UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE INFORMÁTICA
MESTRADO EM INFORMÁTICA

GLEIDSON BERTOLLO

DEFINIÇÃO DE PROCESSOS EM UM AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE

VITÓRIA
2006

GLEIDSON BERTOLLO

DEFINIÇÃO DE PROCESSOS EM UM AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE

Dissertação apresentada ao Mestrado em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Orientador: Prof. Dr. Ricardo de Almeida Falbo.

VITÓRIA
2006

GLEIDSON BERTOLLO

DEFINIÇÃO DE PROCESSOS EM UM AMBIENTE DE DESENVOLVIMENTO DE SOFTWARE

COMISSÃO EXAMINADORA

_____

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

_____

Prof. Giancarlo Guizzardi, PhD.
UFES

_____

Prof. Tânia Barbosa Salles Gava, D.Sc.
Faculdade Salesiana de Vitória

Vitória, 13 de Junho de 2006.

RESUMO

Atualmente, é amplamente reconhecido que a qualidade dos produtos de software depende da qualidade dos processos de software utilizados em seu desenvolvimento e manutenção. Com isso, muito trabalho tem sido feito no sentido de apoiar organizações em seus esforços pela busca da qualidade de processo. Pesquisas na área de processos de software têm explorado duas principais vertentes: (i) abordagens para modelagem, análise e melhoria do processo de software e (ii) tecnologia de apoio ao processo de software.

A primeira vertente focaliza abordagens para estruturação, organização, documentação e descrição de processos de software e inclui normas de qualidade de processo de software. A segunda está voltada para o desenvolvimento de Ambientes de Desenvolvimento de Software (ADSs) Centrado em Processos, que integram ferramentas de apoio ao desenvolvimento de artefatos com ferramentas de apoio à modelagem e execução de processos de software, utilizados na construção desses artefatos. A representação explícita de processos, seus produtos e suas interações é a base sobre a qual modernos ambientes de desenvolvimento são construídos. Provendo formas mais poderosas de descrever e implementar processos de software, ADSs centrado em processos têm provido também um poderoso meio de integrar processos e ferramentas, e de automatizar, pelo menos parcialmente, tarefas.

Esta dissertação apresenta uma infra-estrutura de processos de software que define uma ferramenta de definição de processos construída e integrada ao ambiente ODE (*Ontology-based software Development Environment*). Para apoiar essa integração, a infra-estrutura proposta foi construída tomando por base uma ontologia de processos de software. Essa ontologia foi evoluída no contexto deste trabalho, procurando capturar conceituações importantes definidas em normas e modelos de qualidade de processo recentes.

Palavras-chave: Processos de Software, Modelos de Qualidade de Processos, Ambientes de Desenvolvimento de Software, Ontologias.

ABSTRACT

Nowadays, it is widely recognized that the quality of a software product depends on the quality of the software processes used in its development and maintenance. With this, much work has been done aiming to support organizations in their efforts towards process quality. The software process research area has explored two main directions: (i) approaches for modeling, analyzing and improving software processes, and (ii) support technology for software processes.

The first goal focuses on approaches for structuring, organizing, documenting and describing software processes and includes process quality standards and maturity models. The second goal is concerned with the development of Process-Centered Software Engineering Environments (PSEEs) that integrate tool support for the development of artifacts with tool support for software process modeling and execution. The explicit representation of processes, its products and its interactions are the basis over which the modern development environments are built. Providing more powerful ways of describing and implementing software processes, PSEEs have also provided a powerful way to integrate processes and tools, and to automate, at least partially, tasks.

This work presents a software process infrastructure that includes a tool for defining software processes. This tool was built and integrated to ODE (Ontology-based software Development Environment). To support this integration, the proposed infrastructure was built based on a software process ontology. This ontology was evolved in the context of this work, taking into account recent quality models and standards.

Keywords: Software Process, Process Quality Models, Software Engineering Environments, Ontologies.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução

Com a crescente demanda no setor de desenvolvimento de software, aumenta a necessidade de se dispor de ambientes automatizados que apóiem todo o processo de desenvolvimento. Essa necessidade se deve também à grande complexidade dos sistemas atuais, à expectativa de que eles tenham alta qualidade e que sejam desenvolvidos respeitando os prazos estabelecidos, sem a necessidade de alocação de mais recursos.

Ambientes de Desenvolvimento de Software (ADSs) surgem, então, com o intuito de integrar diferentes ferramentas, construídas para finalidades específicas, mas que operam juntas para apoiar a construção de um produto de software. Tal integração é fundamental para os ADSs e envolve diversas dimensões, tais como apresentação, dados, plataforma, processo, controle e conhecimento (PFLEEGER, 2001).

Contudo, apenas automatizar o desenvolvimento não é suficiente para atingir todos os critérios de qualidade necessários no desenvolvimento de software. Uma outra vertente de evolução é o próprio domínio de processos de software. Nos últimos anos, diversas normas de qualidade de software foram revisadas, tais como a ISO 9001 (ISO, 2000a) e o CMM (PAULK, 1993), e até mesmo elaboradas, como foi o caso da ISO/IEC 15504 (ISO/IEC, 2003). Em geral, essas normas descrevem requisitos de processos para serem implementados nas organizações, além de estabelecerem critérios para avaliação da maturidade e capacidade das organizações em desenvolver e manter software. Essas abordagens tiveram uma grande aceitação do mercado e passaram a ser uma ferramenta, de fato, para avaliar o nível da qualidade de organizações desenvolvedoras de software.

Dessa forma, essas duas vertentes se completam, pois estabelecem uma forma de definir processos de qualidade e automatizar esses processos, buscando garantir eficiência e rapidez no desenvolvimento.

Um problema quando se utiliza diversas normas e modelos de qualidade de processo distintos é que não há um vocabulário comum compartilhado por eles. Cada norma, família de

normas ou modelo define os seus próprios conceitos e trabalha de forma independente. Neste contexto, ontologias tornam-se muito úteis, pois podem ser usadas como uma estrutura unificadora para dar semântica e uma representação comum à informação (FALBO et al., 2004a), podendo servir como uma inter-língua para compartilhamento de informação.

Além disso, ontologias podem ser úteis na construção dos ambientes de desenvolvimento de software. Ontologias são vistas como uma boa alternativa para ajudar na solução do problema da integração, pois padronizam e melhoram a comunicação entre desenvolvedores e ferramentas em um ADS, evitando problemas de interpretação e ajudando a minimizar um dos maiores problemas recorrentes em ADSs - a integração de ferramentas.

1.1 Contexto e Objetivo do Trabalho

Como a integração de ferramentas ainda é um dos maiores desafios na construção de Ambientes de Desenvolvimento de Software (ADS), o uso de ontologias pode trazer muitos benefícios. A partir dessa premissa, foi desenvolvido o ambiente ODE (*Ontology based software Development Environment*) (FALBO et al., 2003), um ADS baseado em ontologias. Essa base ontológica facilita a integração de ferramentas, uma vez que os conceitos das ontologias estão formalmente definidos e são compartilhados entre as ferramentas integradas ao ambiente.

A principal contribuição das ontologias para ADSs é prover um conjunto de conceitos e restrições bem definidos, determinando um vocabulário comum que pode ser compartilhado por pessoas e ferramentas. A premissa de ODE é que, se as ferramentas são construídas baseadas em ontologias, a sua integração é facilitada.

Por ser um ADS centrado em processo e baseado em ontologias, a ontologia de processo de software é a base ontológica principal do ambiente. Elaborada originalmente em (FALBO, 1998), essa ontologia trata dos principais conceitos envolvidos em processos de software e foi a base sobre a qual definiu-se a infra-estrutura de processos de software de ODE, incluindo a definição e a execução de processos (RUY et al., 2004).

Contudo, como recentemente o domínio de processos de software evoluiu bastante, era preciso que a ontologia também evoluísse. A partir do uso de ODE em organizações desenvolvedoras de software, alguns aspectos práticos foram levantados para serem incorporados à ontologia. Em sua primeira versão, a ontologia de processo de software não

capturava conceitos como composição de processos, interação de processos e níveis de abstração de processos, que são essenciais na modelagem de processos.

O objetivo principal deste trabalho é evoluir a infra-estrutura de processos de software de ODE e, por conseguinte, dado que essa infra-estrutura é construída sobre a base ontológica da ontologia de processo de software, evoluir essa ontologia. Assim, a ontologia de processo de software foi revisada para se adequar aos novos conceitos acerca do domínio de processo de software. Para tal, foi necessário estudar o estado da arte e o estado da prática de processos de software e propor uma revisão na ontologia de forma a acompanhar esse novo contexto.

Após a revisão da ontologia de processo de software, torna-se necessário evoluir também a infra-estrutura de processo de software do ambiente, de forma a incorporar os novos conceitos, relações e restrições descritos na ontologia. Assim, a ferramenta de definição de processos de software de ODE foi revisada, de forma a atender à nova conceituação. Essa ferramenta foi desenvolvida já integrada a ODE e estabelece o ponto de partida para a execução de projetos no ambiente.

1.2 Metodologia

Este trabalho teve início com uma revisão bibliográfica sobre os principais temas que o compõem. Como fonte de pesquisa foram utilizados artigos científicos, relatórios técnicos, livros e trabalhos acadêmicos que tratam sobre Ontologias, Ambientes de Desenvolvimento de Software, Processo de Software e Qualidade de Software.

A partir desse estudo, constatou-se a evolução de diversos conceitos relacionados com o domínio de processos de software. Dessa forma, a Ontologia de Processo de Software, publicada originalmente em (FALBO, 1998) e que compõe a base de ontológica de ODE, teve que ser revisada para incluir e adequar-se a novos conceitos.

Um dos objetivos da ontologia é determinar um vocabulário comum acerca de um domínio, descrevendo a conceituação que envolve esse domínio. Dessa forma, para avaliar os conceitos da ontologia e sua aderência à literatura sobre processos de software, foi feito um mapeamento dos conceitos da ontologia com os conceitos envolvidos nas principais normas e modelos de qualidade de software existentes.

O próximo passo foi avaliar a infra-estrutura de processos de software de ODE, analisando o impacto que a revisão da ontologia teria na abordagem de processos de software

adotada em ODE. Outro insumo para a evolução da infra-estrutura de processos de ODE foi a implantação, em caráter experimental, desse ambiente em uma organização de desenvolvimento de software. Do feedback dado por ela, verificou-se a necessidade de incorporar novas funcionalidades. Assim, a ferramenta de definição de processos do ambiente (BERTOLLO e FALBO, 2003) teve que ser revisada para atender aos novos conceitos da ontologia e aos novos requisitos identificados para a infra-estrutura de processos de ODE.

O texto da dissertação foi elaborado à medida que o trabalho avançava, documentando os resultados dos estudos realizados, as técnicas utilizadas e as soluções adotadas, servindo inclusive de base para outros trabalhos.

Parte do trabalho realizado foi publicado nos anais do *I Workshop on Vocabularies, Ontologies and Rules for the Enterprise* (VORTE'2005), realizado em Enschede, Holanda, em setembro de 2005, no artigo intitulado "*Establishing a Common Vocabulary for Software Organizations Understand Software Processes*" (FALBO e BERTOLLO, 2005). Esse trabalho foi convidado para ser submetido em versão estendida para a revista *International Journal of Business Process Integration and Management,* tendo sido submetido sob o título "*A Software Process Ontology as a Common Vocabulary about Software Processes*", em março de 2006. Outro artigo oriundo desta trabalho, intitulado "Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias" (BERTOLLO et al., 2006), foi aceito para publicação nos anais do V Simpósio Brasileiro de Qualidade de Software (SBQS'2006).

1.3 Organização do Trabalho

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

O Capítulo 2 – *Processos de Software* – apresenta os conceitos do domínio de processos de software, descrevendo uma abordagem para tratamento de processos de software em diferentes níveis de abstração. Além disso, discute os principais aspectos relacionados à Qualidade de Software, apresentando as principais normas e modelos de qualidade de processo. Nesse capítulo também é discutida a automatização do processo de software por meio dos Ambientes de Desenvolvimento de Software e é apresentado ODE, ambiente no qual está inserido este trabalho.

No Capítulo 3 – *Evoluindo uma Ontologia de Processo de Software* – são apresentados os novos conceitos, relações e restrições envolvidos na evolução da ontologia de processo de software descrita em (FALBO, 1998). As sub-ontologias que compõem a ontologia de processo de software também foram alteradas e estão documentadas nesse capítulo. A evolução dessas ontologias foi feita seguindo a abordagem proposta pelo método SABiO (*Systematic Approach for Building Ontologies*) (FALBO, 1998) (FALBO, 2004), que utiliza um perfil UML para modelagem dos conceitos, além de axiomas escritos em lógica de primeira ordem para formalização.

O Capítulo 4 – *Uma Ferramenta de Apoio à Definição de Processos em ODE* – discute a infra-estrutura de processo de software de ODE e apresenta a revisão da ferramenta de apoio à definição de processos do ambiente. São descritas as alterações da ferramenta em relação à versão anterior, motivadas, principalmente, pela evolução da ontologia de processo de software descrita no capítulo anterior.

Finalmente, o Capítulo 5 – *Conclusões e Perspectivas Futuras* – contém as considerações finais sobre o trabalho aqui desenvolvido, apresentando suas contribuições e propostas para trabalhos futuros.

Capítulo 2

Processos de Software

Um dos principais objetivos da Engenharia de Software é desenvolver sistemas com qualidade, dentro dos prazos estabelecidos e sem a necessidade de alocação de mais recursos. Para que tal objetivo seja alcançado, o foco não deve estar apenas nos produtos gerados, mas também no seu processo de desenvolvimento. A qualidade de um produto de software depende fortemente da qualidade do processo de software utilizado em seu desenvolvimento (FUGGETTA, 2000). Assim, a definição de um processo de software é um requisito básico para se chegar a produtos de qualidade.

Porém, a definição de um processo de software não é uma tarefa trivial. Processos devem ser definidos caso a caso, levando-se em consideração as especificidades do projeto em questão. Deve-se considerar a adequação às tecnologias envolvidas, ao tipo de software em questão, ao domínio de aplicação, ao grau de maturidade (ou capacitação) da equipe em engenharia de software, às características próprias da organização e às características do projeto e do grupo de desenvolvimento (ROCHA et al., 2001).

Embora diferentes projetos requeiram processos com características específicas para atender às suas particularidades, é possível estabelecer um conjunto de ativos de processo de software (*software process assets*) a ser utilizado na definição de processos de software de uma organização. Essas coleções de ativos de processo constituem os chamados processos padrão de software. Processos para projetos específicos podem, então, ser definidos a partir da instanciação de um processo de software padrão da organização, levando em consideração suas características particulares.

Essa necessidade de estruturação, organização, documentação e descrição padronizada de processos de software vem se tornando prática de mercado, sendo advogada por diversas normas e modelos de qualidade, tais como ISO/IEC 12207 (ISO, 1995), CMMI (CHRISSIS et al., 2003) e o MPS.BR (SOFTEX, 2005).

À medida que aumenta a complexidade dos sistemas a serem desenvolvidos, o processo de software torna-se também mais complexo e cresce a necessidade de automatizá-

lo, através da utilização de ferramentas. Porém, ferramentas isoladas podem oferecer apenas soluções parciais, enquanto o que se deseja é utilizar ferramentas de apoio ao longo de todo o processo de desenvolvimento de software.

Surgem, então, os Ambientes de Desenvolvimento de Software (ADSs), que podem ser descritos como coleções de ferramentas integradas que facilitam as atividades da engenharia de software, durante todo o processo ou pelo menos em porções significativas dele (HARRISON et al., 2000).

Uma evolução desses ambientes são os ADSs Centrados em Processo, que integram ferramentas de apoio ao desenvolvimento de artefatos com ferramentas de apoio à modelagem e execução de processos de software, utilizadas na construção desses artefatos (HARRISON et al., 2000). A representação explícita de processos, seus produtos e suas interações é a base sobre a qual os modernos ambientes de desenvolvimento são construídos. Provendo formas mais poderosas de descrever e implementar processos de software, ADSs centrados em processo têm provido também um poderoso meio de integrar processos e ferramentas, e de automatizar, pelo menos parcialmente, tarefas (HARRISON et al., 2000).

Este capítulo procura dar uma visão geral dos conceitos relativos a processos de software. A seção 2.1 discute aspectos relacionados à definição de processos de software. Na seção 2.2 é discutida a qualidade de processos de software, apresentando os principais modelos e normas de qualidade de processo. Na seção 2.3, o enfoque está na automatização de processos de software, sendo discutidos os ambientes de desenvolvimento de software e a evolução por que tem passado essa classe de sistemas. Apresenta, ainda, o ambiente ODE (*Ontology-based software Development Environment*) (FALBO et al., 2003), ambiente no qual este trabalho foi desenvolvido. Finalmente, a seção 2.4 apresenta as conclusões deste capítulo.

2.1 Definição de Processos 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

requeridos, os artefatos consumidos e produzidos e os procedimentos a serem adotados (métodos, técnicas, modelos de documentos, entre outros) (FALBO et al., 1998).

A área de processos de software, como uma disciplina autônoma, é relativamente nova, tendo surgido em meados da década de 80. Ao longo desses pouco mais de 20 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, 1995) e o modelo de maturidade CMMI (CHRISSIS et al., 2003).

Um elemento chave dessas normas e modelos de maturidade é a definição de um processo padrão descrevendo as atividades que devem ser realizadas em todos os projetos de software de uma organização, bem como os demais ativos de processo envolvidos, dentre eles artefatos, procedimentos, ferramentas e papéis. O uso de um processo padrão como base para o planejamento do processo de software específico de um projeto permite aos gerentes de projeto definir planos em conformidade com os padrões de qualidade e procedimentos da organização (BERGER, 2003).

Entretanto, esse tipo de esforço de padronização sofre com um problema: para acomodar todos os tipos de iniciativas de desenvolvimento em uma organização, o padrão vai inevitavelmente estar em um nível de abstração que atenda às necessidades de todos os projetos, mas não vai ser capaz de fornecer apoio específico às atividades individuais de um dado projeto.

De fato, nenhum projeto é igual ao outro e, portanto, para ser efetivo e conduzir a produtos de qualidade, um processo deve ser adequado às características específicas do projeto, considerando, dentre outros, o tipo de software a ser desenvolvido, o paradigma, o domínio de aplicação, características da equipe, tamanho, complexidade e criticalidade[1] do projeto. Assim, é necessária uma abordagem flexível e configurável para a definição de processos, de modo a facilitar a adaptação de processos padrão às necessidades específicas de cada projeto. A Figura 2.1 mostra esquematicamente essa abordagem (ROCHA et al., 2001).

[1] Quão crítico é software desenvolvido para o domínio de aplicação.

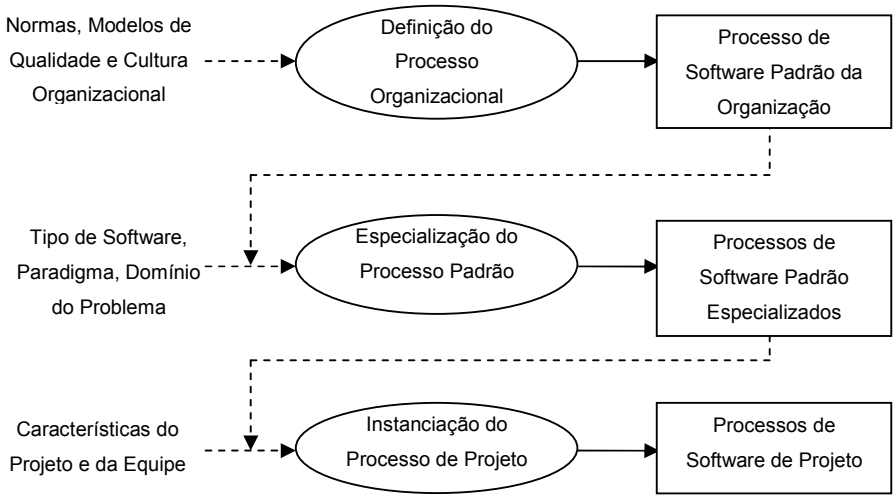


Figura 2.1 – Abordagem em Níveis para a Definição de Processos.

Em uma abordagem de definição de processos em níveis, inicialmente 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. Idealmente, o processo padrão da organização deve ser definido considerando padrões e modelos de qualidade, como CMMI e ISO/IEC 12207. Além disso, a cultura organizacional e a prática da organização em engenharia de software influenciam diretamente na definição do seu processo de software padrão.

Conforme citado anteriormente, para acomodar todos os projetos, esse processo estará em um nível de abstração muito alto, normalmente, ainda dando muito trabalho para que um gerente de projeto o adapte para seu projeto específico. Mas 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 - OO). 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 processo podem ser adicionados ou modificados de acordo com o contexto da especialização.

Por último, o processo padrão especializado mais indicado pode ser instanciado para um projeto específico. Durante a instanciação do processo de projeto, particularidades desse projeto e características da equipe devem ser levados em conta. 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 adicionados.

É importante ressaltar que estabelecer um processo bem definido para um projeto de software não garante a qualidade do produto final, contudo aumenta consideravelmente a probabilidade de se obter, ao término de sua execução, um produto de software que atenda aos requisitos levantados. Conhecer o processo significa conhecer como os produtos são planejados e produzidos. A partir da definição de processo, é possível fazer medições e coletar dados de execução, dando visibilidade aos gerentes do andamento dos projetos.

2.2 Qualidade de Processos de Software

Organizações bem sucedidas melhoram continuamente seus processos. Com a definição de um processo padrão organizacional, a melhoria sistemática dos processos é mais efetiva e eficiente se guiada por normas e modelos de qualidade de processo. A finalidade da maioria desses padrões é ajudar as organizações de software a conseguir a excelência seguindo processos e atividades adotados por organizações com alto grau de maturidade. Contudo, não é fácil selecionar os padrões apropriados. Existem muitas opções com uma grande sobreposição entre elas. Muitas vezes, é vantagem para uma organização de software usar ou implementar mais de um padrão ao mesmo tempo. Nessa situação, é melhor implementá-los simultaneamente. Tal abordagem permite aos engenheiros de processo maximizar os pontos fortes de um padrão e diminuir suas fraquezas (MUTAFELIJA, STROMBERG, 2003). A seguir, é apresentado um breve resumo das principais normas e modelos de qualidade, a saber: ISO 9000, ISO/IEC 12207, ISO/IEC 15504, CMMI e MPS.BR. Além desses modelos e normas, o modelo do Processo Unificado da Rational (*Rational Unified Process* – RUP) é também descrito, dada a sua ampla disseminação e aceitação.

2.2.1 ISO 9000

Em 2000, uma revisão da família de normas ISO 9000 – Sistemas de gestão da qualidade – foi publicada, substituindo a versão anterior, cuja versão em português da ABNT

datava de 1994. Algumas normas da família ISO 9000 foram alteradas, outras canceladas e novas criadas, ficando a ISO 9000 resumida a quatro normas (ISO, 2000b):

- ISO 9000 – Fundamentos e Vocabulário (ISO, 2000b);
- ISO 9001 – Requisitos para sistemas de gestão da qualidade (ISO, 2000a);
- ISO 9004 – Diretrizes para Melhoria de Desempenho (ISO, 2000c).
- ISO 19011 – Diretrizes sobre auditoria de sistemas de gestão da qualidade e ambiental

A principal alteração ocorreu na abordagem adotada: mudou-se de uma perspectiva prescritiva baseada em procedimentos, para práticas de gestão da qualidade baseadas em uma abordagem de engenharia de sistemas, orientada a processos, buscando a satisfação do cliente e a melhoria contínua (MUTAFELIJA, STROMBERG, 2003).

A norma ISO 9001:2000 é a única certificadora, ou seja, as organizações podem obter um certificado de sistema de gestão da qualidade se implementarem seus requisitos. Seu objetivo é estabelecer requisitos genéricos para o sistema de gestão da qualidade, sendo aplicáveis a todas organizações, sem considerar tipo, tamanho ou produto fornecido. O padrão é baseado em princípios de gestão da qualidade. Esses princípios não estão explicitamente na parte normativa da norma, mas são rastreáveis em seus principais requisitos. Os princípios são (ISO, 2000a): foco no cliente, liderança, envolvimento das pessoas, abordagem de processo, abordagem sistêmica para a gestão, melhoria contínua, abordagem factual para tomada de decisão e benefícios mútuos na relação com os fornecedores.

Organizações desenvolvem produtos e prestam serviços através de uma sistemática definida. A complexidade desses produtos e serviços pode requerer processos interdisciplinares que transformam requisitos do cliente, entradas e restrições em um produto, ou mais genericamente, em uma solução. Esses processos inter-relacionados formam um sistema, definido como uma combinação de elementos (humanos, software e hardware) e processos (facilidades, equipamentos, material e procedimentos) que são relacionados para satisfazer necessidades através de um ciclo de vida sustentável do produto (IEEE, 1998).

A mudança de foco na nova versão da norma teve o objetivo de encorajar a adoção da abordagem de processo para a gerência de uma organização. É natural que diferentes processos possam interagir e os mesmos devem ser gerenciados coletivamente para atingir os objetivos organizacionais. Para funcionarem de forma eficaz, as organizações têm que identificar e gerenciar processos inter-relacionados.

2.2.2 ISO/IEC 12207

A norma NBR ISO/IEC 12207 – Tecnologia da Informação – Processos de Ciclo de Vida de Software (ISO/IEC, 1995) estabelece uma estrutura comum para os processos de ciclo de vida de software, com terminologia bem definida, que pode ser referenciada pela indústria de software. A estrutura contém processos, atividades e tarefas que devem ser aplicados na aquisição, fornecimento, desenvolvimento, operação e manutenção de produtos de software. Esse conjunto de processos, atividades e tarefas foi projetado para ser adaptado, de acordo com as características específicas, a uma organização, projeto ou aplicação, o que pode envolver o detalhamento, a adição e a supressão de elementos de acordo com o objetivo de utilização.

Seu modelo de referência de processo fornece indicações de processos, descritos em termos de seus propósitos e resultados esperados, em conjunto com uma arquitetura que descreve as relações entre esses processos. Define, ainda, atividades e tarefas requeridas para implementar em alto nível tais processos, objetivando atingir a capacidade desejada para compradores, fornecedores, desenvolvedores, mantenedores e operadores de sistemas que contêm software.

Na ISO/IEC 12207 são consideradas três categorias de processos: Fundamentais, de Apoio e Organizacionais. Os processos fundamentais constituem um conjunto de cinco processos que atendem às partes fundamentais (pessoa ou organização) durante o ciclo de vida do software. Os processos fundamentais são: Processo de Aquisição, Processo de Fornecimento, Processo de Desenvolvimento, Processo de Operação e Processo de Manutenção. Os processos de apoio auxiliam um outro processo como parte integrante, com um propósito distinto, contribuindo para o sucesso e qualidade do projeto de software. Um processo de apoio é empregado por outro processo, quando necessário. São eles: Processo de Documentação, Processo de Gerência de Configuração, Processo de Garantia da Qualidade, Processo de Verificação, Processo de Validação, Processo de Revisão Conjunta, Processo de Auditoria e Processo de Resolução de Problemas. Os processos organizacionais são empregados por uma organização para estabelecer e implementar uma estrutura subjacente de forma a melhorar continuamente os seus processos. São empregados tipicamente fora do domínio de projetos e contratos específicos; entretanto, ensinamentos desses projetos e

contratos contribuem para a melhoria da organização. São eles: Processo de Gerência, Processo de Infra-estrutura, Processo de Melhoria e Processo de Treinamento .

A norma descreve a arquitetura dos processos de ciclo de vida de software, mas não representa uma abordagem de implementação particular, nem prescreve um modelo de ciclo de vida, metodologia ou técnica específica. O modelo de referência tem o objetivo de ser adaptado para uma organização, considerando suas necessidades de negócio e domínios de aplicação.

A ISO/IEC 12207 foi originalmente publicada em 1995 como uma norma internacional. Em 1998, foi publicada a sua versão brasileira que sustenta o mesmo nome que a internacional, acrescida das iniciais NBR. Em 2002 e 2004, foram feitas atualizações, chamadas de emendas 1 e 2 respectivamente, quando foram inseridas algumas melhorias. Essas melhorias criaram novos ou expandiram o escopo de alguns processos, inseriram para cada processo o seu propósito e resultados esperados e para os novos processos definiram suas atividades e tarefas. Essas modificações têm o objetivo de representar a evolução da área de processos de software, as necessidades vivenciadas pelos usuários da norma e sua harmonização com a série ISO/IEC 15504 - Avaliação de Processos de Software.

2.2.3 ISO/IEC 15504

A norma ISO/IEC 15504 foi publicada pela primeira vez em 1998 com a designação de Relatório Técnico (ISO/IEC TR 15504) (ISO/IEC, 1998). Essa designação é atribuída quando o assunto está ainda em desenvolvimento técnico ou quando existe a possibilidade de se alcançar um entendimento para uma Norma Internacional. Em 2003, a ISO/IEC 15504 (ISO/IEC, 2003) foi publicada como norma, sendo organizada em cinco partes sob o título genérico Tecnologias de Informação – Avaliação de Processos. As partes que a compõem são (ISO/IEC, 2003):

- Parte 1: Conceitos e Vocabulário (Informativa) – estabelece uma introdução geral sobre os conceitos de avaliação de processos e um glossário para os termos relacionados;
- Parte 2: Execução de uma avaliação (Normativo) – define os requisitos mínimos para execução de uma avaliação que garanta consistência e repetição dos processos;

- Parte 3: Guia para execução de uma avaliação (Informativo) – estabelece uma interpretação dos requisitos para execução de uma avaliação;
- Parte 4: Guia para utilização em processos de melhoria e na determinação da capacidade de processos (Informativo) – identifica a avaliação de processos como uma atividade a ser executada como parte de uma iniciativa de melhoria de processos ou como parte de uma abordagem de determinação de capacidade.
- Parte 5: Um exemplo de um modelo de avaliação de processos (Informativo) – contém um exemplo de um Modelo de Avaliação de Processo baseado no Modelo de Referência de Processo definido na ISO/IEC 12207, emendas 1 e 2.

A avaliação de processos é baseada num modelo bi-dimensional, que contém uma Dimensão de Processo e uma Dimensão de Capacidade. A Dimensão de Processo é constituída por processos de ciclo de vida definidos no Modelo de Referência de Processos, o qual define um conjunto de processos caracterizados por uma descrição dos objetivos e dos resultados. Um exemplo de modelo de referência é derivado diretamente da norma ISO/IEC 12207 emendas 1 e 2. Os processos encontram-se agrupados em nove categorias: Aquisição, Fornecimento, Engenharia, Operação, Apoio, Gerência, Melhoria de Processo, Recurso e Infra-estrutura, e Reuso.

A norma permite, ainda, que os modelos de referência de processos possam ser desenvolvidos fora do contexto da normalização ISO, desde que preencham um conjunto de requisitos de conformidade pré-estabelecidos. O patrocinador deve determinar qual modelo de referência de processo melhor atende o requisito especificado ou objetivo de negócio definido, seguindo o guia ISO/IEC 15504-3 para seleção.

A Dimensão de Capacidade consiste numa plataforma de medição composta por seis níveis de capacidade e pelos atributos de processo associados a cada nível. O resultado da avaliação consiste na determinação do perfil de cada um dos processos através da classificação dos respectivos atributos. A avaliação pode incluir, ainda, o nível de capacidade atingido por cada processo (determinado a partir dos atributos). Os níveis de capacidade, ordenados de forma crescente de 0 (zero) a 5 (cinco), são: Incompleto, Executado, Gerenciado, Estabelecido, Previsível e Otimizado. Os atributos de processo são:

- Nível 1 – Execução de Processo;
- Nível 2 – Gerência da Execução e Gerência de Produtos de Trabalho;

- Nível 3 – Definição de Processo e Implementação de Processo;
- Nível 4 – Medição de Processo e Controle de Processo;
- Nível 5 – Inovação de Processo e Otimização de Processo.

