
Universidade Federal de Pernambuco
Centro de Informática

Graduação em Ciência da Computação

# *Promodeller* XMI: Ferramenta de Modelagem de Processo de Software a partir de Arquivos no padrão XMI utilizando Ontologia de Processo

Edvaldo Lopes da Silva Filho

TRABALHO DE GRADUAÇÃO

Recife, junho de 2008


Universidade Federal de Pernambuco
Centro de Informática

Edvaldo Lopes da Silva Filho

# *Promodeller* XMI: Ferramenta de Modelagem de Processo de Software a partir de Arquivos no padrão XMI utilizando Ontologia de Processo

*Monografia apresentada ao Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.*

*Orientador: Alexandre Vasconcelos*

Recife, junho de 2008

*No final tudo dá certo. Se até agora não deu certo, é porque ainda não chegou ao fim.*

*Fernando Sabino*

# Agradecimentos

A meus pais e meus irmãos, pelo apoio e carinho dispensados em todas as etapas de minha vida, sendo, não apenas meros personagens, mas sim protagonistas desta conquista.

A minha avó Fia, meu tio André e meu primo Rodrigo, por sempre terem acreditado no meu potencial e vibrarem com minhas vitórias.

A todos do Centro de Informática, em especial, a meu orientador, Alexandre Vasconcelos e a meu co-orientador Fernando Bione, pela atenção, seriedade e forte contribuição durante este trabalho.

A meus amigos da TCI, principamente a Mano, Renatinha, Jinmi e Tay, pela excelente amizade contruída, pelos conhecimentos repassados e por sempre compreenderem quando foi necessário me ausentar para poder me dedicar a este trabalho.

A meus amigos Rafael, Renan e Siqueira pelos grandes momentos vividos dentro e fora da faculdade. Com certeza continuaremos grandes amigos, mesmo com o fim do curso.

A meus amigos de turma e de farra, Laís, Buarque, Yuri, Botão, Coruja e Aretakis, pelo companheirismo nas noites passadas no CIn, pelas várias viagens que fizemos e por todas as farras que ainda vamos fazer.

A todos vocês, MUITO OBRIGADO!

# Resumo

Para automatizar a elaboração e acompanhamento de um processo de software, é comum as empresas de desenvolvimento utilizarem de serviços oferecidos por ferramentas cases. Cada ferramenta possui diferentes finalidades, provê benefícios distintos, sendo mais adequada para diferentes pontos da criação e apoio do processo de software.

O ideal, na elaboração do processo, é a empresa ter a sua disposição vários aplicativos. Isso deixaria o usuário livre para aproveitar as melhores funcionalidades de cada programa e escolher a ferramenta que estiver mais familiarizado. A formação deste "ambiente", constituído por várias ferramentas cases, só é possível se houver uma interoperabilidade entre elas, com o processo podendo ser visualizado por qualquer programa.

Este trabalho visa inserir na ferramenta de modelagem de processo em SPEM, *Promodeller,* as funções de importar e exportar arquivos que representam modelagens de processo. Este arquivo será escrito no formato definido pela OMG, XMI (XML Metadata Interchange). Assim, além de ler e exibir modelagens vindas de outras ferramentas, o *Promodeller* poderá disponibilizar seus modelos gerados para outras aplicações, desde que estas também utilizem o padrão XMI.

Palavras Chaves: Processo de software, XMI, SPEM.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# 1. Introdução

No contexto atual das empresas de desenvolvimento de software, onde as organizações têm prazos e recursos limitados, elaborar um processo de software que se adéqüe à realidade da empresa é uma atividade que demanda bastante trabalho e envolve praticamente toda a empresa. Além disso, exige muito conhecimento técnico, tanto para definir o fluxo correto de tarefas, quanto para construir a modelagem.

Uma das principais dificuldades encontradas pela da Engenharia de Software é prover os mecanismos apropriados para que o processo de desenvolvimento de software se dê com qualidade e produtividade. Esse processo geralmente é complexo devido à grande quantidade de atividades e informações envolvidas. Diante disso, diversas ferramentas de modelagem surgiram com o objetivo de facilitar seu desenvolvimento e aumentar as chances te torná-lo bem sucedido [1].

Entretanto, verificou-se que a maioria das ferramentas possuem determinadas funcionalidades que apóiam pontos específicos do processo, motivando a necessidade do uso de mais de uma ferramenta para poder contemplar todo o processo. Desta maneira, a utilização de mais de um aplicativo exige a necessidade da troca de informações entre eles para elaborar o processo de software utilizando os melhores recursos de cada programa utilizado.

Tais necessidades motivaram a idéia de um Ambiente de Desenvolvimento de Software, com o objetivo principal de prover a interoperabilidade entre todas as aplicações envolvidas no desenvolvimento do processo. Assim, o processo pode receber o suporte de um conjunto de ferramentas integradas, com os usuários utilizando as melhores características de cada uma.

Construir esse ambiente colaborativo não é uma tarefa simples, pois cada ferramenta possui suas peculiaridades, com diferentes formatos de persistências

e protocolos. Uma das formas de adaptar ferramentas, que desempenham suas funções de forma impendente, para transmitir dados para outros aplicativos, seria com o uso de um formato produzido em uma linguagem que pudesse ser interpretada por todas as ferramentas envolvidas na construção do processo.

XMI é um formato aberto, baseado em XML, criado justamente para evitar os formatos proprietários das ferramentas, permitindo a troca de modelos entre ferramentas de modelagem diferentes. Atualmente, grande parte dos fabricantes de ferramentas de moelagem já tem incluído o XMI aos seus produtos, possibilitando assim o compartilhamento de informações entre os aplicativos [2].

A idéia é que as modelagens feitas em uma ferramenta possam ser mapeadas em um arquivo XMI. Com isso, outras ferramentas conseguiriam ler o arquivo, podendo decodificá-lo e gerar novamente a modelagem de origem. Esta interoperabilidade é fundamental para o processo receber um suporte, aproveitando os melhores recursos de cada ferramenta, além de dar a opção de o usuário escolher o programa que estiver mais familiarizado.

## 1.1 Objetivos

O objetivo deste trabalho é adaptar a ferramenta de modelagem de processos em SPEM, *Promodeller*, para representar suas modelagens em arquivos no formato XMI. Com isso o *Promodeller* deixa de ser uma ferramenta isolada, passando a ter a possibilidade integrar-se com outras ferramentas de modelagem, através das funcionalidades importar/exportar arquivos XMI.

Além de permitir a visualização dos processos por outras ferramentas, o uso do XMI insere o *ImPProS* (Ambiente de Implementação Progressiva de Processo de Software), desenvolvido no CIn/UFPE, em um Ambiente de Desenvolvimento de Software mais completo. Isso é possível, pois tanto o *ImPProS* como o *Promodeller* são capazes de representar seus processos usando ontologias de processo. Assim, um processo elaborado no *ImPProS* pode ser visualizado pelo *Promodeller* através de sua ontologia, e, em seguida, pode ser lido por outros programas de modelagem depois de ser exportado para XMI pelo *Promodeller*.

## 1.2 Estrutura do Trabalho

Este trabalho está dividido da seguinte maneira:

- O Capítulo 2 apresenta a notação usada pelo *Promodeller*, SPEM;
- O Capítulo 3 mostra uma Ontologia de Processo de Software, enfatizando seus aspectos mais relevantes utilizado pela a ferramenta;
- O Capítulo 4 define o formato XMI e como ele é usado pelo *Promodeller.*
- O Capítulo 5 apresenta a estrutura da ferramenta *Promodeller;*
- O Capítulo 6 é reservado para conclusões e trabalhos futuros.

# 2. SPEM – Software Process Engineering Metamodel

Neste capítulo, abordaremos a notação SPEM, que se trata de um meta-modelo de processo criado para suprir as principais necessidades das técnicas de modelagem de processo existentes [3]. Inicialmente, na seção 2.1, será apresentada uma visão geral sobre processo de software; em seguida, na seção 2.2, veremos alguns conceitos sobre os pacotes do meta-modelo; e por fim, na seção 2.3, haverá uma explanação sobre modelagem usando a notação.

## 2.1 Processo de Software: Visão Geral

Um processo é um conjunto de passos parcialmente ordenados, formados por atividades, métodos, práticas e modificações, usado para atingir um objetivo. Este objetivo geralmente está associado a um ou mais resultados concretos finais, que são os produtos da execução do processo [4].

É estudado dentro da área de engenharia de software, e considerado um dos principais mecanismos para se obter software de qualidade e cumprir corretamente os contratos de desenvolvimento, sendo uma das respostas técnicas mais adequadas para resolver os principais problemas inerentes ao desenvolvimento de software [5].

### 2.1.1 Definição de Processo

Em uma organização existem basicamente três níveis de processo [6]:

Processo Padrão - envolve ciclo de vida de processos propostos na literatura e características gerais da organização;

Processo Especializado – é uma instancia do processo padrão da organização, abrangendo os diversos paradigmas e métodos de desenvolvimento de software;

Processo Instanciado – trata-se de uma instancia de um processo especializado, levando em conta as peculiaridades dos níveis superiores e especificas do projeto, como por exemplo: modelos de ciclo de vida,

características do projeto e da equipe, requisitos de qualidade e disponibilidade de recursos.

A figura 2-1 ajuda a entender os três níveis de processo citados, onde cada nível é uma instancia do nível superior.


**Figura 2-1 Níveis de Processo**

### 2.1.2 Características da Notação

SPEM é um meta-modelo desenvolvido pela OMG em 2002 como um padrão que define estereótipos UML para a modelagem de processos de software. Diagramas de classe, de pacote, de atividade, de caso de uso, de seqüência e de estados, com as devidas restrições, são usados para modelar os vários aspectos de um processo de software [7]. Alguns diagramas como os de implementação e de componentes possuem elementos específicos de UML e, por isso, não fazem parte do escopo da notação SPEM [8]. Sua descrição é baseada em uma arquitetura com quatro camadas, definida pela Object Management Group (OMG), conforme a figura 2-2.

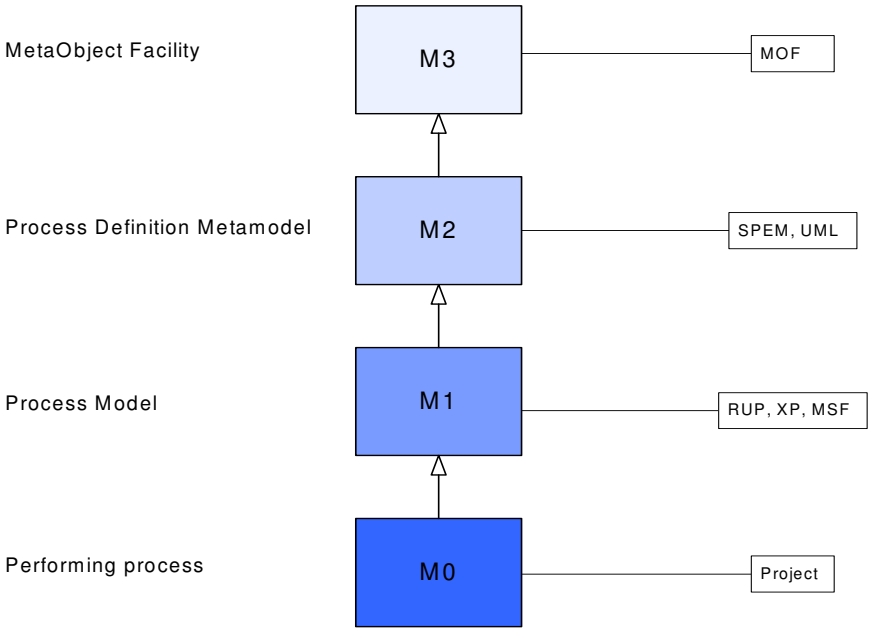


**Figura 2-2 Níveis de modelagem definidos pela OMG [9]**

O nível M0 define um processo instanciado a partir de um modelo de processo para um determinado projeto. Os modelos de processos como o RUP (Rational Unified Process), OPEM, Método IBM SI e XP são descritos pelo nível M1 da arquitetura. O meta-modelo SPEM, que pode ser utilizado para descrever qualquer processo, é definido no nível M2, sendo uma instância do Meta-Object Facility (MOF), descrito no nível M3.

## 2.2 Os Pacotes SPEM

O meta-modelo SPEM é definido pelo pacote “SPEM_Foundation”, uma extensão de um subconjunto do meta-modelo da UML 1.4, e pelo pacote SPEM_Extensions, que adiciona as construções e semânticas necessárias para a engenharia de processos de software, conforme a figura 2-3.

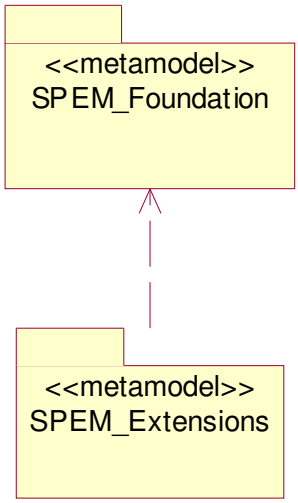


**Figura 2-3 Estrutura de Pacotes do SPEM**

A figura 2-4 apresenta a estrutura interna do Pacote SPEM_Extensions, em termo de seus sub-pacotes, e mostra a sua dependência em relação aos sub-pacotes do SPEM_Foundation:

**Core**: Formado por elementos de modelo, classes, relações, associações, dependências, elementos, elementos auxiliares, etc.

**DataTypes:** É um subconjunto do pacote Data_Types da UML 1.4 e inclui elementos como integer, Boolean, String, enumeração, name, etc.

**Model_Management:** É um subconjunto do pacote Model_Management. Nele, todos os elementos têm visibilidade pública.

**Actions:** É um subconjunto do UML 1.4 Common_Behaviour_Package, e inclui os elementos ModelElement, Action, CallAction, Operation, etc.

**State_Machines:** É um subconjunto do pacote UML State_Machines e apresenta os significados de estado, condição de guarda, transição, estado final e estado composto.

