Assist-Pro: Um Assistente Baseado em Conhecimento para Apoiar a Definição de Processos de Software

Ricardo de Almeida Falbo [1,2]
Crediné Silva de Menezes[2]
Ana Regina C. da Rocha[1]

[1] COPPE Sistemas e Computação - UFRJ
Caixa Postal 68511, CEP 21945-970,
Rio de Janeiro - RJ
email: (rfalbo, darocha)@cos.ufrj.br

[2] Departamento de Informática - UFES
Av. Fernando Ferrari s/nº,
CEP 29060-900, Vitória - ES
email: (falbo, credine)@inf.ufes.br

Abstract

An important requirement for software quality is to define a software development process. However, software process definition is a complex activity that requires intensive use of knowledge. So, it is worth to offer knowledge-based support to this activity. In this paper we present Assist-Pro, a knowledge-based assistant to aid software engineers to define software processes. We also discuss its development process using knowledge components available in a Knowledge Server.

1. Introdução

À medida que cresce a preocupação com a qualidade de software, cada vez mais as organizações de desenvolvimento de software têm procurado seguir orientações de modelos e padrões de qualidade de processo, tais como ISO 9000-3 [1], CMM [2] e SPICE [3]. De forma geral, um requisito importante para a qualidade consiste na definição de um processo de desenvolvimento. Contudo, a definição de processos de software envolve várias e complexas facetas e são poucos os profissionais qualificados para realizar esta tarefa. De fato, para profissionais menos experientes, esta é uma atividade que requer alguma forma de ajuda. Uma maneira de apoiar engenheiros de software na realização de suas atividades consiste em oferecer suporte baseado em conhecimento, através de assistentes inteligentes.

É grande o número de assistentes inteligentes descritos na literatura, cobrindo um amplo espectro de atividades do processo de desenvolvimento, tais como análise de requisitos, projeto de sistema e gerência de riscos, entre outros [4]. Entretanto, pouco se tem tratado a definição de processos de software. Assim, desenvolvemos *Assist-Pro,* um assistente baseado em conhecimento para apoiar a definição de processos de software dentro do contexto de um meta-ambiente de desenvolvimento de software, a Estação TABA [5,6]. Assist-Pro foi desenvolvido utilizando uma infra-estrutura de conhecimento sobre processos de software, o Servidor de Conhecimento de Processo [4,7], existente na Estação TABA.

Este artigo apresenta Assist-Pro e seu processo de construção, utilizando o Servidor de Conhecimento de Processo. A seção 2 apresenta brevemente a Estação TABA e o Servidor de Conhecimento de Processo. A seção 3 apresenta os requisitos de Assist-Pro, enquanto a seção 4 discute o seu processo de construção. Na seção 5, são apresentadas as funcionalidades implementadas de Assist-Pro e seu funcionamento. A seção 6 discute trabalhos correlatos e, finalmente, na seção 7, são apresentadas as conclusões deste trabalho.

2. A Estação TABA e O Servidor de Conhecimento de Processo

A Estação TABA [5,6] é um meta-ambiente de desenvolvimento de software capaz de gerar, por meio de instanciação, ADSs adequados às particularidades de processos de desenvolvimento, domínios de aplicação e projetos específicos. Este meta-ADS possui ferramentas para a definição, execução, alteração e acompanhamento de processos [8]. Entretanto, tais ferramentas não oferecem suporte baseado em conhecimento para o engenheiro de software nestas tarefas que, sem dúvida, são bastante complexas e requerem uma gama variada de conhecimento.

Com o objetivo de apoiar a construção de ferramentas baseadas em conhecimento de processo de software na Estação TABA, foi desenvolvido o Servidor de Conhecimento de Processo [4,7]. Neste contexto, entende-se um Servidor de Conhecimento como uma infra-estrutura capaz de apoiar a construção de sistemas baseados em conhecimento em um domínio de interesse [4], provendo componentes de conhecimento sobre este domínio para serem reutilizados.