2.2.4 CMMI

Em 1995, o *Software Engineering Institute* (SEI) criou um modelo de maturidade e capacidade para organizações de software, o SW-CMM. O modelo contém elementos essenciais para um processo efetivo em diversas disciplinas e descreve um caminho para as organizações evoluírem seus processos de um nível imaturo para um nível disciplinado. Assim como foi elaborado um modelo para software, diversas outras disciplinas criaram os seus modelos de maturidade cobrindo outras áreas de interesse, tais como o SA-CMM (Modelo de Maturidade e Capacidade para Aquisição de Software) e o SE-CMM (Modelo de Maturidade e Capacidade para Engenharia de Sistemas).

Embora esses modelos tenham sido bastante úteis para muitas organizações, o uso de múltiplos modelos tornou-se problemático. Por não serem desenvolvidos de forma integrada, diferenças na arquitetura, conteúdo e abordagem aumentaram os custos de treinamento, avaliações e atividades de melhoria.

Por estas razões, o SEI elaborou o CMMI (*CMM Integration*) (CHRISSIS et al., 2003) com o objetivo de apoiar a melhoria de processos e produtos, diminuindo a redundância e eliminando a inconsistência que surge ao se utilizar diversos modelos independentes. Esse objetivo é atingido através da integração dos diversos CMMs numa estrutura única, todos com a mesma terminologia, processos de avaliação e estrutura. O projeto também se preocupou em tornar o CMM compatível com a norma ISO/IEC 15504, de modo que avaliações em um modelo sejam reconhecidas como equivalentes aos do outro (CHRISSIS et al., 2003).

O CMMI oferece duas abordagens diferentes para a melhoria de processos, sendo conhecidas por “modelo em estágios” e “modelo contínuo”. A representação por estágios prescreve a ordem para implementação de cada área de processo de acordo com níveis de maturidade, que definem um caminho de melhoria para uma organização do nível Inicial para o nível Otimizado. Atingindo um nível de maturidade, garante-se uma base adequada para buscar o próximo estágio.

A representação contínua oferece uma abordagem flexível para melhoria de processos. Uma organização pode escolher melhorar a qualidade de um processo específico ou trabalhar em diversas áreas de forma alinhada aos objetivos de negócio. Os níveis de capacidade são usados para medir uma área, indo de um processo não executado para um processo otimizado. Existem algumas limitações de escolha, devido à dependência entre as áreas de processos.

As duas representações contêm os mesmos componentes, usando os mesmos conceitos. A diferença entre as representações está na abordagem para melhoria de processos. Se uma organização planeja atingir uma maturidade organizacional, deve selecionar a representação por estágios. Se uma organização tem interesse na capacidade de um processo específico, deve selecionar a representação contínua.

O CMMI é estruturado em termos de áreas de processo (*Process Area* – PA), que consistem em práticas relacionadas que coletivamente satisfazem um conjunto de objetivos. Um objetivo genérico descreve a institucionalização requerida para atingir um nível de capacidade (representação contínua) ou de maturidade (representação por estágio). Cada objetivo genérico é associado a um conjunto de práticas genéricas que descrevem atividades requeridas para a institucionalização de processos em uma determinada área de processo. Cada área de processo contém, ainda, objetivos específicos e práticas específicas, que descrevem atividades essenciais no atendimento aos objetivos específicos. Os componentes do modelo CMMI podem, ainda, ser agrupados em três categorias: requerido, esperado e informativo. A Figura 2.2 apresenta esses componentes.

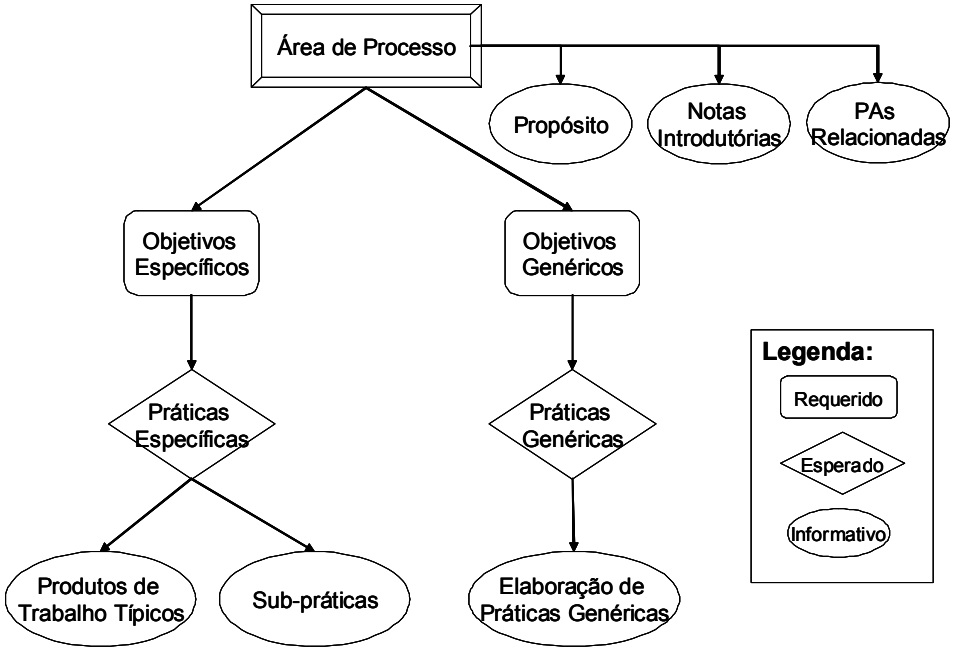


Figura 2.2 – Componentes do Modelo CMMI.

2.2.5 MPS.BR

O MPS.BR (Melhoria de Processo do Software Brasileiro) (SOFTEX, 2005) é um modelo para avaliação e melhoria do processo de software. A base técnica é composta pelas normas NBR ISO/IEC 12207 e suas emendas 1 e 2, e a ISO/IEC 15504, além de cobrir todo o conteúdo do modelo CMMI.

O MPS.BR baseia-se nos conceitos de maturidade e capacidade de processo para a avaliação e melhoria da qualidade e produtividade de produtos de software e serviços correlatos. Dentro desse contexto, o MPS.BR possui três componentes: Modelo de Referência (MR-MPS), que contém os requisitos a serem atendidos pelas organizações, bem como os níveis de maturidade e capacidade, e os processos; Modelo de Avaliação (MA-MPS), que contém o processo de avaliação, os requisitos para os avaliadores e os requisitos para averiguação da conformidade ao MR-MPS; Modelo de Negócio (MN-MPS), que contém as regras para implantação do MPS.BR pelas empresas de consultoria, de software e avaliação.

O MPS.BR é descrito através de documentos em formato de guias:

- **Guia Geral:** contém a descrição geral do MPS.BR e detalha o Modelo de Referência (MR-MPS), seus componentes e as definições comuns necessárias para seu entendimento e aplicação.

- **Guia de Aquisição:** contém recomendações para a condução de compras de software e serviços correlatos. Foi descrito como forma de apoiar as instituições que queiram adquirir produtos de software e serviços correlatos apoiando-se no MR-MPS.
- **Guia de Avaliação:** contém a descrição do processo de avaliação, os requisitos para o avaliador, os requisitos para a avaliação, o método e os formulários para apoiar a avaliação.

O Modelo de Referência MR-MPS define níveis de maturidade como uma combinação entre processos e capacidade de processos. Os níveis de maturidade estabelecem patamares de evolução de processo, caracterizando estágios de melhoria de implementação de processos na organização. O MR-MPS define sete níveis de maturidade: A (Em Otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F (Gerenciado) e G (Parcialmente Gerenciado). A escala de maturidade se inicia no nível G e progride até o nível A. Para cada um desses sete níveis de maturidade foi atribuído um perfil de processos e de capacidade de processos que indica onde a organização tem que colocar esforço para melhoria, de forma a atender os objetivos de negócio.

A capacidade do processo é um conjunto de atributos de processo descrito em termos de resultados. A capacidade estabelece o grau de refinamento e institucionalização com que o processo é executado na organização. À medida que evolui nos níveis, um maior ganho de capacidade para desempenhar o processo é atingido pela organização. A capacidade do processo possui cinco atributos de processo, que são detalhados, por sua vez, em termos dos resultados para alcance completo do atributo. Os atributos de processo do MPS.BR são:

- AP 1.1: O processo é executado;
- AP 2.1: O processo é gerenciado;
- AP 2.2: Os produtos de trabalho do processo são gerenciados;
- AP 3.1: O processo é definido e
- AP 3.2: O processo está implementado.

O progresso e a obtenção de um nível de maturidade ocorrem quando são atendidos todos os resultados e os propósitos dos processos e dos atributos de processo relacionados àquele nível e aos anteriores. A Tabela 2.1 apresenta os processos de cada nível de maturidade e a capacidade exigida.

Tabela 2.1 – Níveis de Maturidade.

Nível	Processo	Capacidade
A	Implantação de Inovações na Organização	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
	Análise de Causas e Resolução	
B	Desempenho do Processo Organizacional	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
	Gerência Quantitativa do Projeto	
C	Análise de Decisão e Resolução	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
	Gerência de Riscos	
D	Desenvolvimento de Requisitos	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
	Solução Técnica	
	Integração do Produto	
	Verificação	
	Validação	
E	Treinamento	AP 1.1, AP 2.1, AP 2.2, AP 3.1 e AP 3.2
	Definição do Processo Organizacional	
	Avaliação e Melhoria do Processo Organizacional	
	Adaptação do Processo para Gerência do Projeto	
F	Medição	AP 1.1, AP 2.1 e AP 2.2
	Gerência de Configuração	
	Aquisição	
	Garantia da Qualidade	
G	Gerência de Requisitos	AP 1.1 e AP 2.1
	Gerência do Projeto	

2.2.6 Processo Unificado da Rational

O Processo Unificado da Rational (*Rational Unified Process* – RUP) é um processo de engenharia de software que determina uma abordagem disciplinada para associar tarefas e responsabilidades com o desenvolvimento organizacional (KRUCHTEN, 1998). O RUP possui uma estrutura em duas dimensões. A primeira é expressa em termos de ciclos, fases, iterações e marcos; a segunda é descrita em termos de componentes de processo, atividades, fluxos de trabalho (*workflows*), artefatos e trabalhadores (*workers*).

Um trabalhador é um papel que um indivíduo ou um grupo de indivíduos representam em um projeto. Uma atividade é uma porção de trabalho que um indivíduo em um determinado papel pode executar. Atividades produzem artefatos e podem ser divididas em etapas. Um artefato é uma informação que é produzida, modificada e usada por um processo, e pode ser composto por outros artefatos. Artefatos são usados como insumo para trabalhadores executarem uma atividade e são o resultado ou produto de uma atividade. Finalmente, um fluxo de trabalho é uma seqüência de atividades que produz um resultado com valor observado.

Esses quatro elementos fundamentais – atividades, fluxos de trabalho, artefatos e trabalhadores – representam a espinha dorsal da estrutura do RUP. Mas outros elementos podem ser adicionados para tornar os processos mais fáceis de serem entendidos e utilizados. Esses elementos adicionais são:

- **Guias (*Guidelines*):** são regras, técnicas, recomendações ou heurísticas que descrevem como executar uma atividade ou uma etapa. Guias podem ser utilizados também para descrever uma técnica específica, avaliar a qualidade de um artefato ou revisar uma atividade. Organizações podem especializar um guia, como, por exemplo, para descrever uma técnica ou uma ferramenta em particular;
- **Modelos de Documento (*Templates*):** são modelos de artefatos. Um artefato pode estar associado a um ou mais modelo. Organizações podem customizar os modelos, incluindo logotipo, identificação de projeto ou alguma outra informação relevante para a organização;
- **Mentores de Ferramenta (*Tool Mentors*):** são usados para descrever um conhecimento adicional para a execução de uma atividade ou etapa usando uma ferramenta de software específica;

- **Conceitos:** são conceitos chaves, como iteração, fase e risco, usualmente definidos em fluxos de trabalho básicos para auxiliar seu entendimento.

2.3 Automatização de Processos de Software

Devido à influência do processo de software no desenvolvimento de produtos de software de qualidade, muito trabalho tem sido feito no sentido de apoiar organizações em seus esforços pela busca da qualidade de processo. Conforme apontado por ARBAOUI et al. (2002), as pesquisas na área de processos de software nos últimos anos têm explorado duas principais vertentes: (i) abordagens para modelagem, análise e melhoria do processo de software; (ii) tecnologia de apoio ao processo de software.

A primeira vertente focaliza abordagens para estruturação, organização, documentação e descrição de processos de software e inclui normas e modelos de qualidade de processo de software, tais como as apresentadas na seção anterior. A segunda vertente está voltada para o desenvolvimento de Ambientes de Desenvolvimento de Software (ADSs), que buscam combinar técnicas, métodos e ferramentas para apoiar o engenheiro de software na construção de produtos de software, abrangendo todas as atividades inerentes ao processo, tais como planejamento, gerência, desenvolvimento e controle da qualidade.

Os benefícios da utilização de ADSs incluem (PRESSMAN, 2005):

- Facilitar a transferência de informação entre ferramentas e, conseqüentemente, entre passos do processo de software;
- Reduzir os esforços para realizar atividades de gerência de configuração, garantia de qualidade e produção de documentação;
- Aumentar o controle do projeto, através de melhor planejamento, monitoramento e comunicação;
- Prover coordenação entre os recursos humanos que estão trabalhando em um grande projeto de software.

A integração de ferramentas ainda é um dos maiores desafios dos ADSs, pois demanda representação consistente da informação, interfaces padronizadas, significados homogêneos para a comunicação entre engenheiros de software e ferramentas, e uma efetiva abordagem que possibilite portar os ADSs para várias plataformas (PRESSMAN, 2005).

THOMAS e NEJMEH (1992), TRAVASSOS (1994), FALBO (1998) e PFLEEGER (2001), dentre outros, propõem que a integração de ferramentas em ADSs seja tratada através de uma infra-estrutura que contemple várias dimensões, dentre elas:

- **Integração de Dados:** trata de serviços de repositório de dados, provendo armazenamento e gerência de objetos, e integração de dados, permitindo o compartilhamento de informações entre as ferramentas;
- **Integração de Apresentação:** diz respeito a serviços de interface com o usuário. As interfaces de um ADS devem ser homogêneas e consistentes, permitindo que os desenvolvedores alternem entre as ferramentas que utilizam, sem mudanças substanciais de estilo;
- **Integração de Controle:** está relacionada aos serviços de gerência de funcionalidades e mensagens, permitindo uma ferramenta notificar ou iniciar uma ação em outra ferramenta, controlando os eventos ocorridos e compartilhando funcionalidades;
- **Integração de Processo:** é obtida pela ligação explícita entre ferramentas e o processo de software. Para integrar ferramentas, é preciso ter um foco forte no gerenciamento do processo de software. O processo deve definir que ferramentas um engenheiro de software pode utilizar e quando deverá ter acesso às mesmas, em função da atividade que está realizando;
- **Integração de Plataforma:** trata da independência da plataforma sobre a qual funcionará a ferramenta.
- **Integração de Conhecimento:** diz respeito aos serviços de gerência de conhecimento, permitindo capturar conhecimento durante os projetos de software e oferecer apoio baseado em gerência de conhecimento aos engenheiros de software durante a realização de atividades do processo.

Devido às dificuldades inerentes ao controle de processos de software, a integração de processo tem se mostrado tão importante que deu origem a uma nova classe de ambientes, os Ambientes de Desenvolvimento de Software Centrados em Processo (ADSCPs). ADSCPs integram ferramentas para apoiar o desenvolvimento do software e para apoiar a modelagem e a execução do processo de software utilizado para desenvolver este produto (HARRISON et al., 2000; FUGGETTA, 2000).

Um ADSCP explora uma definição explícita do processo de software e pode ser visto como a automatização parcial desse processo, incluindo mecanismos para (CHRISTIE, 1995): (i) guiar a seqüência de atividades definida; (ii) gerenciar os produtos que estão sendo desenvolvidos; (iii) executar ferramentas necessárias para a realização das atividades; (iv) permitir comunicação entre as pessoas; (v) colher dados de métricas automaticamente; (vi) reduzir erros humanos e (vii) prover controle do projeto à medida que este vai sendo executado.

Processos modelados em um ADSCP tipicamente descrevem as atividades a serem realizadas, os responsáveis por cada uma delas e as ferramentas de software a serem utilizadas. São criados para vários propósitos, dentre eles: documentação do processo de software, melhoria do processo de software, gerenciamento do fluxo de trabalho e automatização (GRUHN, 2002).

Desta forma, ao prover meios de descrever e implementar processos de engenharia de software, ADSCPs provêem um mecanismo de integração de processo e ferramentas e, parcialmente, de automatização de tarefas (HARRISON et al., 2000).

Alguns exemplos de ADSCPs citados em (HARRISON et al.,2000) incluem: *Adele, Argo, PCTE e SPADE*. Em (ARBAOUI et al., 2002) são citados *TEMPO* e *APEL* (construídos a partir do *Adele)*, *LEU*, *PEACE*, *PIE* e *OZ*. HOLZ et al. (2001) e RICHTER e MAURER (2003) enfocam o ADSCP *MILOS (Modeling Language and Operational Support for Software Processes)*, que provê meios para definir um modelo de processo genérico e configurar planos de projeto concretos baseados neste modelo.

No contexto deste trabalho, dois ADSCPs merecem destaque: o ambiente ODE (FALBO et al., 2003), no qual este trabalho foi desenvolvido, e a Estação TABA (BERGER, 2003) que adota uma filosofia de definição de processos correlata à apresentada neste trabalho. A seguir, esses dois ambientes são apresentados com maiores detalhes.

2.3.1 A Estação TABA

A Estação TABA, desenvolvida desde 1990 (ROCHA *et al.*, 1990), é um meta-ambiente capaz de gerar ambientes de desenvolvimento de software adequados às particularidades de organizações, processos de software e projetos específicos. Para tal, utiliza atualmente a abordagem de geração de ambientes mostrada na Figura 2.3.

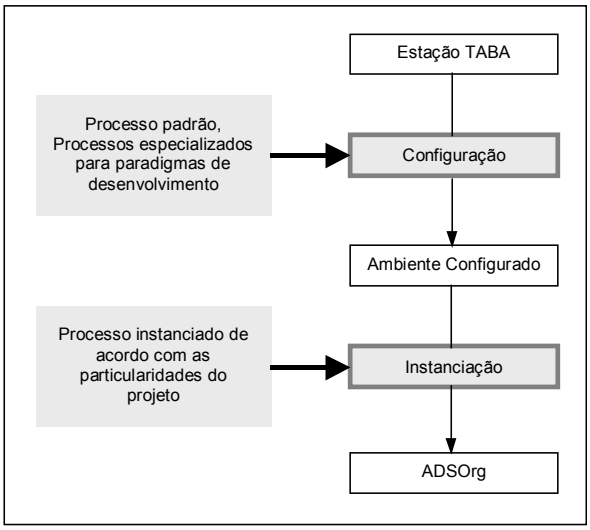


Figura 2.3 - Geração de ambientes a partir da Estação TABA (BERGER, 2003)

Inicialmente, é necessário criar um Ambiente Configurado a partir das características da organização e de sua maturidade em desenvolver software. O ambiente configurado contém o processo padrão da organização e os processos especializados para os paradigmas de desenvolvimento utilizados na organização. Esse ambiente armazena e provê conhecimento capturado pela organização, importante para o desenvolvimento e a manutenção de software.

Um ambiente configurado contém uma ferramenta para apoiar a instanciação de processos, que tem a capacidade de gerar, a partir dos processos instanciados, ambientes de desenvolvimento de software para projetos específicos, os ditos Ambientes de Desenvolvimento de Software Orientados a Organização (ADSOrg). Os ambientes instanciados são os ambientes que apóiam o desenvolvimento dos produtos de software e contêm outras ferramentas de apoio ao desenvolvimento (BERGER, 2003).

A Estação TABA é também um ADS Centrado em Processo, pois possibilita modelar explicitamente um processo de software e controlá-lo no ambiente instanciado. O ambiente possibilita, ainda, a gestão do conhecimento da organização, definindo um repositório de conhecimento que pode ser acessado pelos ADSOrg instanciados.

O processo de instanciação descreve as atividades necessárias para se realizar a adaptação (instanciação) de um processo especializado para um projeto específico, dentro do contexto de um Ambiente Configurado, a saber (BERGER, 2003):

Pré-Condição: Identificação de Atividades Obrigatórias

1. Caracterizar Projeto
    1.1 Definir Projeto
    1.2 Identificar Características do Projeto
2. Planejar Processo
    2.1 Incluir Atividades do Tipo de Software
    2.2 Definir Modelo de Ciclo de Vida
    2.3 Mapear Atividades do Modelo de Ciclo de Vida
        2.3.1 Verificar a Existência de Mapeamento Anterior
        2.3.2 Realizar Mapeamento
    2.4 Excluir Atividades Não Pertinentes
    2.5 Incluir Novas Atividades
    2.6 Selecionar Técnicas de Avaliação
    2.7 Selecionar Ferramentas
3. Instanciação de ADSOrg para o projeto

As atividades descritas são de responsabilidade do gerente do projeto. O resultado da instanciação é um processo instanciado que constitui o Plano do Processo de Desenvolvimento para um projeto específico. A cada novo projeto deve ser instanciado um novo ambiente (BERGER, 2003).

2.3.2 O Ambiente ODE

ODE (*Ontology-based software Development Environment*) (FALBO et al., 2003; FALBO et al., 2004a) é um Ambiente de Desenvolvimento de Software desenvolvido no Laboratório de Engenharia de Software da Universidade Federal do Espírito Santo (LabES/UFES), tendo por base ontologias e buscando tratar características de um ADS Semântico.

A característica marcante, que distingue ODE de outros ADSs, é o fato de ser desenvolvido baseado em ontologias. Uma ontologia é a especificação de uma conceituação (GRUBER, 1995), que inclui um vocabulário de termos e a especificação de seu significado. Inclui também definições e uma indicação de como os conceitos são inter-relacionados, que

coletivamente impõem a estrutura em um domínio e limita as possíveis interpretações dos termos (JASPER et al., 1999).

A premissa do Projeto ODE está baseada no argumento de que, se as ferramentas em um ADS são construídas baseadas em ontologias, a integração delas pode ser mais facilmente obtida. A mesma ontologia pode ser usada para construir diferentes ferramentas que apóiam atividades de engenharia de software correlacionadas. Além disso, se as ontologias são integradas, a integração de ferramentas construídas com base nelas pode ser bastante facilitada (FALBO et al., 2003). Assim, as ontologias nas quais ODE se baseia têm sido construídas procurando reutilizar conceituações previamente estabelecidas por outras ontologias já desenvolvidas no projeto, formando uma grande rede de conceitos. As ontologias que compõem a base ontológica de ODE são as seguintes:

- Ontologia de Processo de Software (FALBO e BERTOLLO, 2005): é a principal ontologia de ODE. Todas as demais utilizam conceitos dela. Dada a complexidade do domínio de processos de software, é dividida em sub-ontologias de atividade, recurso e procedimento. Sua primeira versão foi desenvolvida em (FALBO, 1998), tendo sido objeto de evolução neste trabalho, conforme discutido no capítulo 3. Serve de base para toda a infra-estrutura de processos de software do ambiente, incluindo as ferramentas de definição de processo e acompanhamento de projetos.
- Ontologia de Qualidade de Software (DUARTE e FALBO, 2000): trata de características de qualidade, métricas e medição de produtos e processos de software. A ferramenta de apoio ao controle da qualidade de ODE foi construída tomando-a por base.
- Ontologia de Artefatos de Software (NUNES, 2005): trata de alguns tipos de artefatos, a saber: documento, artefatos de código e diagramas, tendo uma sub-ontologia para cada um desses tipos. Uma ferramenta de apoio ao processo de documentação de software está sendo construída utilizando sua conceituação.
- Ontologia de Gerência de Configuração de Software (GCS) (NUNES, 2005): trata dos conceitos envolvidos no processo de GCS e é utilizada por outras ontologias quando a GCS é utilizada nos domínios das mesmas, como ocorre com artefatos, ferramentas e requisitos. O sistema de GCS de ODE foi construído tomando-a por base.

- Ontologia de Requisitos de Software (NARDI e FALBO, 2006): procura conceituar o que são requisitos de software e trata de tipos de requisitos, interesses e responsabilidades, rastreabilidade, dentre outros. Está sendo usada como base para a ferramenta de apoio à Engenharia de Requisitos de ODE.
- Ontologia de Riscos (FALBO et al., 2004b): trata dos conceitos envolvidos no processo de Gerência de Riscos e serve de base para a ferramenta que apóia esse processo.
- Ontologia de Organizações de Software (RUY, 2006): conceitua organizações de software e sua divisão em unidades organizacionais, tratando, ainda, de competências. Diversos serviços do ambiente estão sendo construídos com base nela, tais como alguns serviços de gerência de conhecimento e alocação de recursos.

Inicialmente, ODE era apenas um Ambiente de Desenvolvimento de Software Centrado em Processo. Posteriormente evoluiu para um Ambiente de Desenvolvimento de Software Orientado a Domínio (ADSOD), quando passou a considerar o conhecimento do domínio, para prover apoio de gerência de conhecimento. Este objetivo foi alcançado quando ODE passou a prover suporte ao desenvolvimento de ontologias de domínio, como forma de permitir a descrição do conhecimento sobre domínios de aplicação no ambiente, através de um editor de ontologias (MIAN et al., 2003).

Além do apoio à captura do conhecimento sobre o domínio, detectou-se a necessidade de apoiar a captura do conhecimento gerado nos processos de software. NATALI (2003) desenvolveu uma infra-estrutura de gerência de conhecimento que foi incorporada a ODE, tornando-o um ambiente capaz de prover aos desenvolvedores conhecimento acumulado no contexto do desenvolvimento e da manutenção de software. A infra-estrutura desenvolvida consiste de uma memória organizacional, que apóia o armazenamento e o compartilhamento do conhecimento, além de serviços de gerência de conhecimento, que provêem ativamente informações úteis a usuários que estejam trabalhando em atividades que exijam conhecimento.

Por ser baseado em ontologias, ODE realiza grande parte de suas tarefas utilizando e respeitando os modelos e restrições impostos pelas ontologias sobre as quais se fundamenta. Além disso, como ODE utiliza a tecnologia de objetos, é seguida uma abordagem sistemática que permite que os elementos definidos em uma ontologia (conceitos, relações, propriedades e

restrições definidas como axiomas) sejam mapeados para um modelo de objetos que é, então, integrado como parte fundamental da estrutura do ambiente (FALBO et al., 2004a).

Com o intuito de manter a amarração semântica entre os objetos de ODE e agregar ontologias ao ambiente, sua arquitetura conceitual[2] foi projetada em três níveis, como mostra a Figura 2.4, discutidos a seguir (FALBO et al., 2004a):

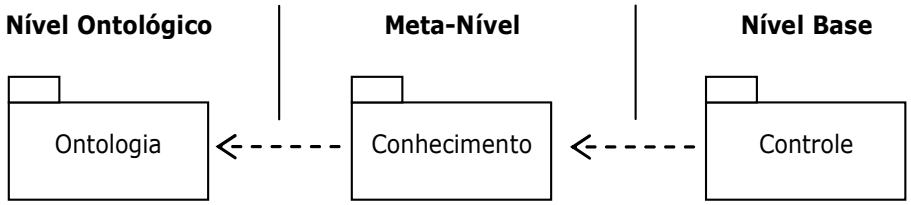


Figura 2.4 – Os três níveis da arquitetura conceitual de ODE.

- O Nível Ontológico, ou pacote *Ontologia*, é responsável pela descrição das ontologias de ODE. Nela encontram-se as classes que definem explicitamente a ontologia de um certo domínio e seus elementos, não sendo de sua responsabilidade prover detalhes de implementação de sistemas nesse domínio. Contudo, vale ressaltar que as instâncias do nível ontológico guiam a definição das classes dos outros dois níveis, originando as principais classes tanto do meta-nível quanto do nível base.
- O Meta-nível, ou pacote *Conhecimento*, abriga as classes que descrevem o conhecimento em relação a um domínio. Suas classes são derivadas das ontologias e as instâncias dessas classes atuam no papel de conhecimento sobre os objetos do nível base. Elas constituem o conhecimento do ambiente, que pode ser utilizado tanto pelo ambiente quanto pelas ferramentas que o compõem.
- O Nível Base, ou de Aplicação (pacote *Controle*), define as classes responsáveis por implementar as aplicações no contexto do ambiente (funcionalidades da infra-estrutura do ambiente e suas ferramentas). Essas classes também são derivadas das ontologias, mas tipicamente incorporam detalhes não descritos por elas, necessários para implementar as aplicações do ambiente. Muitas vezes, é necessária a criação de novas classes, associações, atributos e operações com o intuito de se tratar decisões específicas do nível de aplicação.

[2] Está-se usando o termo "arquitetura conceitual" para indicar uma decomposição de alto nível dos pacotes do ambiente, em contraste à arquitetura de software em camadas utilizada para implementá-lo efetivamente.

Essa arquitetura conceitual facilita o estabelecimento de uma correlação entre objetos dos diferentes níveis de ODE, permitindo anotar os objetos com informação semântica, dada efetivamente pelas ontologias, o que contribui para que ODE possa ser considerado um ADS Semântico (FALBO et al., 2005). Dessa forma, em diversas tarefas do ambiente, os objetos dos níveis base ou mesmo de conhecimento podem ter sua anotação ontológica utilizada para realização de tarefas de forma mais inteligente e consistente, segundo uma visão ontológica.

Conforme citado anteriormente, no mapeamento de ontologias em modelos de objetos, uma abordagem sistemática, que segue a filosofia proposta em (FALBO et al., 2002), é aplicada. Essa abordagem determina que os elementos definidos em uma ontologia (conceitos, relações, propriedades e restrições definidas como axiomas) devem ser mapeados em elementos de modelo de diagramas de classes (classes, associações, atributos e operações). Além disso, deve-se definir em que nível de ODE a correspondente classe será gerada. Na maioria das vezes, pelo menos uma classe, seja no meta-nível seja no nível base, é criada. Contudo, muitas vezes, classes nos dois níveis são necessárias, como é o caso, por exemplo, do conceito de atividade. Ela dá origem a uma classe no pacote *Conhecimento* para representar tipos de atividade (por exemplo, *Especificação de Requisitos*). Por outro lado, uma classe no nível base também é necessária para representar uma atividade específica, concreta (por exemplo, uma atividade de *Especificação de Requisitos Preliminar do Projeto X*).

ODE e suas ferramentas internas têm sido implementados em plataforma livre, utilizando Java como linguagem de programação. As informações ficam armazenadas em um banco de dados relacional – o PostgreSQL e, em alguns casos específicos, em arquivos XML.

2.4 Conclusões do Capítulo

Atualmente, é amplamente reconhecido que a qualidade dos produtos de software depende da qualidade dos processos de software utilizados em seu desenvolvimento e manutenção. Com isso, muito trabalho tem sido feito no sentido de apoiar organizações em seus esforços pela busca da qualidade de processo (FUGGETTA, 2000).

Neste contexto, cresce a demanda por automatizar as tarefas inerentes ao processo de software e surgem os Ambientes de Desenvolvimento de Software, com destaque para os

ambiente centrados em processo, tais como o ambiente ODE e a Estação TABA, discutidos na seção 2.3.