**Activity_Graphs:** É subconjunto do pacote UML Activity_Graph e descreve elementos de modelagem como estado, estado simples, classificador, parâmetro, estado do classificador, etc.

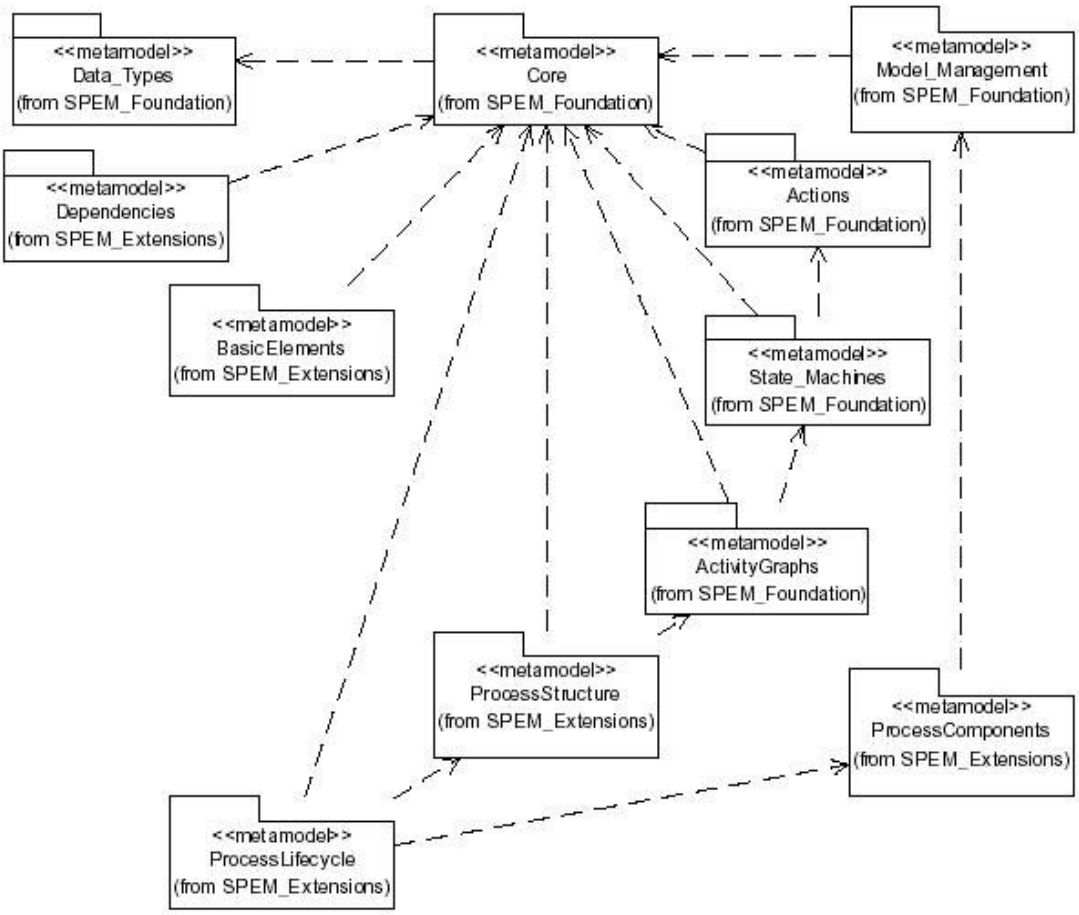


**Figura 2-4 Detalhamento da Estrutura de Pacotes do SPEM**

## 2.3 Modelagem Com SPEM

SPEM é baseado na idéia de que um processo de desenvolvimento de software é uma colaboração entre entidades abstratas chamadas de papel, responsáveis por desempenhar atividades em entidades reais e concretas denominadas Produto de Trabalho, conforme a figura 2-5.

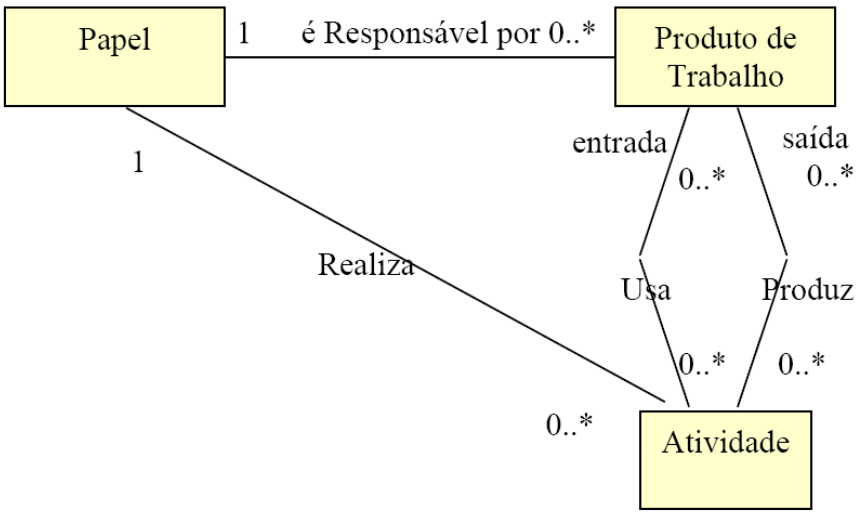


**Figura 2-5 Componentes Básicos do SPEM**

Para os engenheiros de processo e demais usuários do SPEM não é necessário conhecer a fundo todos os conceitos do meta-modelo. Basta apenas entender que SPEM é um perfil da linguagem de modelagem UML, ou seja, uma modelagem de processo utilizando tal notação é apenas um diagrama UML estereotipado. Apresentaremos a seguir um sub-conjunto desses estereótipos utilizados pela meta-linguagem [3].

**WorkProduct:** Tudo que é consumido, produzido, ou alterado por uma atividade é considerado um WorkProduct (artefato). Um WorkProduct pode tanto ser um artefato de entrada (insumo) como um artefato de saída (produto) gerado por uma atividade. Modelos, documentos, código, builds do sistema, planos podem representar artefatos utilizados pelo processo.

**ProcessRole:** São os responsáveis por determinados WorkProducts e por executar as atividades a que são designados no processo.

**ProcessPerformer:** Um ProcessPerformer realiza WorkDefinitions de níveis mais altos, que não podem ser associadas a um único ProcessRoles.

**Dicipline:** Discipline é uma particularização do Package, que divide as atividades dentro de um processo de acordo com um "tema" comum. A divisão das atividades (em seções ou compartimentos) neste modo implica que o Guidance associado e os WorkProducts de saída são categorizados de acordo com o respectivo tema. A inclusão de uma atividade em uma Discipline é representada pela dependência Categorizes, desde que toda atividade seja categorizada por exatamente uma Disciplina.

**Guidance:** Guidance é um elemento que se associa aos demais itens com o objetivo de auxiliar seu desenvolvimento durante o processo. Podem ser representados por Checklists, Guidelines e Templates, etc.

**WorkDefinition:** Uma WorkDefinition tem um ProcessPerformer próprio, representando o papel principal que a executa no processo. Ela pode ser composta de outras WorkDefinitions. Suas subclasses são Activity, Phase, Iteration e LifeCycle.

**Activity:** Atividade é a subclasse mais importante do WorkDefinition e descreve uma parte do trabalho executado por um recurso: as tarefas, operações, e ações que são feitas por ele.

**Phase:** Phase é uma subclasse de WorkDefinition e representa estágios do Ciclo de vida. A Precondição das fases define seu critério de entrada e seu objetivo (geralmente chamado de "milestone" ou "marco") define o critério de saída.

**LifeCycle:** Um LifeCycle é definido como uma seqüência de fases, visando uma meta específica. Ele define o comportamento de um processo completo a ser realizado em um dado projeto.

**Iteration:** Uma Iteração é uma composição de WorkDefinition com algum marco secundário. Estes elementos não descrevem a execução da tarefa em si, são elementos da descrição do processo usados para ajudar a planejar e executá-la.

### 2.3.1 Estereótipos em SPEM

Na seção anterior, apresentamos alguns conceitos de um subconjunto dos estereótipos de SPEM. Na tabela a seguir, mostraremos notações gráficas desse subconjunto, acompanhadas de um resumo explicando o elemento.

**Tabela 1 Principais Estereótipos do SPEM [3]**

| Estereótipo | Resumo | Notação |
|---|---|---|
| WorkProduct | Elementos consumidos ou produzidos por uma atividade. Ex: modelos, documentos, código, builds do sistema, planos, etc. |  |
| WorkDefinition | Elemento que pode ser decomposto em outros elementos. Suas subclasses são: Activity, Phase, Iteration e LifeCycle. |  |
| Guidance | Representa itens que podem auxiliar outros elementos a executar alguma tarefa durante o processo. Ex: guideline, templates, cheklists, ToolMentor, etc. |  |
| Activity | Indica uma tarefa que um ProcessRole realiza no processo. É uma extensão de WorkDefinition. |  |
| ProcessRole | São os responsáveis por executar as atividades dentro do processo. Alguns processRole também possuem responsabilidades sobre determinados WorkProducts. |  |
| ProcessPerformer | São os executores de determinadas tarefas que não podem ser realizadas por um ProcessRoles. |  |
| Phase | Representam as etapas do ciclo de vida utilizadas pelo processo. É uma extensão de WorkDefinition. |  |
| UML Model | São modelos representados em UML. |  |

| | | |
|---|---|---|
| Document | Constituem diversos tipos de WorkProduct, sendo uma extensão dessa entidade Ex: uma planilha, um texto, um diagrama UML, etc. |  |

## 2.4 Considerações Finais do Capítulo

Neste capítulo apresentamos a notação SPEM, o meta-modelo usado pela ferramenta *Promodeller* pra construir modelagens de processo de software. O objetivo maior foi mostrar alguns conceitos como o pacote SPEM, modelagens usando esta notação e seus estereótipos, para que o usuário possa ter um melhor embasamento teórico antes de usar a ferramenta.

No próximo capítulo veremos uma ontologia de processo de software, abordando seus principais conceitos, componentes e como os sistemas podem utilizá-la.

# 3. Ontologia de Processo de Software

Neste capítulo, apresentaremos uma ontologia de processo de desenvolvimento de software, que foi desenvolvida por vários especialistas no assunto, da COPPE/UFRJ (Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa em Engenharia), usando um método chamado Engenharia de Ontologia.

A ontologia foi construída para ajudar no desenvolvimento de ferramentas baseadas no conhecimento sobre processos de software em um meta-ADS, (meta-ambiente). Desta forma, como o *Promodeller* possui estas características, decidimos usar tal formato para tornar persistentes os processos construídos na ferramenta.

O capítulo está estruturado da seguinte maneira: na seção 3.1, apresentaremos uma introdução sobre ontologia de processo de software; na seção 3.2, discutiremos ontologia de atividade; na seção 3.3, falaremos sobre ontologia de procedimento; em seguida, na seção 3.4, será visto ontologia de recurso e finalmente na seção 3.5, abordaremos de forma mais aprofundada ontologia de processo de software, incluindo sua instanciação.

## 3.1 Introdução

A Ontologia de processo de software visa auxiliar a obtenção, coordenação, reuso e a troca de conhecimento sobre processos de desenvolvimento de software. Seus axiomas são mapeados como um conjunto de predicados lógicos que podem ser processados por sistemas programados para tal. Quando se constrói uma ontologia, denota-se que estamos dando um significado a um conjunto de elementos que refletem um processo de software. Sendo assim, a ontologia formada poderá ser usada para compartilhar recursos por ferramentas que sejam capazes de interpretar tal notação.

Como citado nos capítulos anteriores, um processo de software instanciado engloba as atividades realizadas, os insumos consumidos, os

produtos gerados, os recursos utilizados, além de procedimentos que são usados como guia para auxiliar no desempenho de uma atividade. Estes termos são utilizados como base na modelagem de processo de software.

As atividades consistem de um conjunto de passos atômicos que são partes de um trabalho realizado por um ou mais recursos. Elas podem ser alimentadas por algum artefato de entrada (insumo) e devem, obrigatoriamente, gerar pelo menos um artefato de saída (produto). Atividades se comunicam entre si justamente através de produtos/insumos, pois um determinado produto pode servir de insumo para outra atividade.

## 3.2 Ontologia de Atividade

Dentre as características que fazem parte de atividades de processo, de um modo geral, esta ontologia selecionou algumas que considerou mais relevante [10]:

- Taxonomia de atividades;
- Decomposição de atividades;
- Encadeamento de atividades;
- Atividades como primitivas de transformação;
- Recursos requeridos por atividades;
- Adoção de procedimentos na realização de atividades.

### 3.2.1 Taxonomia de Atividades

As atividades de processo de desenvolvimento de software podem ser rotuladas em três grupos: atividades de construção, atividades de gerência e atividades de avaliação da qualidade, conforme é mostrado na figura 3-1. Esses grupos dizem respeito à natureza da atividade.

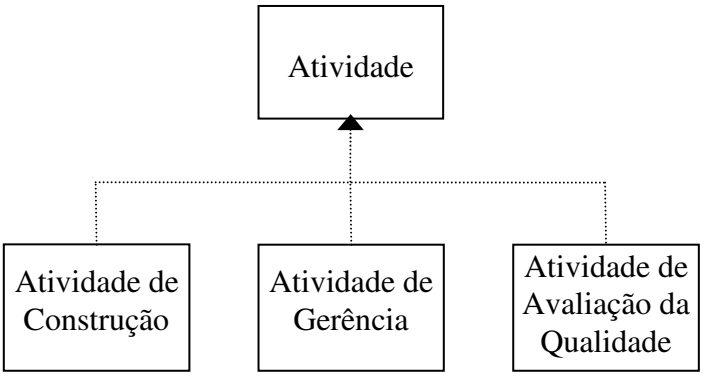


**Figura 3-1 Taxonomia de Atividades [10]**

A ontologia mapeia estes conceitos na seguinte forma de predicados: *atividade(a)*, indicando que *a* é uma atividade; *atconstrução(a)*, denotando que *a* trata-se de atividade de construção; *atgerência(a)*, denotando que *a* é uma atividade de gerência; e *atavqualidade(a)*, denotando que *a* pode ser mapeada como uma atividade de avaliação da qualidade.