A arquitetura básica do Servidor de Conhecimento de Processo (SCP) apoia-se em uma concepção introduzida pela comunidade de Inteligência Artificial para aumentar a reusabilidade do conhecimento e facilitar a manutenção dos sistemas resultantes: a separação entre os conhecimentos de domínio e de tarefa. Esta separação evidencia que há dois aspectos distintos, apesar de relacionados, no projeto de um sistema baseado em conhecimento e que estes podem ser, pelo menos até certo ponto, investigados e tratados separadamente. Desta forma, podem existir modelos estruturados separados para representar o domínio e a tarefa e, portanto, o SCP busca tornar disponíveis componentes de conhecimento destes dois tipos [4], como mostra a figura 1.

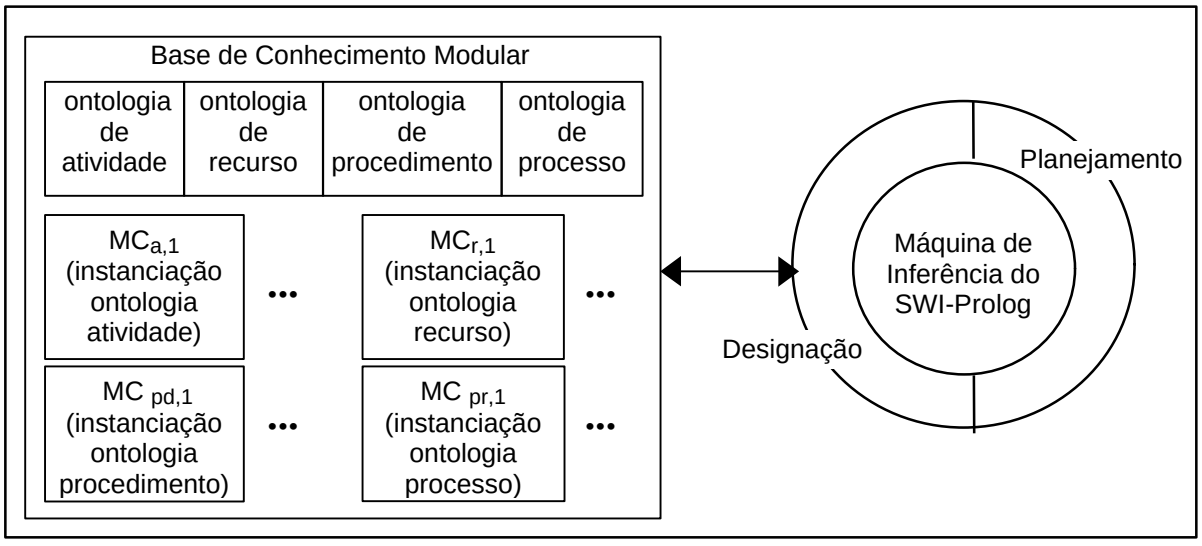

Figura 1 – A Arquitetura do Servidor de Conhecimento de Processo

Os modelos de domínio e respectivos componentes, ditos *módulos de conhecimento*, foram desenvolvidos tendo por base ontologias [4]. Uma ontologia é uma especificação de uma conceituação [9], isto é, uma descrição de conceitos e relações que podem existir para um agente ou uma comunidade de agentes, procurando restringir a estrutura de uma abstração da realidade [10]. Basicamente, uma ontologia consiste de conceitos e relações, e suas definições e restrições expressas na forma de axiomas [11]. Cada ontologia origina um módulo de

conhecimento. Além disso, cada uma das instanciações de uma ontologia específica também dá origem a um componente. Uma instanciação de uma ontologia é uma descrição de fatos do domínio usando os conceitos e as relações descritos na ontologia. Estes módulos de conhecimento foram escritos em Prolog, tendo em vista que a máquina de inferência do SWI-Prolog [12] está integrada à Estação TABA [13]. A base de conhecimento de uma aplicação que se comprometa com uma ou várias ontologias deverá ser a conjunção dos módulos de conhecimento correspondentes, mais o conhecimento específico para esta aplicação.