Acompanhando a evolução da área de processos de software, diversas normas e modelos de qualidade foram criados ou evoluídos, como é o caso da família de normas ISO 9000. De uma foram geral, essas normas e modelos de qualidade de processo advogam uma explícita abordagem de processos, implementando uma definição de processos em níveis. Uma organização deve determinar seus processos padrão, definindo uma sistemática a ser executada em quaisquer projetos. Contudo, as particularidades desses mesmos projetos devem ser consideradas, exigindo uma adaptação do processo padrão organizacional para as características específicas do projeto, como tamanho, complexidade, paradigma, experiência da equipe etc.

As normas e modelos de qualidade definem conceitos e abordagens que contribuem para a melhoria da maturidade e capacidade das organizações em desenvolver e manter software. Contudo, por serem elaboradas de forma independente, essas normas e modelos definem seus próprios termos, não compartilhando um vocabulário comum. Assim, visando a uma uniformização de conceitos, restrições e relações, uma ontologia de processo de software se faz necessária, tal como a ontologia descrita em (FALBO, 1998).

Para acompanhar a evolução da área de processos de software e das normas de qualidade, uma revisão nessa ontologia foi feita buscando contemplar a conceituação mais atual acerca do domínio de processos de software. Assim, o próximo capítulo apresenta a evolução da ontologia de processo de software e mapeia seus conceitos em termos dos conceitos definidos nas principais normas e modelos de qualidade apresentados neste capítulo.

Capítulo 3

Evoluindo uma Ontologia de Processo de Software

Diversos padrões de qualidade de processo, modelos de maturidade e modelos de processo, como os apresentados o Capítulo 2, são usados para guiar os esforços das organizações de software na busca por processos de software de qualidade. Contudo, o vocabulário usado por esses modelos e padrões é variado. Para tratar esse problema, FALBO (1998) desenvolveu uma ontologia de processo de software. Essa ontologia tinha o objetivo de apoiar a aquisição, a organização, o reúso e o compartilhamento de conhecimento sobre processos de software.

Contudo, com a evolução pela qual tem passado a área de processos de software nos últimos anos, notou-se que era necessário evoluir essa ontologia para que ela tivesse uma maior abrangência, incorporando novos conceitos, relações e restrições. Muitos trabalhos foram feitos na vertente de modelos de qualidade de processo, tais como a publicação da família de normas ISO/IEC 15504 (ISO, 2003), as emendas 1 (ISO, 2002) e 2 (ISO, 2004) à ISO/IEC 12207 (ISO, 1995), a evolução do modelo CMM (PAULK, 1993) para CMMI (CHRISSIS et al., 2003) e o desenvolvimento do Modelo de Melhoria do Processo de Software Brasileiro (MPS.BR) (SOFTEX, 2005). Assim, neste trabalho, foi feita uma evolução da ontologia proposta em (FALBO, 1998). Para tal, utilizou-se a versão mais recente do método para construção de ontologias utilizado originalmente, agora denominado SABiO (FALBO, 2004).

Visando a compatibilidade com a versão original, o trabalho de evolução da ontologia de processo de software adotou a mesma abordagem de dividir o domínio de processos de software em sub-domínios para atividades, recursos e procedimentos. Inicialmente, cada uma dessas sub-ontologias foi avaliada e evoluída. A seguir, evoluiu-se a ontologia de processo de software, usando as ontologias de seus sub-domínios de forma integrada.

Este capítulo está estruturado da seguinte forma: na seção 3.1 discute-se o que é uma ontologia e é apresentado o método SABiO, utilizado na evolução da ontologia de processo de software e suas sub-ontologias. A seção 3.2 apresenta a evolução das sub-ontologias que

compõem a ontologia de processo de software. Na seção 3.3 é apresentada a evolução da ontologia de processo de software. Na seção 3.4, é feito um mapeamento entre os termos definidos na ontologia de processo de software e os termos adotados nas principais normas e modelos de qualidade de processo, discutidos na seção 2.2. Com esse mapeamento, espera-se tornar mais fácil o uso da ontologia de processo como uma inter-língua para se falar sobre processos de software. Finalmente, na seção 3.5, são apresentadas as conclusões do capítulo.

3.1 Ontologias

De acordo com CHANDRASEKARAN et al. (1999), uma ontologia é a representação de uma conceituação capturada por um vocabulário, geralmente especializado para um domínio de aplicação. Ontologias são usadas para descrever compromissos ontológicos para um conjunto de agentes (humanos ou aplicações de software), ou seja, acordos para se utilizar um vocabulário compartilhado de forma coerente, de modo que esses agentes possam se comunicar sobre o domínio em questão.

Uma ontologia, como um artefato de engenharia, é constituída por um vocabulário que descreve uma certa realidade, mais um conjunto de suposições explícitas (axiomas formais) considerando o significado pretendido pelos termos do vocabulário (GUARINO, 1998).

Ontologias, como artefatos complexos, devem ser desenvolvidas seguindo métodos e técnicas apropriados. Para evoluir a ontologia de processo de software, neste trabalho foi utilizado SABiO (*Systematic Approach for Building Ontologies*) (FALBO et al., 1998; FALBO, 2004). A escolha de SABiO na formalização da revisão da ontologia se deu por esse método já ter sido utilizado na versão original da ontologia (apesar de ainda não ter esse nome) e por já termos um conhecimento prévio da metodologia. SABiO define as seguintes atividades:

- Identificação do Propósito e Especificação de Requisitos: tem por objetivo identificar claramente o propósito da ontologia e seu uso pretendido, isto é, a competência da ontologia.
- Captura da Ontologia: o objetivo é capturar a conceituação do domínio tomando por base a competência da ontologia. Os conceitos e relações relevantes devem ser identificados e organizados. Um modelo utilizando uma linguagem gráfica e um

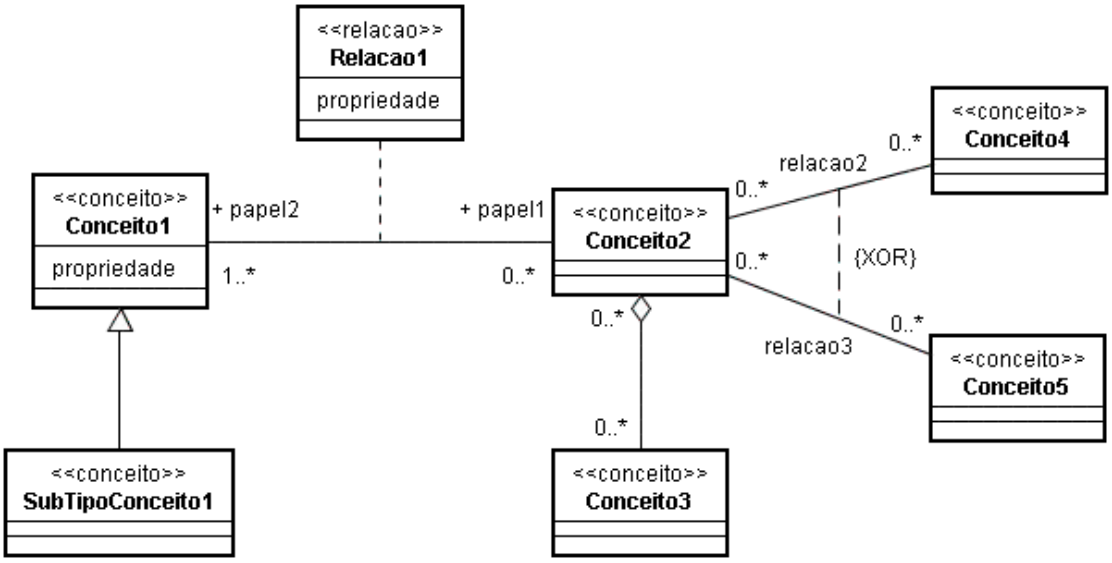
dicionário de termos devem ser produzidos para auxiliar a comunicação com os especialistas do domínio.

- Formalização da Ontologia: objetiva representar explicitamente a conceituação capturada no passo anterior em uma linguagem formal.
- Integração com Ontologias Existentes: durante a captura e a formalização da ontologia, pode ser necessário integrar a ontologia corrente com outras existentes, a fim de reutilizar conceituações previamente estabelecidas.
- Avaliação da Ontologia: a ontologia deve ser avaliada para verificar se satisfaz os requisitos especificados.
- Documentação: toda a ontologia definida deve ser documentada.

Na fase de especificação de requisitos, para estabelecer a competência da ontologia, SABiO utiliza questões de competência, ou seja, as questões que a ontologia deve ser capaz de responder (GRUNINGER e FOX, 1995).

Durante a captura da ontologia, uma linguagem gráfica é usada para facilitar a comunicação entre os engenheiros de ontologia e os especialistas de domínio. Em sua versão atual, SABiO propõe o uso de um perfil UML para a modelagem de ontologias (MIAN e FALBO, 2003). Esse perfil UML estende alguns elementos de modelo da UML, dando a eles a mesma semântica dos elementos de LINGO, a linguagem original proposta em (FALBO et al., 1998). Os elementos de modelo estendidos possuem uma teoria associada, dada por alguns axiomas formalmente definidos. Por exemplo, os axiomas (AE1) a (AE5) na Figura 3.1 são impostos pelas relações todo-parte e assume-se que são incorporados à ontologia quando uma notação de agregação da UML é usada.

A Figura 3.1 mostra um sumário do perfil UML usado para representar ontologias e alguns dos axiomas impostos pela notação correspondente. Quando alguma dessas notações for utilizada, entende-se que os axiomas correspondentes (ditos axiomas independentes de domínio) são incorporados, não sendo necessário escrevê-los explicitamente.



Axiomas:

Todo-parte:

(AE1) ($\forall$ x) $\neg$ parteDe (x,x)

(AE2) ($\forall$ x,y) parteDe (y,x) $\leftrightarrow$ todoDe (x,y)

(AE3) ($\forall$ x,y) parteDe (y,x) $\rightarrow \neg$ parteDe (x,y)

(AE4) ($\forall$ x,y) parteDe (y,x) $\rightarrow$ ($\exists$z) (parteDe (z,x))

(AE5) ($\forall$ x,y,z) parteDe (z,y) $\wedge$ parteDe (y,x) $\rightarrow$ parteDe (z,x)

Sub-tipo-de:

(AE6) ($\forall$ x,y,z) subTipoDe (x,y) $\wedge$ subTipoDe (y,z) $\rightarrow$ subTipoDe (x,z)

(AE7) ($\forall$ x,y) subTipoDe (x,y) $\rightarrow$ superTipoDe (y,x)

Ou-exclusivo (XOR):

(AE8) ($\forall$ a $\in$ C2) (($\exists$b) (b $\in$ C3) $\wedge$ R2(a,b)) $\rightarrow \neg$ (($\exists$c $\in$ C4) $\wedge$ R3(a,c)))

(AE9) ($\forall$ a $\in$ C2) (($\exists$c) (c $\in$ C4) $\wedge$ R3(a,c)) $\rightarrow \neg$ (($\exists$b $\in$ C3) $\wedge$ R2(a,b)))

Figura 3.1 – Perfil UML para construção de ontologias e seus axiomas.

Um modelo gráfico, associado com axiomas independentes de domínio, é útil, mas não é suficiente para capturar uma ontologia por completo. Além dos axiomas independentes de domínio, outros axiomas podem ser usados para capturar conhecimento. Esses axiomas, ditos axiomas dependentes de domínio, podem ser de dois tipos: axiomas de consolidação e

axiomas de derivação. O primeiro impõe restrições que precisam ser atendidas para que uma relação seja estabelecida consistentemente. O último pretende representar o conhecimento declarativo que pode derivar novo conhecimento a partir de conhecimento factual representado na ontologia, mas que não é capturado pela estruturação de conceitos e relações da ontologia.

Na avaliação da ontologia, SABiO sugere checar a ontologia contra suas questões de competência e verificar alguns critérios de qualidade, tais como os propostos por GRUBER (1995): clareza, coerência, extensibilidade, comprometimento de codificação mínimo e comprometimento ontológico mínimo.

3.2 Evoluindo as Sub-ontologias da Ontologia de Processo

Conforme apontado por FALBO (1998), analisando os elementos envolvidos em processos de software, pode-se observar que se trata de um domínio complexo, dificultando a construção de uma ontologia. Como premissa básica, é essencial seguir uma abordagem que foca no comprometimento ontológico mínimo[3] (GRUBER, 1995). Com base nessa abordagem, a ontologia deve descrever somente aspectos gerais, válidos para o domínio em estudo, contendo somente os elementos essenciais. Incluir muitos detalhes em uma ontologia pode torná-la específica demais e, portanto, menos reutilizável.

Contudo, mesmo considerando o critério de comprometimento ontológico mínimo, o domínio de processos de software é, ainda, extremamente complexo. Assim, na construção da versão original da ontologia de processo de software (FALBO, 1998), foi aplicado um mecanismo de decomposição, permitindo construir a ontologia em partes. A estratégia adotada foi definir sub-domínios do domínio de processos de software e construir sub-ontologias para cada sub-domínio. Uma vez definidas as sub-ontologias, elas foram usadas de forma integrada para estabelecer uma conceituação acerca de processos de software.

[3] Uma ontologia deve fazer tão poucas imposições quanto possível sobre o universo de discurso que está sendo modelado, permitindo que as partes comprometidas com a ontologia fiquem livres para especializá-la e instanciá-la na medida do necessário.

A Figura 3.2 mostra a relação entre a ontologia de processo de software e suas três principais sub-ontologias: ontologia de atividade, ontologia de recurso e ontologia de procedimento.

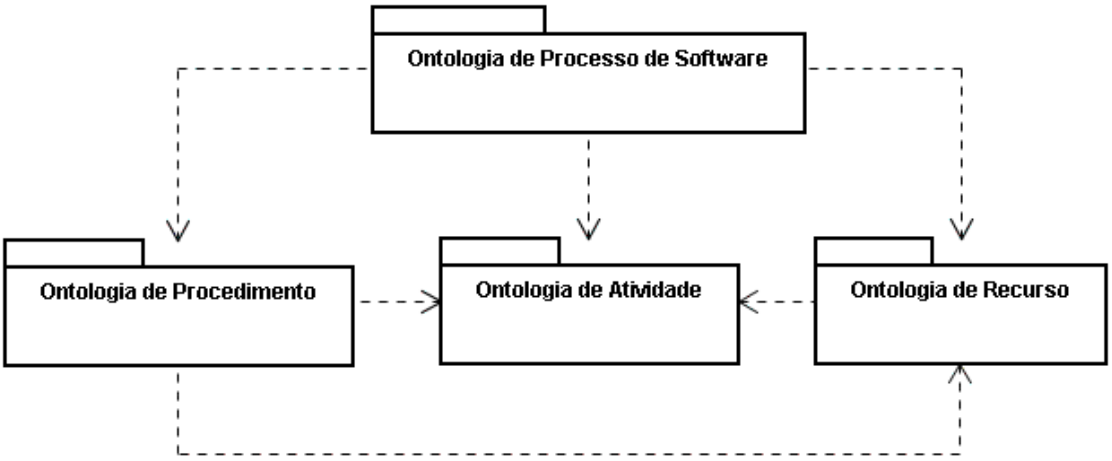


Figura 3.2 – Ontologia de Processo de Software e suas sub-ontologias.

Durante a evolução da ontologia de processo de software, suas sub-ontologias também foram alteradas. Assim, nesta seção, a evolução dessas sub-ontologias é discutida, realçando as alterações feitas e apresentando os modelos gráficos das mesmas, usando o perfil UML proposto pela versão atual do método SABiO. Axiomas e dicionários de termos são mostrados apenas para as alterações propostas. Para ter acesso à versão atual completa da ontologia de processo de software, veja o *site* do Laboratório de Engenharia de Software da UFES (www.inf.ufes.br/~labes).

3.2.1 Ontologia de Atividade

O conceito de atividade está no núcleo de qualquer modelo de processo de software. Atividades podem acontecer em diversos níveis, desde uma tarefa elementar até uma fase de um processo. Uma atividade é uma ação básica que pode utilizar artefatos como entrada e produzir artefatos como saída. Para ser executada, uma atividade pode requerer recursos e procedimentos podem ser adotados (FALBO, 1998).

Basicamente, uma ontologia de atividade deve ser capaz de responder às seguintes questões de competência (FALBO, 1998):

- Qual a natureza de uma atividade?

- Como uma atividade pode ser decomposta?
- Quais atividades devem anteceder uma dada atividade?
- Quais artefatos são insumo para uma determinada atividade?
- Quais artefatos são produzidos por uma determinada atividade?
- Que recursos são necessários para que uma atividade possa ser realizada?
- Que procedimentos podem ser adotados na realização de uma atividade?

A Figura 3.3 apresenta o modelo da ontologia de atividade. Como mostrado na figura, uma atividade pode ser decomposta em outras atividades, ditas sub-atividades. Atividades consomem e produzem artefatos, e algumas atividades dependem da realização de outras atividades, ditas pré-atividades.

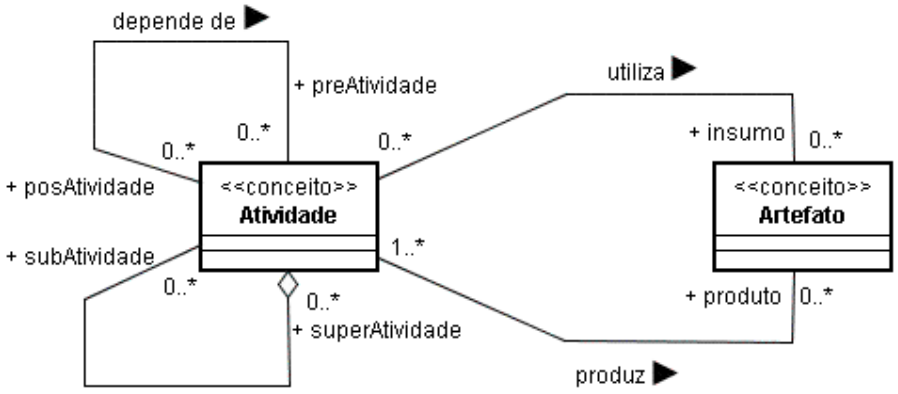


Figura 3.3 – Ontologia de Atividade.

Conforme discutido, o uso de um perfil UML para representar ontologias impõe alguns axiomas que não precisam ser escritos. Para ilustrar a instanciação de axiomas independentes de domínio, considere a relação todo-parte entre atividades. De acordo com o perfil UML adotado, os axiomas abaixo são considerados, sendo eles derivados dos axiomas (AE1) a (AE5) na Figura 3.1, respectivamente:

$$(\forall a)\ \neg\, subAtividade\ (a, a)$$

$$(\forall a1, a2)\ subAtividade\ (a2, a1) \leftrightarrow superAtividade\ (a1, a2)$$

$$(\forall a1, a2)\ subAtividade\ (a1, a2) \rightarrow \neg\, subAtividade\ (a2, a1)$$

$$(\forall a1, a2)\ subAtividade\ (a2, a1) \rightarrow (\exists\, a3)\ subAtividade\ (a3, a1)$$

$(\forall a1, a2, a3)\ subAtividade(a1, a2) \wedge subAtividade(a2, a3) \rightarrow subAtividade(a1, a3)$

Contudo, existem outros axiomas que não são capturados pela notação. Esses axiomas precisam ser escritos, tal como o que define que, se uma atividade *a1* produz um artefato *s* que é requerido por uma atividade *a2*, então *a1* é pré-atividade de *a2*.

$(\forall a1, a2, s)\ produto(s, a1) \wedge insumo(s, a2) \rightarrow preAtividade(a1, a2)$

Não é objetivo deste trabalho reapresentar axiomas e definições de termos que não foram objeto de mudança. Assim, a seguir, discutimos apenas alterações feitas por ocasião da evolução realizada neste trabalho.

A primeira alteração refere-se à questão de competência "Qual a natureza de uma atividade?". Para responder essa questão o conceito Atividade possuía três sub-tipos: Atividade de Construção, Atividade de Gerência e Atividade de Avaliação da Qualidade. Na evolução da ontologia, esses sub-tipos foram excluídos devido a dois fatores: (i) existem atividades que não se enquadram nesses três sub-tipos, por exemplo, atividades de um processo de aquisição; (ii) com a criação do conceito de Categoria de Processo, apresentado na seção 3.3, a categoria de uma atividade segue a categoria do processo a que pertence. Por exemplo, atividades de um processo de Garantia da Qualidade são atividades de Garantia da Qualidade.

Outra mudança diz respeito à restrição de cardinalidade da relação entre Atividade e Produto. Na versão da ontologia definida em (FALBO, 1998), toda atividade tinha de produzir pelo menos um artefato. Além disso, um artefato poderia não ser produzido por nenhuma atividade. Na evolução da ontologia, a cardinalidade da relação foi alterada. Uma atividade pode não gerar um produto, como, por exemplo, uma atividade de um processo de treinamento. Outra mudança está na obrigatoriedade de um artefato ser produzido por pelo menos uma atividade.

Foi excluída a questão de competência "Qual a natureza de um artefato?". Essa pergunta era respondida pela propriedade *tipo* do conceito Artefato, que assumia um dos seguintes valores: Componente, Documento ou Código. Com a elaboração da Ontologia de Artefatos de Software (NUNES, 2005), essa propriedade perdeu a relevância e a questão passou a ser tratada por aquela ontologia. Também foi excluída da Ontologia de Atividade a relação todo-parte que definia a composição de artefatos. Esse aspecto passou a ser discutido

na Ontologia de Artefatos de Software, onde possui uma importância maior. A Figura 3.4 apresenta parte da Ontologia de Artefatos de Software que trata das alterações descritas.

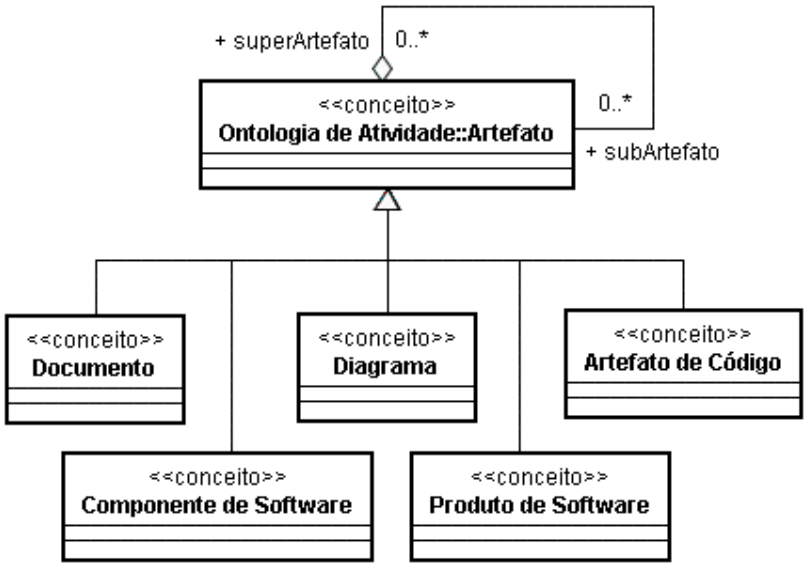


Figura 3.4 – Parte da Ontologia de Artefato de Software (NUNES, 2005).

3.2.2 Ontologia de Recurso

Para realizar uma atividade, além dos artefatos de insumo, outros elementos são necessários. Esses elementos, tais como agentes humanos, equipamentos de hardware e ferramentas de software, atuam ou apóiam a realização da atividade. Contudo, eles apenas auxiliam o processo, mas não são incorporados ao produto de software. Tais elementos são considerados recursos para a atividade.

A ontologia de recurso deve tratar as seguintes questões de competência:

- Quais recursos são requeridos por uma atividade?
- Qual a natureza de um recurso?

A Figura 3.5 apresenta o modelo da ontologia de recurso. Como mostrado na figura, recursos são usados para apoiar a execução de atividades. Esses recursos podem ser agrupados em três categorias principais:

- Recursos Humanos: papéis das pessoas que são requeridos para executar uma atividade, tais como analistas, gerentes de projeto, clientes e usuários. (RUY,

2006) aponta que, no contexto de uma organização de software, podem ser vistos como cargos;

- Recursos de hardware: incluem os equipamentos de hardware necessários para executar uma atividade, tais como computadores e impressoras;
- Recursos de software: produtos de software usados na realização de uma atividade. Dividem-se em *ferramentas de software*, *sistemas de apoio* e *ambientes de software*. Ferramentas de software são recursos de software que apóiam apenas partes específicas de um processo, sendo utilizadas, geralmente, para (semi-) automatizar um procedimento. Sistemas de apoio são recursos de software que não apóiam diretamente atividades do processo de software, mas ainda assim são necessários, tal como um sistema de gerenciamento de redes. Ambientes de software são conjuntos de recursos de software integrados, tal como um ambiente de desenvolvimento de software.

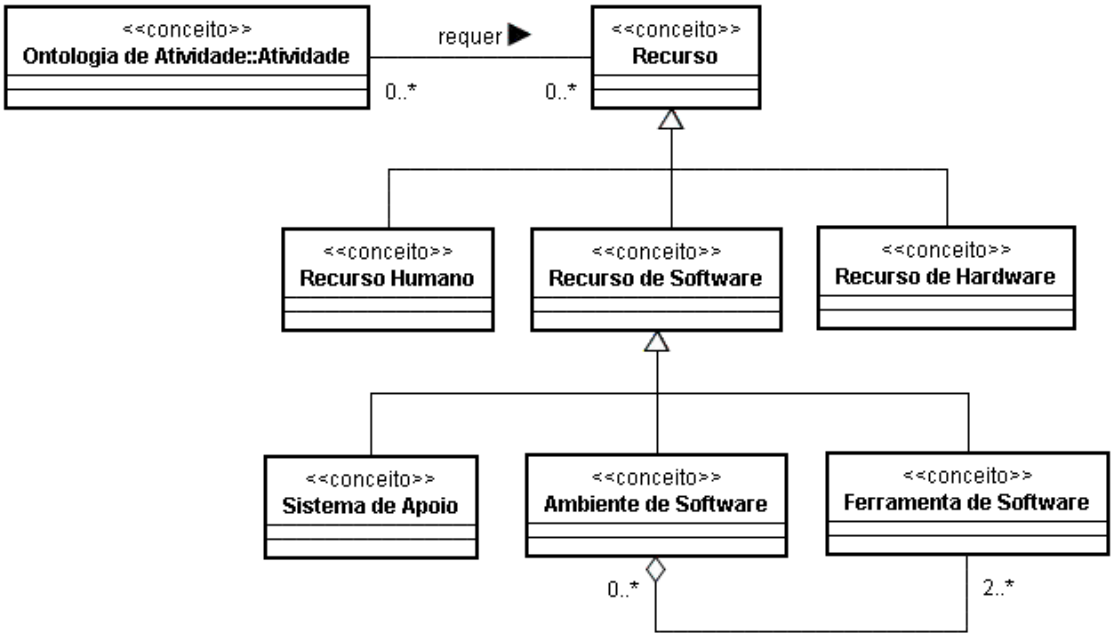


Figura 3.5 – Ontologia de Recursos.

Embora os sub-tipos de recurso (recurso humano, recurso de hardware e recurso de software) possuam características bem distintas e específicas, optou-se por fazer uma simplificação e manter uma relação simples (*requer*) com o conceito *Atividade*. Assim, não é o objetivo dessa ontologia, por exemplo, capturar aspectos de comportamento dos recursos

humanos na execução de uma atividade. Vale ressaltar, ainda, que a concepção adotada para recurso humano é análoga a cargo e não a pessoas (RUY, 2006).

A primeira alteração na Ontologia de Recurso está nos tipos de Ferramentas de Software. Na versão original da ontologia, esse conceito possuía quatro sub-tipos: Ferramenta de Construção, Ferramenta de Gerência, Ferramenta de Avaliação da Qualidade e Ferramenta de Propósito Geral. De forma análoga aos sub-tipos de atividade, na evolução da ontologia os sub-tipos de Ferramentas de Software foram desconsiderados.

O conceito de Ambiente de Software foi introduzido como um tipo de Recurso de Software. Um Ambiente de Software tem por objetivo integrar diversas Ferramentas de Software para automatizar várias atividades do processo de software. Como exemplo, pode-se citar um Ambiente de Desenvolvimento de Software, tal como ODE, descrito no capítulo anterior.

O seguinte axioma ontológico relacionado a esse novo conceito deve ser observado:

Se uma ferramenta de software *f* compõe um ambiente de software *as* e uma atividade *at* requer *f*, então *at* requer *as*.

$$(\forall as, at, f)\ parteDe\ (f, as) \wedge requer\ (at, f) \rightarrow requer\ (at, as)$$

A Tabela 3.1 apresenta parte do dicionário de termos, mostrando apenas o novo conceito adicionado à ontologia de recurso.

Tabela 3.1 – Descrição dos Novos Termos da Ontologia de Recurso.

Ambiente de Software	Tipo de **recurso de software** que agrega um conjunto de **ferramentas de software** com objetivo de automatizar várias atividades do processo de software. Ex: Ambientes de Desenvolvimento de Software.

3.2.3 Ontologia de Procedimento

Durante a definição do processo de software, é importante estabelecer como as atividades serão realizadas. Para isso, é preciso definir os procedimentos a serem adotados na realização das atividades.

Existem vários tipos de procedimentos. Métodos, por exemplo, definem um conjunto de sub-atividades a serem executadas em uma atividade, além de proverem orientações sobre

como realizar essas sub-atividades. Um Mentor de Ferramenta, por outro lado, descreve como usar uma ferramenta de software na realização de uma atividade. Roteiros são diretrizes a serem seguidas na elaboração de um artefato em uma atividade. Assim, a ontologia de procedimento deve ser capaz de responder às seguintes questões de competência:

- Que procedimentos podem ser adotados na realização de uma atividade?
- Qual a natureza de um procedimento?
- Que ferramentas de software podem ser utilizadas para (semi-)automatizar um procedimento?

A Figura 3.6 apresenta o modelo da ontologia de procedimento. Como mostrado nessa figura, procedimentos são adotados para apoiar a execução de atividades. Esses procedimentos são classificados segundo várias categorias. Não é o objetivo fixar uma rígida taxonomia de procedimentos. Algumas classes de procedimentos incluem métodos, técnicas, mentores de ferramentas e roteiros. Mas outras podem ser adicionadas a essa taxonomia, tal como regras de nomenclatura.