### 3.2.2 Decomposição de Atividades

Considerando que as atividades de processo podem ser decomposta em atividades consideradas menores, que quando executadas em conjunto fazem o trabalho da atividade "pai", surge a idéia de macro-atividade(super-atividade) e atividade elementar(sub-atividade). Conforme a ilustra a figura 3-2.

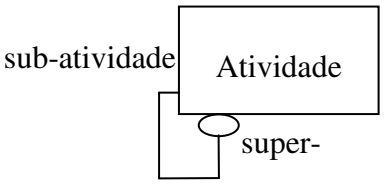


**Figura 3-2 Decomposição de Atividades [10]**

Esta ontologia mapeou o conceito de super-atividade com os predicados *superatividade(a2 ,a1)*, indicando que a atividade *a2* é uma super atividade de *a1* e *subatividade(a1, a2 )* indicando que a atividade *a1* é uma sub-atividade da atividade *a2.*

### 3.2.3 Encadeamento de Atividades

Para definir um processo de produção de software, não basta apenas inserir as atividades que farão parte do processo, é preciso que o fluxo de atividades siga uma seqüência correta, pois há determinadas atividades que dependem de outra. Dessa forma, definiram-se os seguintes predicados: *preatividade(a1 ,a2)* denotando que a atividade *a1* é uma pré-atividade de *a2* e *posatividade(a1 ,a2)* para indicar que a atividade *a1* é uma pós atividade de *a2*, conforme a figura 3-3.

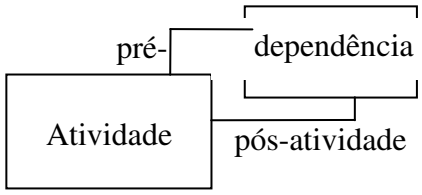


**Figura 3-3 Encadeamento de Atividades [10]**

### 3.2.4 Atividades como Primitivas de Transformação

As atividades de um processo devem produzir um artefato de saída (produto) que pode ser um documento, um modelo, código, builds, releases, componentes, etc. Esses elementos podem, possivelmente, servir de artefato de entrada (insumo) para outra atividade do processo. Ou seja, atividades precisam, obrigatoriamente, gerar um produto e podem, não necessariamente, consumir um insumo. Tais conceitos foram mapeados da seguinte forma por esta ontologia: *insumo(s, a)*, indicando que o artefato *s* é um insumo para a atividade *a* e *produto(s, a)*, indicando que o artefato *s* é um produto gerado pela atividade *a*, , conforme mostra a figura 3-4.

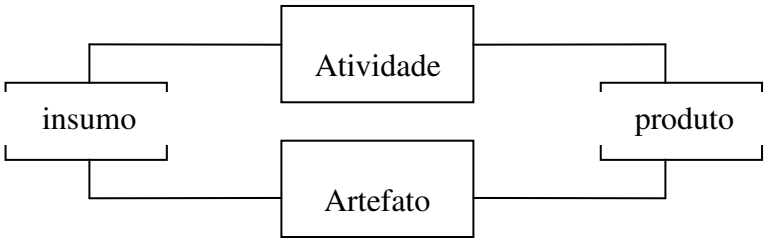


**Figura 3-4 Atividades como Primitivas de Transformação [10]**

## 3.3 Ontologia de Procedimento

Da forma análoga ao que foi feito com as atividades, esta ontologia selecionou algumas características mais relevantes dos procedimentos para fazer parte do seu escopo [10]:

- Taxonomia de procedimentos;
- Adequação de procedimentos;
- (Semi-)Automatização de procedimentos.

### 3.3.1 Taxonomia de Procedimentos

Existem três tipos de procedimentos: método, técnica e diretriz. Os métodos podem ser classificados em métodos de construção, métodos de gerência e métodos de avaliação. As técnicas se subdividem em técnicas de construção, técnicas de gerência e técnicas de avaliação. Já as diretrizes podem ser de dois tipos: roteiro ou norma, conforme a figura 3-5.

Esta ontologia mapeou tais procedimentos da seguinte maneira: *procedimento(p)*, indicando que *p* é um procedimento; *método(m)*, indicando que *m* é um método; *técnica(t)*, indicando que *t* é uma técnica; *diretriz(d)*, indicando que *d* é uma diretriz; *metconstrução(m)*, indicando que *m* é um método de construção; *metgerência(m)*, indicando que *m* é um método de gerência; *metavqualidade(m)*, indicando que *m* é um método de avaliação da qualidade; *tecconstrução(t)*, indicando que *t* é uma técnica de construção;

*tecgerência(t)*, indicando que *t* é uma técnica de gerência; *tecavqualidade(t)*, indicando que *t* é uma técnica de avaliação da qualidade; *roteiro(r)*, indicando que *r* é um roteiro; e por fim, *norma(n)*, indicando que *n* é uma norma [10].

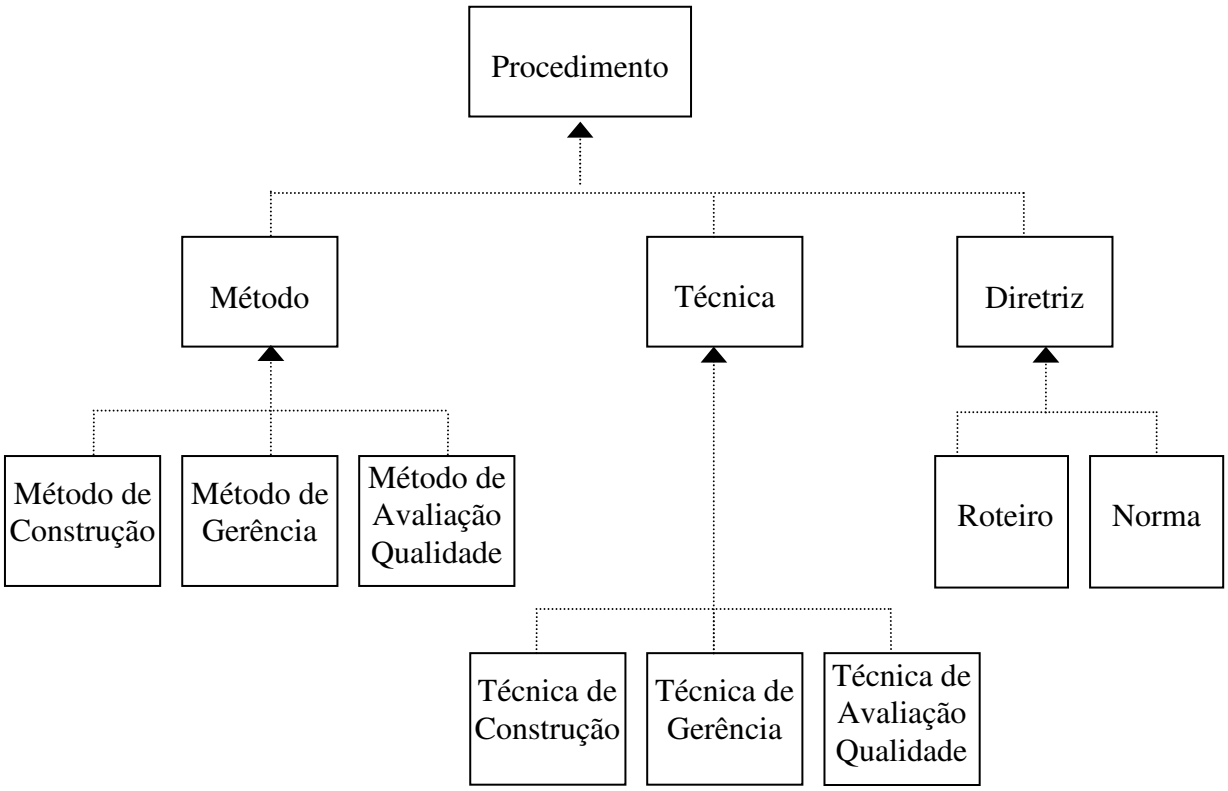


**Figura 3-5 Taxonomia de Procedimentos [10]**

### 3.3.2 Adequação de Procedimentos

No momento em que se vai elaborar o processo de software, é preciso analisar e definir quais, dentre uma gama de procedimentos existentes no processo, farão parte de uma atividade. Para tal, esta ontologia mapeou a possibilidade de uma atividade adotar um procedimento da seguinte forma: *possíveladoção(p,a)*, para indicar que o procedimento *p* pode ser adotado pela atividade *a*, conforme mostra a figura 3-6.

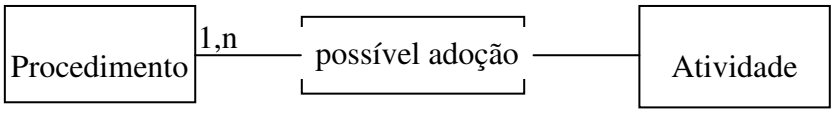


**Figura 3-6 Adequação de Procedimentos [10]**

## 3.4 Ontologia de Recurso

Assim como nos itens de processo avaliados anteriormente, esta ontologia separou as principais características de recursos de processo de desenvolvimento de software para fazer parte do seu escopo [10]:

- Recursos requeridos por atividades;
- Taxonomia de recursos;
- (Semi-)Automatização de procedimentos.

### 3.4.1 Taxonomia de Recursos

Os recursos de processo podem ser classificados em três grupos: recursos humanos, recursos de hardware e recursos de software. Os recursos de software dividem-se em dois conjuntos: ferramentas de software e sistemas de apoio. As ferramentas de software podem ser instanciadas por ferramentas de construção, ferramentas de gerência, ferramentas de avaliação da qualidade e ferramentas de propósito geral, conforme mostra a figura 3-7.

Esta ontologia mapeou estes conceitos na seguinte forma de predicados: *recurso(r)*, indicando que *r* é um recurso; *rechumano(r)*, indicando que *r* é um recurso humano; *rechardware(r)*, indicando que *r* é um recurso de hardware; *recsoftware(r)*, indicando que *r* é um recurso de software; *ferramenta(f)*, indicando que *f* é uma ferramenta de software; *sistemaapoio(s)*, indicando que *s* é um sistema de apoio; *ferconstrução(f)*, indicando que *f* é uma ferramenta de construção; *fergerência(f)*, indicando que *f* é uma ferramenta de gerência; *feravqualidade(f)*, indicando que *f* é uma ferramenta de avaliação da qualidade; e *ferpropgeral(f)*, indicando que *f* é uma ferramenta de propósito geral.

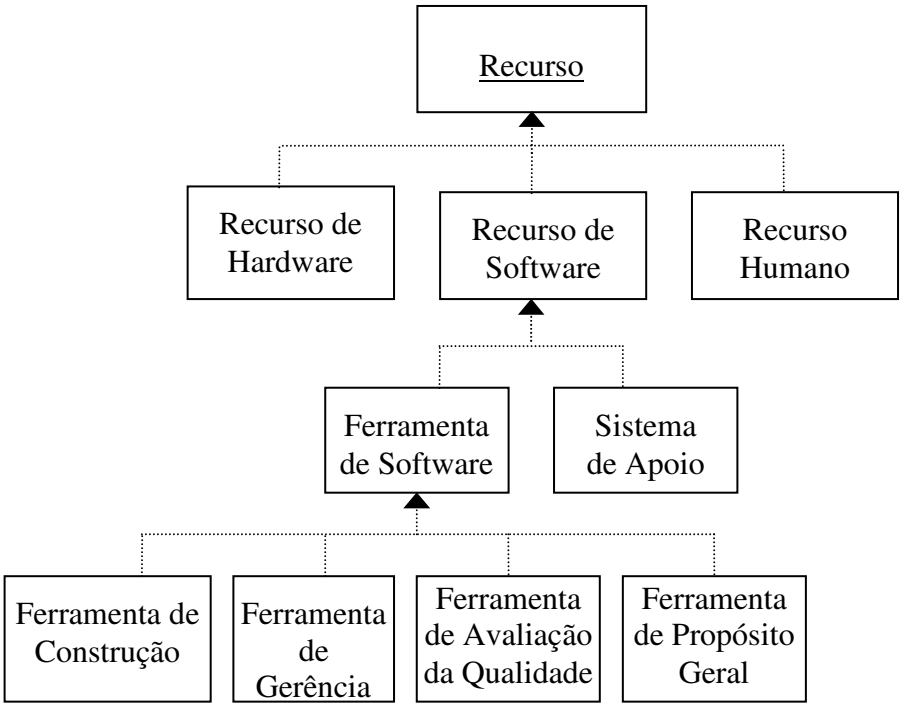


**Figura 3-7 Taxonomia de Recursos [10]**

### 3.4.2 Recursos Requeridos por Atividades

Alguns recursos são considerados como pré-requisitos para realizar uma atividade, pois, em alguns casos, para transformar insumos em produtos, torna-se necessária a participação obrigatória de outros elementos (recursos), que podem ser um ou mais dos citados na seção anterior, conforme a figura 3-8. Esta ontologia mapeou tais conceitos na seguinte forma de predicado: *uso(r,a)*, indicando que é preciso do recurso *r* para realizar a atividade *a*.

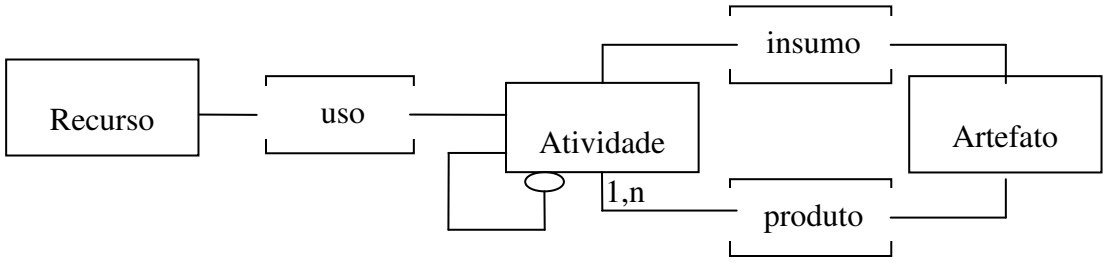


**Figura 3-8 Recursos Requeridos por Atividades[10]**

### 3.4.3 (Semi-) Automatização de Procedimentos

Como sabemos, é comum em um processo de software o uso de diversas ferramentas para (semi-)automatizar um ou mais procedimentos, agilizando o andamento do processo. Dessa forma, esta ontologia definiu o seguinte predicado: *possívelautomatização(f,p)* indicando que a ferramenta *f* pode ser usada para (semi-)automatizar o procedimento *p*, como mostra figura 3-9.