Os modelos de tarefa e respectivos componentes de conhecimento, os *templates de resolvedores de problemas*, foram construídos utilizando a modelagem de tarefas de CommonKADS [14]. Alguns dos modelos de tarefa da biblioteca de CommonKADS, tipicamente aqueles projetados para os tipos de problemas que ocorrem com maior freqüência no universo de discurso de processos de software, foram selecionados, adaptados e implementados na forma de templates de resolvedores de problemas, formando uma camada sobre a máquina de inferência geral do SWI-Prolog.

Os módulos de conhecimento e os templates de resolvedores de tipos de problema são os componentes reutilizáveis que o Servidor de Conhecimento oferece para auxiliar o processo de construção de SBCs no universo de discurso de processos de software. Além disso, guardam estreita relação entre si: os papéis de conhecimento considerados em um template de resolvedor de problema devem ser preenchidos com o conhecimento de domínio descrito nos módulos de conhecimento.

Com estes componentes disponíveis, é possível mudar o enfoque na construção de assistentes inteligentes. Ao invés de iniciar o processo de aquisição de conhecimento a partir do nada, uma aquisição de conhecimento preliminar pode ser feita, sem consumir muito tempo dos especialistas. Navegando as ontologias do domínio, o engenheiro de software ganha um entendimento do problema e pode montar uma base de conhecimento inicial. Os modelos de tarefa, por sua vez, podem ser usados para estabelecer uma estratégia inicial para a resolução do problema. Adaptando um modelo de tarefa para o problema em mãos e fazendo um mapeamento de seus papéis de conhecimento com o conhecimento de domínio correspondente, tem-se um protótipo inicial do sistema. Além disso, ao propor esta solução preliminar, o engenheiro de software pode identificar os tipos de conhecimento que estão faltando e, assim, tem um mecanismo eficiente para guiar a segunda etapa da aquisição de conhecimento, agora junto aos especialistas.

3. Requisitos de Assist-Pro

De maneira geral, a definição de um processo de software envolve as seguintes tarefas:

- Definição de um ciclo de vida para o processo;
- Detalhamento das fases do ciclo de vida em atividades;
- Definição de como as atividades devem ser realizadas;
- Definição dos artefatos requeridos e produzidos por uma atividade;
- Definição dos recursos para as atividades;

Diferentes aplicações possuem diferentes perfis e estas diferenças têm influência sobre seus processos de desenvolvimento. Para que as tarefas anteriormente relacionadas possam ser realizadas, é imprescindível que algumas características gerais relacionadas ao processo de software sejam observadas. Desta forma, uma etapa preliminar para capturar estas

características foi adicionada. A seguir, os passos anteriormente relacionados são discutidos com mais detalhes.

Determinação das Características do Projeto

Nesta etapa, basicamente, deseja-se obter algumas informações sobre o projeto, que sejam relevantes para a definição de um processo, entre elas:

- *Complexidade do problema a ser tratado*: notadamente a complexidade do software a ser desenvolvido tem um grande impacto no ciclo de vida a ser adotado. Além disso, deve ser observado se o problema a ser tratado é bem definido, isto é, pode ser totalmente especificado no início do desenvolvimento.
- *Características do desenvolvimento*: deve-se definir o paradigma a ser adotado no esforço de desenvolvimento e observar se o software precisa ser colocado em uso rapidamente.
- *Características da equipe*: a formação, atualização e experiência da equipe são fatores importantes a serem considerados na definição do processo.

Definição do Ciclo de Vida para o Projeto

A definição do ciclo de vida é um dos passos mais importantes na definição de um processo de software. Para auxiliar esta etapa, existem vários modelos de ciclo de vida descritos na literatura. Assim, o ciclo de vida é definido com base em um modelo de ciclo de vida adequado à situação (dada pelas características discutidas anteriormente) e adaptado a ela.

Detalhamento das Fases do Ciclo de Vida em Atividades

Uma vez definido o ciclo de vida do projeto, tem-se apenas as macro-atividades que compõem o processo. É necessário, portanto, refinar estas macro-atividades em sub-atividades para se obter o conjunto total de atividades do processo.