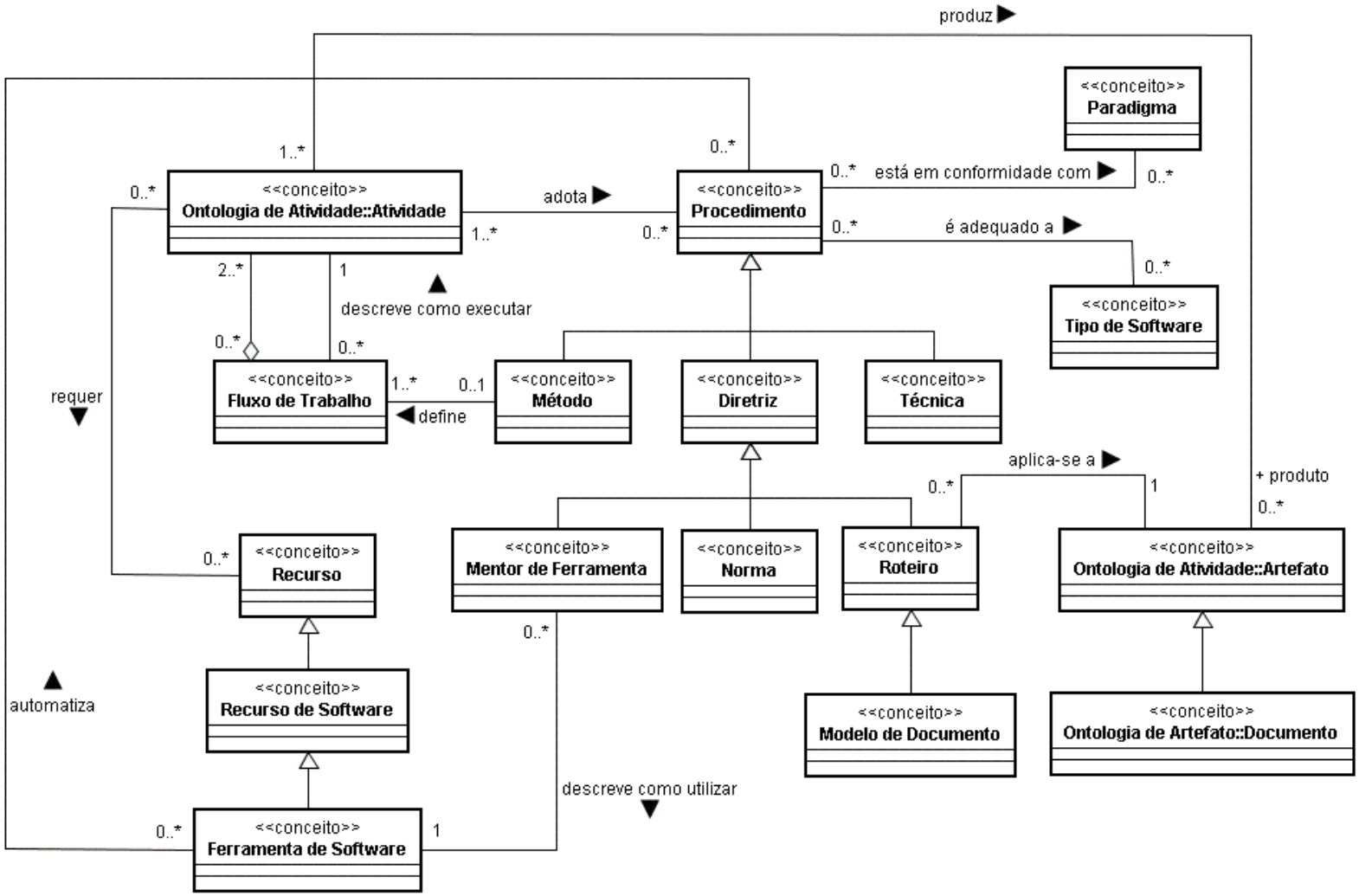


Figura 3.6 – Ontologia de Procedimento.

Um método é um procedimento sistemático que define fluxos de trabalho (*workflow*), dados por conjuntos de atividades, e heurísticas para executar uma ou mais atividades. Quando um método pode ser adotado na execução de mais de uma atividade, há um fluxo de trabalho para cada uma delas. Por exemplo, muitos métodos de desenvolvimento descrevem diferentes fluxos de trabalho para as atividades de análise e projeto.

Uma técnica é um procedimento para executar uma atividade, sendo menos rígido e detalhado que um método, pois não descreve um fluxo de trabalho. Contudo, ainda provê heurísticas para executar uma atividade. Técnicas de teste caixa branca e de teste caixa preta são exemplos de técnicas.

Um mentor de ferramenta descreve como usar uma ferramenta de software e, finalmente, um roteiro tem objetivo de fornecer orientações para a elaboração de um artefato.

Na versão original da ontologia de procedimento, os conceitos Método e Técnica possuíam, respectivamente, os seguintes sub-tipos: Método de Construção, Método de Gerência e Método de Avaliação da Qualidade; e Técnica de Construção, Técnica de Gerência e Técnica de Avaliação da Qualidade. De forma análoga aos sub-tipos de atividade, na evolução da ontologia de procedimento, os sub-tipos de Método e Técnica foram desconsiderados.

O termo Padrão de Atividades, usado para designar o conjunto de atividades prescrito por um método para realizar uma atividade, foi alterado para Fluxo de Trabalho. Um Método possui um ou mais Fluxos de Trabalho, que descrevem como uma atividade deve ser executada segundo aquele método. Um Fluxo de Trabalho define um conjunto de Atividades. Neste contexto, são definidos os seguintes axiomas de consolidação:

(i) Se um fluxo de trabalho *f* descreve como executar a atividade *a*, então *a* não pode ser parte do conjunto de atividades definido por *f*.

$$(\forall f, a)\ descreveComoExecutar\ (f, a) \rightarrow \neg parteDe(a, f)$$

(ii) Se um fluxo de trabalho *f* apóia a adoção de um método *m* na execução de uma atividade *a*, então *m* deve poder ser adotado por *a*.

$$(\forall m, f, a)\ define\ (m, f) \wedge descreveComoExecutar\ (f, a) \rightarrow adota\ (a, m)$$

A partir das relações entre os conceitos de Atividade e Fluxo de Trabalho, é possível derivar a seguinte informação: as atividades que compõem um fluxo de trabalho são sub-atividades da atividade para a qual o fluxo de trabalho está sendo definido. O seguinte axioma de derivação, depende do domínio, formaliza essa declaração:

Se um fluxo de trabalho *f* descreve como executar uma atividade *a1* e uma atividade *a2* é parte do fluxo de trabalho *f*, então *a2* é uma sub-atividade de *a1*.

$$(\forall a1, a2, f)\ descreveComoExecutar\ (f, a1) \wedge parteDe\ (a2, f) \rightarrow subAtividade\ (a2, a1)$$

O conceito Mentor de Ferramenta foi criado como um sub-tipo de Diretriz. Um Mentor de Ferramenta tem por objetivo descrever como usar uma Ferramenta de Software específica. Por exemplo, pode-se elaborar um mentor de ferramenta para a utilização de uma ferramenta de modelagem. O seguinte axioma ontológico deve ser observado:

Se um mentor de ferramenta *m* é adotado por uma atividade *a* e descreve a utilização de uma ferramenta de software *f*, então *a* pode requerer *f*.

$$(\forall m, a, f)\ adota\ (a, m) \wedge descreveComoUtilizar\ (m, f) \rightarrow requer\ (a, f)$$

Na ontologia de artefato de software, NUNES (2005) define uma relação entre Roteiro e Documento, denominada *aplica-se a*, indicando que um roteiro se aplica na elaboração de um documento. Contudo, essa relação é mais genérica, pois um roteiro pode se aplicar à elaboração de qualquer artefato. Por exemplo, um roteiro pode ser usado na elaboração de um diagrama. Na revisão da ontologia de procedimento essa relação foi definida como sendo entre os conceitos Roteiro e Artefato. Neste contexto, se o roteiro for um Modelo de Documento (conceito criado na ontologia de documento (NUNES, 2005)), o artefato deve ser um Documento. Essa restrição é descrita pelo seguinte axioma de consolidação:

Se um roteiro *r* é utilizado na elaboração do artefato *a* e *r* é um modelo de documento, então *a* tem de ser um documento.

$$(\forall r, a)\ aplicaSeA\ (r, a) \wedge modeloDocumento\ (r) \rightarrow documento\ (a)$$

Um procedimento pode ser adotado por uma determinada atividade. Com a definição do conceito Ambiente de Software, descrito na ontologia de recurso, um novo axioma é necessário.

Se uma Ferramenta de Software *f* compõe um Ambiente de Software *a* e *f* pode (semi-) automatizar um Procedimento *p*, então *a* pode (semi-) automatizar *p*.

$$(\forall a,\ f, p)\ parteDe\ (f, a) \wedge automatiza\ (f, p) \rightarrow automatiza\ (a, p)$$

Finalmente, o termo Tecnologia de Desenvolvimento da ontologia de processo, usado pela ontologia de procedimento, foi substituído pelo termo Tipo de Software, designando o mesmo conceito. A alteração foi apenas no termo, permanecendo a mesma semântica. A mudança ocorreu devido ao uso atual do termo Tecnologia estar bastante amplo, não atendendo ao objetivo inicial do conceito.

A Tabela 3.2 apresenta parte do dicionário de termos, mostrando apenas os conceitos e relações novos e alterados na evolução da ontologia de procedimento.

Tabela 3.2 –Termos Novos e Alterados da Ontologia de Procedimento.

aplica-se a ***(novo)***	relação entre um **artefato** e um **roteiro**, indicando que um **roteiro** provê orientações para a elaboração de um **artefato**.
define ***(novo)***	relação entre um **método** e um **fluxo de trabalho**, indicando que um **fluxo de trabalho** define um conjunto de **atividades** para apoiar a adoção de um **método** na execução uma **atividade**.
descreve como executar ***(novo)***	relação entre um **fluxo de trabalho** e uma **atividade**, indicando que uma **atividade**, que adota um determinado **método**, é executada segundo um determinado **fluxo de trabalho**.
descreve como utilizar ***(novo)***	relação entre uma **ferramenta de software** e um **mentor de ferramenta**, indicando que um **mentor de ferramenta** descreve como utilizar uma **ferramenta de software**.
Fluxo de Trabalho ***(alterado)***	Descreve um conjunto de sub-atividades a serem realizadas quando um **método** for usado na realização de uma **atividade**.
Mentor de Ferramenta ***(novo)***	Tipo de **diretriz** que determina um guia para a utilização de uma **ferramenta de software** específica. Ex: Guia para uso de uma ferramenta de modelagem, guia para uso de um ambiente de programação etc.

3.3 Ontologia de Processo de Software

A ontologia de processo de software originalmente definida em (FALBO, 1998) foi evoluída para atender às novas demandas da área de processos de software. Assim, as seguintes questões de competência foram consideradas no desenvolvimento da nova versão da ontologia de processo de software:

QC1: Como um processo pode ser decomposto?

QC2 : Quais são os ativos que compõem um processo de software?

QC3: Quais são as entradas e saídas de um processo de software?

QC4: Qual a natureza de um processo?

QC5: Qual o nível de abstração de um processo?

QC6: Como um processo padrão pode ser adaptado?

QC7: Como se dá a interação entre processos?

QC8: Como as atividades de um processo de software de projeto são organizadas?

Para tratar essas questões de competência, alguns aspectos devem ser considerados:

- Decomposição e Interação entre Processos (QC1 e QC7);
- Definição de Processos (QC2 e QC3);
- Tipos e Nível de Abstração de Processos (QC4, QC5 e QC6);
- Modelo de Ciclo de Vida de Processo de Projeto (QC8).

Nas seções seguintes, cada aspecto listado acima é discutido, apresentando os modelos e axiomas correspondentes.

3.3.1 Decomposição e Interação entre Processos

Um processo é definido para estabelecer uma abordagem sistemática no desenvolvimento e manutenção de software, e pode ser decomposto em atividades ou outros processos. Por exemplo, de acordo com a ISO/IEC 12207, o processo de software pode ser decomposto em processos para aquisição, fornecimento, desenvolvimento, operação e manutenção, entre outros. O processo de desenvolvimento pode, então, ser decomposto em outros processos, como um processo de engenharia de requisitos etc. O processo de engenharia de requisitos, por sua vez, pode ser decomposto em atividades, como levantamento de requisitos, análise e negociação, modelagem, documentação etc. Atividades, por sua vez, podem ser decompostas em sub-atividades, conforme aponta a ontologia de atividade.

Além disso, um processo de software pode interagir com outros processos. Essa interação pode se dar de várias formas, entre elas: um processo pode preceder a execução de outro, dois processos podem ser executados em paralelo, ou um processo pode ser executado em um momento específico durante a execução de outro processo. Não é objetivo da ontologia proposta definir os tipos de interação entre processos. A Figura 3.7 mostra o modelo que trata dessas questões.

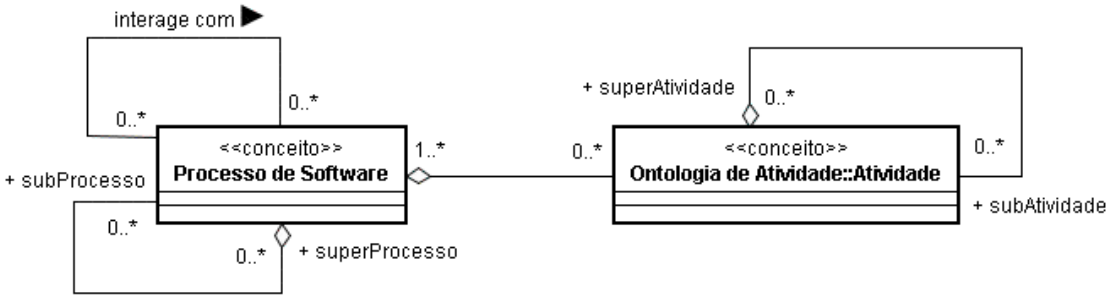


Figura 3.7 – Decomposição e Interação de processos.

Um super-processo é composto por outros processos. Um sub-processo é um processo de software que compõe um processo maior, seu super processo.

3.3.2 Definição de Processo

Como discutido anteriormente, um processo pode ser decomposto em processos ou atividades. Durante a definição de um processo, muitos outros ativos de processo devem também ser definidos. Para cada atividade do processo de software, deve-se definir suas sub-atividades, pré-atividades, artefatos de insumo e produto, recursos requeridos (humano, software e hardware) e os procedimentos (métodos, técnicas, etc) a serem adotados na realização da atividade. A Figura 3.8 apresenta os ativos de processo envolvidos na definição de processo de software.

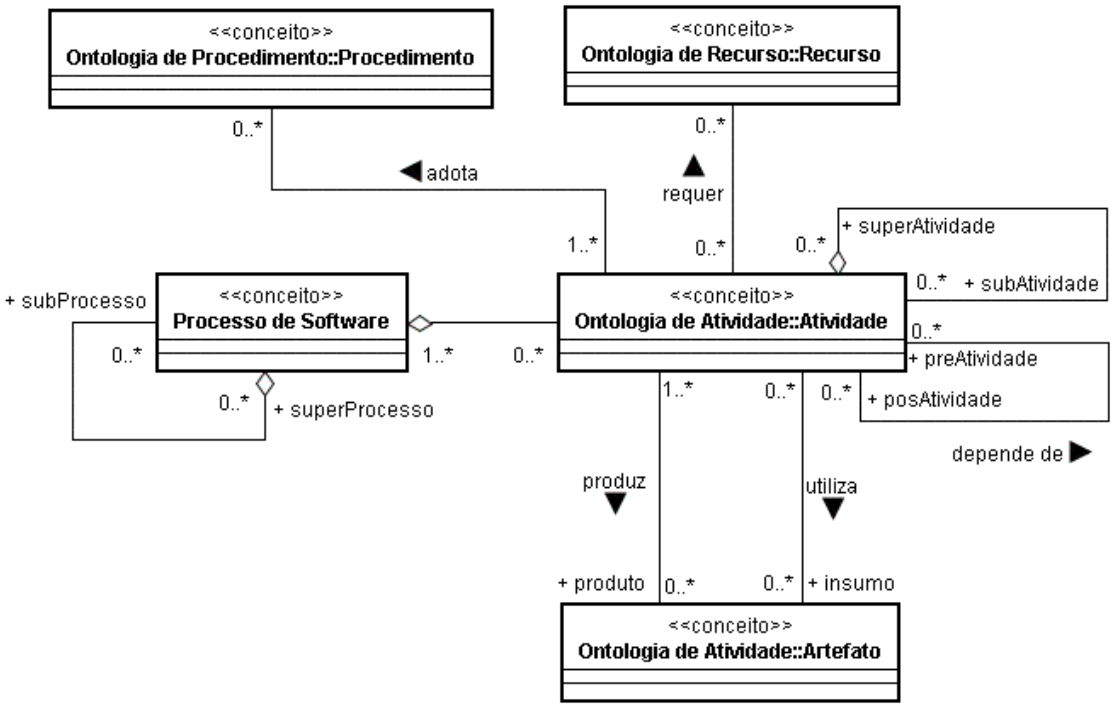


Figura 3.8 – Definição de Processos.

A maior parte desse modelo corresponde às sub-ontologias apresentadas na seção anterior, com destaque para a ontologia de atividade. Assim, são enfatizados, a seguir, os tópicos referentes a processos.

Conforme apresentado, uma atividade é uma ação que pode produzir artefatos. Para ser executada uma atividade pode requerer recursos, adotar procedimentos e utilizar artefatos previamente produzidos.

De forma similar, pode-se dizer que um processo de software possui entradas e saídas. Seus insumos e produtos são diretamente relacionados com os insumos e produtos de suas atividades. Assim, se uma atividade *a1*, parte de um processo de software *p*, requer como insumo um artefato *s*, e não existe outra atividade *a2*, parte do mesmo processo *p*, que produz esse artefato, então *s* é dito insumo para *p*.

$$\forall (p, a1, s)\ atividade\ (a1) \wedge processo\ (p) \wedge parteDe\ (a1, p) \wedge insumo(s, a1) \wedge ((\neg \exists a2)\ parteDe(a2, p) \wedge produto\ (s, a2)) \rightarrow insumo\ (s, p)$$

Considerando produtos, pode-se dizer que os produtos de um processo correspondem aos produtos de suas atividades.

$$\forall (p, a1, s)\ atividade(a1) \wedge processo(p) \wedge parteDe\ (a1, p) \wedge produto\ (s, a1) \rightarrow produto\ (s, p)$$

De forma análoga, um super-processo tem seus insumos e produtos definidos através dos insumos e produtos de seus sub-processos, conforme descrito pelos seguintes axiomas:

$$\forall (p1,p2,s)\ subProcesso\ (p2,p1) \wedge insumo\ (s, p2) \wedge$$
$$((\neg \exists p3)\ subProcesso\ (p3, p1) \wedge produto\ (s, p3)) \rightarrow insumo\ (s, p1)$$

$$\forall (p1,p2,s)\ (subProcesso(p2,p1) \wedge produto\ (s, p2) \rightarrow produto\ (s, p1)$$

3.3.3 Tipos e Níveis de Abstração de Processos

Conforme apresentado na Figura 3.9, processos podem ser classificados em categorias de processos. Por exemplo, se uma organização segue a classificação da norma ISO/IEC 12207, as categorias podem ser Processos Fundamentais, Processos de Apoio e Processos Organizacionais. Além disso, processos estão em diferentes níveis de abstração. Um processo padrão refere-se a um processo genérico institucionalizado em uma organização, estabelecendo requisitos básicos para processos serem executados nessa organização. Processos de Projeto referem-se a processos definidos para projetos específicos, considerando as particularidades desses projetos.

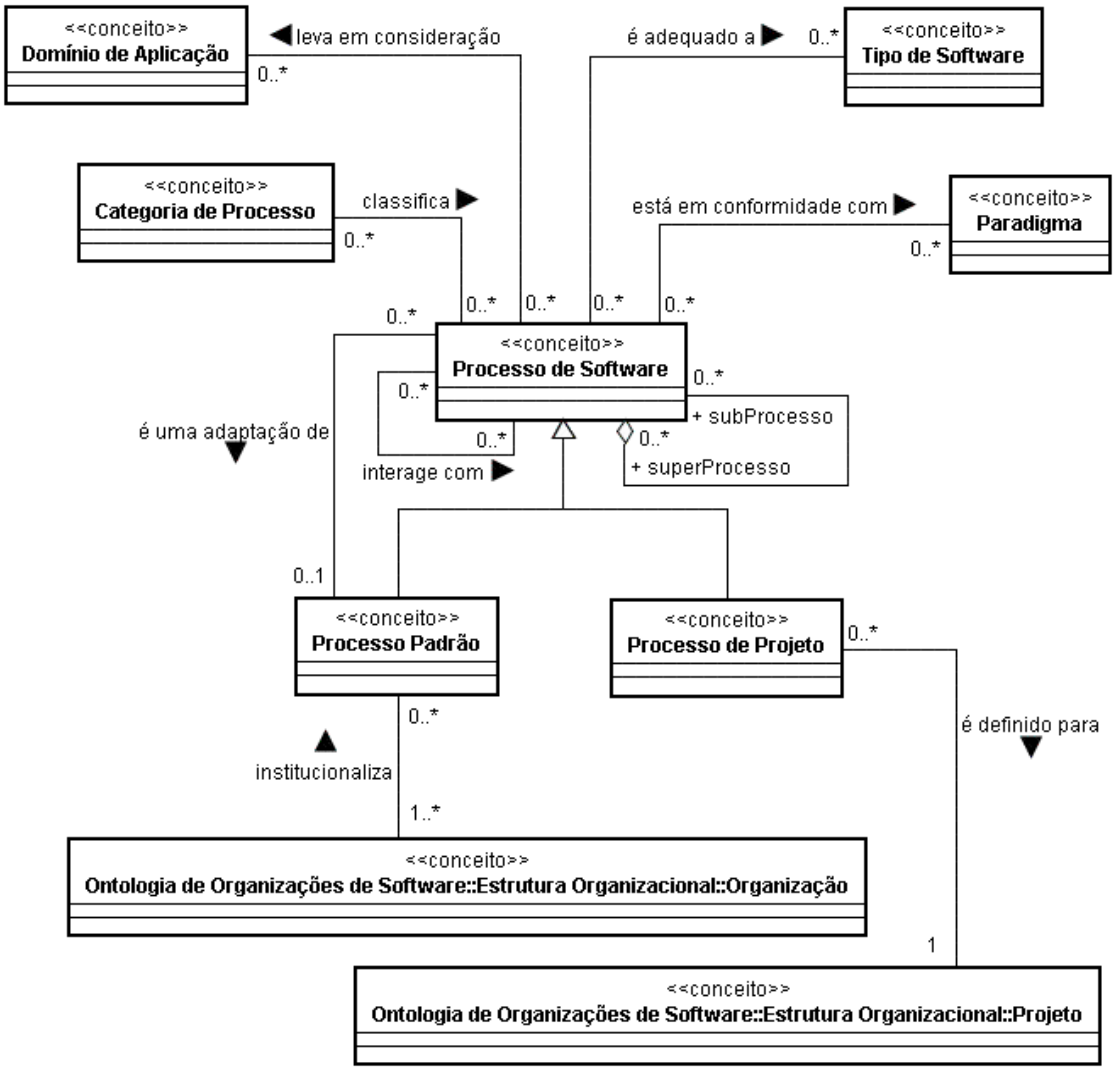


Figura 3.9 – Tipos de Processo e Níveis de Abstração.

Processos de software (padrão ou de projeto) podem ser definidos adaptando-se um processo padrão. Quando um processo padrão é adaptado para outro processo padrão, o processo adaptado é chamado processo especializado e todos os ativos de processo definidos para o processo padrão tornam-se parte do processo especializado. Contudo, novos ativos podem ser incluídos para tratar um tipo de software, paradigma ou domínio de aplicação específico.

Quando um processo de projeto é definido a partir da adaptação de um processo padrão, o mesmo será composto pelos ativos do processo padrão e novos ativos podem ser incluídos considerando características do projeto, como complexidade, tamanho, experiência da equipe etc.

Vale ressaltar que a interação entre processos deve ocorrer no mesmo nível de abstração, ou seja, um processo padrão pode interagir somente com outro processo padrão e um processo de projeto pode interagir somente com outro processo de projeto.

$$\forall(p1,p2)\ (interageCom\ (p1,p2) \rightarrow (processoPadrao\ (p1) \wedge processoPadrao\ (p2)) \vee (processoProjeto\ (p1) \wedge processoProjeto\ (p2))$$

3.3.4 Modelo de Ciclo de Vida de Processo de Projeto

A definição do processo de projeto começa com a escolha do modelo de ciclo de vida a ser usado como referência. Um modelo de ciclo de vida estrutura as atividades do projeto em fases (ou macro-atividades), estabelecendo uma abordagem para organização dessas macro-atividades. Considerando os principais modelos de ciclo de vida descritos na literatura, pode-se notar que as macro-atividades são agrupadas em combinações que seguem duas estratégias básicas: seqüencial ou iterativa. Em combinações seqüenciais, as fases são executadas apenas uma vez, retornando à fase anterior apenas para correção de possíveis problemas detectados. Em combinações iterativas, um conjunto de fases é executado diversas vezes, de acordo com algum critério estabelecido (FALBO, 1998).

O modelo de ciclo de vida cascata (ou modelo seqüencial linear) (PRESSMAN, 2005), por exemplo, pode ser descrito como uma combinação seqüencial simples de todas as fases. O modelo espiral (PRESSMAN, 2005) pode ser descrito também com apenas uma combinação, mas neste caso uma combinação iterativa. Outros modelos de ciclo de vida, como o modelo paralelo/recursivo (PRESSMAN, 2005), podem ser descritos com várias combinações de atividades, algumas delas seqüenciais e outras iterativas. Assim, a maioria dos modelos de ciclo de vida pode ser mapeada como combinações híbridas (seqüencial e iterativa) de macro-atividades. Dessa forma, um modelo de ciclo de vida define um conjunto de macro-atividades (ou fases) que um processo pode apresentar e a ordem de execução em forma de combinações, como mostra a Figura 3.10.

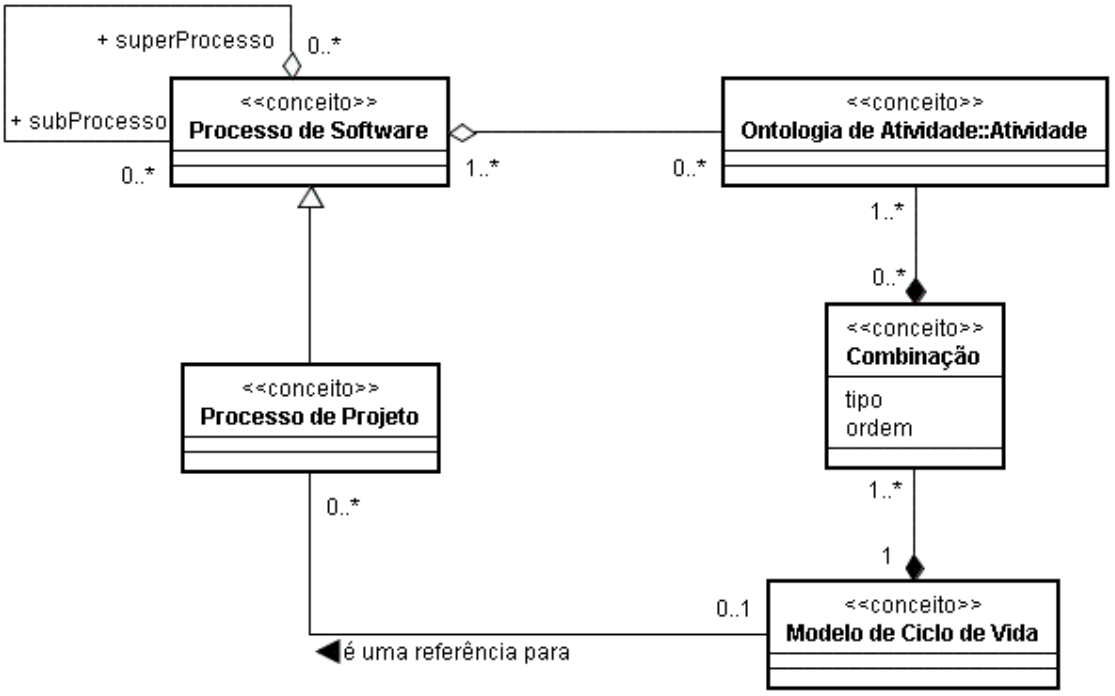


Figura 3.10 – Modelo de Ciclo de Vida de Processo de Projeto.

O ponto de partida para a definição de processos de projeto é o modelo de ciclo de vida adotado. Assim, a estrutura inicial do processo de projeto deve corresponder ao conjunto de macro-atividades que compõe o modelo de ciclo de vida. Ou seja, se um processo de projeto *p* adota como referência o modelo de ciclo de vida *m*, então cada atividade *a* que é parte de uma combinação *c* do modelo de ciclo de vida *m* deve também ser parte de *p*.

$$\forall(p, m, c, a, n)\ (ehUmaReferenciaPara\ (m, p) \wedge parteDe\ (c,m) \wedge parteDe\ (a, c)) \rightarrow parteDe\ (a, p))$$

A Tabela 3.3 apresenta parte do dicionário de termos, mostrando apenas os termos novos e alterados na evolução ontologia de processo de software.

Tabela 3.3 - Dicionário de Termos da Ontologia de Processo de Software.

Categoria de Processo ***(novo)***	categorização de **processos de software**. Ex: Fundamental, de Apoio ou Organizacional.
classifica ***(novo)***	relação entre **processo de software** e **categoria de processo**, indicando a classificação de um **processo de software** em uma determinada **categoria de processo**.
é adequado a ***(novo)***	relação entre um **processo de software** e um **tipo de software**, indicando que um **processo de software** é adequado ao **tipo de software**.
é definido para ***(novo)***	relação entre **processo de projeto** e **projeto**, indicando que um **processo de projeto** é definido especificamente para um **projeto**.
é uma adaptação de ***(novo)***	relação entre **processo padrão** e **processo de software**, indicando que um **processo padrão** pode ser adaptado para outro **processo de software** (podendo ser padrão ou de projeto).
é uma referência para ***(alterado)***	relação entre um **processo de projeto** e um **modelo de ciclo de vida**, indicando que um **processo de projeto** organiza suas **atividades** segundo um **modelo de ciclo de vida**.
está em conformidade com ***(alterado)***	relação entre **processo de software** e **paradigma**, indicando que um **processo de software** segue os princípios do **paradigma**.
interage com ***(novo)***	relação entre dois **processos de software**, indicando que **processos de software** podem interagir de alguma forma.
institucionaliza ***(novo)***	relação entre **processo padrão** e **organização**, indicando que um **processo padrão** é institucionalizado em uma **organização**, ou seja, **projetos** da **organização** seguem os padrões definidos no **processo padrão.**
leva em consideração ***(alterado)***	relação entre **processo de software** e **domínio de aplicação**, indicando que um **processo de software** tem elementos específicos voltados para atender a características de um **domínio de aplicação**.

Tabela 3.3 - Dicionário de Termos da Ontologia de Processo de Software (continuação).

Modelo de Ciclo de Vida ***(alterado)***	estrutura que define as **macro-atividades** de um **processo de projeto**, organizadas na forma de **combinações** que são realizadas de forma seqüencial ou iterativa.
Organização ***(novo)***	grupo organizado para um propósito comum. Ex: LabES-UFES, Empresa SPI-Quality.
Processo de Software ***(novo)***	conjunto de **sub-processos** ou **atividades** inter-relacionadas que transforma insumos em produtos. Ex: Processo de Desenvolvimento de Software
Processo de Projeto ***(novo)***	processo definido para um **projeto** específico, considerando suas particularidades. Pode ou não ser definido a partir de uma adaptação de um **processo padrão**. Ex: Processo de Desenvolvimento de Software definido para o Projeto do Portal do LabES.
Processo Padrão ***(novo)***	processo definido em um nível organizacional, contendo os ativos de processo básicos que devem fazer parte de qualquer **processo de projeto** de uma **organização**. Ex: Processo de Desenvolvimento de Software do LabES.
sub-Processo ***(novo)***	faceta da relação de composição entre dois **processos de software** p1 e p2. Se p2 é parte de p1, então p2 é dito **sub-processo** de p1.
Tipo de Software ***(alterado)***	tipo do software a ser desenvolvido ou mantido em um **projeto**. Ex.: sistema de informação, sistema baseado em conhecimento, aplicação *Web* etc.

3.4 Mapeando Normas e Modelos de Qualidade na Ontologia

Uma vez definida a ontologia de processo de software, é interessante mapear os termos usados pelos principais modelos e normas de qualidade de processos nos conceitos da ontologia. Deste modo, a ontologia pode ser usada para facilitar a aplicação conjunta desses modelos, eliminando confusões terminológicas e servindo como uma inter-língua entre eles, bem como é possível avaliar o grau de cobertura da ontologia em relação aos principais conceitos adotados pelas normas. É importante apontar que alguns padrões, como a ISO

9001:2000 (ISO, 2000) e o CMMI, não são específicos para a área de software. Contudo, o mapeamento enfoca somente organizações de software e, então, o foco está centrado apenas nessa perspectiva.