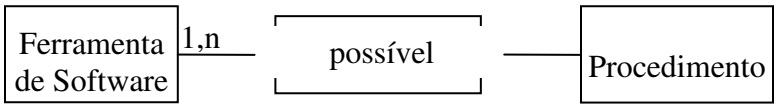


**Figura 3-9 (Semi-) Automatização de Procedimentos [10]**

## 3.5 Ontologia de Processo

Definir as ontologias de atividade, procedimento e recurso, que são as ontologias essenciais para a formação do processo de software, ainda não é o suficiente para que a ontologia possa mapear um processo inteiro de software, pois algumas dúvidas ainda ficam pendentes, como por exemplo: as atividades que devem pertencer ao processo, os procedimentos e recursos que devem fazer parte do processo.

Devido a essas pendências, foi necessário esta ontologia analisar as seguintes características [10]:

- Modelos de ciclo de vida;
- Definição de um processo de software;
- Adequação à tecnologia de desenvolvimento e ao paradigma adotado no desenvolvimento.

### 3.5.1 Modelos de Ciclo de Vida

O modelo de ciclo de vida é uma forma de representar de maneira abstrata o processo de desenvolvimento de software, incluindo todas as fases,

com suas respectivas ordens e interdependências existentes. A ordem em que as fases serão executadas é definida através de combinações que caracterizam a estrutura: seqüencial ou interativa, conforme a figura 3-10.

Esta ontologia definiu os seguintes predicados para dar suporte a tais conceitos: *mciclovida(mcv,e)*, denotando que *mcv* é um modelo de ciclo de vida, cuja forma de estruturação básica é *e*; *combinação(c,e)*, denotando que *c* é uma combinação, cuja forma de estruturação é *e*; *mcv-composição(c,mcv,n)*, indicando que *c* é a n-ésima combinação do modelo de ciclo de vida *mcv*; e *comb-composição(a,c,n)*, denotando que *a* é a n-ésima fase da combinação *c*. O terceiro argumento em *mcv-composição(c,mcv,n)* e *comb-composição(a,c,n)*, *(n)* é necessário para definir a ordem das combinações de um modelo de ciclo de vida e a ordem das fases dentro de uma combinação. O segundo argumento em *mciclovida(mcv,e)* e *combinação(c,e)* representa a estrutura e só pode assumir um dos seguintes valores: {Seq, Iter}, denotando as organizações seqüencial e iterativa, respectivamente [10].

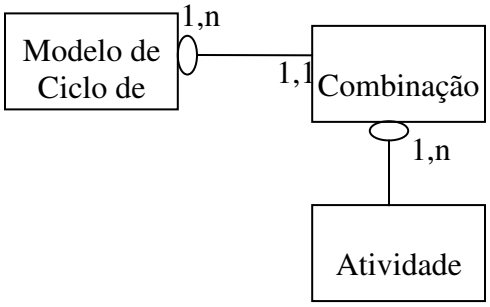


**Figura 3-10 Modelos de Ciclo de Vida [10]**

### 3.5.2 Definição de um Processo de Software

Processo de software constitui-se das diversas fases necessárias para produzir e manter um produto de software. Exigem a organização lógica de diversas atividades técnicas e gerenciais envolvendo recursos, atividades, métodos, ferramentas, artefatos e restrições que possibilitam disciplinar, sistematizar e organizar o desenvolvimento e manutenção de produtos de software [11]. Para tal os predicados *processo(pr)* e *proc-composição(a,pr)*

foram definidos pela ontologia e indicam respectivamente que pr é um processo e que a atividade a faz parte do processo pr. Como mostra a figura 3-11.

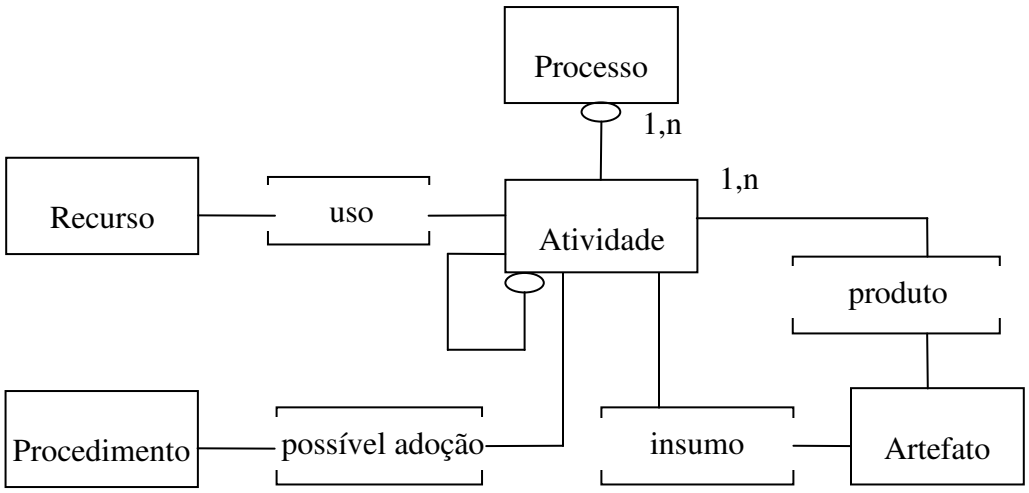


**Figura 3-11 Processo de Software [10]**

### 3.5.3 Adequação à Tecnologia de Desenvolvimento e ao Paradigma

A construção de um processo de software é intimamente influenciada pela tecnologia e paradigma escolhidos. Dessa forma a ontologia definiu o predicado *processo-adequação(pr,td)* para indicar que o processo *pr* é adequado à tecnologia de desenvolvimento *td* e *processo-conformidade(pr,pd)* para indicar que o processo *pr* está em conformidade com o paradigma *pd*, conforme mostra a figura 3-12.

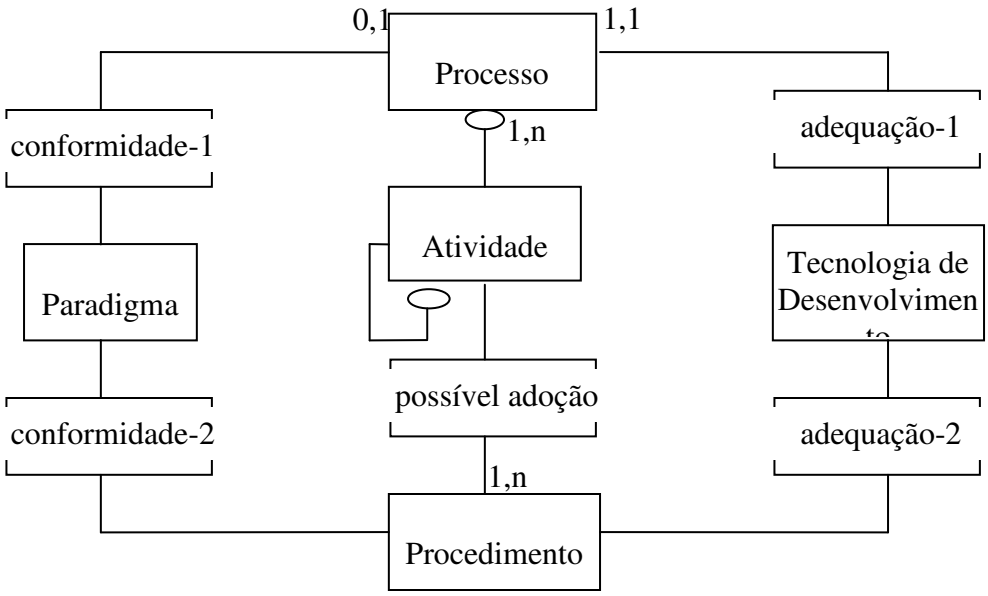


**Figura 3-12 Adequação a Tecnologias de Desenvolvimento e Paradigmas [10]**

### 3.5.4 Instanciação da Ontologia de Processo

Após apresentar todos os elementos presentes, mostraremos uma instanciação desta ontologia de processo. O exemplo na figura 3-13, corresponde a uma instanciação de um processo em SPEM gerado pelo *Promodeller* e representado na forma de predicados descritos pela ontologia. Tal ferramenta utiliza um subconjunto dos predicados visto neste capítulo, para auxiliar na construção e armazenamento dos seus processos.

```
mciclovida(Gestão de Configuração, Seq)
combinacao(Gestão de Configuração, Seq, 1)
mcv-composicao(Gestão de Configuração, Gestão de Configuração, 0)
comb-composicao(Identificar Itens de Configuração,Gestão de Configuração, 0)
insumo(Especificação de Requisitos, Identificar Itens de Configuração)
insumo(Plano de Projeto, Identificar Itens de Configuração)
produto(Lista dos Itens de Configuração, Identificar Itens de Configuração)
uso(Gerente de Configuração, Identificar Itens de Configuração)
comb-composicao(Estabelecer Sistema de Gestão de Configuração,Gestão de
Configuração, 1)
insumo(Plano de Atividades, Estabelecer Sistema de Gestão de Configuração)
insumo(Lista dos Itens de Configuração, Estabelecer Sistema de Gestão de Configuração)
insumo(Plano de Projeto, Estabelecer Sistema de Gestão de Configuração)
```

produto(Plano de Configuração, Estabelecer Sistema de Gestão de Configuração)
produto(Repositorio do Projeto, Estabelecer Sistema de Gestão de Configuração)
uso(Gerente de Configuração, Estabelecer Sistema de Gestão de Configuração)
preatividade(Identificar Itens de Configuração, Estabelecer Sistema de Gestão de Configuração)
comb-composicao(Criar Baselines,Gestão de Configuração, 2)
insumo(Lista dos Itens de Configuração, Criar Baselines)
insumo(Versões atuais dos itens de configuração, Criar Baselines)
produto(Nova Baseline, Criar Baselines)
produto(Branch de Desenvolvimento, Criar Baselines)
uso(Gerente de Configuração, Criar Baselines)
uso(Conselho de Configuração, Criar Baselines)
preatividade(Identificar Itens de Configuração, Criar Baselines)
comb-composicao(Liberar baselines,Gestão de Configuração, 3)
insumo(Nova Baseline, Liberar baselines)
produto(Avaliação da Baseline, Liberar baselines)
produto(Liberação da Baseline, Liberar baselines)
uso(Gerente de Testes, Liberar baselines)
uso(Gerente de Configuração, Liberar baselines)
uso(Conselho de Configuração, Liberar baselines)
preatividade(Criar Baselines, Liberar baselines)
comb-composicao(Rastrear Solicitações de Alteração,Gestão de Configuração, 4)
insumo(Solicitação de Alteração, Rastrear Solicitações de Alteração)
insumo(Lista dos Itens de Configuração, Rastrear Solicitações de Alteração)
insumo(Avaliação da Baseline, Rastrear Solicitações de Alteração)
produto(Rastreamento das Solicitações de Alteração, Rastrear Solicitações de Alteração)
uso(Gerente de Configuração, Rastrear Solicitações de Alteração)
preatividade(Identificar Itens de Configuração, Rastrear Solicitações de Alteração)
preatividade(Liberar baselines, Rastrear Solicitações de Alteração)
comb-composicao(Controlar Itens de Configuração,Gestão de Configuração, 5)
insumo(Lista dos Itens de Configuração, Controlar Itens de Configuração)
insumo(Plano de Configuração, Controlar Itens de Configuração)
insumo(Rastreamento das Solicitações de Alteração, Controlar Itens de Configuração)
insumo(Itens de Configuração Alterados, Controlar Itens de Configuração)
produto(Histórico de Revisões de itens de configuração, Controlar Itens de Configuração)
produto(Arquivos das baselines, Controlar Itens de Configuração)
uso(Gerente de Configuração, Controlar Itens de Configuração)
uso(Conselho de Configuração, Controlar Itens de Configuração)
preatividade(Identificar Itens de Configuração, Controlar Itens de Configuração)
preatividade(Estabelecer Sistema de Gestão de Configuração, Controlar Itens de Configuração)
preatividade(Rastrear Solicitações de Alteração, Controlar Itens de Configuração)
comb-composicao(Realizar Auditoria de Configuração,Gestão de Configuração, 6)
insumo(Plano de Configuração, Realizar Auditoria de Configuração)
insumo(Histórico de Revisões de itens de configuração, Realizar Auditoria de Configuração)

insumo(Repositorio do Projeto, Realizar Auditoria de Configuração)
insumo(Lista dos Itens de Configuração, Realizar Auditoria de Configuração)
produto(Relatório de Auditoria, Realizar Auditoria de Configuração)
uso(Gerente de Configuração, Realizar Auditoria de Configuração)
preatividade(Estabelecer Sistema de Gestão de Configuração, Realizar Auditoria de Configuração)
preatividade(Identificar Itens de Configuração, Realizar Auditoria de Configuração)
preatividade(Controlar Itens de Configuração, Realizar Auditoria de Configuração)
preatividade(Estabelecer Sistema de Gestão de Configuração, Realizar Auditoria de Configuração)
preatividade(Identificar Itens de Configuração, Realizar Auditoria de Configuração)

**Figura 3-13 Instanciação da Ontologia de Processo Gerada Pelo *Promodeller***

## 3.6 Considerações finais do Capítulo

Neste capítulo apresentamos uma ontologia de processo de desenvolvimento de software construída por especialistas da COPPE/UFRJ. O objetivo foi mostrar as ontologias dos vários componentes existentes em um processo: atividade, recurso, procedimento, incluindo, a instanciação de alguns itens. Com isso, o usuário consegue adquirir um melhor embasamento, para entender como o *Promodeller* usa esta ontologia de processo.

