Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias

Article · January 2006

CITATIONS
3

READS
57

3 authors, including:

Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo
172 PUBLICATIONS **1,661** CITATIONS

SEE PROFILE

Definição de Processos de Software em um Ambiente de Desenvolvimento de Software Baseado em Ontologias

Gleidson Bertollo, Bruno Segrini, Ricardo de Almeida Falbo

Departamento de Informática – Universidade Federal do Espírito Santo (UFES)
Av. Fernando Ferrari s/n, Campus de Goiabeiras – 29.060-900 – Vitória – ES – Brasil
{gleidsonbertollo, brunosegrini}@yahoo.com.br, falbo@inf.ufes.br

Abstract. *Software process definition is not an easy task. It can be done in several levels, starting by defining a standard software process for the organization, specializing it to consider certain paradigm, software type or domain application, and finally tailoring the specialized standard processes to projects. Since it is a very complex, knowledge-intensive and error prone task, it is useful to provide automated tools to support it. In ODE, an ontology-based process-centered software engineering environment, there is a tool that supports this leveled approach for process definition. This paper presents an evolution of this tool and discusses the design rationale behind this evolution.*

Resumo. *A definição de processos de software não é uma tarefa fácil. Ela pode ser feita em diversos níveis, começando pela definição de um processo padrão para a organização, passando pela especialização deste processo em outros processos padrão especializados para paradigmas, tipos de software ou domínios de aplicação específicos, até chegar à instanciação de um processo para um projeto específico, a partir de um processo padrão, especializado ou não. Uma vez que esta é uma tarefa complexa, propensa a erros e que requer muito conhecimento e experiência, é muito importante prover ferramentas automatizadas para apoiá-la . Em ODE, um ambiente de desenvolvimento de software centrado em processo e baseado em ontologias, há uma ferramenta que apóia essa abordagem em níveis para a definição de processos. Este artigo apresenta uma evolução dessa ferramenta e discute as razões por detrás dessa evolução.*

1. Introdução

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. 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 formal de processos de software e inclui normas de qualidade de processo de software, tal como a ISO/IEC 12207 [ISO, 1995], e modelos de

qualidade de processo, tais como o CMMI [Chrissis et al., 2003] e o MPS.BR [Softex, 2005].

A segunda vertente 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, utilizadas na construção desses artefatos [Harrison et al., 2000]. A representação explícita de processos, seus produtos e suas interações é o fundamento no 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 [Harrison et al., 2000].

Mas, 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. Esse fato também se reflete nos ADSs Centrados em Processo e seus modelos de representação de processo.

Contudo, ainda que diferentes projetos requeiram processos com características específicas, conforme apregoado pela imensa 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 processo 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].

Visando a apoiar essa abordagem de definição de processos em níveis no ADS Centrado em Processos ODE (*Ontology-based software Development Environment*) [Falbo et al., 2003], foi desenvolvida uma ferramenta, cuja versão anterior, apresentada em [Bertollo e Falbo, 2003], permitia a definição de processos padrão, sua especialização em processos padrão especializados e finalmente a instanciação de processos de projeto. Vale destacar que ODE, como o próprio nome indica, é um ambiente baseado em ontologias. Assim essa ferramenta foi construída fundamentada em uma ontologia de processo de software, definida em [Falbo, 1998].

Em outubro de 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 enveredada 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. No

que concerne à 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 um processo poder ser composto por outros processos, também foram detectadas.

Na busca por aperfeiçoar a ferramenta com base no feedback dado, percebeu-se que, de fato, a própria ontologia que servia de fundamentação para o ambiente precisava evoluir. Muitos trabalhos haviam sido feitos na vertente 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 [Fiorini et al., 1998] para CMMI [Chrissis et al., 2003] e o desenvolvimento do Modelo de Melhoria do Processo de Software Brasileiro (MPS.BR) [Softex, 2005]. Assim, era necessário, primeiro, evoluir a ontologia 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.

Este artigo discute a evolução da infra-estrutura de processos de software do ambiente ODE e de sua ferramenta de definição de processos. A seção 2 discute brevemente o tema central deste trabalho: processos de software e ADSs centrados em processo. A seção 3 apresenta o ambiente ODE e sua estruturação baseada em ontologias. A seção 4 aborda a definição de processos em ODE e as limitações das primeiras versões da ferramenta de apoio a essa atividade, bem como discute a evolução da infra-estrutura de processos de ODE e de sua ferramenta de definição de processos. A seção 5 discute trabalhos correlatos e, finalmente, a seção 6, apresenta as conclusões do trabalho e perspectivas futuras.

2. Processo de Software e sua Automatização

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).

