M. (1998) “REP – ChaRacterizing and Exploiting Process Components: Results of Experimentation”. In: Working Conference on Reverse Engineering, Honolulu, United Sates. pp.20-29.

GARY, K. A., LINDQUIST, T. E. (1999) “Cooperating Process Components”. In: International Computer Software and Applications Conference (COMPSAC), Phoenix, United States. Pp.218-223.

GIMENES, I. M. S., HUZITA, E. H. M. (2006) “Desenvolvimento Baseado em Componentes: Conceitos e Técnicas”, Ciência Moderna, 1ª Ed., 2006.

GRISS, M. (1995) “Systematic Software Reuse: Objects and Frameworks are not enough”. In: SSR’95 - ACM Symposium on Software Reusability, Seattle, EUA. Anais… Seattle, 1995.

GRUHN, V. (2002) “Process-Centered Software Engineering Environments: A Brief History and Future Challenges”. In: Kluwer Academic Publishers. Annals of Software Engineering 14. p. 363-382.

GUIZZARDI, G. (2000) “Uma abordagem metodológica de desenvolvimento para e com reuso baseada em Ontologias formais de Domínio", Dissertação de Mestrado, Mestrado em Informática, UFES.

GUIZZARDI, G., FALBO, R. A., GUIZZARDI, R. S. S. (2008) “Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology”. In: XI Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS 2008), Recife, Brazil. Proceedings of the XI Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS 2008).

GROENEWEGEN, L., LICHTNOW, D., WANGENHEIM, A. (1996) “Reuse of software process fragments is reuse of software too”. In Proceedings of the 10th International Software Process Workshop (ISPW '96), Dijon, France.

HENNINGER, S. (1999) “Using Software Process to Support Learning Software Organizations”. In Workshop on Learning Software Organizations, LSO'99, Kaiserslautern, Germany. Proceedings…1999.

HOLANDA, C. B. S, SOUZA, C. A. A., MELO, W. L. (2001) “ProReuso: Um Repositório de Componentes para Web Dirigido por um Processo de Reuso”, In: Anais do XV Simpósio Brasileiro de Engenharia de Software, SBES’2001, Rio de Janeiro, Brasil. Anais... 2001.

HUANG, H., ZHANG, S. (2003) "Hierarchical Process Patterns: Construct Software Processes in a Stepwise Way". In: IEEE International Conference on Systems, Man and Cybernetics, v. 2, pp. 1353-1358, Washington, United States, October.

ISO/IEC (2008) “ISO/IEC 12207 – Systems and software engineering - Software Life Cycle Processes”.

ISODA, S. (1992) “Experience Report on Software Reuse Project: Its Structure, Activities and Statistical Results”. In: Proc. of International Conference on Software Engineering, 14.

JAUFMAN, O. (2005) “Process Line Framework for the Domain of Automotive System Development”, SPICE 2005 Conference.

JAUFMAN, O., MÜNCH, J. (2005) "Acquisition of a Project-Specific Process". In: Product Focused Software Process Improvement, Oulu, Finland, June. pp. 328-342

KELLNER, M. I. (1996) “Connecting Reusable Software Process Elements and Components”. In: Proceedings of the 10th International Software Process Workshop (ISPW '96), Dijon, France.

KRUEGER, C. (1992) “Software Reuse”. In: ACM computing surveys. v.24, n.2.

MAURER, F., DELLEN, B. (1999) “Process Support for Virtual Software Organizations”. In Workshop on Learning Software Organizations, LSO'99, Kaiserslautern, Germany. Proceedings…1999.

MARTINS, A. F., NARDI, J. C., FALBO, R. A. (2006) “ReqODE: Uma Ferramenta de Apoio à Engenharia de Requisitos Integrada ao Ambiente ODE”. In: XX Simpósio Brasileiro de Engenharia de Software, SBES’2006, Florianópolis, Brasil. Anais da XIII Sessão de Ferramentas do SBES 2006, 2006. p. 31-36.