3.4.1 As Normas ISO e a Ontologia de Processo de Software

A Tabela 3.4 mostra o mapeamento básico entre o vocabulário usado pelas normas ISO/IEC 12207, ISO 9001:2000 e ISO/IEC 15504, e os conceitos da ontologia de processo de software.

Tabela 3.4 – ISO X Ontologia de Processo de Software.

ISO	Ontologia de Processo de Software
Processo	Processo de Software
Propósito	Propósito
Resultado	Resultado Esperado
Interação de Processos	Interação
Processo Padrão	Processo Padrão
Processo Adaptado	Processo de Projeto
Atividade / Tarefa / Prática	Atividade
Produto de Trabalho / Registro	Artefato
Modelo de Ciclo de Vida	Modelo de Ciclo de Vida
Projeto	Projeto
Organização	Organização

Embora cada norma possua seu objetivo específico, grande parte dos conceitos é compartilhada pelas três normas, ainda que outros possam aparecer mais fortemente em uma norma específica. Por serem normas que utilizam uma abordagem de processo, o conceito Processo aparece sob a mesma definição em todas, assim como o conceito Interação de Processos. Os conceitos Propósito e Resultado são descritos nas normas ISO/IEC 15504 e ISO/IEC 12207 com a mesma semântica desses conceitos na ontologia de processo de software.

O conceito Processo Padrão aparece mais explicitamente na norma ISO/IEC 15504. O conceito de Processo Adaptado é tratado nas normas ISO/IEC 15504 e ISO/IEC 12207, enquanto a ISO 9001 apenas obriga a definição dos processos da organização, não fazendo uma distinção clara entre Processo Padrão e Processo Adaptado.

Atividade, como sendo o detalhamento de um processo, e Produto de Trabalho, como sendo uma saída de uma atividade ou processo, são conceitos comum às três normas. Contudo, a ISO/IEC 15504 define atividades como Práticas e a ISO 9001 define Produto de Trabalho como Registro. Por ser uma norma que define Processos de Ciclo de Vida de Software, na ISO/IEC 12207 o conceito de Modelo de Ciclo de Vida é bem explorado. Porém essa ênfase não se repete nas demais normas.

O conceito Projeto é utilizado com a mesma semântica da ontologia de processo de software nas normas ISO/IEC 15504 e ISO/IEC 12207, consistindo em um conjunto gerenciado de recursos inter-relacionados que entrega produtos a um cliente ou usuário final e tipicamente opera de acordo com um plano.

Deve-se tomar cuidado, contudo, com o emprego do termo Projeto por essas normas, sobretudo a ISO 9001:2000, pois, em sua tradução para o português, o termo Projeto é utilizado algumas vezes com a semântica da atividade Projeto (em inglês *Design*).

Por fim, o conceito Organização é compartilhado pelas três normas.

3.4.2 O CMMI e a Ontologia de Processo de Software

A Tabela 3.5 apresenta o mapeamento básico entre o vocabulário usado pelo CMMI e os conceitos da ontologia de processo de software.

O conceito de Área de Processo do CMMI está diretamente relacionado com o conceito Processo de Software da ontologia, inclusive no que se refere à propriedade Propósito, que descreve o propósito de um processo. Contudo, o CMMI tem um uso especial do conceito de Processo na descrição de objetivos e práticas genéricas, se referindo ao processo ou processos que as implementam em uma área de processo. As áreas de processo são agrupadas em Categorias e, como o conceito de Processo de Software na ontologia, interagem entre si.

Tabela 3.5 – CMMI x Ontologia de Processo de Software.

CMMI	Ontologia de Processo de Software
Área de Processo / Processo	Processo de Software
Propósito	Propósito
Interação	Interação
Categoria	Categoria de Processo
Processo Padrão	Processo Padrão
Adaptação de Processo	Adaptação
Processo Definido / Processo Definido de Projeto	Processo de Projeto
Prática / Sub-Prática	Atividade
Produto de Trabalho	Artefato
Modelo de Ciclo de Vida	Modelo de Ciclo de Vida
Projeto	Projeto
Organização	Organização

Devido ao fato do modelo CMMI ser uma importante base técnica para a evolução da ontologia de processo, muitos dos conceitos envolvidos estão alinhados, mas não necessariamente utilizam o mesmo termo. Por exemplo, Produto de Trabalho e Artefato são termos diferentes usados para o mesmo conceito, compartilhando, portanto, a mesma semântica. A definição de um Processo Padrão e sua adaptação para um Processo Definido (ou Processo Definido de Projeto) é análogo à adaptação de um Processo Padrão para um Processo de Projeto na ontologia.

O conceito Projeto possui uma particularidade, visto que o modelo CMMI permite composição de projetos. Essa abordagem não é compartilhada pela ontologia de processo de software.

3.4.3 O MPS.BR e a Ontologia de Processo de Software

A Tabela 3.6 apresenta o mapeamento básico entre o vocabulário usado pelo MPS.BR e os conceitos da ontologia de processo de software.

Tabela 3.6 – MPS.BR x Ontologia de Processo de Software.

MPS.BR	Ontologia de Processo de Software
Processo	Processo de Software
Propósito do Processo	Propósito
Resultado Esperado do Processo	Resultado Esperado
Interação	Interação
Classe de Processo	Categoria de Processo
Processo Padrão	Processo Padrão
Adaptação de Processo	Adaptação
Processo Definido	Processo de Projeto
Atividade / Tarefa	Atividade
Produto de Trabalho	Artefato
Recurso	Recurso
Interessado	Recurso Humano
Procedimento	Procedimento
Método	Método
Técnica	Técnica
Ciclo de Vida	Modelo de Ciclo de Vida
Projeto	Projeto
Organização	Organização

O MPS.BR é o modelo de qualidade com maior aderência à ontologia de processo de software. Esse fato se deve à utilização das mesmas bases técnicas na elaboração dos modelos. A abordagem de definição de um Processo Padrão para a Organização e a Adaptação para um Processo de Projeto é utilizada tanto no modelo MPS.BR quanto na ontologia de processo de software.

Com a mesma semântica, em ambos os modelos, são citados ativos de processo relacionados com Atividades, tais como Artefatos (ou Produtos de Trabalho), Recursos e Procedimentos. Contudo, no modelo MPS.BR, não há uma relação explícita de sub-tipo entre Interessado e Recurso; e entre Técnica / Método e Procedimento, como ocorre na ontologia de processo de software.

Projeto, Organização e Ciclo de Vida (como uma necessidade explícita para o Projeto) também são conceitos descritos nos dois modelos.

3.4.4 O RUP e a Ontologia de Processo de Software

A Tabela 3.7 apresenta o mapeamento básico entre o vocabulário usado pelo RUP e os conceitos da ontologia de processo de software.

Tabela 3.7 – RUP X Ontologia de Processo de Software.

RUP	Ontologia de Processo de Software
Processo / Fluxo de Trabalho	Processo de Software
Interação de Fluxos de Trabalho	Interação
Arcabouço de Processo	Processo Padrão
Atividade / Etapa	Atividade
Artefato	Artefato
Trabalhador	Recurso Humano
Procedimento	Procedimento
Mentor de Ferramenta	Mentor de Ferramenta
Modelo de Documento	Modelo de Documento

O conceito de Fluxo de Trabalho (*Workflow*) do RUP está diretamente associado ao conceito de Processo de Software na ontologia, inclusive no que se refere à relação de interação entre processos. No RUP ainda são definidos seis fluxos de trabalho básicos e três fluxos de trabalho de apoio. Essa distinção de tipos de fluxos de trabalho é mapeada para o conceito de Categoria de Processo na ontologia. Contudo, na ontologia de processo de software não são instanciadas categorias ou processos de software. Os elementos base do RUP determinam um Arcabouço (*Framework*) de Processo, que é análogo ao conceito de Processo Padrão da ontologia.

Para demonstrar a influência do RUP na revisão da ontologia de processo de software, podemos citar o conceito Mentor de Ferramenta que não estava na versão original da ontologia e foi incorporado na evolução proposta.

Artefato e Atividade são termos que possuem a mesma semântica.

Embora tenha sido feito um mapeamento dos conceitos aderentes aos dois modelos, vale ressaltar um conceito que não é compartilhado na ontologia, mas que tem uma grande importância no RUP: o conceito de fase. Cada processo pode ter diversas iterações e essas iterações são agrupadas por fases, a saber Concepção, Elaboração, Construção e Transição. Cada fase possui um objetivo específico. Esse conceito não é tratado explicitamente na

ontologia, mas entendemos que o conceito de Modelo de Ciclo de Vida cobre parcialmente essa idéia.

3.5 Conclusões do Capítulo

Uma ontologia descreve um vocabulário acerca de um domínio, definindo seus conceitos, relações e restrições. Com a evolução da área de processos de software, uma revisão na ontologia de processo de software descrita em (FALBO, 1998) foi necessária. Utilizando o método para construção de ontologias SABiO, a ontologia foi evoluída.

Dentre as principais alterações, destaca-se o conceito de níveis de processos, tratando de processos padrão de uma organização e processos adaptados a um projeto específico. Outra modificação importante é a composição de processos.

ODE é um ambiente de desenvolvimento de software baseado em ontologias. A principal ontologia em que ODE se baseia é a ontologia de processo de software. Com a evolução dessa ontologia, se fez necessário revisar a infra-estrutura de controle de processo de ODE, incluindo a ferramenta de definição de processos, de forma a adequá-la à nova conceituação capturada. O capítulo 4 apresenta a nova versão da ferramenta de definição de processos, construída para torná-la aderente à revisão da ontologia de processo de software.

Capítulo 4

Uma Ferramenta de Apoio à Definição de Processos de Software em ODE

Uma vez revisada a ontologia de processos de software para atender a novas conceituações, tem-se a base necessária para evoluir a infra-estrutura de processos de software de ODE e sua ferramenta de apoio à definição de processos.

A infra-estrutura de processos de software de ODE compreende os pacotes de classes responsáveis por promover a integração de processos de ODE. Esses pacotes são usados pelo ambiente para controlar o acesso a ferramentas, tomando por base um processo definido na ferramenta de definição de processos, bem como para permitir o acompanhamento de projetos, usando ControlPro, a ferramenta de acompanhamento de projetos de ODE (MORO et al., 2005). Neste trabalho, a evolução de ControlPro não foi realizada e, portanto, essa ferramenta não será apresentada.

No que tange à ferramenta de definição de processos, a primeira versão provia apoio apenas para a definição do processo de desenvolvimento de um projeto específico, tomando por base um conhecimento previamente cadastrado no ambiente que era organizado segundo a ontologia de processo de software definida em (FALBO, 1998). Em uma segunda versão, apresentada em (BERTOLLO e FALBO, 2003), essa abordagem foi estendida para permitir a definição de processos padrão e especializados, além da instanciação destes para um projeto específico. Porém, apenas o processo de desenvolvimento continuava sendo contemplado.

Em 2004, ODE foi experimentalmente implantado em uma organização de software que buscava a certificação ISO 9001. A certificação foi obtida e essa mesma organização está atualmente envolvida em um esforço para se certificar CMMI Nível 2, o que está previsto para ocorrer no segundo semestre deste ano.

Do uso, mesmo em caráter experimental, de ODE e mais precisamente de sua ferramenta de definição de processos, oportunidades de melhoria foram apontadas. Considerando a definição de processos, a abordagem em níveis foi apontada como um ponto

muito positivo. Por outro lado, algumas fraquezas, como a inexistência do conceito de composição de processos, também foram detectadas.

Na busca por aperfeiçoar a ferramenta com base no feedback dado, optou-se por, primeiro, evoluir a ontologia que servia de fundamentação para o ambiente, conforme discutido no capítulo anterior, e depois utilizar a nova versão produzida para fundamentar a evolução da infra-estrutura de processos de software do ambiente ODE e sua ferramenta de definição de processos, objeto de discussão deste capítulo que está estruturado da seguinte forma: na seção 4.1 é apresentada a antiga infra-estrutura de processos de software de ODE e sua ferramenta de definição de processos, descrevendo as abordagens utilizadas. A seção 4.2 discute a nova infra-estrutura de processos de ODE, definida com base na revisão da ontologia de processo de software apresentada no capítulo 3, e apresenta, também, uma descrição passo-a-passo da definição de processos nos diferentes níveis de abstração de processo. Finalmente, na seção 4.3, são apresentadas as conclusões do capítulo.

4.1 A Antiga Infra-estrutura de Processos de Software de ODE

A infra-estrutura de processos de software de ODE foi originalmente desenvolvida com base na primeira versão da ontologia de processo proposta em (FALBO, 1998). Essa infra-estrutura era composta de apenas dois níveis, *Conhecimento* e *Controle* (FALBO et al., 2003), que tinham os mesmos propósitos dos homônimos da atual arquitetura conceitual de ODE em três níveis (*Ontologia*, *Conhecimento* e *Controle*), conforme apresentado na seção 2.3.2. A ferramenta de definição de processos, por sua vez, apoiava apenas a definição de processos de projeto. Instâncias pré-cadastradas de classes do nível de *Conhecimento* serviam de guia para essa definição, feita no nível de *Controle*.

A Figura 4.1 apresenta o modelo original do pacote *Conhecimento*.

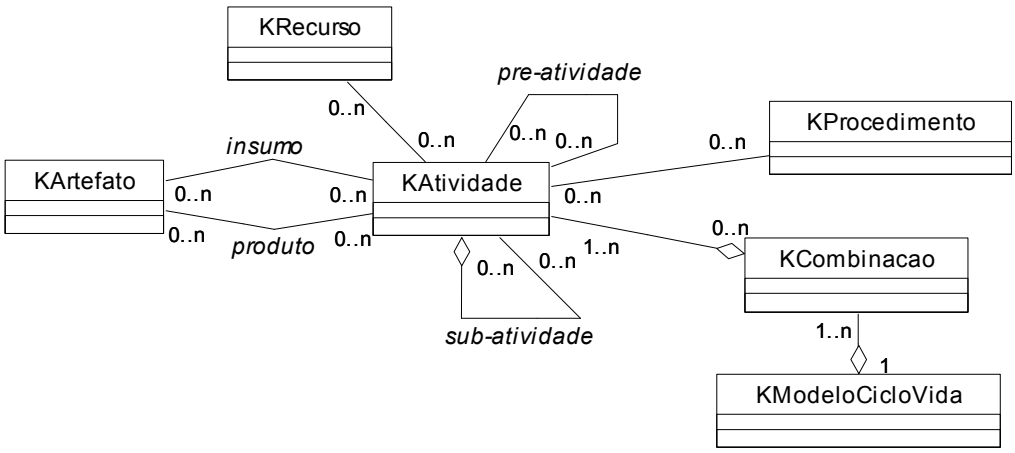


Figura 4.1 – Modelo original do pacote *Conhecimento*.

A classe *KAtividade* descreve o conhecimento acerca das atividades do ambiente. As classes *KArtefato*, *KRecurso* e *KProcedimento* representam, respectivamente, os possíveis artefatos, recursos e procedimentos associados com uma atividade. Um artefato pode ser um insumo ou um produto da atividade.

Um modelo de ciclo de vida, representado pela classe *KModeloCicloVida*, constitui a organização das atividades de um processo de desenvolvimento. Assim, um MCV possui combinações de atividades. Essas combinações (representadas pela classe *KCombinacao*) podem ser seqüenciais, quando as atividades são executadas em uma única seqüência, ou iterativas, quando for necessário definir mais de uma iteração.

A Figura 4.2 apresenta o modelo original do pacote *Controle*.

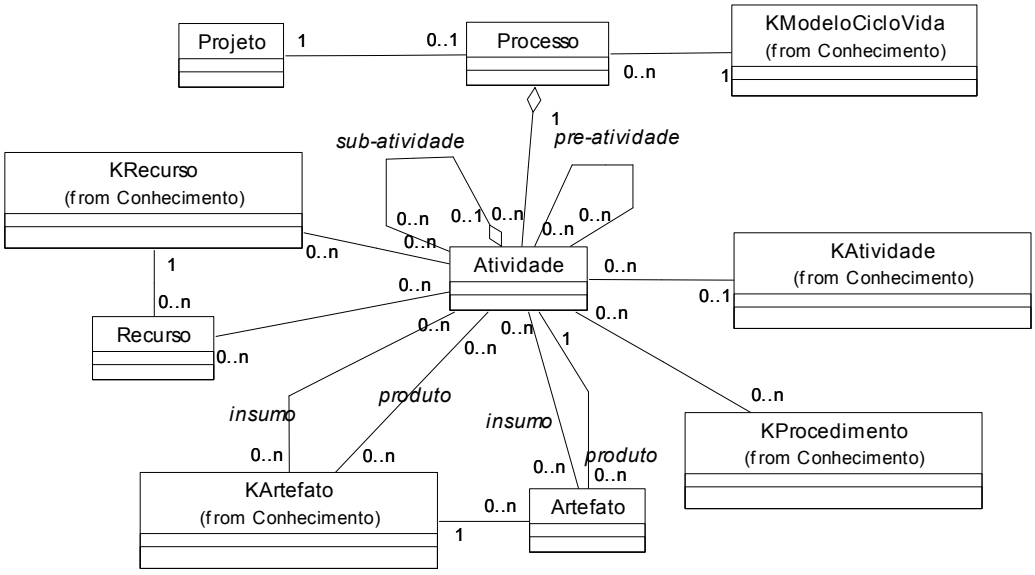


Figura 4.2 – Modelo original do pacote *Controle*.

Todo projeto (classe *Projeto*) possui um processo (classe *Processo*) que determina as atividades (classe *Atividade*) a serem seguidas no desenvolvimento de software. Essas atividades são organizadas segundo um modelo de ciclo de vida.

No contexto de um projeto, atividades consomem e produzem artefatos. Durante a definição do processo são apenas determinados os "tipos" de artefatos (associação com a classe *KArtefato*), que serão realmente elaborados na execução das atividades. De forma análoga, na definição dos recursos necessários para uma atividade, são definidos os tipos de recursos (por exemplo, um Analista de Sistemas), para, num momento posterior, alocar o recurso que vai efetivamente realizar a atividade. Atividades podem seguir um determinado procedimento, sendo representado pela associação com *KProcedimento*. A Figura 4.3 apresenta uma tela da definição de um processo na primeira versão da ferramenta.

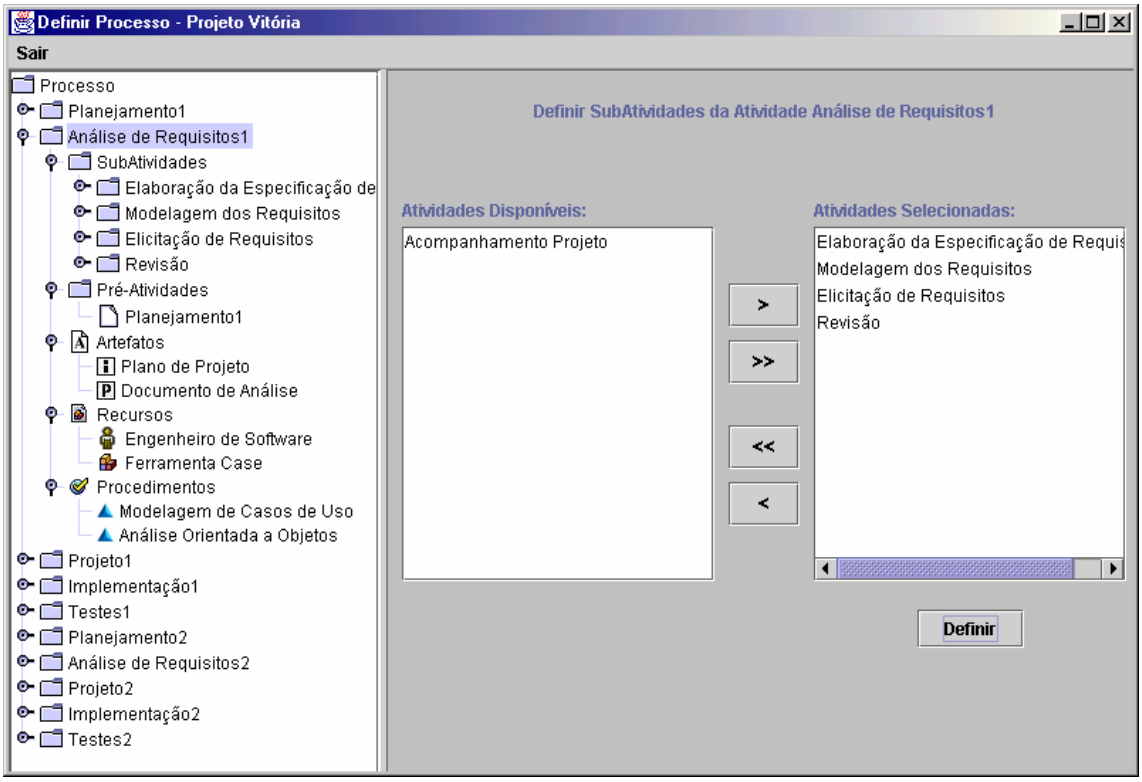


Figura 4.3 – Definição de processos na primeira versão da ferramenta.

Com o crescente reconhecimento da importância da definição de processos padrão, passou-se a utilizar as classes de *Conhecimento* para descrever processos padrão. Contudo, essa abordagem só permitia a definição de um único processo padrão, bem como este era somente implicitamente definido na ferramenta.

Essas limitações motivaram o desenvolvimento de uma segunda versão da infra-estrutura de processos de software de ODE, apresentada em (BERTOLLO e FALBO, 2003).

Nessa versão, foi incorporada a definição explícita de processos padrão, incluindo processos especializados. Instâncias pré-cadastradas de *Conhecimento* eram usadas como guia para a definição do processo padrão. No caso de processos especializados e processos de projeto, o nível superior correspondente (processo padrão organizacional ou processo especializado, respectivamente) era usado como guia. Vale destacar que, ainda que essa versão contemplasse processos padrão e especializados, a ontologia que fundamentava a infra-estrutura de processos de software de ODE não tratava desses conceitos e, portanto, a infra-estrutura não estava aderente à filosofia apregoada no Projeto ODE.

A Figura 4.4 apresenta o modelo da segunda versão do pacote *Conhecimento*, destacando as alterações feitas.

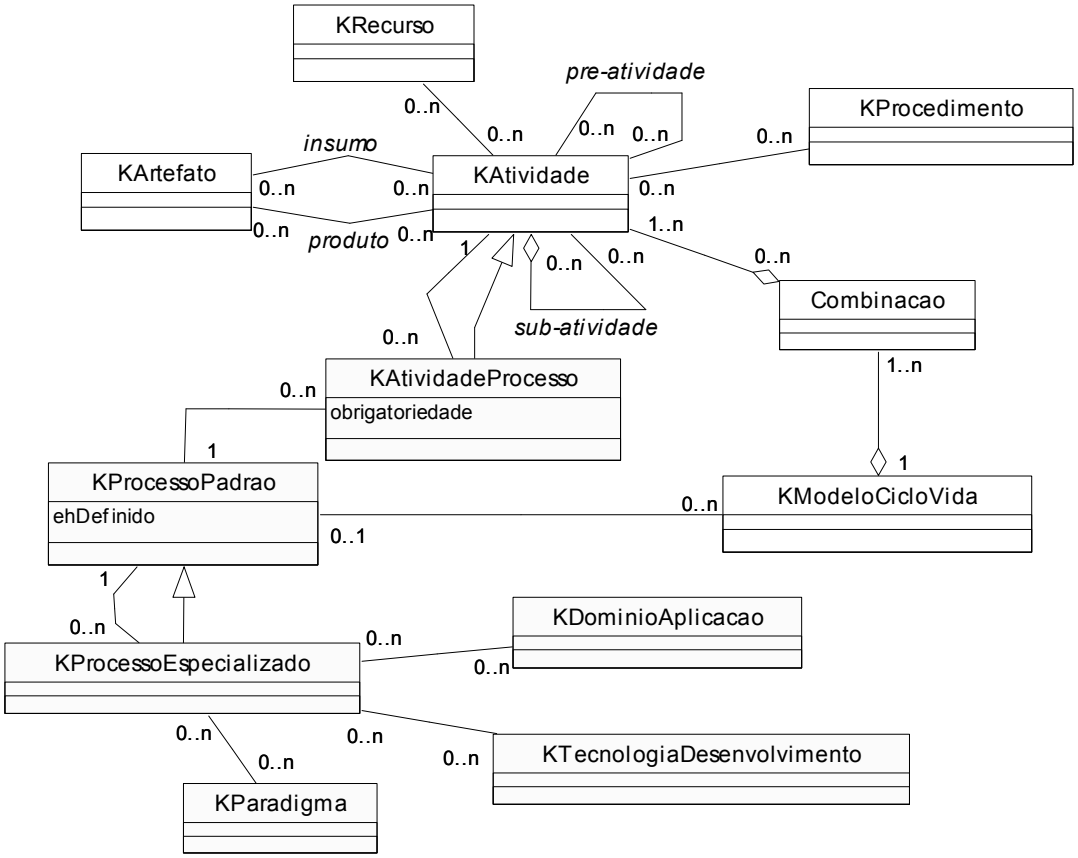


Figura 4.4 – Modelo de *Conhecimento* da segunda versão da ferramenta.

Na evolução do modelo foram criadas classes que contemplam a definição de processos padrão e especializados. Foi adicionada a classe *KProcessoPadrao,* que representa os processos padrão de uma organização. Cada processo padrão pode ser especializado, incluindo características relacionadas com paradigmas, tecnologias e domínios específicos. Os processos especializados são representados pela classe *KProcessoEspecializado* e suas

características representadas pelos relacionamentos com as classes *KParadigma*, *KTecnologiaDesenvolvimento* e *KDomínioAplicacao*.

Devem ser definidas, ainda, as atividades pertencentes a um processo padrão ou especializado. Essas atividades são representadas pela classe *KAtividadeProcesso* que herda de *KAtividade*. Cada atividade possui seus insumos e produtos, assim como os recursos e procedimentos necessários para sua execução. A obrigatoriedade de uma atividade indica que, ao se definir um processo de projeto a partir de um processo padrão ou especializado, essa atividade não pode ser excluída do processo.

Um processo padrão ou especializado possui, ainda, os modelos de ciclo de vida associados ao processo. Esses modelos são criados durante a definição do processo padrão ou especializado.

A Figura 4.5 apresenta a definição de processos padrão ou especializados na segunda versão da ferramenta de definição de processos de ODE.

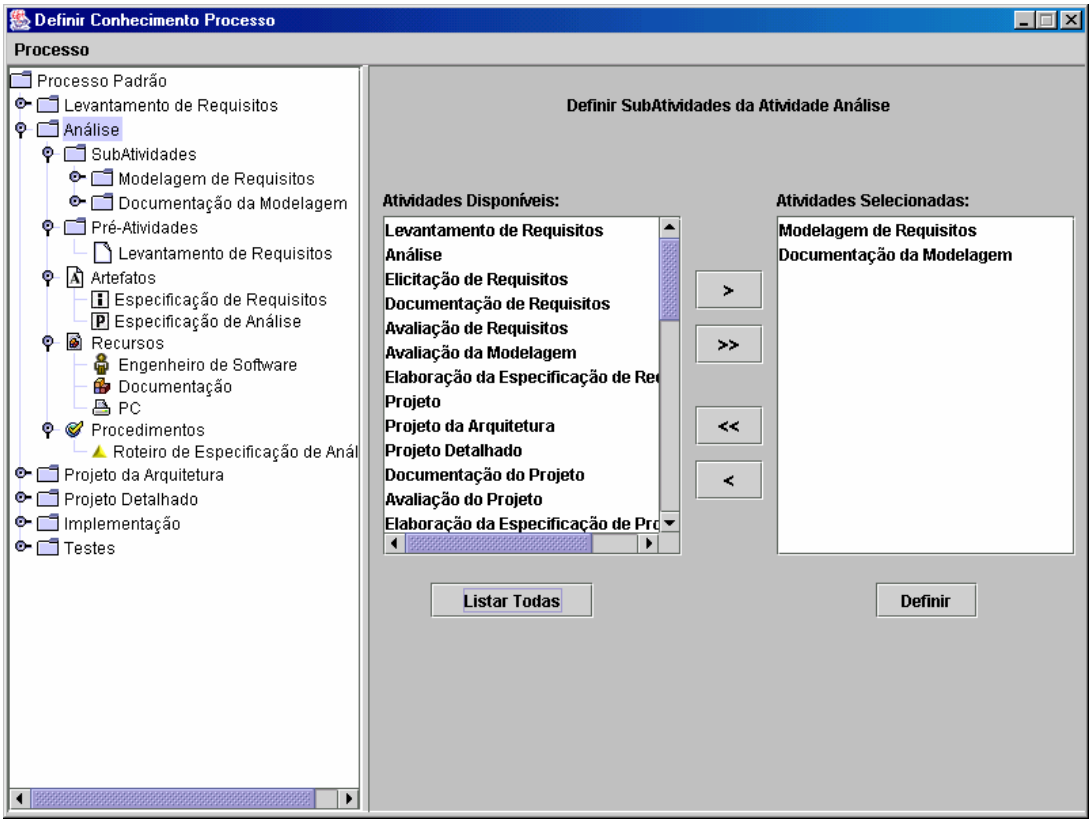


Figura 4.5 – Definição de processos padrão e especializado.

A Figura 4.6 apresenta o modelo de classes do pacote *Controle*, destacando as alterações em relação à versão anterior da ferramenta.

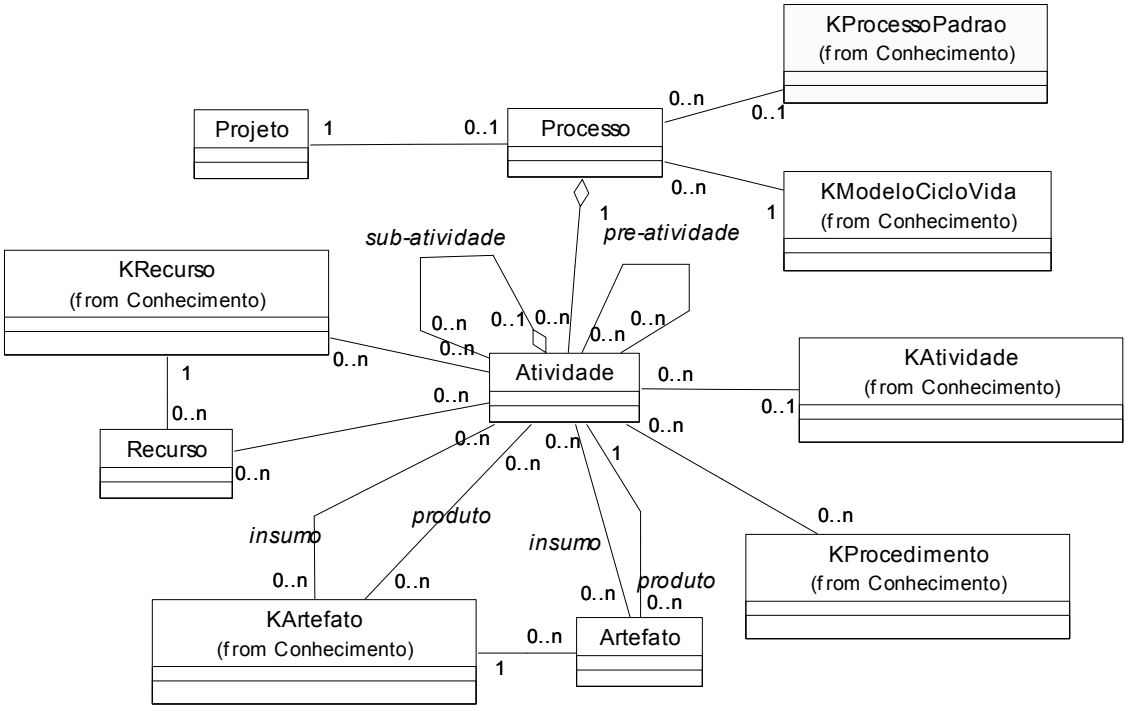


Figura 4.6 – Modelo de *Controle* da segunda versão da ferramenta.