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 do 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 e complexidade, e criticalidade do projeto. Assim, é necessária uma abordagem flexível e configurável para a definição de processos, de modo a facilitar a adaptação de processos às necessidades específicas de cada projeto. A Figura 1 mostra esquematicamente essa abordagem [Rocha et al., 2001].

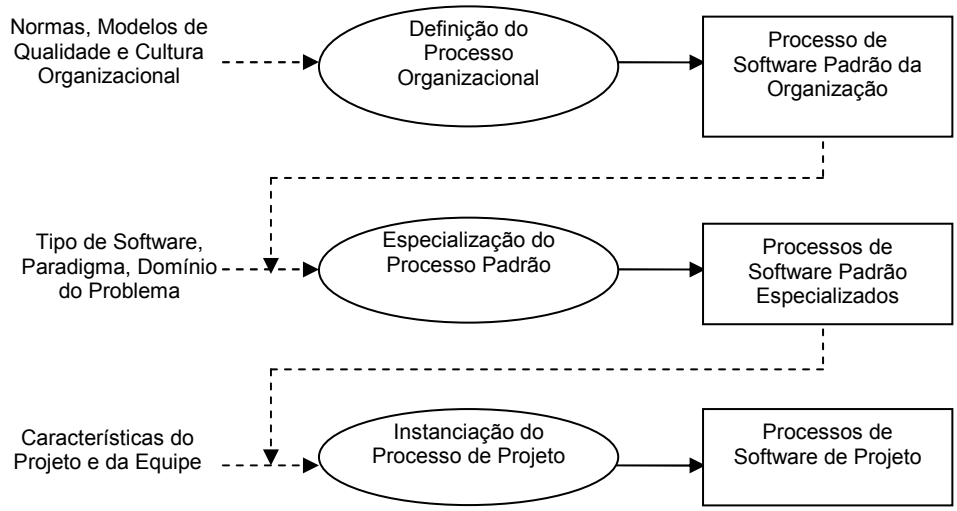

Figura 1 – Abordagem em Níveis para a Definição de Processos.

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, sem desconsiderar, no entanto, a cultura organizacional.

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 processos 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.

A ênfase no conceito de processo de software motivou diversas iniciativas relacionadas, com destaque para a modelagem e automatização do processo [Fuggetta, 2000]. A primeira geração de ferramentas de apoio ao processo de software apoiava apenas atividades isoladas, sem considerar a integração de ferramentas e a execução coordenada do processo. A partir da identificação da necessidade de apoio integrado às atividades de engenharia de software ao longo de todo o ciclo de vida, surgiram os Ambientes de Desenvolvimento de Software (ADSs). Contudo, os primeiros ADSs não suportavam exatamente a noção de processo de software. Apenas mais tarde, com os ADSs centrados em processos, a representação explícita dos processos de software, seus produtos e suas interações tornou-se a base sobre a qual os ambientes passaram a ser construídos. ADSs centrados em processos passaram, então, a integrar ferramentas de apoio ao desenvolvimento de software com ferramentas de modelagem e execução do processo de software definido [Harrison et al., 2000].

Para representar processos de software e seus ativos, tornou-se necessária a criação de formalismos de modelagem de processos [Fuggetta, 2000]. No contexto do ambiente ODE [Falbo et al., 2003], uma ontologia de processo de software é usada como formalismo para modelagem de processos, já que as ferramentas e o ambiente precisam compartilhar um entendimento comum sobre o que é um processo de software.

Uma ontologia define um vocabulário específico usado para descrever uma certa realidade, mais um conjunto de decisões explícitas fixando o significado pretendido para o vocabulário. Uma ontologia envolve, então, um vocabulário de representação que captura conceitos, relações e propriedades em algum domínio e um conjunto de axiomas, estabelecendo restrições que têm de ser respeitadas [Guarino, 1998].

3. O Ambiente ODE

ODE (***O****ntology-based software* ***D****evelopment* ***E****nvironment*) [Falbo et al., 2003] é um Ambiente de Desenvolvimento de Software (ADS) centrado em processo, desenvolvido no Laboratório de Engenharia de Software (LabES) da Universidade Federal do Espírito Santo. Como o próprio nome indica, ODE tem seu projeto baseado em ontologias, de modo que as ferramentas compartilham uma conceituação comum acerca do domínio de Engenharia de Software. Ou seja, um conjunto de ontologias integradas desse domínio é utilizado na construção das diferentes ferramentas do ambiente e, uma vez que essas ontologias são integradas, a integração das ferramentas é facilitada [Falbo et al., 2003]. 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 recentemente, conforme discutido na seção 4. Serve de base para toda a infra-estrutura de processo 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., 2004]: 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: 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.