No próximo capítulo veremos o formato de intercâmbio para metadados, XMI (XML-based Metadata Interchange). Este formato foi usado pelo *Promodeller* para interagir com outras ferramentas de modelagem que também adotem XMI.

# 4. XMI (XML Metadata Interchange)

Grande parte do tempo gasto, durante a elaboração de um processo de desenvolvimento de software, se deve à modelagem gráfica do processo. Como existem diversas ferramentas de modelagem e todas possuem pontos positivos e negativos, é importante que os usuários aproveitem as melhores características fornecidas por cada uma, para construir um processo de forma mais rápida e eficiente usando um ambiente colaborativo mais poderoso. Desta maneira, é necessário existir um intercâmbio de informações, para que uma modelagem feita em uma determinada ferramenta seja visualizada sem perda de dados em outro ambiente de desenvolvimento.

Uma das formas de prover essa correspondência de informações seria com a existência de um arquivo em um formato que pudesse ser gerado e decodificado por diferentes ferramentas envolvidas na formação do processo. Esse arquivo, escrito em uma linguagem comum, deveria ser exportado por uma aplicação e importado por outra. Uma linguagem tem sido bastante utilizada por sua portabilidade e foi usada pelo *Promodeller*, para realizar a troca de dados com diferentes programas: a XMI (XML Metadata Interchange), definida pela OMG (Object Modeling Group).

Neste capítulo, onde será apresentada a linguagem XMI, está estruturando da seguinte maneira: na seção 4.1 veremos fundamentos da linguagem; na seção 4.2 mostraremos a notação SPEM representada com XMI; na seção 4.3 discutiremos características do arquivo XMI; e finalmente na seção 4.4 falaremos sobre vantagens e desvantagens do seu uso.

## 4.1 Introdução

XMI - XML Metadata Interchange - é um formato para representação e intercâmbio de objetos utilizando XML [12]. É baseado no Meta Object Facility (MOF), tendo como principal meta permitir o compartilhamento de metadados entre ferramentas de modelagem baseadas em UML, e entre repositórios de

metadados baseados no MOF. De forma resumida, pode-se dizer que XMI combina os três diferentes padrões citados: XML (eXtensible Markup Language, padrão da W3C), UML (Unified Modeling Language, padrão da OMG) e MOF (Meta Object Facility, também um padrão da OMG).

Os diagramas em UML podem ser representados em formato XMI, pois UML, que é uma instancia do MOF, encontra-se no nível 2 na arquitetura de camadas da OMG. Desta forma, o XMI pode ser abertamente usado, de maneira padronizada, para o intercâmbio de modelos UML.

XMI utiliza XML como linguagem base, seguindo uma rigorosa abordagem para representar as informações extraídas dos metamodelos construídos. Sendo assim, é possível haver um mapeamento bidirecional entre os modelos e arquivos no formato XMI. Os arquivos podem ser gerados a partir de um metamodelo e, também, podem ser interpretados, e gerar novamente o modelo de origem. Estas características permitem que diversas ferramentas incluam entre suas funcionalidades a opção de importar/exportar XMI.

Por tratar-se de uma notação que visa promover a interoperabilidade entre diversas ferramentas, o uso XMI é visto como um procedimento bastante útil para arquitetos e desenvolvedores manipularem suas modelagens. O XMI possibilita, também, que determinados programas, não necessariamente de modelagem, possam extrair alguma informação de modelos construídos por outros aplicativos. Um exemplo que mostra essa interoperabilidade é mostrado na figura 4-1.

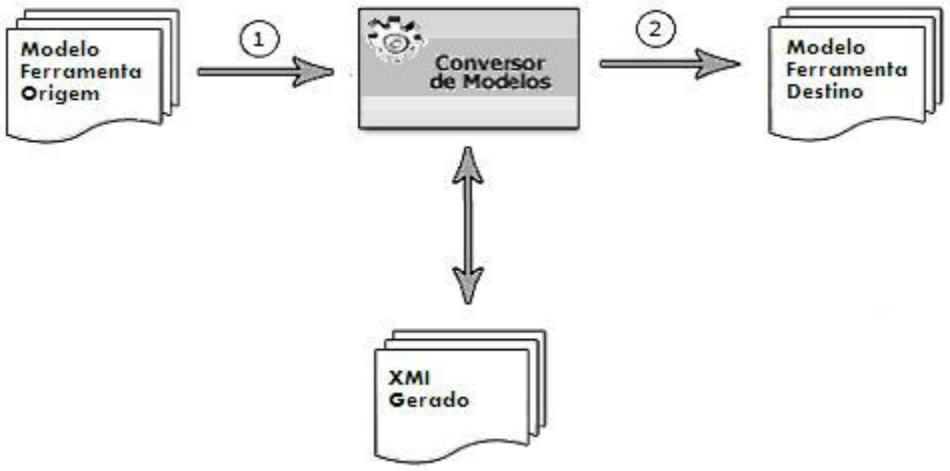


**Figura 4-1 Adequação a Tecnologias de Desenvolvimento e Paradigmas**

## 4.2 Modelagem em SPEM Representada com XMI

Conforme citado anteriormente, o SPEM é uma notação, baseada no MOF, para modelagem de processos de software. Esta notação possui diversos diagramas adaptados de UML para modelar seus processos. Por exemplo, para modelagem que descreve o fluxo de atividades do processo em SPEM (Diagrama de Atividades), pode ser utilizado o Diagrama de Atividades de UML, e para a modelagem dos papéis associados a cada atividade (Diagrama de Atividades), pode ser utilizado o Diagrama de Casos de Uso. [13]

Os elementos de modelagem em SPEM, como atividades (Activity), macro-atividades (WorkDefinition), Fases (Phases), recursos (ProcessRole), etc., possuem um ícone personalizado. Como esta notação trata-se de um perfil da linguagem UML, há um mapeamento destes estereótipos de SPEM para elementos UML. Desta forma, é possível representar os processos modelados usando SPEM em um arquivo XMI, permitindo a integração de diversos programas de modelagem de processo, como mostra figura 4-2.

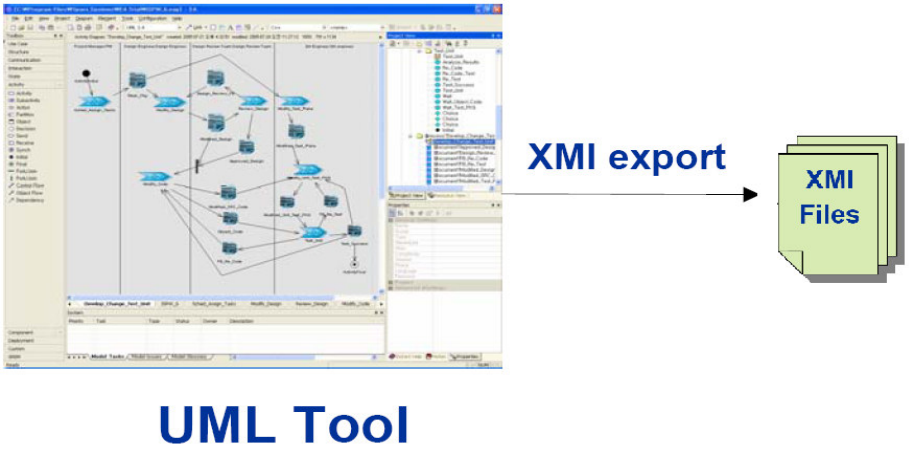


**Figura 4-2 Geração de arquivo XMI a partir de um processo modelado em SPEM [14]**

## 4.3 Elementos do XMI

Existem várias formas de se construir um documento XMI, porém todos os documentos nesse formato devem seguir um padrão descrito pela OMG, com sua composição básica estruturada semelhantemente a uma árvore. Esse padrão começa com um elemento raiz chamado XMI, que possui um atributo para controlar a data de criação do documento, um atributo para controle de versão e um terceiro atributo para controle de verificação.

O atributo de verificação é responsável por indicar se a ferramenta que gerou o arquivo XMI examinou o documento seguindo os padrões definidos pela linguagem. Com isso, apenas erros de sintaxe podem existir no arquivo, não havendo possibilidade de haver erros de semântica [10].

A raiz XMI possui os seguintes elementos “filhos”, mostrados em um exemplo de forma simplificada na figura 4-3:

**XMI.header:** No XMI.header existem informações que identificam o modelo, o metamodelo e o meta-metamodelo de um metadado. Também contém informações da ferramenta que gerou o arquivo XMI.

**XMI.content:** Esta seção concentra a maior quantidade de informações do arquivo, pois contém elementos que representam os modelos transferidos, incluindo dados como: a posição do elemento na modelagem, tamanho do elemento, etc. Cada item da modelagem possui atributos e variáveis correspondentes dentro desta seção.

**XMI.difference:** Esta seção pode estar ou não dentro de content. Basicamente, ela indica diferenças em relação a um modelo anterior, evitando que o arquivo seja totalmente refeito, caso haja necessidade de atualização.

**XMI.extensions:** Qualquer informação adicional ao modelo é colocada nessa seção. È bastante útil para ferramentas inserirem dados extras que precisem ser processados de alguma maneira diferente do padrão do documento.

```
<XMI xmi.version="1.1" xmlns:UML="omg.org/UML1.3" timestamp="2008-04-19 22:43:47">
        <XMI.header>
                <XMI.documentation>
                        <XMI.exporter>Enterprise Architect</XMI.exporter>
                        <XMI.exporterVersion>2.5</XMI.exporterVersion>
                </XMI.documentation>
        </XMI.header>
        <XMI.content>
           <UML:Class name="EARootClass" xmi.id="A7F4" isRoot="true" isLeaf="false" isAbstract="false"/>
           <UML:Comment name="Note" xmi.id="45d8" visibility="public" namespace="EAPK">
             ...
           </UML:Comment>
      </XMI.content>
      <XMI.difference/>
      <XMI.extensions xmi.extender="Enterprise Architect 2.5">
           <EANoteLink xmi.id="8FF8" source="052E" target="B38E"/>
           ...
      </XMI.extensions>
</XMI>
```

**Figura 4-3 Exemplo resumidode um arquivo XMI**

## 4.4 Vantagens e Desvantagens

Analisando as seções anteriores, podemos enfatizar algumas vantagens e desvantagens do uso de XMI, [12]:

### 4.4.1 Vantagens

As seguintes vantagens em utilizar XMI podem ser citadas:

1- Tem como base XML, uma linguagem bastante difundida;
2- Integra diferentes aplicações, formando um ambiente de desenvolvimento colaborativo;
3- Formato suportado por diversas ferramentas;
4- È um padrão aberto;
5- O usuário pode gerar estes arquivos abstraindo sua estrutura;
6- Possibilita o reuso de modelos.

### 4.4.2 Vantagens

Apenas as seguintes desvantagens foram detectadas:

1- Possui uma especificação relativamente complexa;
2- Ainda existem poucos metamodelos prontos para utilizar o XMI.

## 4.5 Considerações Finais do Capítulo

Neste capítulo apresentamos o formato XMI e como ele é usado para integrar ferramentas de modelagem em geral. Mostramos algumas características da linguagem, exemplos do seu uso e como SPEM (notação usada pelo *Promodeller*) utiliza este formato. Constatamos, a partir destes estudos, que XMI facilita consideravelmente o trabalho dos engenheiros de processo, pois possibilita a existência da interoperabilidade entre ferramentas em um ambiente de contrução de processo.

No próximo capítulo, veremos a estrutura da ferramenta *Promodeller* de um modo geral, mostraremos como este sistema faz para mapear a ontologia vista no capítulo anterior e gerar uma modelagem gráfica do processo

correspondente. Também será visto no capítulo seguinte alguns cenários de uso da ferramenta, incluindo a funcionalidade importar/exportar XMI.

# 5. O *Promodeller*

Neste capítulo, onde abordaremos os principais componentes e cenários de uso da ferramenta de modelagem de processo *Promodeller*, encontra-se estruturado da seguinte forma: na seção 5.1 apresentaremos uma idéia geral do aplicativo; em seguida, na seção 5.2, mostraremos como o *Promodeller* se conecta com o *ImPProS* - Ambiente de Implementação Progressiva de Processo de Software - e com outros programas de modelagem de processo; e, finalmente, na seção 5.3 veremos cenários de utilização da da ferramenta.

## 5.1 Visão geral do *Promodeller*

O *Promodeller* é uma ferramenta desenvolvida por alunos do CIn - Centro de informática da UFPE - com o objetivo maior de servir como um suporte para o *ImPProS* modelar e refinar graficamente seus processos desenvolvidos. Essa modelagem é baseada na ontologia de processo visto no capítulo anterior e usa a notação SPEM.

Além da possibilidade de ler os processos vindo do ImPPoS, através de ontologias de processo de desenvolvimento de software, é possível elaborar processos especializados ou instanciados do "zero", usando a notação SPEM. O Promodeller também permite a integração com outras ferramentas de modelagem de processo, através do uso de XMI, descrito no capitulo anterior.

A figura 5-1 ilustra um exemplo de um processo simples modelado no *Promodeller*. Dependendo de suas necessidades, usuário pode visualizar a ferramenta sob diversas perspectivas. As seguintes necessidades são providas pelo *Promodeller:*

- Exibir graficamente os processos construídos no *ImPProS*;
- Refinar os processos feitos no *ImPProS*;
- Modelar um processo do zero;

- Ser um elo entre o *ImPProS* e as diversas ferramentas de processo;
- Conhecer a notação SPEM;
- Construir ontologia de processo.