Um processo de projeto (representado pela classe *Processo*) pode ser definido a partir de um processo padrão ou de um processo especializado (representados pela classe *KProcessoPadrao*), ou a partir do repositório de conhecimento. Dessa forma, o processo de projeto definido pode ter, ou não, um conhecimento de processo padrão relacionado. Essa foi a única alteração no modelo de *Controle*, pois a segunda versão da ferramenta teve o objetivo de criar o nível de processo padrão.

Conforme brevemente comentado na introdução deste capítulo, a partir de 2004, quando ODE foi experimentalmente implantado em uma organização de software que buscava a certificação ISO 9001, e mais tarde CMMI Nível 2, uma série de pontos fortes e oportunidades de melhoria (OM) foram apontados. Dentre os pontos positivos, merece destaque a abordagem de definição de processos em níveis, juntamente com o fornecimento de orientações, através dos objetos de *Conhecimento*, para a definição de processos padrão Vale destacar que esses ativos de processo pré-registrados eram definidos com base na norma ISO/IEC 12207. Além disso, para considerar aspectos organizacionais, ativos de processo típicos da organização também podiam ser previamente cadastrados. As principais oportunidades de melhoria apontadas foram:

OM1. A ferramenta não permitia que um processo fosse definido como sendo composto de outros processos. Um processo era decomposto somente em

atividades, não sendo possível capturar que um processo de software é, de fato, decomposto em outros processos que se inter-relacionam. Muitas normas e modelos de qualidade propõem a definição de diversos processos, em que cada processo possui um propósito e resultados esperados específicos. Assim, os processos deveriam ser definidos, executados e controlados de forma independente. Por exemplo, um Processo de Gerência de Projeto pode ser controlado independentemente de um Processo de Desenvolvimento de Software. Uma grande limitação da segunda versão da ferramenta de definição de processos de ODE estava, então, na impossibilidade de definir mais de um processo, sendo necessário coexistir atividades de gerência de projeto, desenvolvimento de software, garantia da qualidade etc, em um único processo.

OM2. Uma vez que ODE não tratava múltiplos processos, não havia como tratar a interação entre processos. Assim, tudo que se podia fazer era estabelecer uma ordem de precedência entre as atividades de diferentes processos, todas colocadas como parte do grande processo de software definido.

OM3. Não era permitida a classificação de processos, tal como faz a norma ISO/IEC 12207, que classifica processos em processos fundamentais, organizacionais e de apoio.

OM4. Na definição de processos, algumas informações acerca de artefatos não eram diretamente definidas, tais como o roteiro e a ferramenta a serem usados na elaboração dos mesmos. Registrava-se, apenas, que um procedimento e uma ferramenta eram usados em uma atividade. Mas se a atividade produzia mais de um artefato, não era possível saber qual havia usado o quê.

Com base no feedback dado, um projeto de melhoria da infra-estrutura de processos de ODE foi estabelecido. Nesse projeto percebeu-se que, de fato, a própria ontologia que servia de fundamentação para o ambiente precisava evoluir. A área de processos de software tinha evoluído bastante e, portanto, decidiu-se evoluir a ontologia, conforme apresentado no Capítulo 3, e depois utilizar a nova versão produzida para fundamentar a evolução da infra-estrutura de processos de software do ambiente ODE.

4.2 A Nova Infra-estrutura de Processos em ODE

A nova versão da infra-estrutura de processos de software de ODE passou a seguir fielmente o esquema de anotação conceitual discutido no Capítulo 3. Além disso, a arquitetura interna da infra-estrutura foi remodelada, para separar os aspectos que descrevem processos padrão (pacote *ProcessoPadrao*), incluindo processos especializados, dos objetos de conhecimento de processo (pacote *ConhecimentoProcesso*) e aqueles que representam ativos de um processo de projeto (pacote *ControleProcesso*). A Figura 4.7 apresenta o novo diagrama de pacotes da ferramenta.

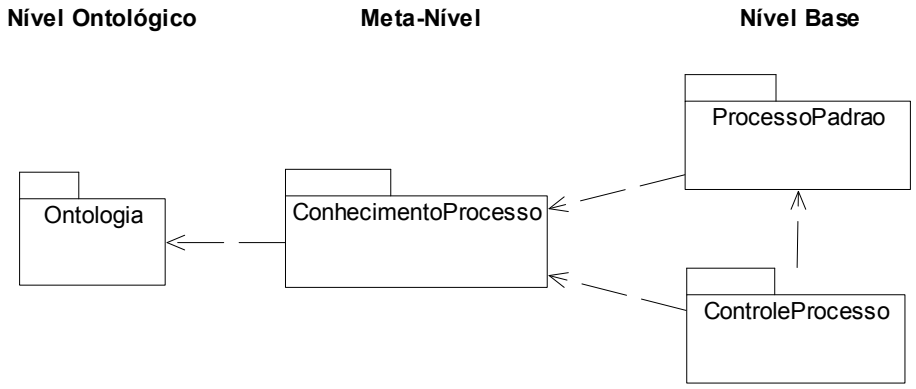


Figura 4.7 – Diagrama de Pacotes.

O Nível Ontológico é responsável pela descrição das ontologias. O pacote Ontologia contém as classes que definem uma ontologia de um certo domínio e seus elementos. As instâncias das classes do nível ontológico servem de base para a definição das classes dos outros níveis. No caso da ontologia de processo de software, conceitos como Processo de Software e Atividade estão definidos nesse nível como instâncias de um classe Conceito (FALBO et al., 2004a).

Conforme discutido no Capítulo 2, o Meta-Nível abriga as classes que descrevem o conhecimento em relação a um domínio de aplicação. Suas classes são derivadas das ontologias e as instâncias dessas classes atuam no papel de conhecimento sobre os objetos do nível base. Elas constituem o conhecimento do ambiente, que pode ser utilizado por qualquer ferramenta que o compõe. No caso da infra-estrutura de processos de software, o conhecimento tratado é precisamente aquele relacionado a processos de software (*ConhecimentoProcesso*)

O Nível Base define as classes responsáveis por implementar as aplicações no contexto do ambiente (funcionalidades da infra-estrutura e suas ferramentas). Essas classes são também derivadas das ontologias, mas tipicamente incorporam detalhes não descritos por elas, necessários para implementar as aplicações. Nesse nível, no caso da infra-estrutura de processos de software de ODE, encontram-se dois pacotes: *ProcessoPadrao* e *ControleProcesso*. O pacote *ProcessoPadrao* contém as classes relacionadas com a definição dos processos padrão de uma organização. Esses processos servem de base para a definição de processos para projetos específicos, que consideram as características particulares do projeto. O pacote *ControleProcesso* contém as classes que modelam a definição de processos de projeto.

Na segunda versão da ferramenta, o processo padrão era modelado a partir de classes do meta-nível. Na evolução da ferramenta proposta neste trabalho, essa concepção foi alterada. As classes do nível base descrevem objetos concretos do mundo real. Assim, o processo padrão passou a ser modelado no nível base, por representar objetos reais de um processo definido para uma determinada organização. Esses objetos possuem um tipo, que descrevem o conhecimento acerca do objeto e são representados pelas classes do meta-nível.

Uma vez definida uma ontologia, é possível derivar modelos que compõem os outros dois níveis da arquitetura conceitual de ODE, utilizando parcialmente a abordagem de derivação proposta em (FALBO et al., 2002), na qual conceitos e relações são mapeados em classes e associações em um modelo de objetos, propriedades de conceitos e relações são mapeadas em atributos das classes originadas a partir do respectivo conceito/relação e axiomas, por sua vez, são mapeados em métodos.

A Tabela 4.1 apresenta os conceitos da ontologia de processo de software e sua derivação para o Meta-Nível e o Nível Base. De forma análoga, as tabelas 4.2, 4.3 e 4.4 apresentam os mapeamentos das ontologias de atividade, recurso e procedimento, respectivamente.

Tabela 4.1 – Derivação de conceitos da ontologia de processo de software.

Nível Ontológico	Meta-Nível	Nível Base	
		Pct *ProcessoPadrao*	**Pct *ControleProcesso***
Processo de Software	KProcesso		
Processo Padrão		ProcessoPadrao ProcessoPadraoGeral ProcessoPadraoEspecifico	
Processo de Projeto			ProcessoProjeto ProcessoProjetoGeral ProcessoProjetoEspecifico
Interação	KInteracaoProcesso KTipoInteracao		
Categoria de Processo	KCategoriaProcesso		
Paradigma	KParadigma		
Domínio de Aplicação	KDominioAplicacao		
Tipo de Software	KTipoSoftware		
Modelo de Ciclo de Vida	KModeloCicloVida	ModeloCicloVidaProcessoPadrao	
Combinação		Combinacao	
Projeto			Projeto

Tabela 4.2 – Derivação de conceitos da ontologia de atividade.

Nível Ontológico	Meta-Nível	Nível Base	
		Pct *ProcessoPadrao*	**Pct *ControleProcesso***
Atividade	KAtividade	AtividadeProcessoPadrao	Atividade
Artefato	KArtefato		Artefato

Tabela 4.3 – Derivação de conceitos da ontologia de recurso.

Nível Ontológico	Meta-Nível	Nível Base	
		Pct *ProcessoPadrao*	**Pct *ControleProcesso***
Recurso	KRecurso		Recurso
Recurso Humano	KRecursoHumano		
Recurso de Hardware	KRecursoHardware		RecursoHardware
Recurso de Software	KRecursoSoftware		RecursoSoftware
Sistema de Apoio	KSistemaApoio		SistemaApoio
Ambiente de Software	KAmbienteSoftware		AmbienteSoftware
Ferramenta de Software	KFerramentaSoftware		FerramentaSoftware

Tabela 4.4 – Derivação de conceitos da ontologia de procedimento.

Nível Ontológico	Meta-Nível	Nível Base	
		Pct *ProcessoPadrao*	**Pct *ControleProcesso***
Procedimento	KProcedimento		
Técnica	KTecnica		
Método	KMetodo		
Fluxo de Trabalho	KFluxoTrabalho		
Diretriz	KDiretriz		
Mentor de Ferramenta	KMentorFerramenta		
Norma	KNorma		
Roteiro	KRoteiro		

Conforme apresentado nas tabelas 4.1 a 4.4, a derivação dos conceitos do nível ontológico pode originar classes nos diversos níveis da arquitetura conceitual de ODE. Dessa forma, durante o processo de derivação, um determinado conceito pode derivar uma classe somente no nível de conhecimento, somente no nível base ou em ambos os níveis (FALBO et al., 2004a).

Classes em ambos os níveis

Muitas vezes, um conceito de uma ontologia é necessário nos dois níveis: no nível de conhecimento, determinando o tipo dos objetos concretos do mundo real, e no nível de

aplicação, representando os próprios objetos do mundo real. Nessa situação, os objetos do nível de conhecimento são utilizados para descrever os objetos do nível de aplicação. Como exemplo, tomemos o conceito *Atividade,* da ontologia de atividade, que dá origem a três classes: *KAtividade* (meta-nível); *AtividadeProcessoPadrao* e *Atividade* (nível base). As classes do nível base descrevem atividades concretas de um projeto (instâncias de *Atividade*) ou de um processo padrão (instâncias de *AtividadeProcessoPadrao*) e se utilizam de instâncias da classe *KAtividade* para descrever tipos de atividade. Por exemplo, a atividade *Planejamento Inicial do projeto X* é uma atividade do tipo *Planejamento*.

Também pode ocorrer de um conceito da ontologia derivar classes em apenas um dos pacotes do nível base. Isso ocorre quando o conceito é relevante apenas no contexto de uma organização, para o caso da classe ser do pacote *ProcessoPadrao*, ou no contexto de um projeto, no caso da classe pertencer ao pacote *ControleProcesso*. Como exemplo, podemos citar o conceito *Modelo de Ciclo de Vida* que é definido para um processo padrão e o conceito *Artefato*, que se torna uma instância real apenas em um projeto.

Classes somente no Meta-Nível

Como o meta-nível descreve o conhecimento acerca dos objetos de uma aplicação, determinando o tipo do objeto, alguns conceitos podem ser derivados apenas para esse nível. Como exemplo pode-se citar o conceito *Procedimento* da ontologia de procedimento. Instâncias desse conceito incluem, dentre outros, *Análise Estruturada*, *Análise Orientada a Objetos*, *Inspeção de Código* etc. Essas instâncias são suficientes para ambos os níveis de conhecimento e de aplicação.

Tomando exemplos de atividades, é possível dizer, no nível de conhecimento, que a atividade de *Especificação de Requisitos* pode adotar como procedimento para sua execução o método da *Análise Orientada a Objetos*. Por outro lado, pode-se dizer, também no nível de aplicação, que a atividade *Especificação de Requisitos Preliminar do Projeto X* é conduzida seguindo o método da *Análise Orientada a Objetos*. Ou seja, apenas uma classe é suficiente para tratar os dois níveis. Neste caso, como a classe derivada é importante para o nível de conhecimento, isto é, o que está sendo descrito por ela é, de fato, um conhecimento sobre os procedimentos que podem ser adotados no desenvolvimento de software, ela é criada no nível de conhecimento (*KProcedimento*). Como o nível base tem acesso ao nível de conhecimento, ele também pode utilizá-la quando necessário.

Classes somente no Nível Base

Esta situação ocorre quando os conceitos possuem instâncias relevantes apenas no nível de aplicação, como o caso em que não é necessário um "tipo", mas são necessários apenas os objetos concretos. Tomando o exemplo da ontologia de processo de software, este é o caso do conceito *Projeto*. Quando se fala em um projeto, está se referindo a um projeto específico tal como o *Projeto X*, que possui as atividades *Planejamento Inicial do Projeto X*, *Especificação de Requisitos Preliminar do Projeto X* etc. Assim, esse conceito é derivado somente para o nível base, dando origem à classe *Projeto*.

4.2.1 O Pacote *ConhecimentoProcesso*

Com a evolução da ferramenta, o pacote *ConhecimentoProcesso* passou a ter os aspectos relacionados apenas ao conhecimento genérico acerca de processos de software. A Figura 4.8 apresenta o modelo parcial do pacote *ConhecimentoProcesso*, destacando as alterações realizadas na nova versão.

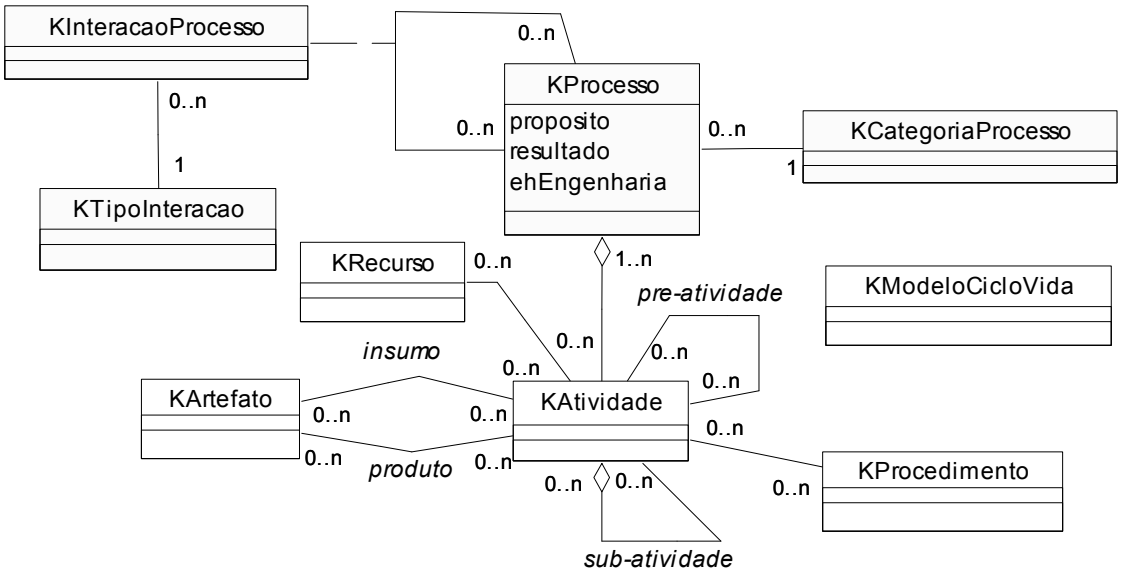


Figura 4.8 – Modelo Parcial do Pacote *ConhecimentoProcesso*.

A classe *KProcesso*, derivada do conceito *Processo de Software*, representa tipos de processos, tais como Processo de Desenvolvimento de Software, Processo de Gerência de Projeto etc. Cada processo possui seus propósitos e resultados específicos e é classificado em uma categoria de processo (*KCategoriaProcesso*). Por ser um ambiente de desenvolvimento

de software, o principal objetivo de ODE é apoiar a construção de software. Dessa forma, embora seja permitida a definição de vários processos de diferentes tipos, é necessário que um processo defina a espinha dorsal do desenvolvimento (ou manutenção), descrevendo as atividades que elaboram os artefatos que darão origem ao produto de software (ou à sua alteração). Esse processo possui a característica peculiar de ser o cerne do processo do software e é definido como sendo um processo de engenharia. Todo processo definido em ODE possui um, e apenas um, processo de engenharia.

Conforme descrito na ontologia de processo de software, os processos interagem entre si. Em ODE, a interação entre processos é definida através do processo de engenharia. Dessa forma, os processos que não são de engenharia interagem com o processo de engenharia de alguma forma. Vale ressaltar que essa abordagem é uma limitação da versão atual da ferramenta de definição de processo, pois processos que não são de engenharia também podem interagir entre si. A classe *KTipoInteracao* define os tipos de interação definidos na infra-estrutura de processos de ODE, a saber:

(i) Seqüencial: os processos têm uma relação de término-início, ou seja, um processo só pode ser iniciado após a conclusão do outro. Por exemplo, um Processo de Manutenção inicia-se após a conclusão do Processo de Desenvolvimento de Software. A Figura 4.9 ilustra esse tipo de interação;

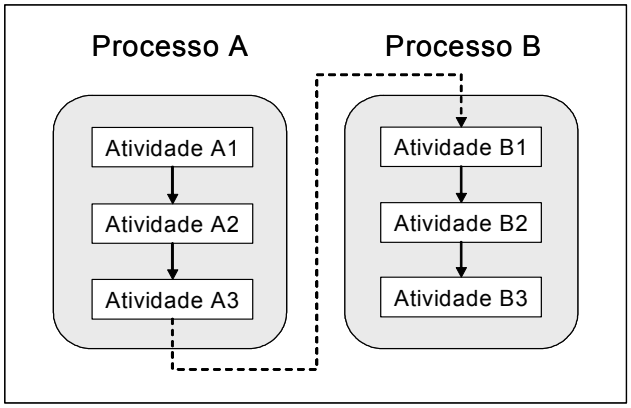


Figura 4.9 – Interação Seqüencial.

(ii) Paralelo Independente: um processo B é iniciado após a conclusão de uma determinada atividade de outro processo A. Contudo, B é executado independentemente de A. Como exemplo, após a conclusão da atividade de Testes do Processo de Desenvolvimento de Software, inicia-se um Processo de Treinamento com os usuários-chave, enquanto segue a

atividade de Implantação do sistema no ambiente de produção. A Figura 4.10 ilustra esse tipo de interação;

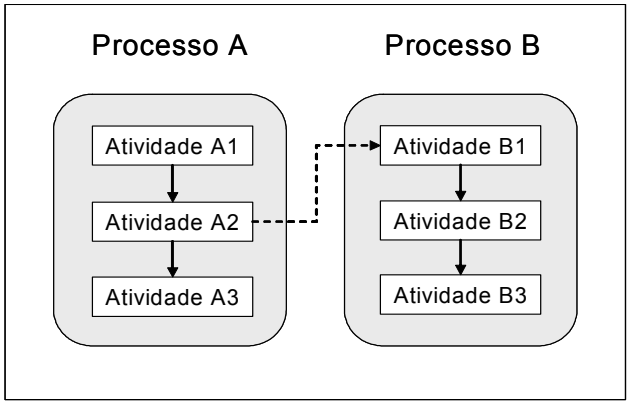


Figura 4.10 – Interação Paralelo Independente.

(iii) Paralelo Dependente: idem o tipo anterior, contudo o processo B pode influenciar o processo A. Por exemplo, o Processo de Desenvolvimento de Software inicia-se após a conclusão da atividade de Planejamento do Processo de Gerência de Projeto. Por ser uma interação dependente, o Processo de Gerência de Projeto pode interagir com o Processo de Desenvolvimento de Software através de uma atividade de Acompanhamento de Projeto, por exemplo, que pode depender, por sua vez, de atividades do Processo de Desenvolvimento. A Figura 4.11 ilustra a interação paralelo dependente;

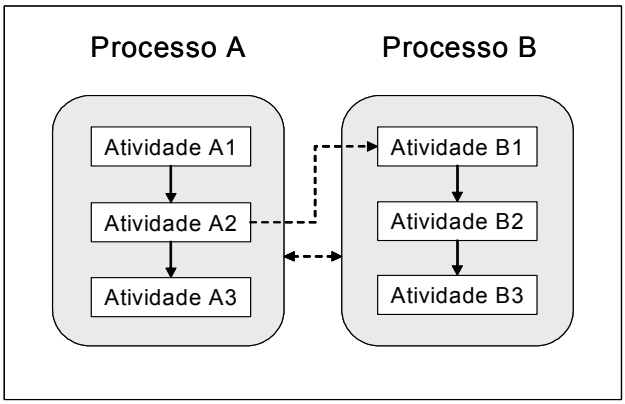


Figura 4.11 – Interação Paralelo Dependente.

(iv) Pontual: um processo A interage com um processo B em um ponto específico. Como exemplo, após a atividade de Análise de Requisitos do Processo de Desenvolvimento de Software ocorre uma atividade de Verificação e Validação da Análise, parte de um Processo de Verificação e Validação. A Figura 4.12 ilustra a interação pontual;

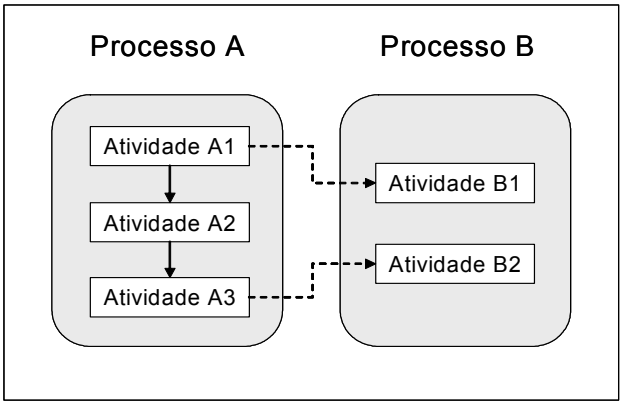


Figura 4.12 – Interação Pontual.

(v) Sob-demanda: um processo A interage com um processo B sob-demanda, ou seja, não existe um momento pré-definido. Por exemplo, o Processo de Gerência de Configuração interage com Processo de Desenvolvimento de Software em qualquer ponto que se deseje colocar um artefato sob gerência de configuração. A Figura 4.13 ilustra esse tipo de interação.

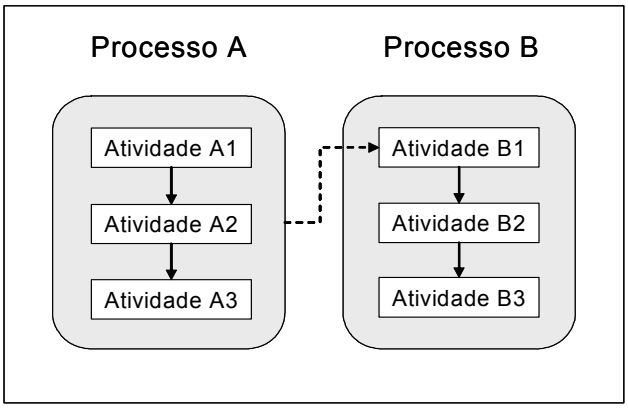


Figura 4.13 – Interação Sob-demanda.

A classe *KModeloCicloVida*, apesar de não ser uma classe nova, teve sua concepção alterada. Um modelo de ciclo de vida (MCV) contém a organização das atividades de um processo na execução de um projeto. Na versão anterior da ferramenta, os possíveis modelos de ciclo de vida eram definidos no pacote *Conhecimento*. Na nova versão, os modelos de ciclo de vida são definidos no pacote *ProcessoPadrao*, criando combinações de atividades definidas para o processo padrão que está sendo definido. Ainda se mantém a concepção de tipos de combinação, a saber, seqüencial, quando atividades são executadas em uma única seqüência, ou iterativo, quando atividades podem ser executadas em várias iterações.

Contudo, agora as combinações de atividades são construídas tomando-se por base apenas as macro-atividades do processo de engenharia.

Considerando as classes já existentes nas versões anteriores da ferramenta, a classe *KAtividade* representa o conhecimento acerca das atividades que podem fazer parte de um determinado processo. Para uma atividade neste nível, são definidos os possíveis artefatos (*KArtefato*) consumidos ou produzidos, recursos (*KRecurso*) utilizados e procedimentos (*KProcedimento*) adotados que podem ser selecionados na definição do processo.

4.2.2 O Pacote *ProcessoPadrao*

O pacote *ProcessoPadrao* contém as classes envolvidas na definição de processos padrão de uma organização. A Figura 4.14 apresenta um modelo de classes parcial desse pacote. A classe *ProcessoPadrao* representa os processos padrão e especializados. Um processo pode ser especializado levando em consideração um tipo de software (*KTipoSoftware*), um domínio de aplicação (*KDominioAplicacao*) ou um paradigma (*KParadigma*).

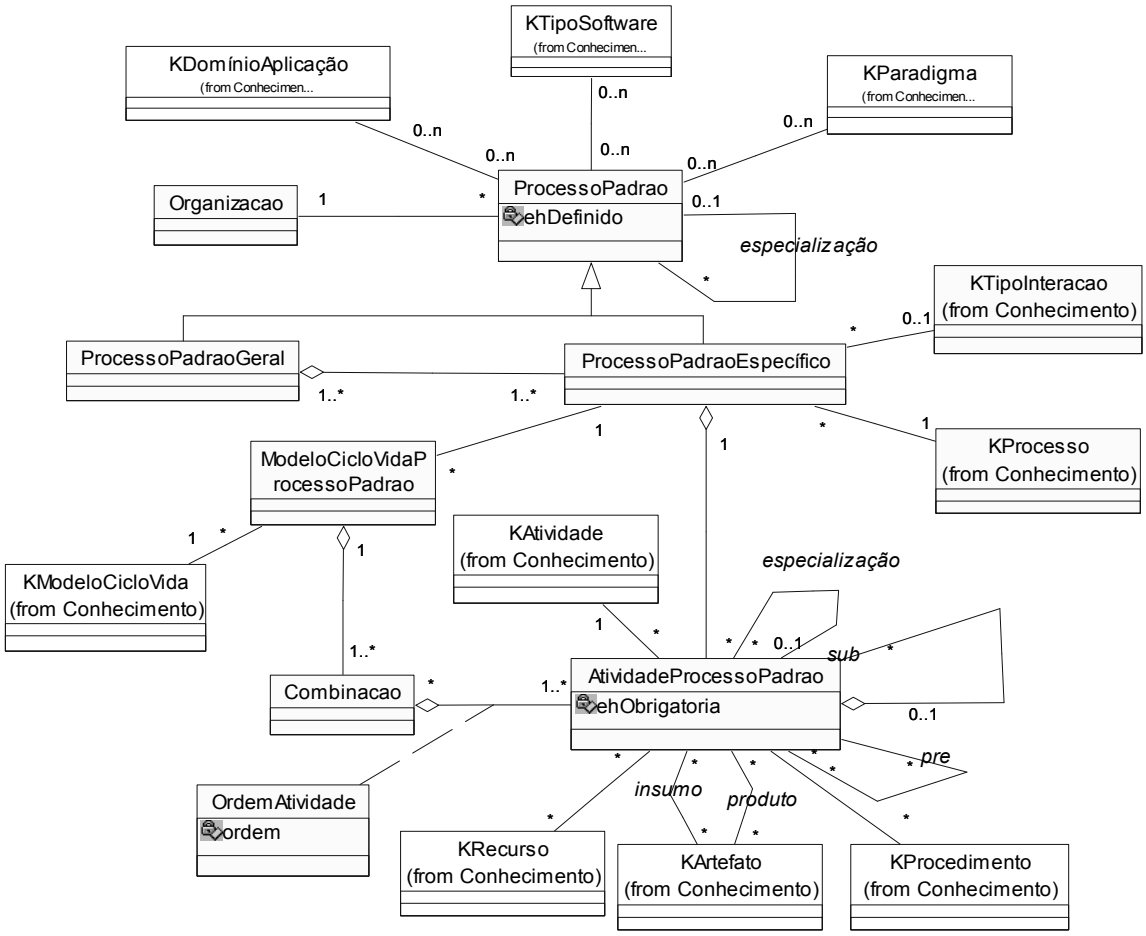


Figura 4.14 – Modelo Parcial do Pacote *ProcessoPadrao*.

Um processo padrão pode ser composto por processos ou por atividades. Quando um processo padrão é composto por processos (ditos sub-processos), ele deve ser uma instância da classe *ProcessoPadraoGeral*. Quando um processo padrão é composto por atividades, deve ser uma instância da classe *ProcessoPadraoEspecifico*. Em ODE, é permitido apenas um nível de composição de processos. Tomemos como exemplo, a definição do processo padrão do Laboratório de Engenharia de Software da UFES (LabES). Foi definido um processo de software padrão, chamado Processo LabES, que é composto por três processos padrão específicos: Desenvolvimento de Software, Gerência de Projeto e Garantia da Qualidade. Uma restrição a ser respeitada é que só é possível ter um único processo padrão de engenharia, que pode ser de desenvolvimento ou manutenção. O processo de engenharia é considerado a espinha dorsal do processo padrão geral definido. Vale destacar que um processo padrão geral pode ser composto por processos específicos definidos para um outro processo geral, ou seja,

é possível reutilizar processos já definidos. Nesse caso, o processo específico não pode ser modificado.

Como descrito na seção anterior, a interação de processos ocorre entre o processo de engenharia e os processos de não-engenharia, através da associação com *KTipoInteracao*. A sistemática para definição das interações entre os processos que compõem o processo padrão é diferente para cada tipo. Em alguns tipos de interação é suficiente escolher o tipo de interação, em outros é necessário o ponto exato da interação através da definição de precedência entre as atividades dos processos. Seguem as regras para definição, sempre considerando A o processo de Engenharia e B o processo Não-Engenharia:

(i) *Seqüencial*: o Gerente de Processo define que B possui uma interação *Seqüencial* com A.

(ii) *Paralelo Independente:* o Gerente de Processo define que B possui uma interação *Paralelo Independente* com A. Contudo, é necessário definir explicitamente o momento da interação. Dessa forma, na definição das pré-atividades de uma determinada atividade, tanto do processo A quanto de B, são sugeridas atividades de A e B.

(iii) *Paralelo Dependente:* Idem ao tipo *Paralelo Independente*.

(iv) *Pontual:* o Gerente de Processo define que B possui uma interação *Pontual* com A. Nesse tipo de interação também é necessário definir o momento da interação. Assim, na definição das pré-atividades de uma atividade do processo B, são sugeridas as atividade dos processos A e B, definindo o ponto exato da interação.