Todas essas ontologias têm sido construídas procurando reutilizar conceituações previamente estabelecidas por outras ontologias já desenvolvidas no projeto, formando uma grande rede de conceitos.

Para se obter os maiores benefícios do uso de ontologias, é necessário estabelecer uma ligação semântica entre os objetos de ODE e os conceitos definidos nas ontologias. Para tal, uma abordagem de anotação conceitual [Falbo et al., 2005], esquematizada na Figura 2, foi utilizada. O nível ontológico, implementado no pacote *Ontologia*, corresponde à meta-ontologia de ODE e o principal objetivo desse nível é registrar as ontologias no ambiente. Suas instâncias são usadas para guiar a definição dos outros níveis, originando as principais classes em ambos, meta-nível e nível base.

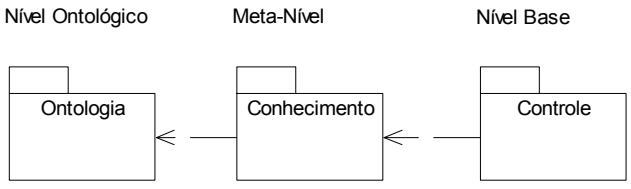

Figura 2 – Esquema de Anotação Conceitual de ODE.

O meta-nível (pacote *Conhecimento*) define as classes que descrevem conhecimento acerca de domínios de interesse, com destaque para o domínio da Engenharia de Software. Suas classes derivam dos conceitos das ontologias e as

instâncias dessas classes agem como conhecimento sobre os objetos do nível base. Como forma de padronização, classes do pacote *Conhecimento* são precedidas pela letra K [Falbo et al., 2005]. Tomando por exemplo a ontologia de processos de software, há classes *KAtividade* e *KArtefato*, *KRecurso* que correspondem aos conceitos de atividade, artefato e recurso, respectivamente.

Por fim, o nível base (pacote *Controle*) define as classes que implementam as aplicações de ODE, ou seja, suas funcionalidades e ferramentas. Muitas de suas classes também derivam diretamente das ontologias, mas, geralmente, elas incorporam detalhes específicos, não descritos nas ontologias, necessários para implementar as aplicações. Além disso, esse pacote contém, ainda, classes, associações, atributos e operações que não possuem uma contrapartida no nível ontológico e no meta-nível [Falbo et al., 2005].

Essa abordagem facilita a correlação entre objetos dos três níveis, pois os objetos em um nível servem de meta-dados para outros nos níveis abaixo.

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 do conceito de atividade. Ela deve dar 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*).

4. Definição de Processos de Software em ODE

Conforme citado na seção 1, 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, Controle e Conhecimento, e a ferramenta apoiava apenas a definição de processos de projeto. Instâncias pré-cadastradas de classes de *Conhecimento* serviam de guia para essa definição.

Com o crescente reconhecimento da importância da definição de processos padrão organizacionais, passou-se a utilizar as classes de *Conhecimento* para descrever o processo 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, conforme discutido na seção 2.

Nesta versão, 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 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.

A partir de outubro 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 sendo apontados. Dentre os pontos positivos, a abordagem de definição de processos em níveis foi considerada um dos pontos mais fortes, juntamente com o fornecimento de orientações, através dos objetos de *Conhecimento*. Vale destacar que esses ativos de processo pré-registrados eram definidos com base na norma ISO/IEC 12207. Além disso, para permitir considerar a cultura organizacional, ativos de processo típicos da organização também podiam ser previamente cadastrados. Contudo, algumas oportunidades de melhoria foram apontadas, a saber:

OM1. A ferramenta não permitia que um processo fosse definido como sendo composto de outros. 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 devem poder 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 dessa versão 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 ISO 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. Neste 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, primeiro, evoluir a ontologia e depois utilizar a nova versão produzida para fundamentar a evolução da infra-estrutura de processos de software do ambiente ODE. A Figura 3 mostra parte da nova versão do modelo da ontologia de processos, que pode ser vista com maiores

detalhes em [Falbo e Bertollo, 2005], onde suas questões de competência e axiomas são apresentados. Os novos conceitos e relações introduzidos estão destacados.