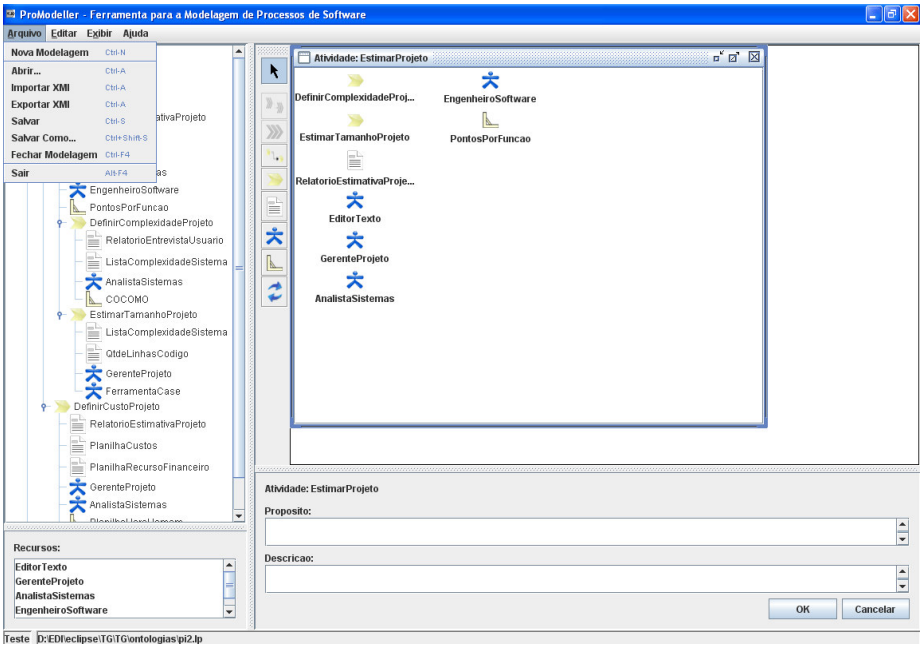


**Figura 5-1 Interface do *Promodeller***

## 5.2 Integração do *Promodeller*, *ImPProS* e Ferramentas de Modelagem de Processo

Conforme citado na seção anterior, o Promodeller se conecta com o *ImPProS*, através de arquivos de ontologia de processo de software, gerados após o usuário montar seu processo usando toda gama de recursos disponibilizada pelo *ImPPros*.

### 5.2.1 *ImPProS* e *Promodeller*

A Figura 5-2 ilustra como ocorre essa ligação entre os dois aplicativos.

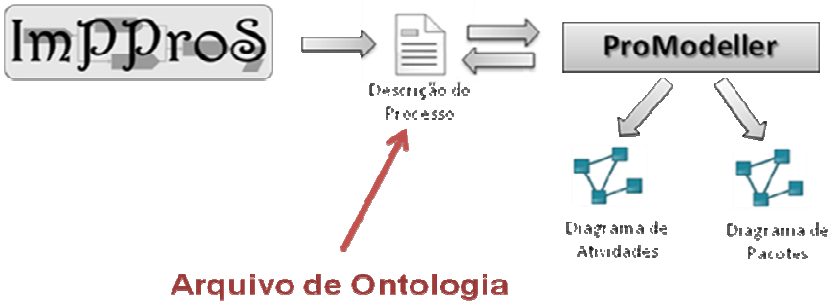


**Figura 5-2 Integração entre o *ImPProS* e o *Promodeller [bione]***

Como pode ser visto, o arquivo de ontologia é o elo entre o *ImPProS* e o Promodeller. Além de representar todo o processo montado no *ImPProS*, a ontologia pode ser lida e processada pelo Promodeller, que gera diagramas de pacote e Atividades, usando a notação SPEM.

O diagrama de atividades mostra, basicamente, o fluxo de tarefas alocadas para cada recurso, juntamente com seus produtos, insumos e procedimentos. Já o diagrama de pacotes exibe pedaços do processo (atividades e macro-atividades) de forma hierárquica, com as ligações e dependências existentes entre eles, pois pacotes podem depender de outros pacotes.

### 5.2.2 Formação de Ambiente Colaborativo

Seria interessante, também, que o Promodeller pudesse trocar informações com outros aplicativos. Isso contribuiria para formação de um ambiente colaborativo, onde o usuário ficaria livre para escolher em qual ferramenta construirá seu processo. Neste ambiente, um processo elaborado usando uma aplicação poderia ser migrado sem perdas para outro programa, quando necessário.

No Promodeller, essa interoperabilidade foi conseguida com a capacidade dele importar/exportar arquivos XMI, uma vez que um grande número de ferramentas também dá suporte a este formato. A figura 5-3 ilustra

como ocorre essa integração. Como pode ser visto, os arquivos de ontologias interligam o *ImPProS* e o Promodeller, enquanto o formato XMI comunica o Promodeller com demais ferramentas de modelagem de processo.

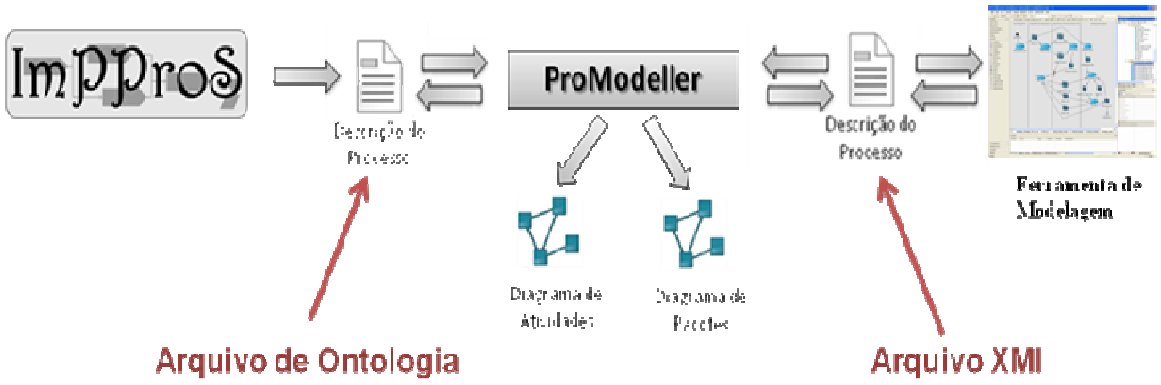


**Figura 5-3 Integração entre várias ferramentas de modelagem de Processo**

## 5.3 Cenários de Uso

Nas próximas subseções serão mostrados alguns cenários de uso do Promodeller, incluindo os passos necessários para realizá-lo.

### 5.3.1 Criando um processo do "Zero"

As telas a seguir mostrarão os passos mais importantes para a criação de um processo instanciado desde o seu início.

#### Nova modelagem

Inicialmente, o usuário terá que escolher a opção "Nova Modelagem" no menu Arquivo.

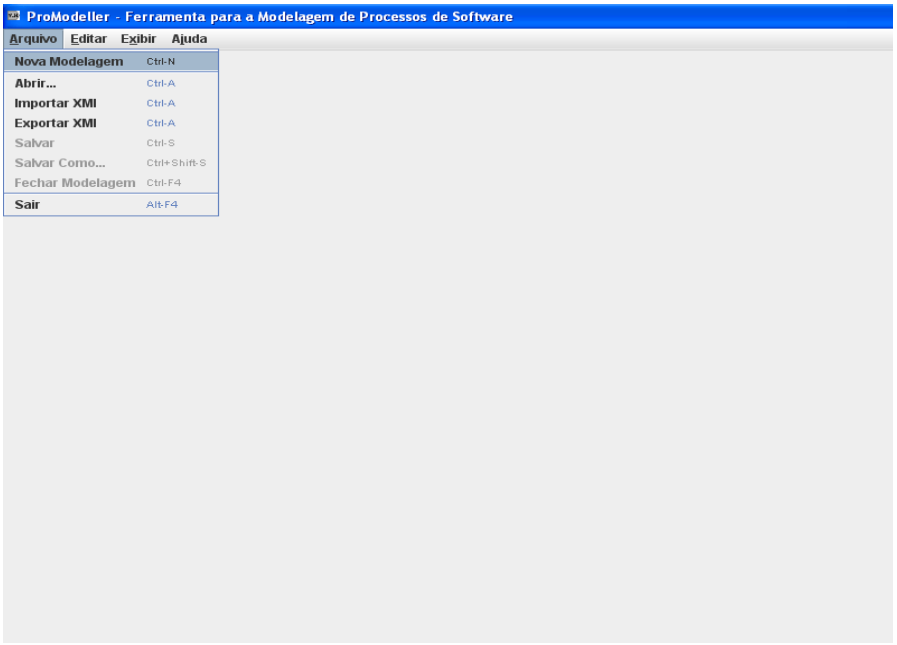


**Figura 5-4 Criando uma nova modelagem**

## Tipo de Processo

Em seguida, o usuário optará entre um processo especializado e instanciado. Para exemplo, selecionamos um processo instanciado.

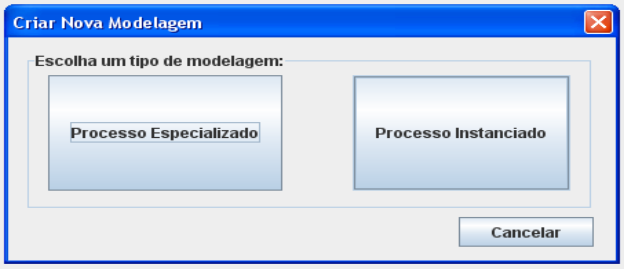


**Figura 5-5 Escolhendo o tipo de processo**

## Inserindo o nome do modelo e tipo de ciclo de vida

Após escolher o tipo de processo, o usuário dará um nome a seu modelo e escolherá o seu tipo (Seqüencial ou Iterativo).


**Figura 5-6 Inserindo o nome do modelo e o tipo de Ciclo de Vida**

Aparecerá a tela a seguir, que possui basicamente quatro áreas principais:

- A área (1) é reservada para exibição da estrutura do processo na forma de árvore;
- Na área (2) serão exibidos os diagramas de pacotes das macro-atividades (Atividades e Fases) do processo. Para que os diagramas apareçam, o usuário deve clicar com o botão direito na árvore do processo;
- Uma lista com todos os recursos envolvidos no processo está na área (3);
- E finalmente a área (4) contém informações mais detalhadas do pacote exibido.

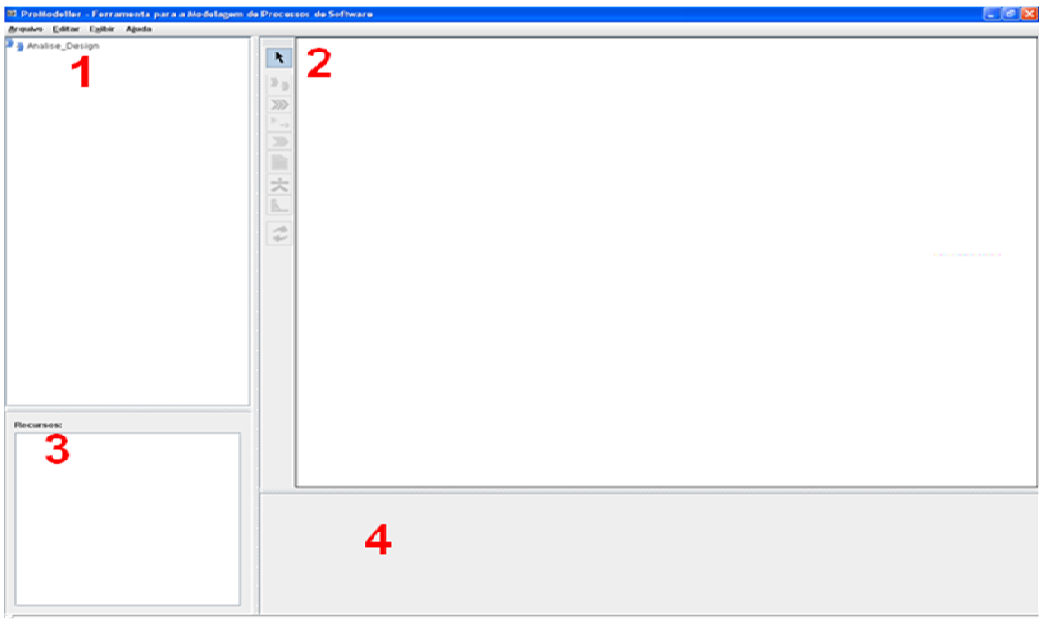


**Figura 5-7 Tela inicial de um processo**

## Elementos do Processo

Para inserir os elementos do processo nos diagramas, o Promodeller disponibiliza botões que representam um subconjunto de itens da notação SPEM. Além desses itens, existe um botão chamado “Realizar Merge de Atividades”. Essa função pode ser utilizada quando o usuário detecta que no processo existem atividades semelhantes e ache viável realizar merge entre elas, refinando os elementos participantes de ambas e gerando uma só atividade. A figura5-8 mostra tais botões com suas respectivas funcionalidades.

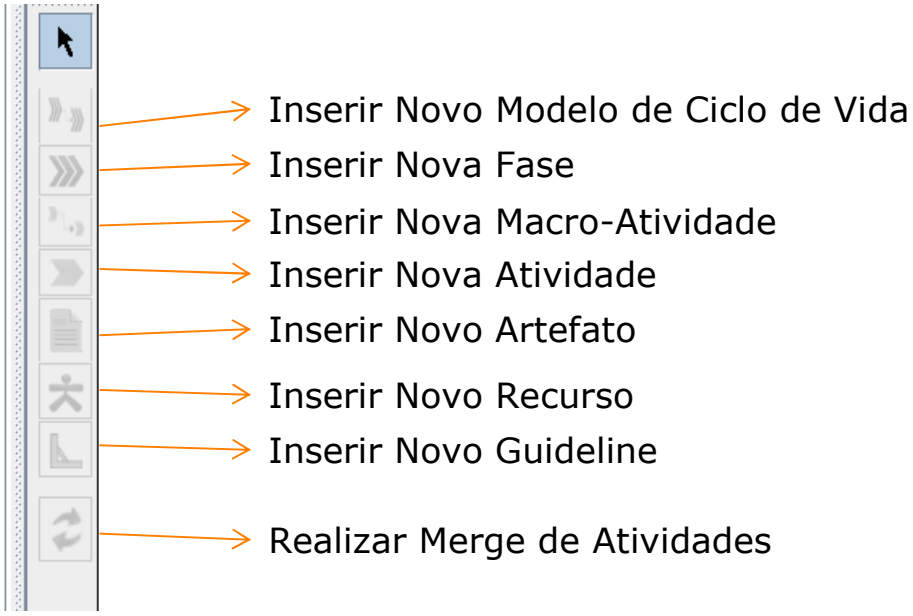


**Figura 5-8 Botões dos elementos do processo**

### Nova Fase

Para o usuário criar uma fase no ciclo de vida, basta clicar com o botão direito na raiz da árvore e, após abrir a janela do ciclo, clicar no botão Inserir Nova Fase, escolhendo o seu tipo (seqüencial ou iterativo) e o número de iterações, conforme a figura 5-9.