(v) *Sob-demanda:* o Gerente de Processo define que B possui uma interação *Sob-demanda* com A.

Para cada processo padrão específico, deve-se definir as atividades que o compõem. A classe *AtividadeProcessoPadrao* define as atividades que compõem um processo padrão específico. Essas atividades podem ser definidas como sendo obrigatórias ou não. Se for obrigatória, a atividade não pode ser excluída em uma especialização desse processo. Para cada atividade, deve-se definir suas pré-atividades, sub-atividades, artefatos, recursos e procedimentos. A Figura 4.15 apresenta a tela de definição do processo padrão da nova versão da ferramenta.

Figura 4.15 – Definição de Processo Padrão.

Finalmente, deve-se definir os possíveis modelos de ciclo de vida do processo padrão. Isso é necessário para garantir uma restrição de um axioma da ontologia de processos de software que define que se um processo de software referencia um modelo de ciclo de vida (MCV), as atividades que compõem o MCV devem compor também o processo de software. Vale ressaltar que somente processos padrão de engenharia têm modelos de ciclo de vida associados. As atividades de um modelo de ciclo de vida são agrupadas em combinações seqüenciais ou iterativas, ordenadas em conformidade com a precedência entre as atividades estabelecida no processo padrão correspondente. Por exemplo, suponha um processo padrão com as seguintes atividades: Especificação de Requisitos, Análise, Projeto, Implementação e Teste, nessa ordem de precedência. Pode-se definir um modelo de ciclo de vida incremental com duas combinações: a primeira seqüencial, incluindo Especificação de Requisitos e Análise; a segunda iterativa, incluindo Projeto, Implementação e Teste.

4.2.3 O Pacote *ControleProcesso*

O pacote *ControleProcesso*, parcialmente mostrado na Figura 4.16, trata da definição de processos de projeto. Um processo de projeto é definido para um projeto (classe *Projeto*), tomando por base um processo padrão (classe *ProcessoPadrao*) e um modelo de ciclo de vida (*ModeloCicloVidaProcessoPadrao*) para o processo de engenharia do processo padrão. De forma análoga ao processo padrão, um processo de projeto pode ser composto por processos ou por atividades. Quando um processo de projeto é composto por processos, deve ser uma instância da classe *ProcessoProjetoGeral*, definido com base em um *ProcessoPadraoGeral*. Quando o processo é composto por atividades, deve ser uma instância da classe *ProcessoProjetoEspecifico*, tendo como base um *ProcessoPadraoEspecifico*.

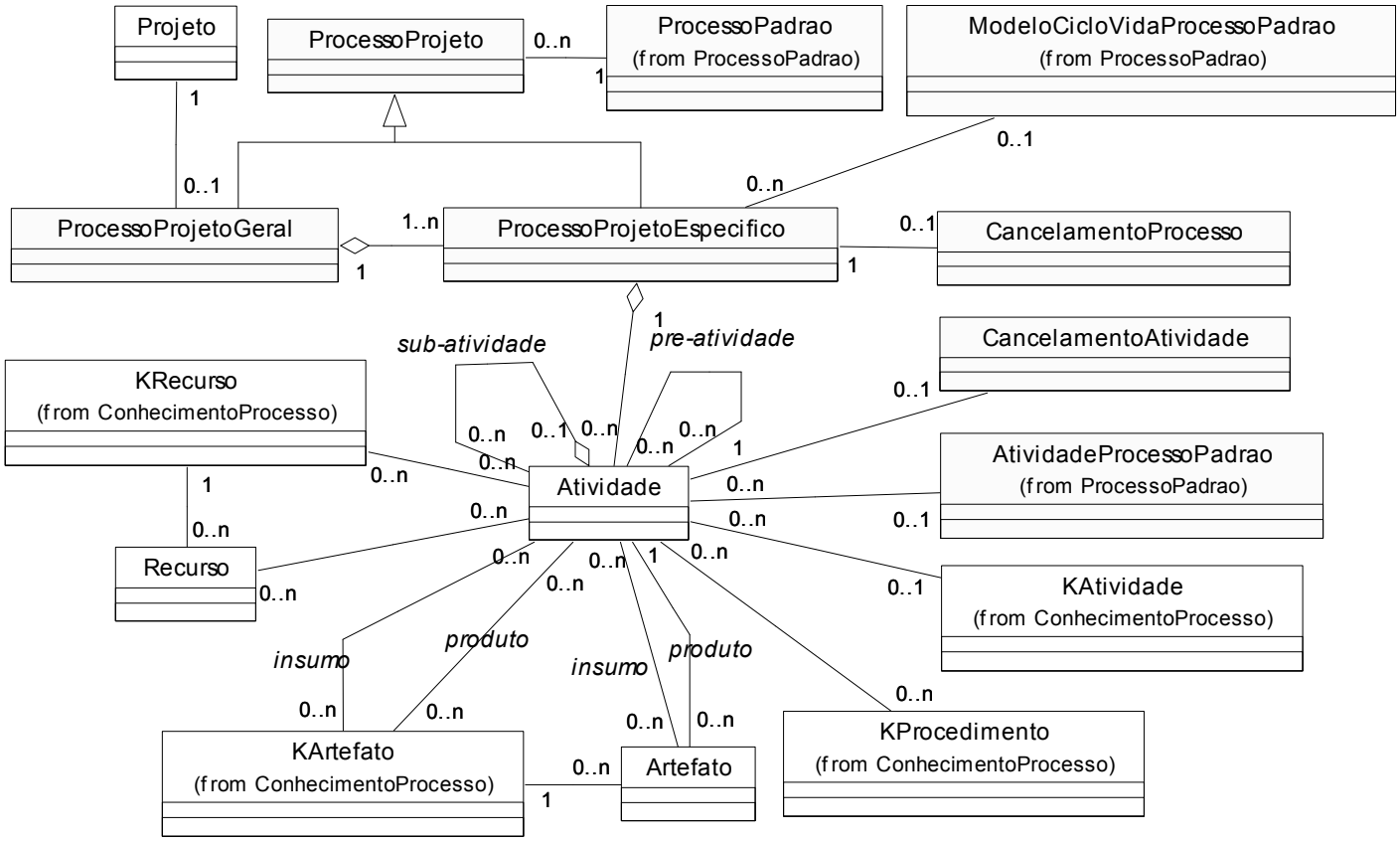


Figura 4.16 – Modelo Parcial do Pacote *ControleProcesso*.

Escolhidos o processo padrão e o modelo de ciclo de vida, o processo de projeto é automaticamente montado com os ativos de processo definidos no processo padrão. Os tipos de interação definidos influenciam a instanciação do processo de projeto da seguinte forma, sempre considerando A o processo de Engenharia e B o processo Não-Engenharia:

(i) *Seqüencial*: as macro-atividades do processo A que não possuem pós-atividades são automaticamente definidas como pré-atividades das macro-atividades de B que não possuem pré-atividades.

(ii) *Paralelo Independente:* se a atividade do processo A estiver em uma combinação iterativa do modelo de ciclo de vida escolhido, serão instanciadas n atividades de todo o processo B, onde n é o número de iterações.

(iii) *Paralelo Dependente:* Idem ao tipo *Paralelo Independente*.

(iv) *Pontual:* as atividades do processo B são instanciadas conforme número de iteração do modelo de ciclo de vida escolhido. Por exemplo, suponha a atividade V&V Análise (Processo de Garantia da Qualidade) foi escolhida como interação pontual de Análise (Processo de Desenvolvimento). Se Análise tiver em uma combinação iterativa, serão criadas várias instâncias de V&V Análise, conforme número de iterações.

(v) *Sob-demanda:* nenhuma ação ocorre na instanciação para o processo de projeto.

Definido o processo de projeto inicial com base no processo padrão, o Gerente de Projeto pode alterar os elementos do processo, considerando as características específicas do projeto. Assim, é possível incluir, alterar ou excluir processos, atividades, artefatos, recursos e procedimentos.

Na abordagem de definição de processos de ODE, o Gerente de Projeto tem total liberdade para alterar os ativos de processo previamente definidos com base no processo padrão. Contudo, alguns eventos são registrados de forma a permitir avaliar a conformidade do processo definido com o processo padrão. Dessa forma, as exclusões de processos inteiros e atividades obrigatórias descritas no processo padrão devem conter uma justificativa do Gerente de Projeto descrevendo os motivos da exclusão. As classes *CancelamentoProcesso* e *CancelamentoAtividade* foram criadas para conter o registro de exclusão de processo e atividade, respectivamente.

4.2.4 Definindo Processos em ODE

A definição de processos em ODE ocorre em três níveis de abstração, conforme abordagem de processos em níveis apresentada no capítulo 2.

Primeiramente, se define o processo padrão organizacional, contendo os ativos de processo desejáveis em qualquer processo dessa organização. Essa etapa é fortemente influenciada pela cultura organizacional e pode ser apoiada pelas normas e modelos de garantia da qualidade. Para definir o processo padrão, o Gerente de Processo, recurso humano responsável por essa tarefa, deve informar o nome e a descrição do novo processo, conforme a Figura 4.17.

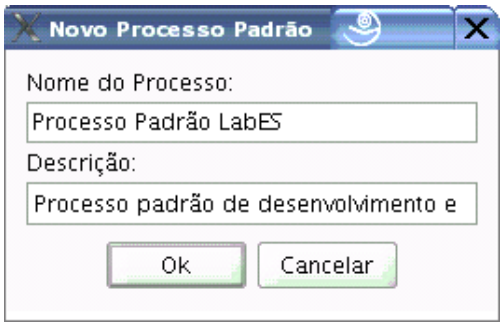


Figura 4.17 – Novo processo padrão.

O próximo passo é escolher os sub-processos que fazem parte do processo padrão. Deve-se definir o processo de engenharia (desenvolvimento ou manutenção) e os demais processos. A Figura 4.18 apresenta a tela para escolha. Os processos disponíveis para escolha são instâncias do Meta-Nível (pacote *Conhecimento*), sendo considerados tipos de processos, sendo que o Gerente de Processo deve selecionar aqueles tipos que farão parte do processo padrão em definição. Na definição dos processos, pode-se escolher um processo já definido para outro processo padrão, ou seja, reutilizar processos. Contudo, os ativos do processo reutilizado não podem ser alterados. Na medida em que os processos são definidos, a árvore no lado esquerdo da janela apresenta os ativos de processo (sub-processos, atividades, artefatos, recursos e procedimentos) que já foram definidos para o processo padrão.

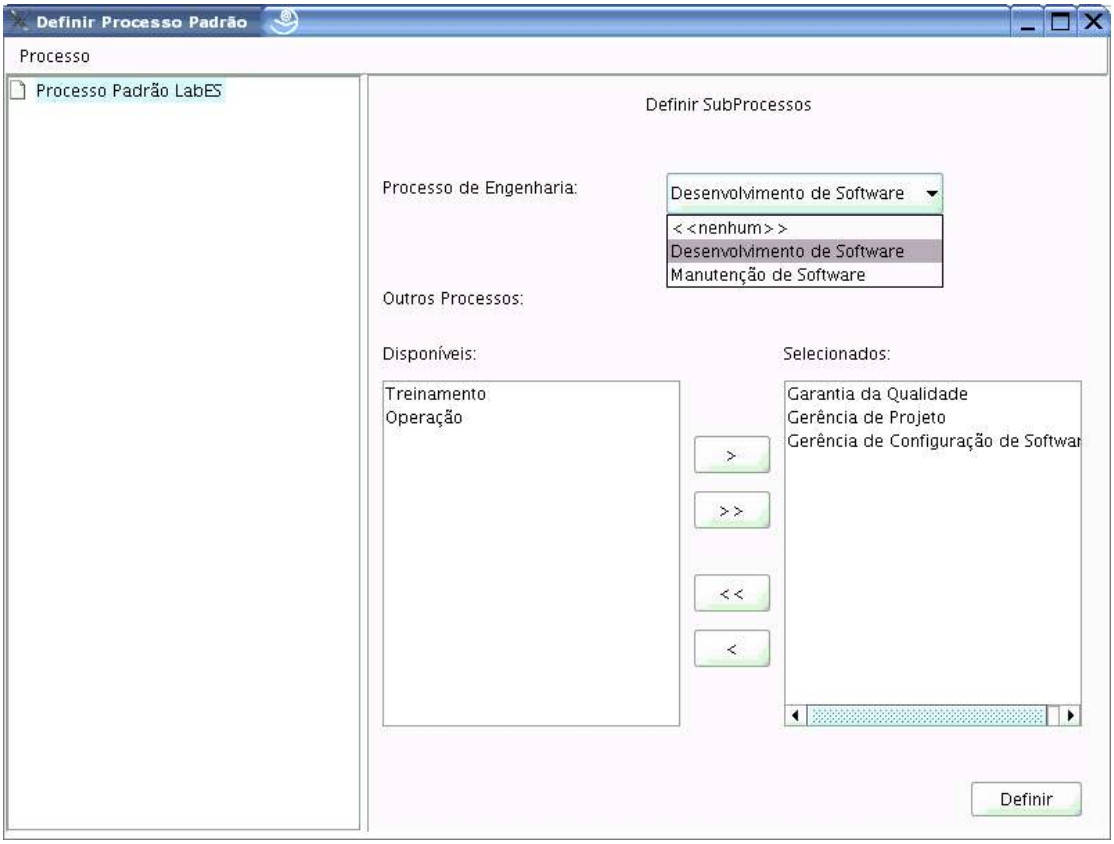


Figura 4.18 – Definição dos sub-processos.

Para cada processo que não seja de engenharia, deve-se definir o tipo de interação que o mesmo terá com o processo de engenharia. Essa interação será replicada para o processo de projeto no momento da instanciação. Os tipos de interação são mapeados em termos da ordem de precedência das atividades. Para cada processo, devem ser definidas suas macro-atividades. Para cada atividade, definem-se as sub-atividades, pré-atividades, artefatos consumidos (insumos) e produzidos (produtos), recursos (humano, hardware e software) utilizados e procedimentos adotados. A Figura 4.19 apresenta a definição das macro-atividades do Processo de Desenvolvimento de Software.

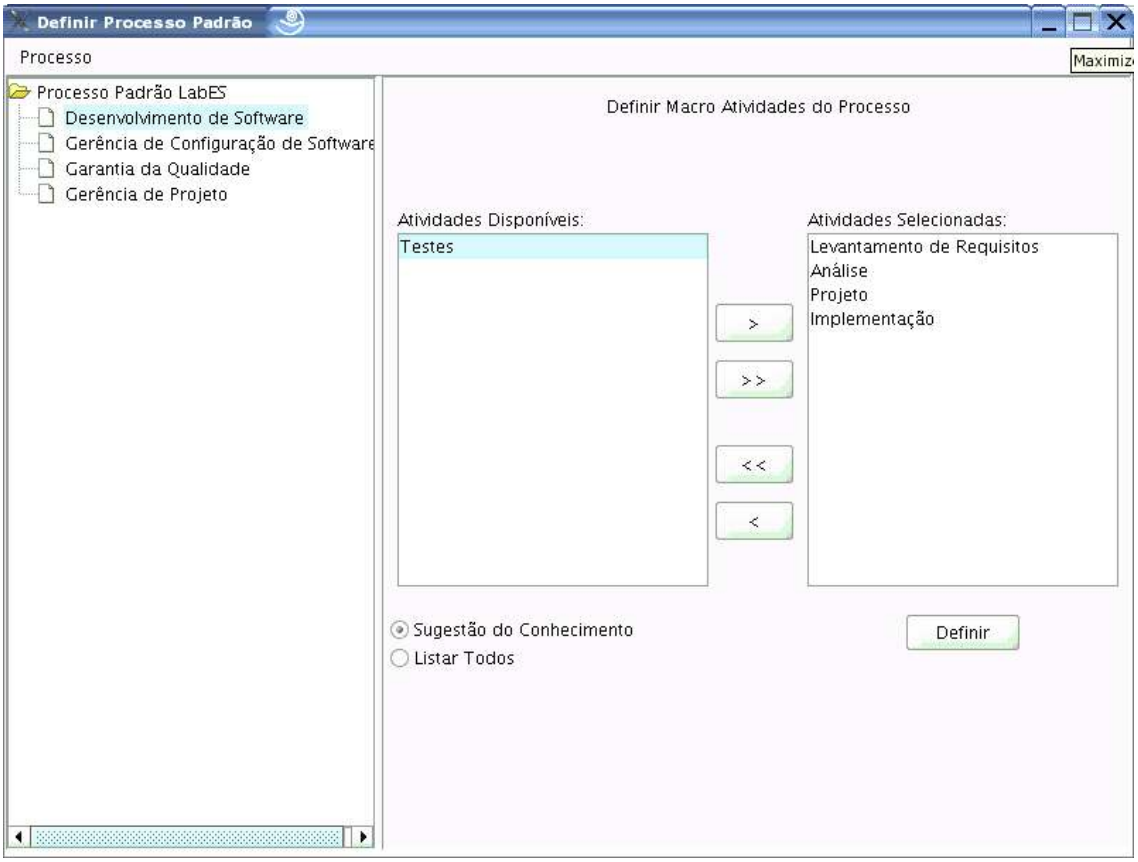


Figura 4.19 – Definição de macro-atividades.

Uma atividade pode ser definida como obrigatória no processo padrão. A atividade obrigatória não pode ser excluída na especialização desse processo, ou seja, ela deve pertencer ao processo especializado. Na adaptação do processo padrão para um processo de projeto, a atividade poderá ser excluída. Contudo, esse evento é registrado para permitir avaliar a conformidade entre o processo de projeto definido e o processo padrão que teve por base. Nas propriedades da atividade pode-se, ainda, alterar seu nome e descrição. A Figura 4.20 apresenta a tela de propriedades de atividades.

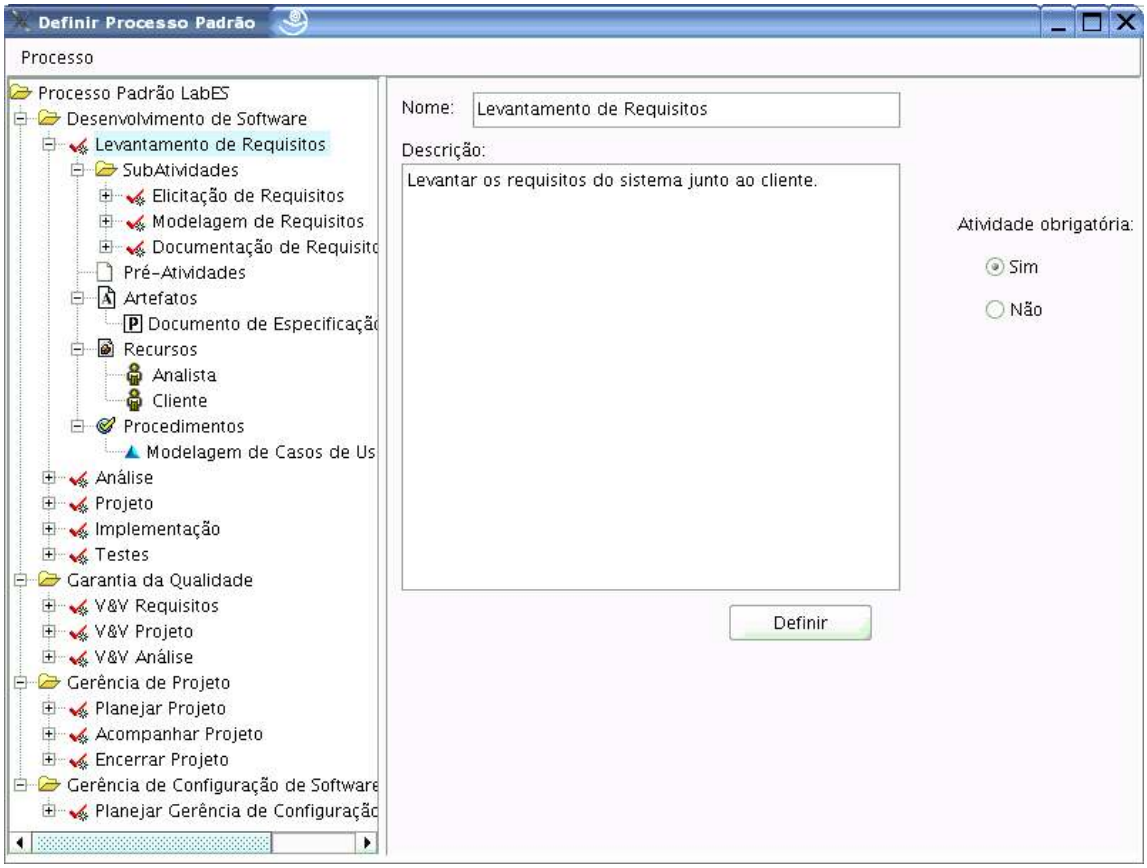


Figura 4.20 – Propriedades de uma atividade.

Quando o processo padrão estiver completamente definido, deve-se finalizar sua definição. A conclusão da definição do processo possui duas implicações: o processo não poderá mais ser alterado e podem ser definidos os modelos de ciclo de vida (MCV) do mesmo. Os MCVs são definidos para o processo de engenharia com base em suas macro-atividades. Os modelos definidos são apresentados em uma janela conforme a Figura 4.21. Pode-se, então, incluir, alterar, consultar e excluir MCVs para o processo.

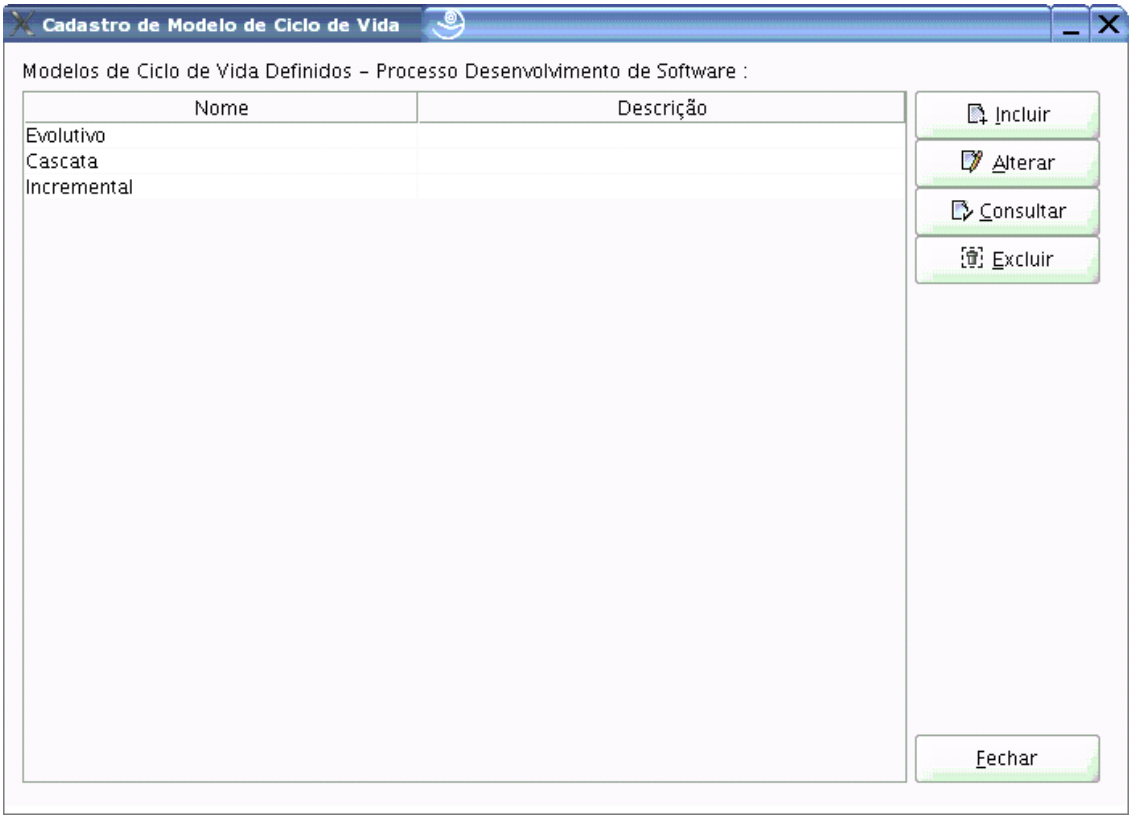


Figura 4.21 – Modelos de ciclo de vida de um processo padrão.

Um MCV é definido com base nas macro-atividades do processo de engenharia do processo padrão e na ordem de precedência dessas atividades. A Figura 4.22 apresenta a inclusão de um novo MCV. Devem ser definidas quantas combinações de atividades fazem parte do MCV. Para cada combinação, define-se o seu tipo como sendo seqüencial ou iterativa. Por fim, são definidas as macro-atividades que compõem cada combinação. Esse passo é muito importante, pois a ordem das atividades em uma combinação deve estar coerente com a ordem de precedência das macro-atividades no processo padrão. A ferramenta faz esse controle para o Gerente de Processo.

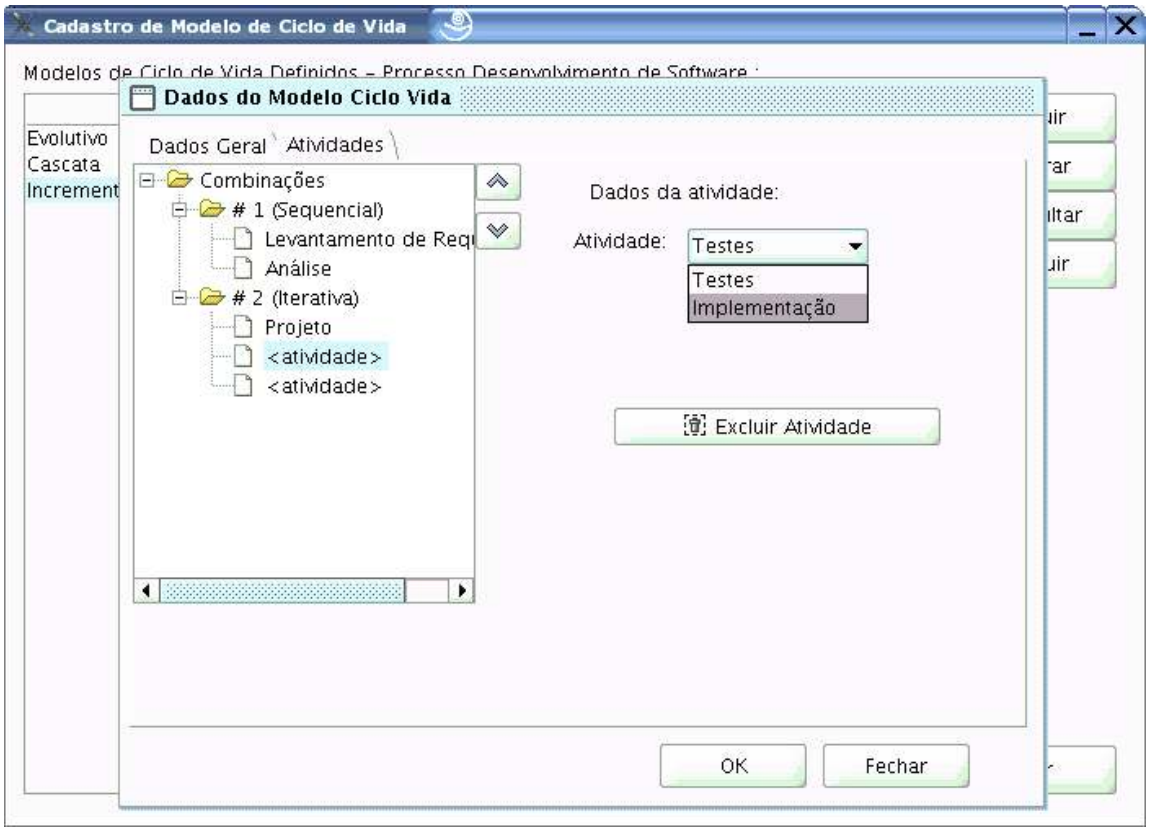


Figura 4.22 – Inclusão de um modelo de ciclo de vida de um processo padrão.

Na abordagem de tratamento de processos de ODE, um processo padrão pode ser especializado. A especialização de processos se dá para um paradigma de desenvolvimento de software, um tipo de software específico ou um domínio de aplicação. A definição de um processo especializado segue os mesmos passos descritos anteriormente. Porém, como um processo especializado é definido a partir de um processo padrão, todos os ativos de processo são replicados para o processo especializado. O processo especializado é definido também pelo Gerente de Processo por se tratar de um processo da organização. A Figura 4.23 apresenta a caracterização de um processo especializado para um paradigma, tipo de software ou domínio de aplicação específico.

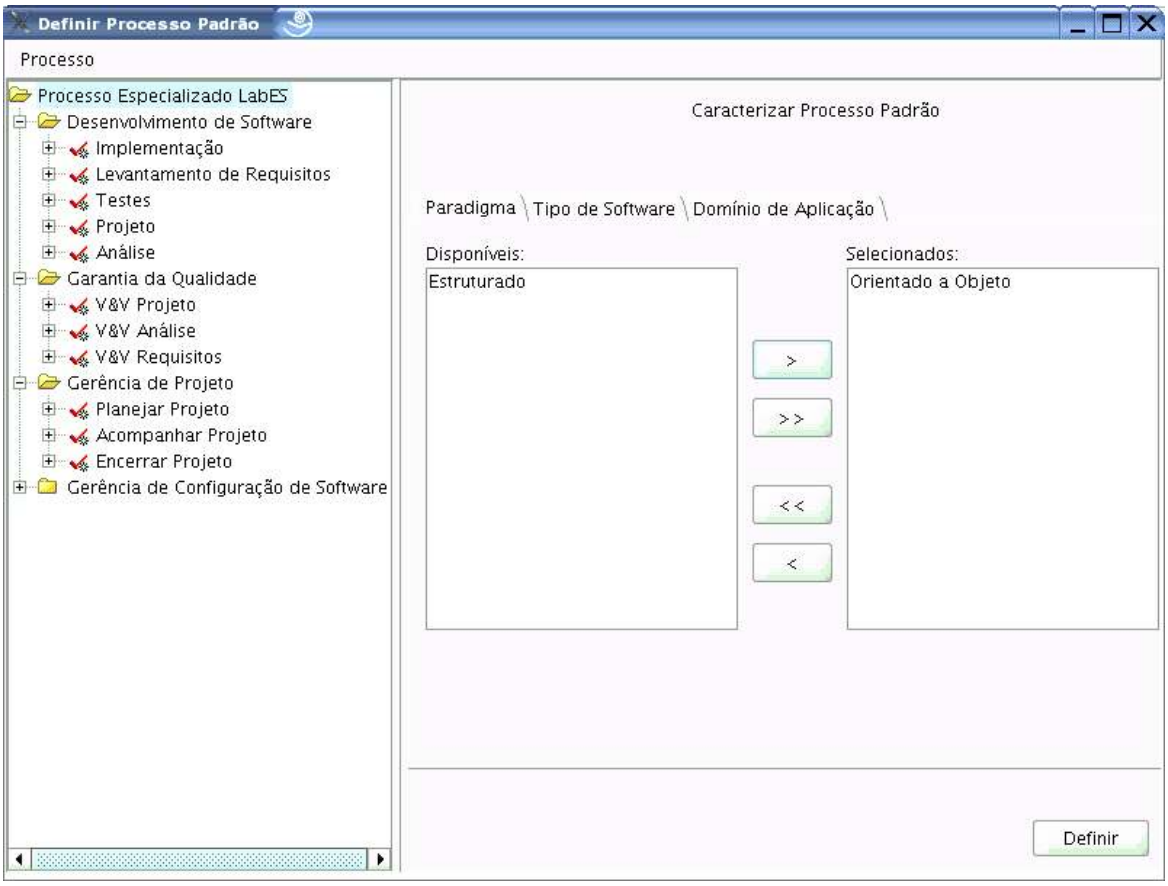


Figura 4.23 – Caracterização de um processo especializado.