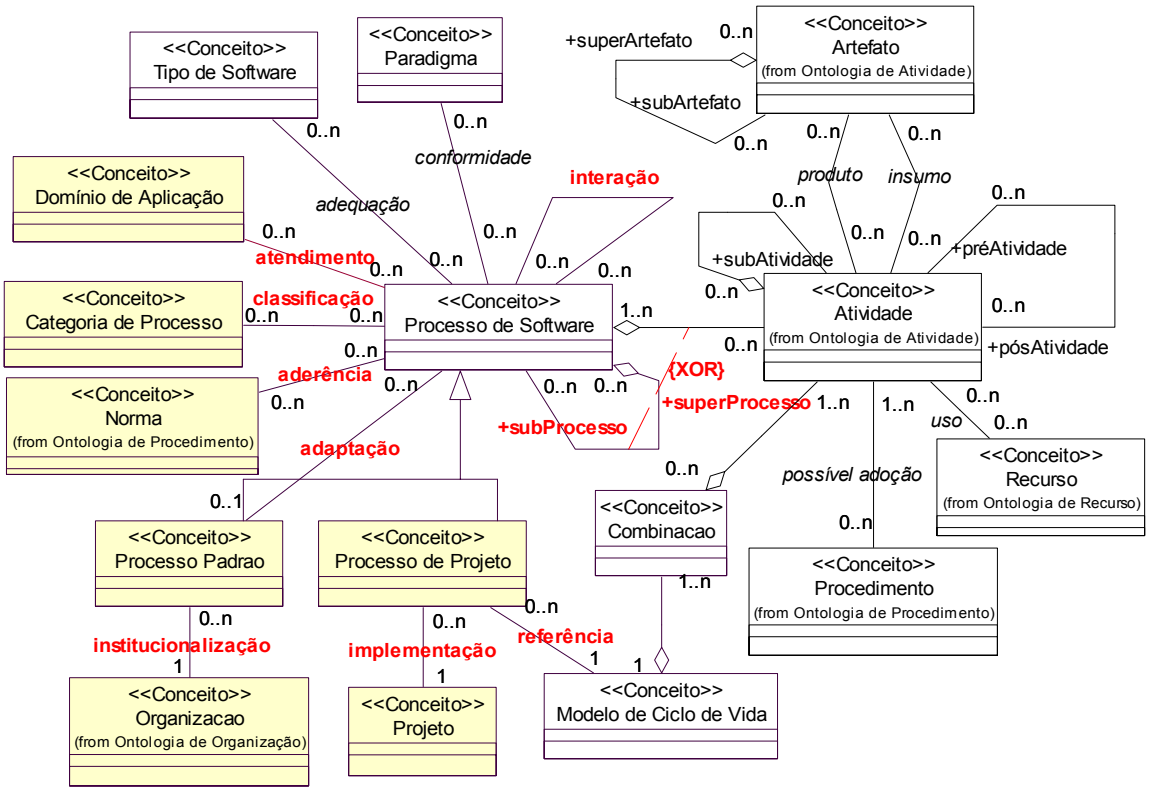

Figura 3 – Modelo Parcial da Nova Versão da Ontologia de Processos de Software.

Foram introduzidos os conceitos de Processo Padrão e Processo de Projeto, como sub-tipos de Processo. Processos Especializados não aparecem explicitamente no modelo, pois são, na verdade, processos padrão adaptados a partir de outros processos padrão. Processos Padrão estão sempre associados a uma Organização, enquanto Processos de Projeto a um Projeto. Um Processo de Projeto é uma adaptação de um Processo Padrão, seja ele especializado ou não. Além disso, um Processo de Projeto define sempre um Modelo de Ciclo de Vida como referência. Processos, de maneira geral, são classificados em Categorias de Processos, podem ou não ser aderentes a Normas e podem ser definidos levando em consideração um Paradigma, Tipo de Software ou Domínio de Aplicação específico. O conceito Paradigma já existia na versão original da ontologia, assim como o conceito de Tipo de Software, mas este último usava o termo Tecnologia de Desenvolvimento para designá-lo, tendo sido alterado.

Finalmente, processos podem ser decompostos em outros processos ou em atividades. Além disso, processos podem ter interações com outros processos de mesmo nível. Os demais conceitos e relações mantiveram-se os mesmos da primeira versão da ontologia.

4.1. A Nova Versão da Infra-estrutura de Processos de ODE

A nova versão da infra-estrutura de processos de software de ODE passou a seguir fielmente o esquema de anotação conceitual discutido na seção 3 e ilustrado pela Figura

2. Além disso, a arquitetura interna da infra-estrutura foi remodelada, para separar os aspectos que descrevem processos padrão, incluindo processos especializados (pacote *ProcessoPadrao*), dos objetos de conhecimento de processo (pacote *ConhecimentoProcesso*) e aqueles que representam ativos de um processo de projeto (pacote *ControleProcesso*). A seguir, o mapeamento de conceitos em classes é apresentado, mostrando-se os diagramas de classes de cada um desses pacotes, começando pelo pacote *ConhecimentoProcesso*, cujo modelo parcial é mostrado na Figura 4, destacando as novas alterações.