MCILROY, M. D. (1968) “Mass-produced Software Components”. In North Atlantic Treaty Organization Conference on Software Engineering, Garmisch-Partenkirchen.

MURTA, L. G. P. (2006) “Gerência de Configuração no Desenvolvimento baseado em Componentes”. Tese (Doutorado em Engenharia de Sistemas e Computação) - Universidade Federal do Rio de Janeiro

NARDI, J. C., FALBO, R. A. (2006) “Uma Ontologia de Requisitos de Software”. In: IX Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software, La Plata, Argentina.

NATALI, A. C. C., FALBO, R. A. (2003) “Gerência de Conhecimento em ODE”. In: XVII Simpósio Brasileiro de Engenharia de Software, SBES’2003, Manaus, Brasil. Anais...2003.

OMG - OBJECT MANAGEMENT GROUP (2007) “Software & systems process engineering metamodel specification 2.0”. http://www.omg.org/cgi-bin/doc?formal/2008-04-02.

OSTERWEIL, L. (1987) “Software Processes Are Software Too”. In: International Conference on Software Engineering. Monterey, Estados Unidos. pp.2-13.

PRESSMAN, R. S. (2006) “Engenharia de Software”. 6º edição. Rio de Janeiro: McGrawHill, 2006.

REIS, R. Q. (2002) “APSEE-REUSE- Um Meta-Modelo para Apoiar a Reutilização de processos de Software”. Tese (Doutorado em Ciência da Computação) - Instituto de Informática, UFRGS.

ROCHA, A. R. C., MADONADO, J. C., WEBER, K. C. (2001) “Qualidade de Software: Teoria e Prática”, Prentice Hall.

RU-ZHI, X., TAO, H., DONG-SHENG, C., YUN-JIAO, X., LE-QIU, Q. (2005) “Reuse-Oriented Process Component Representation and Retrieval”. In: International Conference on Computer and Information Technology, Shangai, China.

SAMETINGER, J. (1997) “Software Engineering with Reusable Components”, Springer, New York.

SEACORD, R. (1999) “Software engineering componentes repositoiries”. In: International Workshop on Component-Based Software Engineering, Los Angeles, EUA.

SEGRINI, B. M., BERTOLLO, G., FALBO, R. A. (2006) “Evoluindo a Definição de Processos de Software em ODE”. In: XX Simpósio Brasileiro de Engenharia de Software, SBES’2006, Florianópolis, Brasil. Anais da XIII Sessão de Ferramentas do SBES 2006, 2006. p. 109-114.

SEI (2006) “CMMI for Development – v1.2”, Pittsburgh: Software Engineering Institute.

SIMIDCHIEVA, B.I., CLARKE, L.A., OSTERWEIL, L. (2007) "Representing Process Variation with a Process Family". In: International Conference on Software Process, v. Lecture Notes in Computer Science 4470, Minneapolis, Estados Unidos, Maio. pp. 109-120.

SOFTEX (2009) “MPS.BR – Melhoria de Processo do Software Brasileiro: Guia Geral”.

TRACZ, W. (1995) “Confessions of a Used Program Salesman: Instituionalizing Software Reuse”. Addison-Wesley.

WANG, Y., KING, G. (2000) “Software Engineering Processes: Principles and Aplications”. Boca Raton: CRC Press.

WASHIZAKI, H. (2006) "Building Software Process Line Architectures from Bottom Up". In: Product-Focused Software Process Improvement, Amsterdam, Netherlands, June. pp. 415-421.

WERNER, C. M. L., BRAGA, R. M. M. (2006) “A Engenharia de Domínio e o Desenvolvimento Baseado em Componentes”. In: Desenvolvimento Baseado em Componentes: Conceitos e Técnicas. 1ª Ed. Editora Ciência Moderna. P 57-103.