O último nível na abordagem de processo utilizada em ODE é a definição de um processo de projeto. O processo de projeto é definido pelo Gerente de Projeto no contexto de um projeto específico. Esse processo é definido com base em um processo padrão (especializado ou não), considerando as particularidades do projeto em questão. O Gerente de Projeto tem liberdade de alterar os ativos de processo, incluindo, alterando ou excluindo elementos do processo de forma a adaptar às características do projeto.

O primeiro passo é escolher o processo padrão que se deseja adaptar. Deve-se, então, escolher o modelo de ciclo de vida para o processo de engenharia do processo padrão escolhido. São apresentados para escolha os MCVs definidos no processo padrão. Se o MCV possuir alguma combinação iterativa, deve-se informar o número de iterações. Com base no processo padrão e no MCV selecionados, são replicados para o processo de projeto todos os elementos definidos no processo padrão. Na Figura 4.24 foi escolhido o MCV Incremental, com duas iterações para a combinação iterativa. A árvore à esquerda da janela já apresenta os elementos do processo padrão replicados para o processo de projeto que está sendo definido.

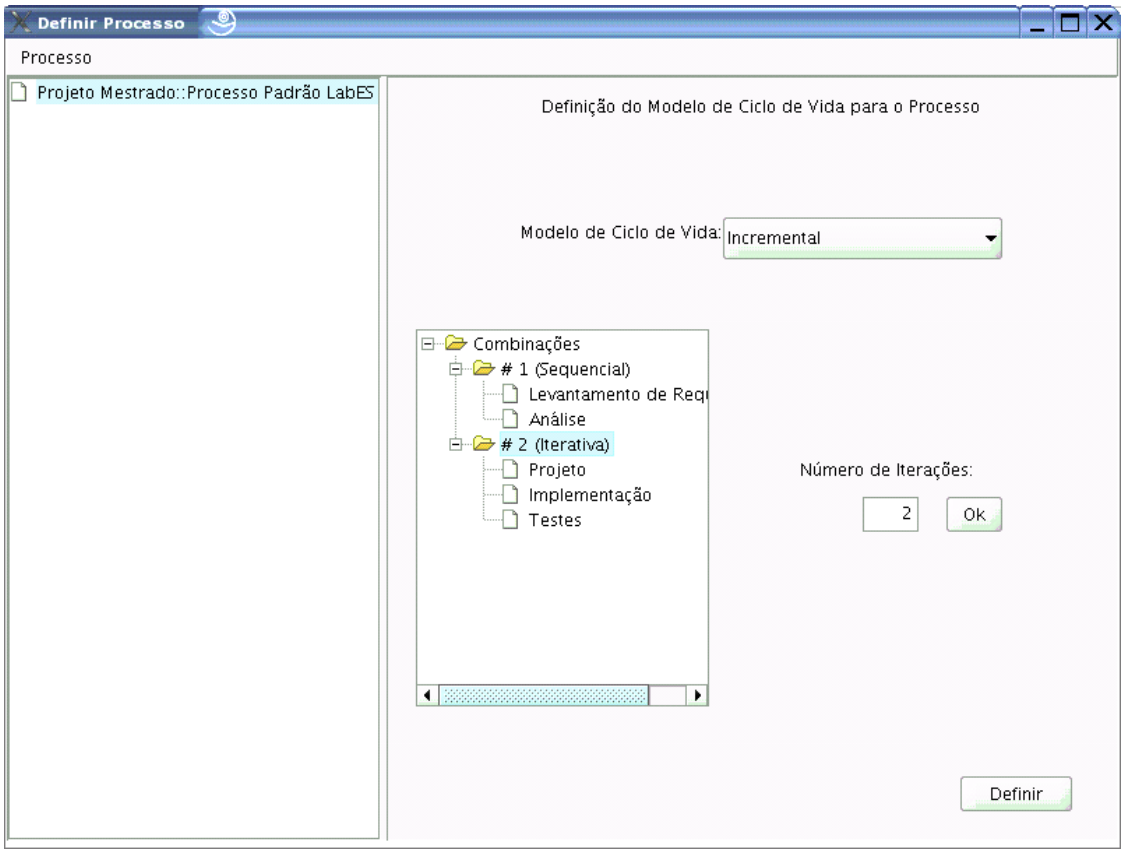


Figura 4.24 – Selecionar modelo de ciclo de vida para um processo de projeto.

O Gerente de Projeto pode, então, adaptar o processo ao projeto específico, incluindo, alterando e excluindo processos, atividades, artefatos, recursos e procedimentos. A Figura 4.25 apresenta a definição de sub-atividades da atividade *Projeto1*.

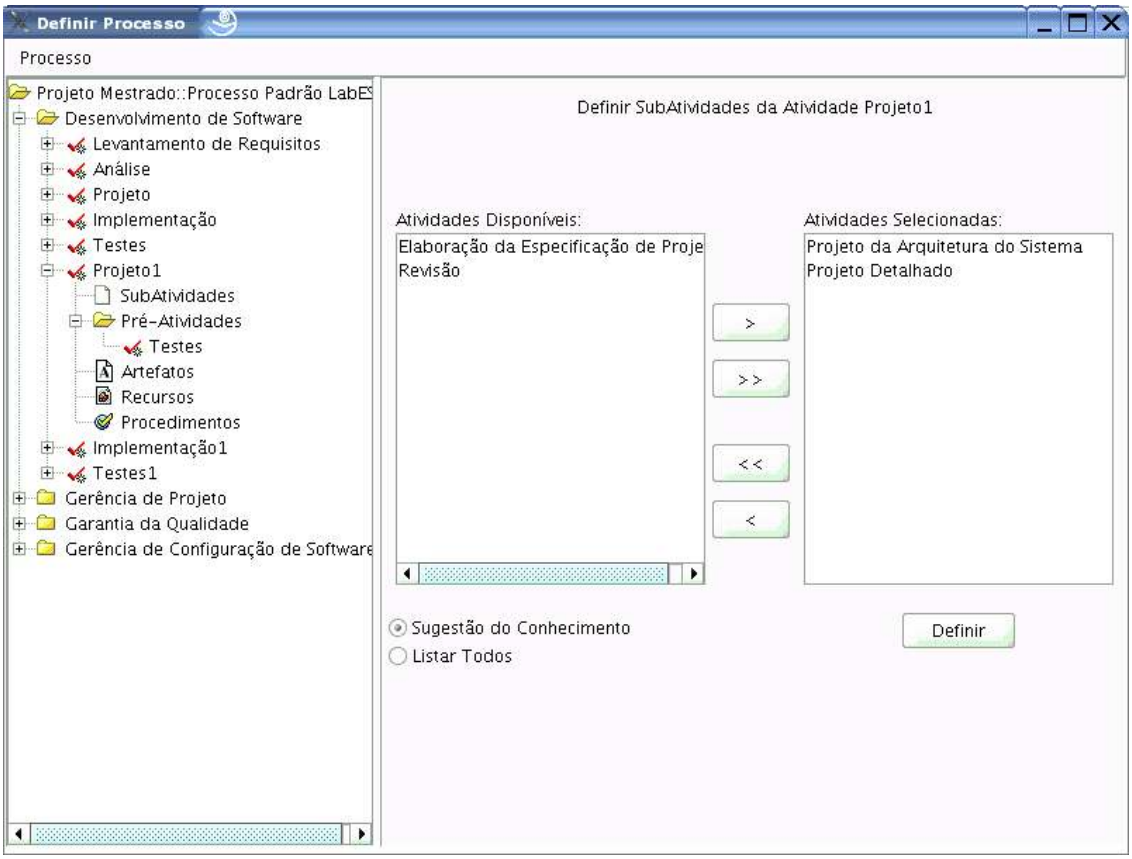


Figura 4.25 – Definir sub-atividades em um processo de projeto.

A conformidade do processo de projeto definido com o processo padrão deve ser buscada. Visando a facilitar a identificação de pontos de não conformidade, para cada processo ou atividade obrigatória definidos no processo padrão e excluídos no processo de projeto, deve-se descrever uma justificativa e registrar a sua exclusão. A Figura 4.26 apresenta a tela de justificativa para a exclusão de um processo.

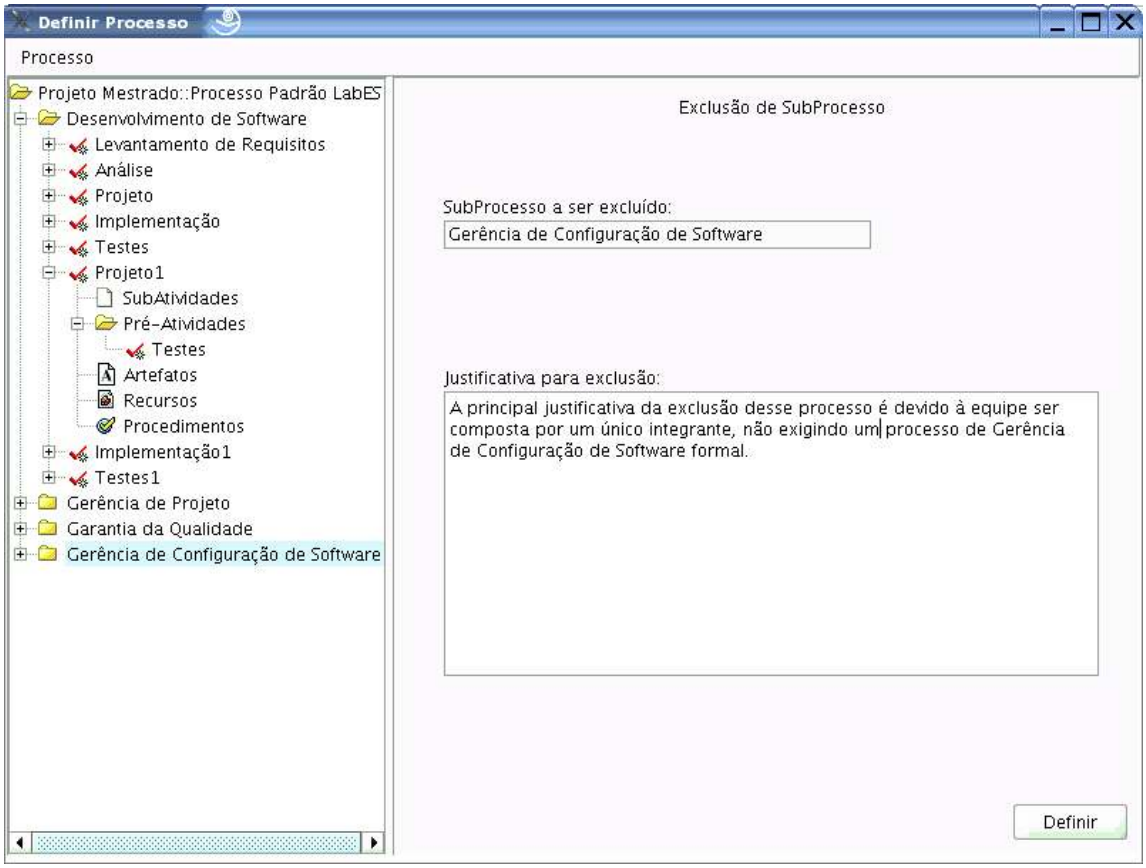


Figura 4.26 – Justificativa de exclusão de processo.

4.3 Conclusões do Capítulo

Nos Ambientes de Desenvolvimento de Software Centrados em Processos, a infra-estrutura de processo utilizada tem uma grande influência no funcionamento do ambiente. Nesse tipo de ADS, conforme abordado pela dimensão de integração de processo, deve-se permitir explicitamente definir e modelar os processos de software.

Por ser um ambiente centrado em processos, ODE possui uma ferramenta de definição de processos. Essa ferramenta permite definir processos em diversos níveis de abstração (processo padrão, processo especializado e processo de projeto), conforme estabelecido na infra-estrutura de processo do ambiente.

ODE é também um ADS baseado em ontologias, tendo como ontologia base a ontologia de processo de software. Dessa forma, após a revisão da ontologia de processo de software, foi reconstruída a infra-estrutura de processos de software de ODE e feita uma nova

versão da ferramenta da ferramenta de definição de processos, de modo a adequá-las aos novos conceitos, relações e restrições da ontologia.

Capítulo 5

Considerações Finais

Neste capítulo são apresentadas as considerações finais a respeito do trabalho desenvolvido nesta dissertação de mestrado. A seção 5.1 apresenta as principais conclusões, destacando as contribuições do trabalho, enquanto na seção 5.2 são enfocadas as perspectivas futuras para a continuidade deste trabalho.

5.1 Conclusões

O desenvolvimento de produtos de software confiáveis e dentro do cronograma e custos planejados sempre foi um desafio para a Engenharia de Software. Há indicações claras na literatura de que uma das maiores causas desses problemas é a falta de um processo de desenvolvimento de software claramente definido e efetivo. A qualidade de um produto de software depende fortemente da qualidade do processo de software utilizado em seu desenvolvimento (FUGGETTA, 2000) e, portanto, definir processos é de extrema importância para se obter produtos de qualidade.

Contudo, definir processos de software não é uma tarefa simples. Mesmo tendo à disposição farto material indicando 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.

Nos últimos anos, a área de processo de software teve um grande avanço. Muitos trabalhos têm sido feitos no contexto de modelos de qualidade de processo, tais como a publicação da norma ISO/IEC 15504 (ISO, 2003), as emendas 1 (ISO, 2002) e 2 (ISO, 2004) à ISO/IEC 12207 (ISO, 1995), a evolução do modelo CMM (PAULK, 1993) para CMMI (CHRISSIS et al., 2003) e o desenvolvimento do Modelo de Melhoria do Processo de Software Brasileiro (MPS.BR) (SOFTEX, 2005).

Ainda que diferentes projetos requeiram processos com características específicas, conforme apregoado pela maioria das normas e modelos de qualidade de processo, é possível estabelecer um conjunto de ativos de processo organizacionais, que podem ser usados na definição de processos de software para projetos específicos. Esses ativos de processo de software são usualmente organizados em um processo padrão organizacional. Dessa forma, a definição de processos de software pode ser feita em diferentes níveis de abstração. Primeiro, um processo de software padrão é definido para a organização. Baseado nesse processo organizacional, processos padrão especializados podem ser definidos considerando paradigma, tecnologia ou domínio de aplicação específicos. Finalmente, processos de projeto podem ser instanciados a partir de processos padrão (especializados ou não) (ROCHA et al., 2001).

Para ser eficiente, o processo tem que ser automatizado. Um dos grandes desafios é fazer com que os desenvolvedores possam trabalhar em um ambiente único, com todas as ferramentas necessárias integradas, compartilhando o conhecimento adquirido e cometendo o mínimo possível de erros. Desta forma, ter-se-ia um processo de desenvolvimento e seus produtos gerados com maior grau de qualidade.

A questão de se ter um ambiente único remete aos Ambientes de Desenvolvimento de Software (ADSs), que provêem suporte para o desenvolvimento e a manutenção de produtos de software e para o gerenciamento dessas atividades (LIMA, 2004). ADSs têm sido cada vez mais utilizados por auxiliar o desenvolvimento de software seguindo um processo bem definido, facilitando a transferência de informação entre ferramentas, provendo coordenação entre os desenvolvedores e aumentando o controle do projeto, através de melhor planejamento, monitoramento e comunicação (PRESSMAN, 2005).

Entretanto a integração de ferramentas tem sido, ao longo dos anos, um grande obstáculo, uma vez que são disponibilizadas no mercado diversas ferramentas, de diversos fabricantes, porém com conceituações e padrões diferentes, que, muitas vezes, conflitam entre si.

Ontologias merecem bastante destaque neste sentido, pois se referem a um entendimento compartilhado de algum domínio de interesse, podendo ser usadas para resolver problemas como comunicação precária, dificuldade na identificação de requisitos, interoperabilidade, reuso e compartilhamento de informações, dentre outros (USCHOLD et al., 1996).

O ambiente ODE (*Ontology based software Development Environment*) (FALBO et al., 2003) (FALBO et al., 2004a) é um exemplo de ADS que tem buscado, através da utilização de ontologias, um conceito mais amplo de integração, envolvendo integração de dados, apresentação, controle, processo, plataforma e conhecimento.

Neste contexto de processos de software e automatização de processos de software, o presente trabalho teve por objetivo evoluir a infra-estrutura de processos de software de ODE, deixando-a mais alinhada com as abordagens mais atuais de definição de processos. Foram contribuições deste trabalho:

- A evolução da ontologia de processo de software proposta em (FALBO, 1998), incorporando novos conceitos acerca do domínio de processos de software. Foram também alteradas as sub-ontologias de atividade, recurso e procedimento. Conceitos como processo padrão, processo de projeto, adaptação de processos, composição de processos, interação entre processos, entre outros, que são advogados pelas principais normas de qualidade de processo, foram incorporados à nova versão da ontologia de processo de software.
- O mapeamento entre os conceitos das principais normas de qualidade de processo e os conceitos da ontologia de processo de software, buscando facilitar o uso dessa ontologia como uma inter-língua para compartilhamento de informações sobre modelos de qualidade de processo.
- A evolução da infra-estrutura de processo de software de ODE. Por ser um ambiente centrado em processo e baseado em ontologias, a evolução dessa infra-estrutura foi feita a partir da revisão da ontologia de processo de software.
- A melhoria da ferramenta de apoio à definição de processos de ODE, incorporando novas funcionalidades, tais como as dedicadas à decomposição de processos e à interação entre processos.

Em suma, esse trabalho buscou colaborar para a evolução da área de processo de software em duas vertentes: a primeira conceitual, através da revisão de uma ontologia de processo de software; a segunda prática, propondo uma ferramenta de apoio à definição de processos de software nos níveis organizacional, com a definição de processos padrão (especializados ou não) e de projeto, por meio da adaptação de um processo padrão às características de um projeto específico.

5.2 Perspectivas Futuras

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

Embora a revisão da ontologia de processos de software tenha sido uma das principais contribuições deste trabalho, identificamos outros aspectos que podem ser evoluídos, mas que não foram discutidos no contexto desta dissertação. Dentre as possíveis melhorias, podemos citar:

- incorporar aspectos já tratados em ontologias genéricas, como, por exemplo, ordenação de atividades e interação entre processos, tomando por base ontologias genérica de atividade e processo, respectivamente;
- detalhar a interação entre processos, incluindo níveis de interação entre atividades;
- evoluir a ontologia de recurso, em especial no que tange a recurso humano. O uso de recursos humanos, de software e de hardware por uma atividade possui características bastante distintas. Assim, é importante ampliar as distinções ontológicas entre esses conceitos, tratando, por exemplo, aspectos comportamentais inerentes a recursos humanos.

A ferramenta de definição de processos possui algumas limitações que podem ser evoluídas. A abordagem definida para interação entre processos centraliza as interações no processo de engenharia. Dessa forma, não é permitido definir interações entre processos não-engenharia. Contudo, isso não é uma realidade. Por exemplo, suponhamos que os processos de Gerência de Projeto e Gerência de Configuração de Software estejam definidos como processos não engenharia, e se deseje colocar o Plano de Projeto sob gerência de configuração. Na abordagem atual da ferramenta isso não é permitido.

A ferramenta ControlPro (MORO et al., 2005) é utilizada para acompanhamento de projeto em ODE. Essa ferramenta não foi evoluída nesse trabalho, portanto não está aderente à nova infra-estrutura de processos. É essencial evoluir ControlPro, pois o acompanhamento do processo é fundamental na utilização de ODE.

Por ser um trabalho elaborado por especialistas com grande conhecimento técnico e prático, as normas de qualidade de processo representam uma grande fonte de pesquisa para a definição de processos organizacionais. Inclusive é advogado nesse trabalho o uso, em conjunto com a cultura organizacional, de diversas normas simultaneamente na definição do processo padrão, aproveitando o ponto forte de cada modelo. Contudo, em ODE, não é clara a utilização dessas normas e modelos durante a definição do processo. Esse apoio poderia vir da experiência e conhecimento do Gerente de Processo e do Gerente de Projeto durante a definição do processo. A oportunidade de melhoria está em apoiar explicitamente a definição dos processos com os requisitos dessas normas e modelos de qualidade do processo. Também é interessante poder atestar a aderência de um processo a uma norma de qualidade.

Um processo de software não deve ser estático. Um processo deve acompanhar as mudanças que naturalmente ocorrem nas organizações. Ou seja, um processo deve estar em um ciclo de melhoria contínua. As oportunidades de melhoria de um processo devem ser capturadas na execução desses processos na organização. Por exemplo, uma alteração em um processo de projeto pode não ser específica para o projeto em questão, podendo se refletir no processo padrão organizacional. Esses eventos devem ser capturados e registrados como sugestões de melhoria a serem avaliadas pelo Gerente de Processo.

Contudo, a alteração de um processo, seja padrão ou de projeto, deve vir acompanhado por uma Gerência de Configuração, de forma a controlar e permitir coexistir diferentes versões de um mesmo processo. Em (NUNES, 2005) foi proposta uma infra-estrutura para gerência de configuração de software em ODE. Um trabalho futuro é utilizar essa infra-estrutura, permitindo controlar a versão de um processo padrão ou de projeto.

Uma outra oportunidade de melhoria é utilizar métricas de software para apoiar a melhoria de processos. A melhoria de processos pode se tornar muito mais efetiva se acompanhada de indicadores e medidas que provejam meios de se avaliar a produtividade dos processos. A sugestão é permitir definir métricas para os processos definidos, estabelecendo metas para os indicadores e avaliando continuamente seus resultados. Uma abordagem desta natureza dá uma referência quantitativa da produtividade dos processos e evidências de onde o processo pode ser melhorado.

Em (NATALI, 2003) foi proposta uma infra-estrutura de Gerência de Conhecimento. A Gerência de Conhecimento consiste no gerenciamento formal que facilita a criação, o acesso e o reuso do conhecimento, tipicamente usando tecnologia avançada. É a

administração, de forma sistemática e ativa, dos recursos de conhecimento de uma organização, utilizando tecnologia apropriada e visando a fornecer benefícios à organização. Um trabalho futuro é utilizar a infra-estrutura de Gerência de Conhecimento para apoiar a definição de processos de software em ODE, apoiando o Gerente de Processo ou de Projeto durante a definição. O objetivo seria permitir criar, capturar, acessar, disseminar, usar, preservar e evoluir o conhecimento organizacional acerca de processos de software.

Referências Bibliográficas

ARBAOUI, S., DERNIAME, J. C., OQUENDO, F., VÉRJUS, H. *A Comparative Review of Process-Centered Software Engineering Environments.* Annals of Software Engineering 14, 311–340, 2002.

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

BERTOLLO, G., FALBO, R.A. *Apoio Automatizado à Definição de Processos em Níveis*, II Simpósio Brasileiro de Qualidade de Software, 77 – 91, Fortaleza, Brasil, 2003.

BERTOLLO, G., SEGRINI, B., FALBO, R.A. *Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias*, V Simpósio Brasileiro de Qualidade de Software, 61 – 75, Vila Velha, Brasil, 2006.

CHANDRASEKARAN, B., JOSEPHSON, J.R., BENJAMINS, V.R. *What Are Ontologies, and Why Do We Need Them?*. IEEE Intelligent Systems 14, 20-26, 1999.

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

CHRISTIE, A. M. S*oftware Process Automation - The Technology and its Adoption*, Peittsburghm Pennsylvannia, Springer-Verlag Berlin Heidelberg, 1995.

DUARTE, K., FALBO, R.A. *Uma Ontologia de Qualidade de Software*, VII Workshop de Qualidade de Software, João Pessoa, Brasil, p. 275-285, 2000.

FALBO, R. A. *Integração de Conhecimento em um Ambiente de Desenvolvimento de Software*. Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, Dezembro de 1998.

FALBO R.A., MENEZES, C.S., ROCHA, A.R.C. *A Systematic Approach for Building Ontologies*. Proceedings of the 6th Ibero-American Conference on Artificial Intelligence. Lisbon, Portugal, Lecture Notes in Computer Science, vol. 1484, 1998.

FALBO R.A., GUIZZARDI, G., DUARTE, K.C. *An Ontological Approach to Domain Engineering*. Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering. SEKE'2002, pp. 351- 358, Ischia, Italy, July 2002.

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

FALBO, R. A. *Experiences in Using a Method for Building Domain Ontologies*. Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering, SEKE'2004, International Workshop on Ontology In Action, OIA'2004. Banff, Alberta, Canada, June 2004.

FALBO, R.A., RUY, F. B., PEZZIN, J., MORO, R. D. *Ontologias e Ambientes de Desenvolvimento de Software Semânticos.* Actas de las IV Jornadas Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento, JIISIC'2004, Volumen I, pp. 277-292. Madrid, España, Noviembre 2004a.

FALBO, R.A., RUY, F. B., BERTOLLO, G., TOGNERI, D. *Learning How to Manage Risks Using Organizational Knowledge.* Advances in Learning Software Organizations (Proceedings of the 6th International Workshop on Learning Software Organizations – LSO'2004), Melnik G. and Holz, H. (Eds.): LNCS 3096, pp. 7-18. Springer-Verlag Berlin Heidelberg, Banff, Canada, June 2004b.

FALBO, R.A., RUY, F.B., MORO, R.D. *Using Ontologies to Add Semantics to a Software Engineering Environment,* 17th International Conference on Software Engineering and Knowledge Engineering, SEKE'2005, p.151-156, Taipei, China, July 2005.

FALBO, R. A., BERTOLLO, G. *Establishing a Common Vocabulary for Software Organizations Understand Software Processes*. EDOC International Workshop on Vocabularies, Ontologies and Rules for The Enterprise, VORTE'2005, Enschede, The Netherlands, September 2005.

FUGGETTA, A. *Software Process: A Roadmap.* In *Proc.* of The Future of Software Engineering, ICSE'2000, Ireland, 2000.

GRUBER, T.R., "Towards principles for the design of ontologies used for knowledge sharing". International Journal of Human-Computer Studies, v. 43, n. 5/6, 1995.

GRUHN, V. *Process-Centered Software Engineering Environments - A Brief History and Future Challenges*. Annals of Software Engineering 14, 363–382, 2002.

GRUNINGER, M., FOX, M. *Methodology for the Design and Evaluation of Ontologies*, Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing – IJCAI´1995, Montreal, Canada, 1995.

GUARINO, N. *Formal Ontology and Information Systems*. In: Proceedings of the First International Conference on Formal Ontology in Information Systems (FOIS'98), Trento, Italy, June 1998.

HARRISON, W., OSSHER, H., TARR, P. *Software Engineering Tools and Environments: A Roadmap.* In *Proc.* of The Future of Software Engineering, ICSE'2000, Ireland, 2000.

HOLZ, H., KÖNNECKER, A., MAURER, F. *Task-Specific Knowledge Management in a Process-Centred SEE*. In: Advances in Learning Software Organizations, v. 2176, Lecture Notes in Computer Science, Springer, pp. 163-177, 2001.

IEEE *Padrão para Aplicação e Gerência de Processo de Engenharia de Sistemas*, IEEE Std 1220 – 1998, Nova York: IEEE, 1998.

ISO 9001, *Sistema de Gestão da Qualidade – Requisitos*, ISO 9001:2000, Geneva, Suíça: ISO, 2000a.

ISO 9000, *Sistema de Gestão da Qualidade – Fundamentos e Vocabulário*, ISO 9000:2000, Geneva, Suíça: ISO, 2000b.

ISO 9004, *Sistema de Gestão da Qualidade – Diretrizes para Melhoria de Desempenho*, ISO 9004:2000, Geneva, Suíça: ISO, 2000c.

ISO/IEC 12207, *Information Technology - Software life cycle processes*. *Amendment 1:*2002, *Amendment 2:* 2004, 1995.

ISO/IEC TR 15504, *Information Technology – Software Process Assessment*, ISO/IEC TR 15504, Geneva, Switzerland: ISO, 1998.

ISO/IEC 15504, *Information Technology – Process Assessment*, ISO/IEC 15504, Geneva, Switzerland: ISO, 2003.

JASPER, R., USCHOLD, M. *A Framework for Understanding and Classifying Ontology Applications.* Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management – KAW'99, Alberta, Canada, 1999.

KRUCHTEN, P. *The Rational Unified Process: an Introduction.* Masschusetts: Addison-Wesley, 1998.

LIMA, K.V.C., *Definição e Construção de Ambientes de Desenvolvimento de Software Orientados a Organização.* Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, Brasil, 2004.

MIAN, P.G., FALBO, R.A. *Supporting Ontology Development with ODEd.* Journal of the Brazilian Computer Science, vol. 9, no. 2, pp 57-76, November 2003.

MORO, R.D., NARDI, J.C., FALBO, R.A. *ControlPro: Uma Ferramenta de Acompanhamento de Projetos Integrada a um Ambiente de Desenvolvimento de Software*, XII Sessão de Ferramentas do Simpósio Brasileiro de Engenharia de Software, SBES'2005, Uberlândia, Brasil, Outubro 2005.

MUTAFELIJA, B. and STROMBERG, H. *Systematic Process Improvement Using ISO 9001:2000 and CMMI*, Artech House, 2003.

NARDI, J.C., FALBO, R.A. *Uma Ontologia de Requisitos de Software*, In: IX Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes de Software – IDEAS´2006, La Plata, Argentina, 2006.

NATALI, A. C. C. *Uma infra-estrutura para gerência de conhecimento em um ambiente de desenvolvimento de software.* Tese de Mestrado. UFES, Vitória, 2003.

NUNES, V. B. *Integrando Gerência de Configuração de Software, Documentação e Gerência de Conhecimento em um Ambiente de Desenvolvimento de Software*. Dissertação de Mestrado, UFES, Vitória, 2005.

PAULK, M. C., et al., *Key Practices of the Capability Maturity Model for Software, Version1.1,* CMU/SEI-93-TR-25, DTIC Number ADA263432, Pittsburgh: Software Engineering Institute, 1993.

PFLEEGER, S.L. *Software Engineering: Theory and Practice*. 2nd edition, New Jersey: Prentice Hall, 2001.

PRESSMAN, R.S. *Software Engineering: A Practitioner's Approach.* 6th Edition, New York: McGraw-Hill, 2005.

RICHTER, M. M., MAURER, F., *MILOS and MASE: Past & Present*, Technical Report, University of Calgary, 2003. Disponível em: http://ebe.cpsc.ucalgary.ca/ebe/Wiki.jsp?page=Root.Publications

ROCHA, A. R. C., AGUIAR, T. C., SOUZA, J. M. *TABA: A Heuristic Workstation for Software development*, In: Proceedings of COMPEURO 90, Tel Aviv, Israel, 1990.

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

RUY, F.B., BERTOLLO, G., FALBO, R.A. *Knowledge-based Support to Process Integration in ODE.* Clei Electronic Journal, vol. 7, n. 1, paper 3, 2004.

RUY, F. B. *Semântica em um Ambiente de Desenvolvimento de Software.* Tese de Mestrado. UFES, Vitória, 2006.

SOFTEX, *MPS.BR – Melhoria de Processo do Software Brasileiro: Guia Geral*, Versão 1.0, disponível em www.softex.br/mpsbr, 2005.

THOMAS I., NEJMEH, B.A. "Definitions of Tool Integration for Environments", IEEE Software, 29-35, March 1992.

TRAVASSOS, G. H. *O Modelo de Integração de Ferramentas da Estação TABA.* Tese de Doutorado. COPPE/UFRJ, Rio de Janeiro, Brasil, 1994.

USCHOLD, M., GRUNINGER, M. *Ontologies: Principles, Methods and Applications.* Knowledge Engineering Review, Vol. 11, No. 2, June 1996.