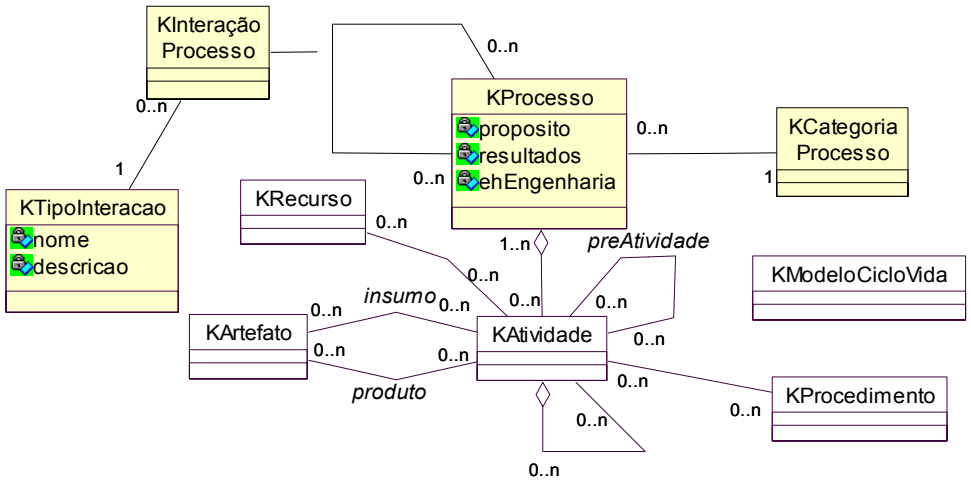

Figura 4 – Modelo Parcial do Pacote *ConhecimentoProcesso*.

A classe *KProcesso*, derivada do conceito processo, representa tipos de processos, tais como processos de Desenvolvimento de Software, Gerência de Projeto etc. Cada processo possui seus propósitos e resultados específicos e é classificado em uma categoria de processo (*KCategoriaProcesso*). Se um processo for definido como sendo um processo de engenharia, então ele será o principal processo a ser seguido, determinando a "espinha dorsal" do processo de software. Os processos que não são de engenharia interagem com o processo de engenharia de alguma forma. A classe KTipoInteracao define os tipos de interação aceitos, a saber: (i) Seqüencial: os processos têm uma relação de término-início, ou seja, um processo B só pode ser iniciado após a conclusão do processo A; (ii) Paralelo Independente: o processo B pode ser iniciado juntamente ou após a conclusão de uma dada atividade do processo A. Contudo, B é executado independentemente de A; (iii) Paralelo Dependente: idem o anterior, mas A pode influenciar B; (iv) Pontual: um processo A interage com um processo B em um ponto específico; (v) Sob-demanda: um processo A interage com um processo B sob-demanda, ou seja, não existe um momento pré-definido.

O pacote *ProcessoPadrão* (Figura 5) contém as classes envolvidas na definição de processos padrão de uma organização. A classe *ProcessoPadrao* representa os processos padrão e especializados. Esses processos podem ser classificados de acordo com categorias de processos. Um processo especializado deve estar levando em consideração um tipo de software, um domínio de aplicação ou um paradigma.