Definição da Realização das Atividades

Para cada atividade do processo, deve-se definir como esta será realizada, isto é, que procedimentos serão adotados. É importante notar que, uma vez que métodos impõem uma particular decomposição de uma atividade em sub-atividades, a escolha dos métodos deve anteceder o completo detalhamento das atividades do processo. Técnicas, roteiros e normas, por sua vez, são aplicáveis apenas a atividades elementares e, portanto, sua escolha pressupõe a conclusão do detalhamento das atividades.

Definição dos Artefatos Requeridos e Produzidos pelas Atividades

Para cada atividade do processo é necessário definir que artefatos são requeridos pela atividade e que artefatos são produzidos por ela.

Definição dos Recursos Necessários para a Realização das Atividades

Finalmente, uma vez definidas todas as atividades que compõem o processo, deve-se determinar que recursos serão necessários para a realização das mesmas, incluindo recursos humanos, de hardware e de software.

4. O Processo de Construção de Assist-Pro

Uma vez que o Servidor de Conhecimento de Processo oferece um conjunto de componentes de conhecimento para serem reutilizados, foi possível utilizar uma abordagem de Engenharia de Conhecimento baseada em Reutilização [4,7], envolvendo os seguintes

passos: (i) Levantamento de Requisitos, (ii) Seleção de Ontologias, (iii) Seleção de Modelos de Tarefa, (iv) Compatibilização de Ontologias e Modelos de Tarefa, (v) Aquisição e Modelagem do Conhecimento Específico da Aplicação, (vi) Projeto, (vii) Implementação e Testes.

Inicialmente, procuramos determinar os passos envolvidos na definição de um processo de software e de que forma um assistente baseado em conhecimento poderia apoiar esta tarefa. A seguir, selecionamos, dentre os componentes de conhecimento do SCP, as ontologias e os modelos de tarefa apropriados ao desenvolvimento de Assist-Pro e especificamos de que forma eles deveriam ser combinados. Uma vez delineado o esqueleto da aplicação, foi necessário instanciar as ontologias com o conhecimento necessário para a apoiar a definição de processos. Além disso, alguns papéis de conhecimento, específicos da aplicação, tiveram de ser elicitados, modelados e instanciados. Contudo, as ontologias guiaram fortemente a realização desta atividade.

Na etapa de projeto, foram considerados aspectos de projeto da interface com o usuário, persistência dos objetos, gerência das tarefas do sistema, bem como de adaptação dos modelos do domínio do problema para a plataforma de implementação, o ambiente ISE-Eiffel [15]. Finalmente, o sistema foi implementado e testado, procurando-se observar se a assistência oferecida estava condizente com o comportamento dos especialistas consultados.

A seguir, os principais passos da construção de Assist-Pro são descritos mais detalhadamente.

4.1 – Levantamento de Requisitos

Com base em entrevistas com especialistas do domínio e em estudo da bibliografia especializada no assunto, incluindo dentre outros [1], [2], [3], [16] e [17], chegamos ao seguinte conjunto de funcionalidades a serem providas por Assist-Pro:

- *Entrada de dados sobre características do projeto para o qual o processo está sendo definido*. Uma vez que características do processo são fortemente dependentes das especificidades do projeto, é necessário capturar as características determinantes na definição de processo, enumeradas na seção 3.
- *Apoio à Seleção de um Modelo de Ciclo de Vida para o Processo*. Com base nas características do projeto, o assistente deve ser capaz de sugerir modelos de ciclo de vida adequados, cabendo ao engenheiro de software, usuário da ferramenta, a decisão final do modelo a ser adotado.
- *Adaptação do Modelo de Ciclo de Vida Selecionado*. Um modelo de ciclo de vida apresenta um conjunto de macro-atividades e estabelece uma forma de estruturar estas atividades. Contudo, dependendo do projeto, pode haver necessidade de se adaptar o modelo selecionado para comportar características específicas. Desta forma, Assist-Pro deve permitir que o engenheiro de software usuário altere a estrutura básica do modelo de ciclo de vida, adicionando novas atividades ou descartando algumas das propostas.
- *Apoio à Seleção de Métodos para a Realização de Atividades.* Métodos impõem um particular refinamento para macro-atividades do processo. Ao se escolher o Método de Coad & Yourdon [17], por exemplo, para a realização da atividade de Análise e Especificação de Requisitos, a seguinte decomposição em sub-atividades deve ser observada: Localização de Classes & Objetos, Identificação de Estruturas,