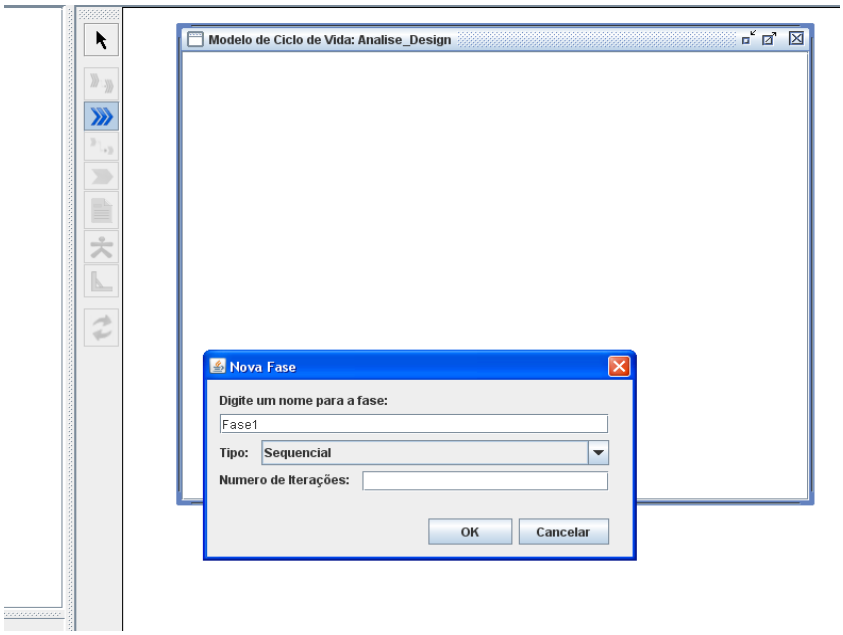


**Figura 5-9 Inserindo uma fase**

## Nova Atividade

As atividades podem ser inseridas em uma fase semelhantemente à função anterior: deve-se clicar com o botão direito sobre a fase desejada na árvore, e, em seguida, clicar no botão Inserir Macro-atividade ou Atividade. A diferença entre Macro-atividade e atividade é que as macro-atividades podem ter atividades como filha. Para exemplificar, escolhemos adicionar quatro atividades, conforme a figura 5-10.

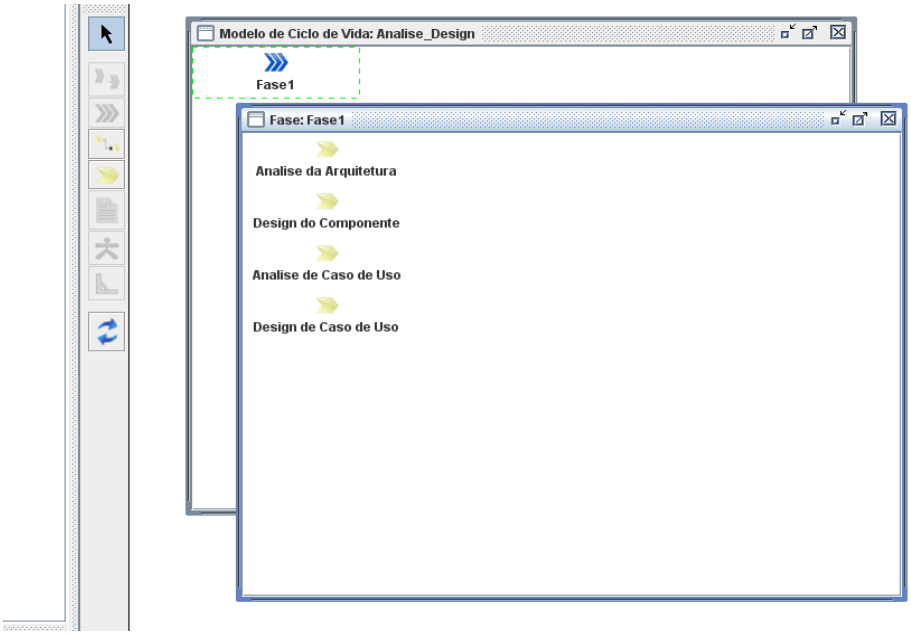


**Figura 5-10 Inserindo atividades**

## Novo Artefato

As telas a seguir mostram que para adicionar um artefato (Insumo ou produto) a uma atividade, deve-se seguir o mesmo padrão, selecionando a atividade desejada na árvore e clicando no botão Inserir Artefato.

## Criando e inserindo um insumo

Um insumo pode ser associado a uma atividade de duas maneiras: quando ele é criado a partir do zero, escolhendo-se a opção “criar novo” conforme mostra a figura 5-11 ou utilizando um artefato já existente no processo, resultado de um produto de alguma atividade, como mostra a figura 5-12.

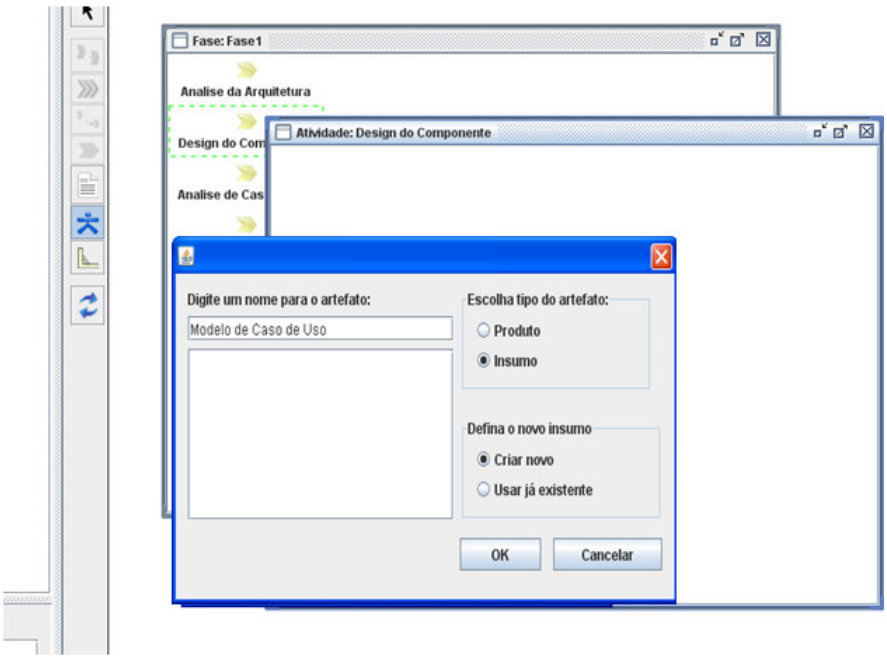


**Figura 5-11 Criando um insumo**

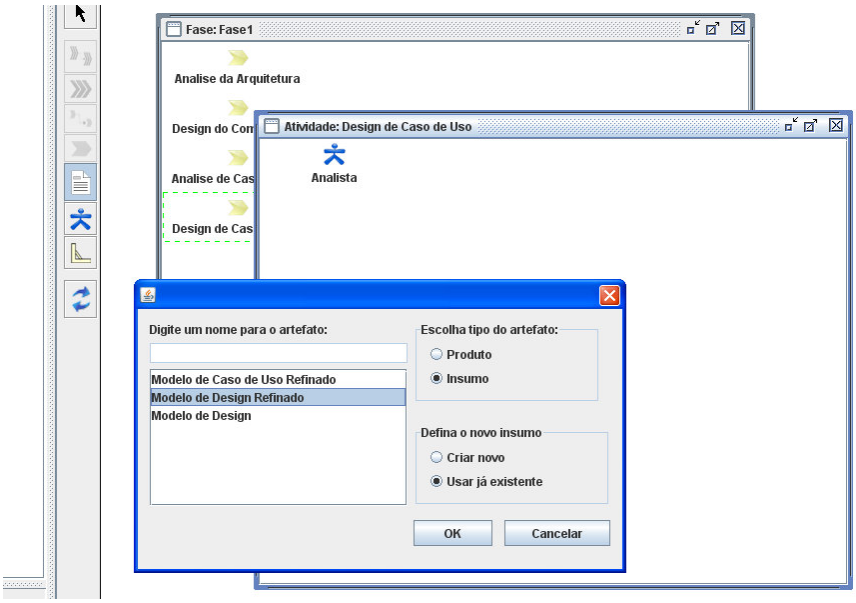


**Figura 5-12 Inserindo um insumo já existente**

## Inserindo um produto

Diferente dos insumos, não faz sentido criar um produto já existente, pois tais artefatos são resultados de uma seqüência de tarefas feitas durante uma

atividade. Sendo assim, os produtos são inseridos nas atividades sempre do zero. Conforme a figura 5-13.

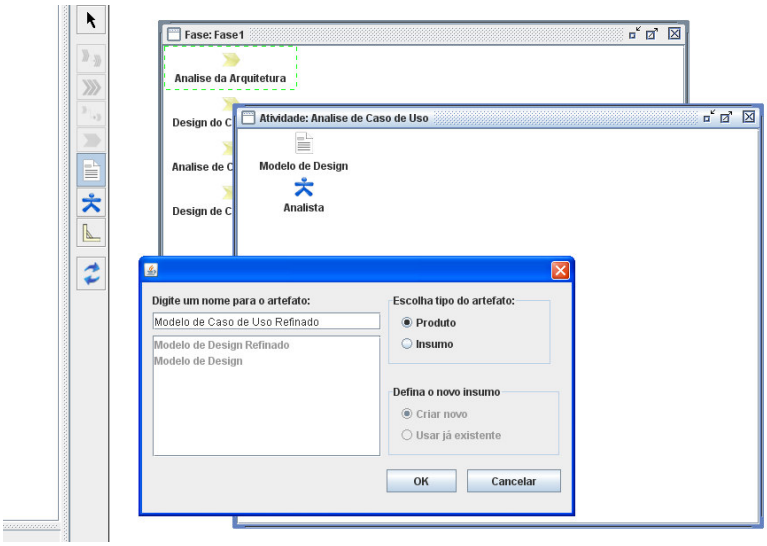


**Figura 5-13 Inserindo um produto**

## Criando e inserindo um recurso

Os recursos seguem a mesma lógica dos insumos: podem ser inseridos do zero, figura 5-14, ou reaproveitados, caso já existam no processo, figura 5-15.

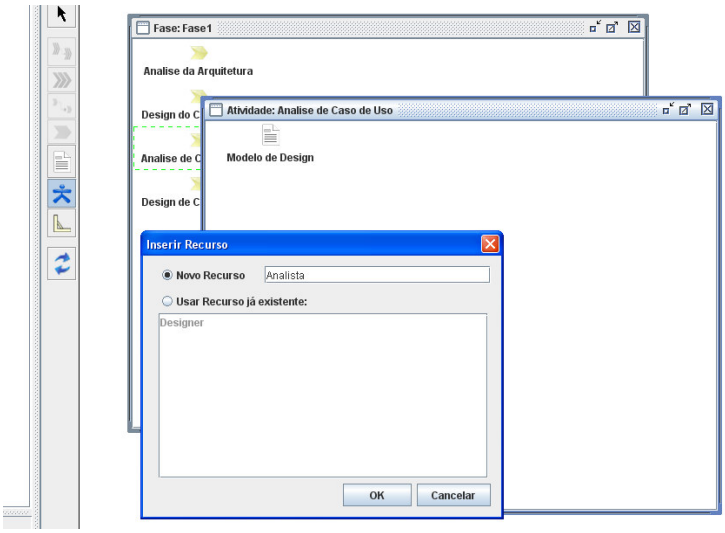


**Figura 5-14 Criando um recurso**

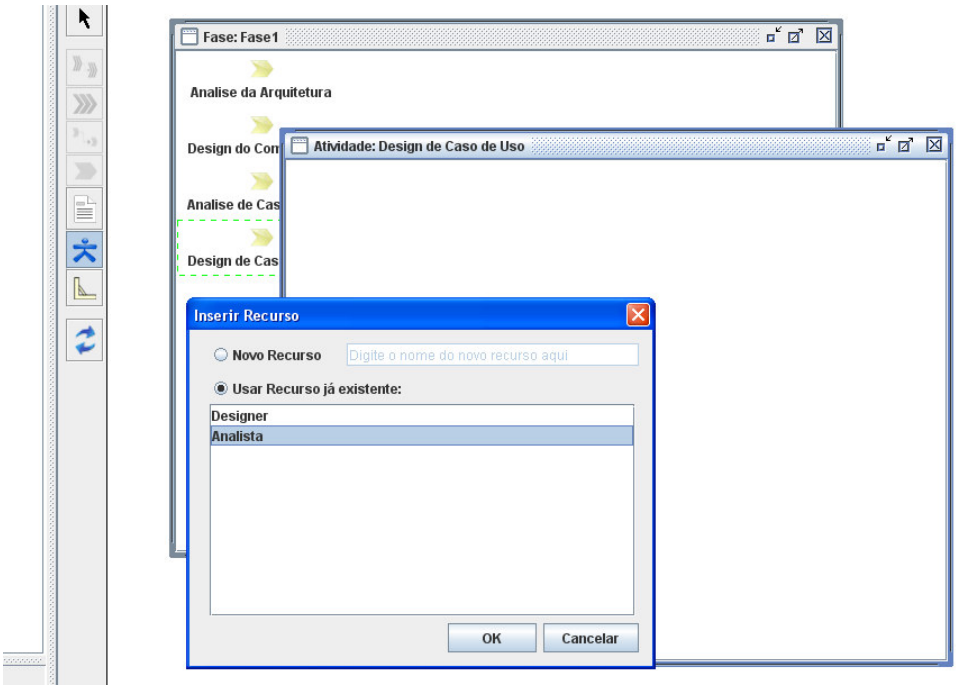


**Figura 5-15 Inserindo um recurso já existente**

## Abrindo o diagrama de atividades

O Promodeller possibilita, também, visualizar o processo através de diagrama de atividades, clicando Ctrl + t ou no menu exibir → diagrama de atividades. O diagrama de atividades será exibido no mesmo local do diagrama de pacotes da macro-atividade escolhida. A figura 5-16 mostra um trecho de um diagrama de atividades.