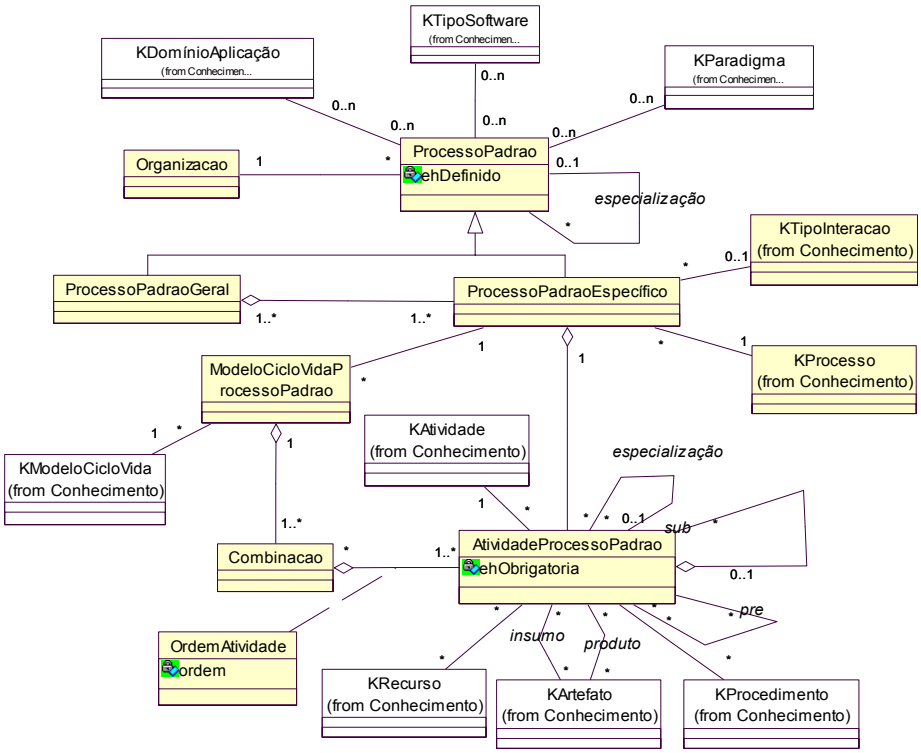

Figura 5 – Modelo Parcial do Pacote *ProcessoPadrão*.

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*. Tomemos como exemplo, a definição do processo padrão do LabES. Podemos definir um processo de software padrão, chamado Processo LabES, que é composto por diversos processos padrão específicos, por exemplo, 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 (desenvolvimento ou manutenção). Os demais processo devem ter sua interação com o processo de engenharia definida através da associação com *KTipoInteracao*. 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.

Para cada processo padrão específico, devem ser definidas as atividades que o compõem. A classe *AtividadeProcessoPadrao* define as atividades que compõem um processo padrão específico. Essas atividades podem ser definidas como obrigatórias ou não. Se for obrigatória, a atividade não pode ser excluída em uma especialização desse processo ou em uma instanciação para um projeto específico. Para cada atividade, deve-se definir suas pré-atividades, sub-atividades, artefatos, recursos e procedimentos. A Figura 6 apresenta a tela de definição do processo padrão da nova versão da ferramenta. A árvore no lado esquerdo da tela apresenta os ativos já definidos para o processo. No lado direito da figura é permitida a escolha de um ativo de processo específico, no caso a escolha dos sub-processos que farão parte do processo padrão. Os ativos disponíveis

são sugeridos conforme itens cadastrados no repositório de conhecimento de ODE e o gerente escolhe aqueles que farão parte do processo. Essa abordagem se repete para os demais ativos de processo, como sub-atividades, artefatos, recursos e procedimentos.

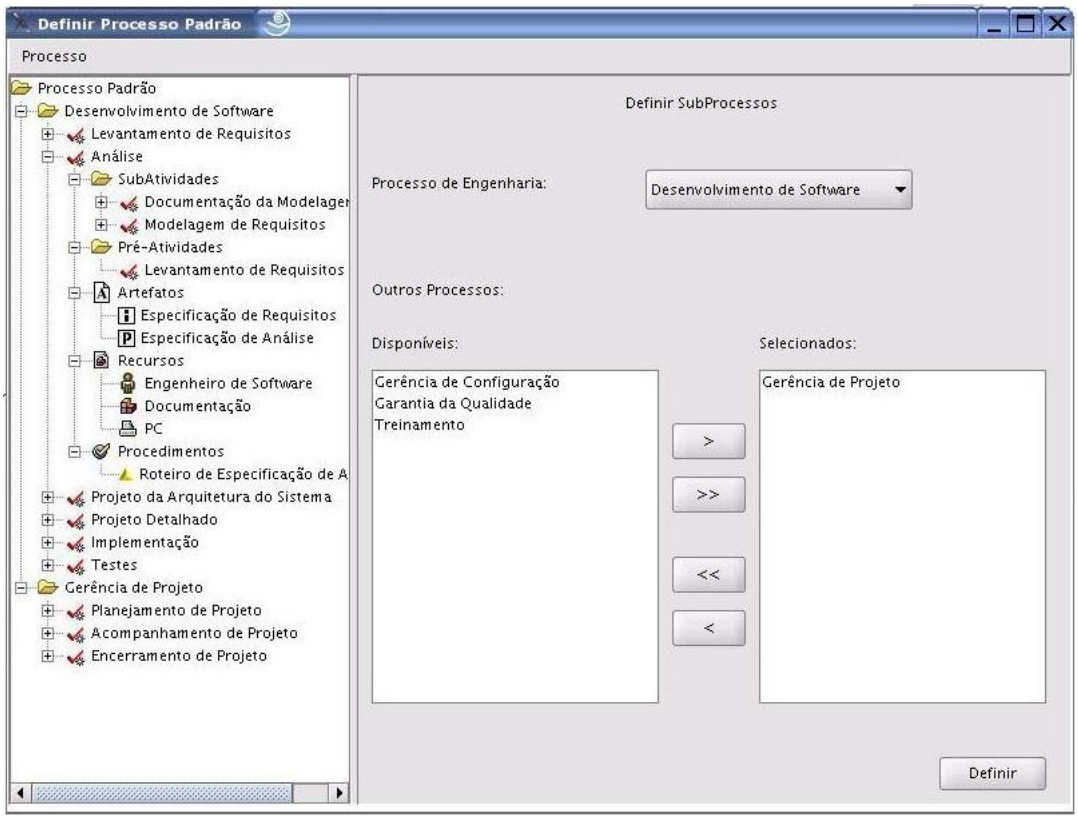

Figura 6 – Definição de Processo Padrão.