Identificação de Assuntos, Definição de Atributos e Definição de Serviços. Assim, antes de se proceder um detalhamento das atividades, é necessário selecionar os métodos a serem utilizados na sua realização. De forma análoga à seleção de um modelo de ciclo de vida, Assist-Pro deve oferecer apoio ao processo de escolha dos métodos, tendo por base a atividade e o paradigma adotado no desenvolvimento.

- *Apoio ao Detalhamento das Atividades.* O nível de macro-atividade é muito geral para a definição de um processo. É necessário decompor estas atividades básicas do processo em sub-atividades, até um nível no qual seja possível gerenciar, alocar recursos e, de fato, executar tais atividades. Assist-Pro deve ser capaz de apoiar esta tarefa, oferecendo sugestões de detalhamento. Vale ressaltar que a última palavra deve ser dada pelo engenheiro de software. Isto é, Assist-Pro apenas sugere. Cabe ao engenheiro de software acatar ou não a sugestão.
- *Apoio à Designação de Técnicas, Artefatos, Roteiros e Recursos às Atividades do Processo.* Uma vez que se tenha definido o conjunto final de atividades, resta definir, para cada atividade, que técnicas serão utilizadas na sua realização, que artefatos serão requeridos e produzidos, que roteiros podem ser utilizados na elaboração dos artefatos e que recursos são necessários para realizar a atividade.
- *Estabelecimento das Relações de Precedência entre Atividades.* Com a designação de artefatos requeridos e produzidos a atividades, é possível estabelecer a precedência entre as atividades. Isto pode ser feito de forma parcialmente automática por Assist-Pro, pelo menos para as atividades que requerem artefatos produzidos por outras atividades.
- *Definição do Processo de Software no Componente Controle de Processo da Estação TABA.* Uma vez concluída a definição do processo, é necessário criar sua estrutura no Componente de *Controle de Processo* [8] da Estação TABA, para permitir a instanciação de um ambiente de desenvolvimento adequado ao processo definido.

4.2 – Seleção de Ontologias

Nesta etapa, procurou-se identificar os componentes de conhecimento de domínio, disponíveis no Servidor de Conhecimento de Processo, passíveis de serem reutilizados no contexto de Assist-Pro.

Dentre estes componentes, foram selecionadas as ontologias de atividade, recurso, procedimento e processo. A figura 2 mostra a ontologia de atividade, que será utilizada neste artigo para ilustrar o processo de construção de Assist-Pro. Esta figura está expressa em LINGO [4, 11] – LINguagem Gráfica para expressar Ontologias, cuja notação está mostrada na própria figura 2. Além da representação gráfica, alguns axiomas e os respectivos módulos de conhecimento derivados deles são mostrados, respectivamente, nas figuras 3 e 4. A especificação completa desta e das demais ontologias pode ser encontrada em [4].

4.3 – Seleção e Adaptação de Modelos de Tarefa

Da mesma forma como foi feito para a perspectiva de domínio, procurou-se identificar componentes de conhecimento de tarefa do Servidor de Conhecimento de Processo, passíveis de reuso na construção de Assist-Pro.