**Figura 5-16 Abrindo Diagrama de Atividades**

### 5.3.2 Visualizar a ontologia

Indo no menu Exibir → Ontologia, é possível, também, visualizar a ontologia a modelagem que está aberta no momento. Assim, o usuário pode ver na própria ferramenta o passo a passo da ontologia gerada, conforme a figura 5-17.

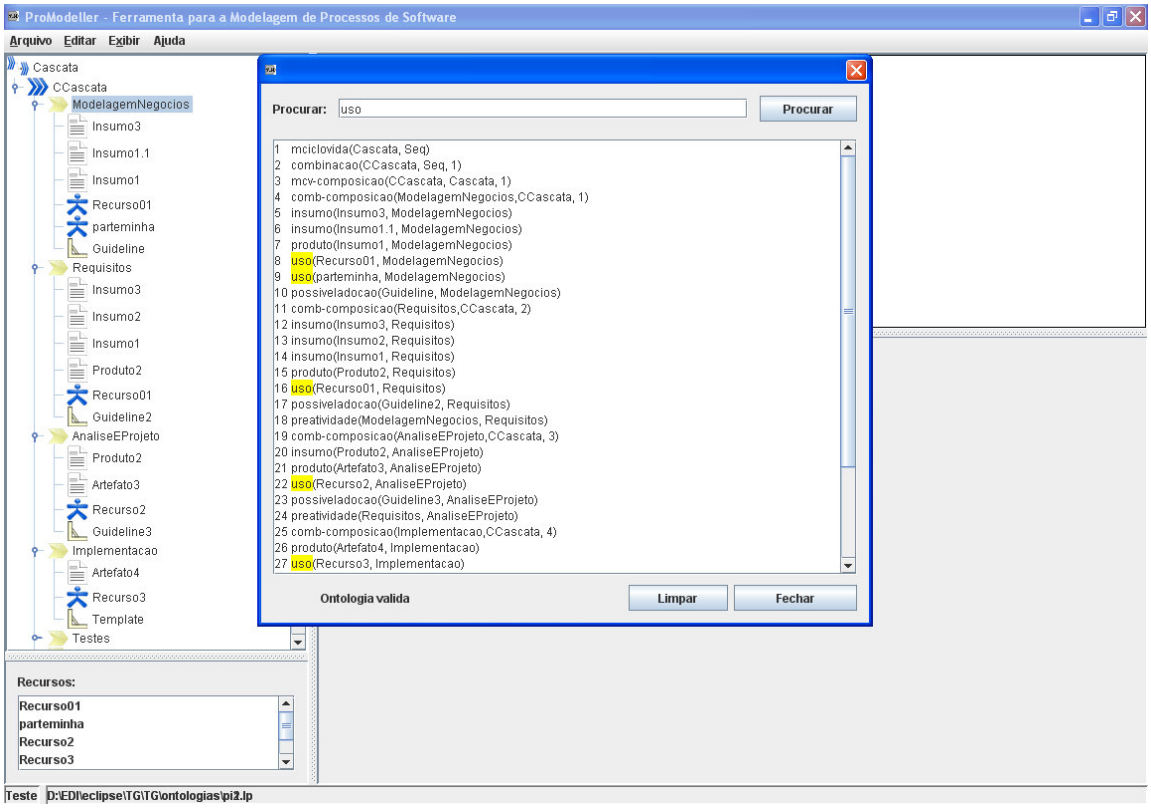


**Figura 5-17 Visualizar Ontologia**

### 5.3.3 Importar e Exportar XMI

A figura 5-18 mostra as funcionalidades de importar (1) um processo e exportar (2) um processo em XMI. Essas duas funcionalidades são as responsáveis pela a interoperabilidade do programa com demais aplicativos de processo, que também utilizem o formato XMI.

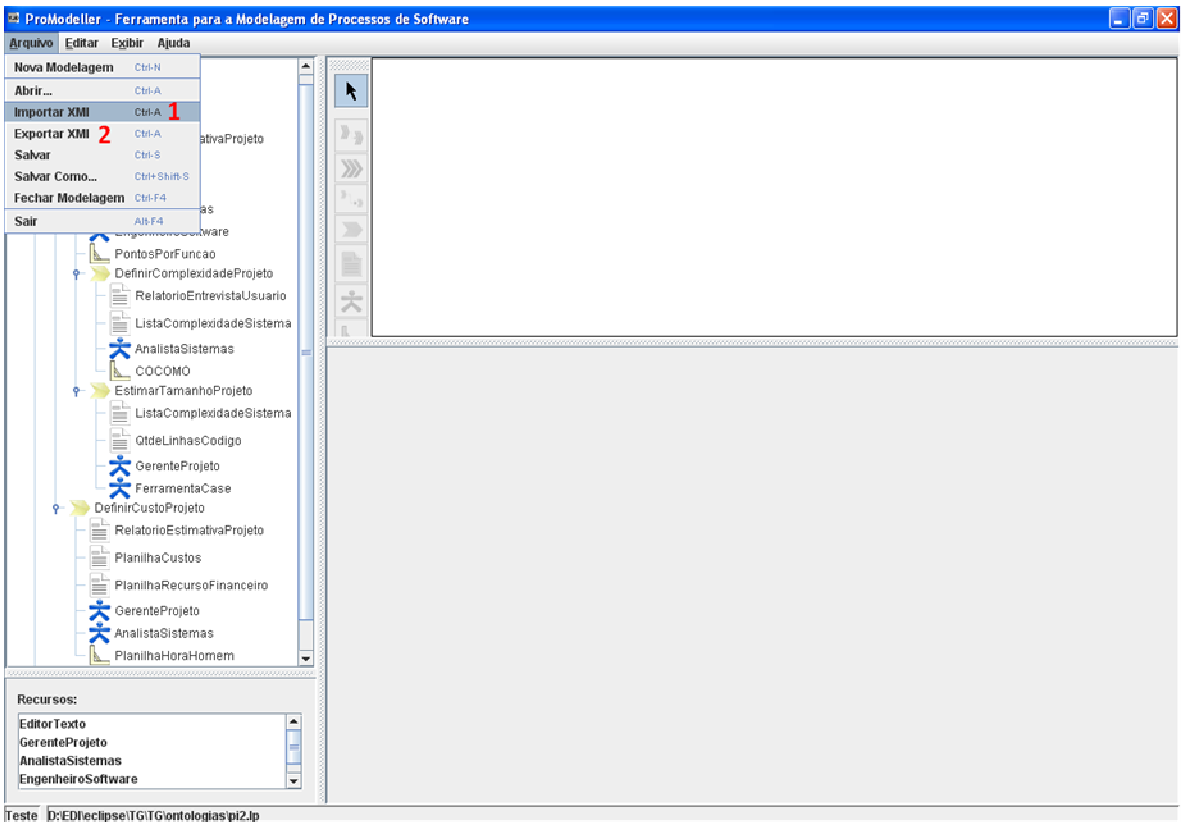


**Figura 5-18 Importar e exportar XMI**

## 5.4 Considerações Finais do Capítulo

Apresentamos, no presente capítulo, conceitos e um subconjunto das funcionalidades mais importantes do *Promodeller*. Nosso objetivo principal foi mostrar cenários de uso da ferramenta e como ela se comunica com outros aplicativos, formando um ambiente colaborativo mais interativo. Suas funcionalidades mostram que, apesar de o Promodeller ter sido desenvolvido academicamente, ele consegue dar o suporte necessário à modelagem de processo e ser útil para usuários com as mais diversas necessidades.

O próximo capítulo conterá as conclusões e experiências obtidas durante este trabalho, juntamente com os trabalhos que poderão ser realizados em projetos futuros.

# 6. Conclusão e Trabalhos Futuros

É grande a necessidade da existência um ambiente de desenvolvimento integrado que suporte, a partir do uso de um protocolo, a troca de informações entre as ferramentas envolvidas. No contexto de processo de software, observou-se a grande importância desse ambiente, onde seja possível visualizar e alterar, sem perdas, modelagens de processo na ferramenta que o usuário julgar mais adequada.

Neste trabalho apresentamos a ferramenta de modelagem de processo *Promodeller*. Mostramos, também, conceitos e componentes de SPEM, Ontologia de Processo e XMI. Estas são as principais notações utilizadas pelo aplicativo para gerar suas modelagens de processo.

Também focamos em mostrar a importância do uso do *Promodeller* junto com o *ImPProS*, pois a implementação da função importar/exportar XMI possibilita ao *ImPProS* interagir com demais ferramentas de modelagem. Com isso, o usuário pode utilizar os poderosos recursos do *ImPProS* para elaborar seu processo e fica livre para escolher uma ferramenta de modelagem para exibir seu processo graficamente.

Para dar continuidade a este trabalho, listamos um conjunto de propostas para trabalhos futuros, dentre as quais, pode-se citar:

- A inclusão das funções importar/exportar XMI diretamente no *ImPProS*;
- A possibilidade de inserir informações extras do processo nos arquivos XMI. Tais arquivos possuem uma seção chamada. XMI.extensions, que suporta a inserção de dados adicionais;
- Foi detectada a necessidade de criar algum mecanismo, no *Promodeller*, que dê suporte à troca de dados de forma remota. Como

sugestão, pode-se vislumbrar o uso de ferramentas já existentes no ambiente WEB.

# Referências Bibliográficas

[1] BERTOLLO et al, ODE – Um Ambiente de Desenvolvimento de Software Baseado em Ontologias, XVI Simpósio Brasileiro de Engenharia de Software. Disponível em: http://www.lbd.dcc.ufmg.br:8080/colecoes/sbes/2002/036.pdf Acessado em: 27/03/2008.

[2] MOSCARDINI, C, Mapeamento UML para XML. Disponível em: http://www.inf.pucpcaldas.br/eventos/seminarios/2003_1/xml/MINICurso.doc Acessado em: 28/03/2008.

[3] Object Management Group (OMG), Software & Systems Process Engineering Metamodel specification (SPEM) Version 2.0, Formal specification. Disponível em: http://www.omg.org/spec/SPEM/2.0/PDF Acessado em: 04/04/2008.

[4] Universidade do Estado da Bahia (UNEB), PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE. Disponível em: http://www.vqv.com.br/uneb/AnaliseSistemasII.pdf. Acessado em: 21/03/2008.

[5] Editora Regularize Segurança Eletrônica LTDA, PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE. Disponível em: http://processo-de-desenvolvimento-de-software.cartilha.regularize.com.br/list/18/cartilha/post/73/ Acessado em: 27/03/2008.

[6] VILLELA, Karina et al, Definição de Processos em Ambientes de Desenvolvimento de Software Orientados a Organização.
Disponível em: http://www.sbc.org.br/bibliotecadigital/download.php?paper=231
Acessado em: 16/04/2008.

[7] KOLCZ, K., Using SPEM/UML profile to specification of IS development processes.
Disponível em:
http://www.bth.se/fou/cuppsats.nsf/all/907192bc71e590e9c12571ff0049ef2e/$file/Using%20SPEM%20UML%20profile%20to%20specification%20of%20IS%20development%20processes.pdf
Acessado em: 22/05/2008

[8] SANT´ANNA, N.; GENVIGIR, E.C.; BORREGO FILHO, L.F. "Modelagem de Processos de Software Através do SPEM - Software Process
Engineering Metamodel - Conceitos e Aplicação. São Paulo, III WORCAP – Workshop dos Cursos da Computação Aplicada do INPE, 2003.

[9] Object Management Group (OMG), MetaObject Facility.
Disponível em: http://www.omg.org/mof/
Acessado em 12/03/2008.

[10] FALBO, R. A., Integração de Conhecimento em Ambientes de Desenvolvimento de Software, tese de doutorado, Universidade Federal do Rio de Janeiro (UFRJ), 1998.

[11] CORDEIRO, E., Introdução a Processos de Software.
Disponível em:
http://www.cordeiro.pro.br/aulas/engenharia/processoDeSoftware/Processo.pdf.
Acessado em: 29/04/2008.

[12] Object Management Group (OMG), MOF 2.0/XMI Mapping, Version 2.1.1. Disponível em: http://www.omg.org/docs/formal/07-12-02.pdf.
Acessado em: 06/04/2008.

[13] LOPES, L., MURTA, L., WERNER C., Controle de Modificações em Software no Desenvolvimento Baseado em Componentes.
Disponível em:
http://reuse.cos.ufrj.br/prometeus/publicacoes/wmswm2005-cr.pdf
Acessado em: 12/05/2008

[14] CHOI, K., Research Progress on Software Process Simulation Modeling.
Disponível em:
http://spic.kaist.ac.kr/~selab/html/Study/Lab%20Seminar/Backup_2002,%202003,%202004,%202005/Deriving%20Software%20Process%20Simulation%20Model%20from%20UML.pdf
Acessado em: 20/05/2008

[15] Schuppenies, R., Steinhauer, S., Software Process Engineering Metamodel.
Disponível em: http://www.sigmadelta.de/data/files/spem_paper.pdf
Acessado em: 01/06/2008

**Orientador**

_____

**Prof. Ph.D. Alexandre Marcos Lins de Vasconcelos**

**Aluno**

_____

**Edvaldo Lopes da Silva Filho**