Finalmente, deve-se definir os possíveis modelos de ciclo de vida do processo padrão. Isso é necessário para garantir uma restrição imposta por um axioma da ontologia de processos de software que define que as atividades de um modelo de ciclo de vida também devem fazer parte do processo. 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.

O pacote *ControleProcesso*, parcialmente mostrado na Figura 7, 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*). 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 de *ProcessoProjetoEspecifico*, tendo por base um *ProcessoPadraoEspecifico*.

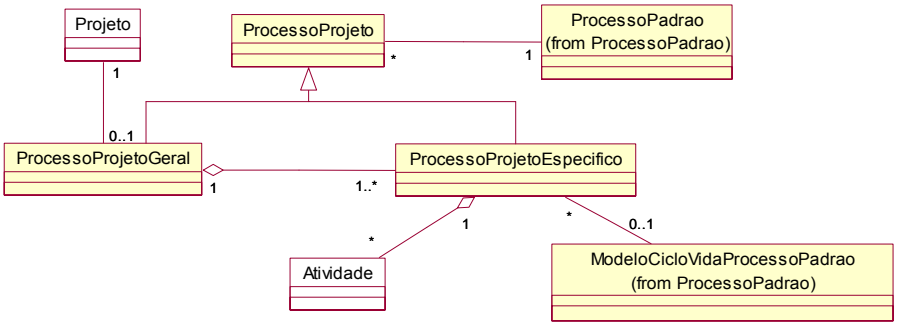

Figura 7 – Modelo Parcial do Pacote *ControleProcesso*.

Escolhidos o processo padrão base e o modelo de ciclo de vida, o processo de projeto é automaticamente montado com os ativos de processo definidos no processo padrão. Contudo, é possível incluir ou modificar atividades e outros ativos de processo.

5. Trabalhos Correlatos

Arbaoui et al. (2002) comparam oito ambientes centrados em processos, observando três aspectos principais: objetivos, características da linguagem de modelagem de processo e características arquiteturais. Considerando apenas o formalismo usado para modelagem de processos, como os oito ambientes citados, ODE também usa um formalismo, no nosso caso, baseado em ontologias. O uso desse formalismo de modelagem tem se mostrado bastante útil, uma vez que não está orientada a muitos detalhes, pois ontologias são elaboradas buscando um comprometimento ontológico mínimo.

Em relação à adoção de uma abordagem em níveis para a definição de processos, encontramos apenas a Estação TABA utilizando uma abordagem semelhante [Berger, 2003]. Inclusive a Estação TABA é baseada em uma ontologia de organização que engloba e estende a ontologia de processo definida em [Falbo, 1998], ou seja a primeira versão sobre a qual a infra-estrutura de processos de ODE foi construída. No que tange, à definição de processos, a Estação TABA permite definir apenas processos de engenharia, incluindo as atividades de outros processos neles, sem tratar explicitamente da interação entre processos.

6. Conclusões e Perspectivas Futuras

Este artigo apresentou a evolução da infra-estrutura de processos de ODE. Nessa evolução, atenção especial foi dada à arquitetura interna da infra-estrutura, bem como ao esquema de anotação conceitual definido recentemente para ODE [Falbo et al., 2005]. Do ponto de vista funcional, ODE passou a oferecer novas funcionalidades, além de preservar características apontadas como importantes em um feedback de uso por uma organização de software. Em relação às oportunidade de melhoria (OM) apontadas na seção 4, as seguintes soluções foram desenvolvidas:

OM1.*A ferramenta não permitia que um processo fosse definido como sendo composto de outros processos*: Com a evolução de ODE, passou-se a permitir a decomposição de processos em outros processos, bem como em atividades. O

processo de software tem de ter um processo de engenharia (desenvolvimento ou manutenção) que serve de espinha dorsal. Outros processos (tais como gerência de projeto, gerência de configuração, garantia da qualidade) podem ser definidos.

OM2. *ODE não tratava a interação entre processos*: Junto com a possibilidade de se definir processos compostos, passou-se a permitir a definição do tipo de interação entre processos.

OM3. *Não era permitida a classificação de processos*: Novas funcionalidades foram adicionadas permitindo à organização definir as categorias de processo que adota, bem como classificar seus processos segundo elas.

OM4. *Falhas na definição de processos, no que tange a artefatos*: Agora é possível, durante a definição de processos, definir que roteiro e/ou ferramenta serão usados na elaboração de um artefato. Essas informações são posteriormente usadas para guiar o acesso a ferramentas e roteiros por ocasião da realização da atividade.

Em relação a trabalhos futuros, a melhoria baseada em gerência de conhecimento é o próximo passo. Quando um processo é executado, muitas lições podem ser aprendidas. Um processo padrão deve evoluir para contemplar novas características. Assim, pode-se trabalhar para evoluir a ferramenta de definição de processo de ODE para apoiar a melhoria de processos de software. Não se pode controlar o que não se pode medir, e então métricas de processo devem ser tratadas. Uma vez consideradas essas métricas, pode-se melhorar o processo de software a partir de dados coletados. Nesse contexto, gerência de conhecimento está sendo considerada um mecanismo para apoiar tarefas de melhoria de processo.

Agradecimentos

Este trabalho foi realizado com o apoio do CNPq e da CAPES, entidades do Governo Brasileiro dedicadas ao desenvolvimento científico e tecnológico, da FAPES, Fundação de Apoio à Ciência e Tecnologia do Espírito Santo, e das empresas VixTeam e Projeta, parceiras que têm financiado o projeto e dado feedback de sua aplicação a casos reais.

Referências

Arbaoui, S., Derniame, J., Oquendo, F. (2002) "Comparative Review of Process-Centered Software Engineering Environments", *Annals of Software Engineering* 14, p. 311-340.

Berger, P. (2003) *Instanciação de Processos de Software em Ambientes Configurados na Estação TABA*. Dissertação de Mestrado, COPPE/UFRJ.

Bertollo, G. e Falbo, R.A. (2003) "Apoio Automatizado à Definição de Processos em Níveis", II Simpósio Brasileiro de Qualidade de Software, 77 – 91, Fortaleza, Brasil.

Chrissis, M.B., Konrad, M., Shrum, S. (2003) *CMMI: Guidelines for Process Integration and Product Improvement*, Addison Wesley.

Duarte, K., Falbo, R.A. (2000) "Uma Ontologia de Qualidade de Software", VII Workshop de Qualidade de Software, João Pessoa, Brasil, p. 275-285.

Falbo, R.A. (1998) Integração de Conhecimento em um Ambiente de Desenvolvimento de Software, Tese de Doutorado, COPPE/UFRJ.

Falbo, R.A., Guizzardi, G., Duarte, K.C. (2002) "An Ontological Approach to Domain Engineering". Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, SEKE'2002, Ischia, Italy, p. 351- 358.

Falbo, R. A., Natali, A. C. C., Mian, P.G., Bertollo, G., Ruy, F.B. (2003) "ODE: Ontology-based software Development Environment", In: Memórias de IX Congreso Argentino de Ciencias de la Computación, p. 1124-1135, La Plata, Argentina.

Falbo, R.A, Ruy, F.B., Bertollo, G., Togneri, D.F. (2004) "Learning How to Manage Risks Using Organizational Knowledge", 6th International Workshop on Learning Software Organization, LSO'2004, Banff, Canada, p. 7-18.

Falbo, R.A., Bertollo, G. (2005) "Establishing a Common Vocabulary for Software Organizations Understand Software Processes", International Workshop on Vocabularies, Ontologies and Rules for the Enterprise, Enschede, The Netherlands.

Falbo, R.A., Ruy, F.B., Dal Moro, R. (2005) "Using Ontologies to Add Semantics to a Software Engineering Environment" Proc. of the 17th International Conference on Software Engineering and Knowledge Engineering, Taipei, China.

Fiorini, S. T., Staa, A., Baptista, R. M. (1998) *Engenharia de Software com CMM*, Brasport.

Fuggetta, A. (2000), "Software Process: A Roadmap", In: Proc. of the Future of Software Engineering, ICSE'2000, Ireland.

Guarino, N. (1998) "Formal Ontology and Information Systems". First International Conference on Formal Ontology in Information Systems, Trento, Italy.

Harrison, W., Ossher, H., Tarr, P. (2000) "Software Engineering Tools and Environments: A Roadmap". *Proceedings of The Future of Software Engineering (ICSE'2000)*. Limerick, Ireland, p. 263 – 277.

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

ISO/IEC 15504 (2003) *Information Technology – Process Assessment.*

Nardi, J.C., Falbo, R.A. (2006) "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.

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

Rocha, A.R.C., Maldonado, J.C., Weber, K.C. (2001) *Qualidade de Software: Teoria e Prática*, Prentice Hall.

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