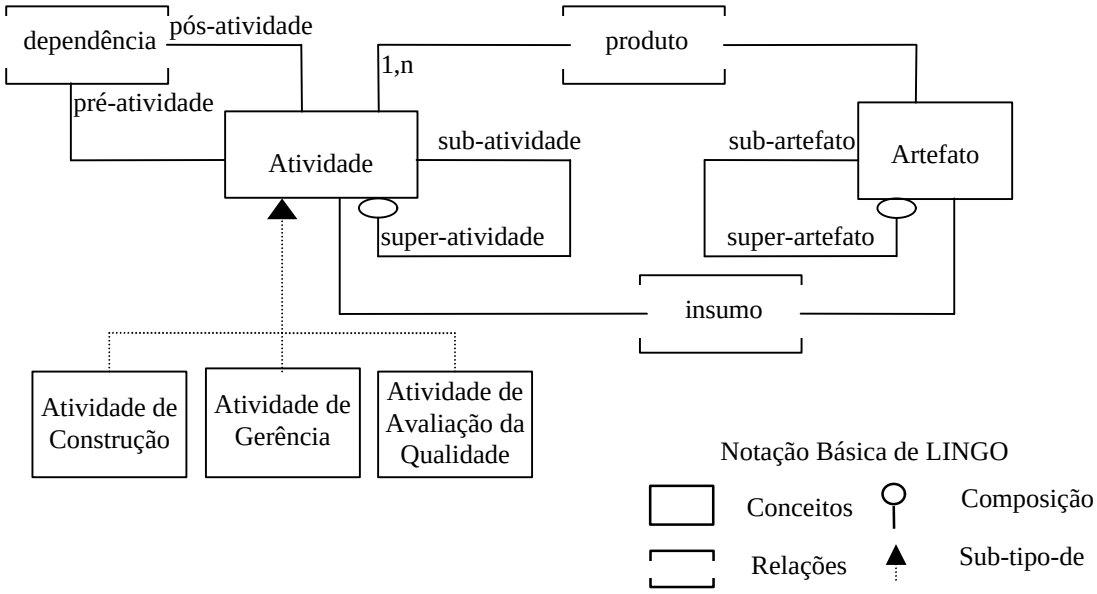

Figura 2 – A Ontologia de Atividade

(" a_1 , a_2 , a_3) (subatividade(a_1 , a_2) Ù subatividade(a_2 , a_3) ® subatividade(a_1 , a_3))
(" a_1 , a_2) (subatividade(a_1 , a_2) « superatividade(a_2 , a_1))
(" a_1 ,a_2) (preatividade(a_1 ,a_2) « ($ s) (insumo(s,a_2) Ù produto(s,a_1))
(" a_1 , a_2 , a_3)(preatividade(a_1 , a_2) Ù preatividade(a_2 , a_3) ® preatividade(a_1 , a_3))
(" a_1 , a_2) (preatividade(a_1 , a_2) « posatividade(a_2 , a_1))

Figura 3 – Alguns Axiomas da Ontologia de Atividade.

```
subatividade(X,Y) :- subatividade(X,Z), subatividade(Z,Y).
subatividade(X,Y) :- superatividade(Y,X).
superatividade(X,Y) :- subatividade(Y,X).
preatividade(X,Y) :- insumo(S,Y), produto(S,X).
preatividade(X,Y) :- preatividade(X,Z), preatividade(Z,Y).
preatividade(X,Y) :- posatividade(Y,X).
posatividade(X,Y) :- preatividade(Y,X).
```

Figura 4 –Módulo de Conhecimento derivado a partir da Ontologia de Atividade.

A definição de um processo de software é uma tarefa que mescla problemas dos tipos planejamento e designação. As funcionalidades de seleção e adaptação de um modelo de ciclo de vida e de refinamento das macro-atividades em sub-atividades representam problemas de planejamento. As demais funcionalidades representam problemas de designação, como mostra a figura 5. Assim, foram selecionados modelos de tarefa de planejamento, mostrado na figura 6, e designação.

Os modelos de tarefa apresentados neste trabalho estão expressos na notação de CommonKADS [14] e correspondem a adaptações feitas para o contexto do desenvolvimento de software, por ocasião da construção do Servidor de Conhecimento de Processo [4]. Estes modelos são descritos em termos de estruturas de funções, sendo que uma função corresponde a uma descrição de uma (sub-)tarefa através de sua meta e de uma relação de entrada-saída. A entrada e a saída de uma função são representadas por papéis que o conhecimento de domínio

pode desempenhar na resolução de problemas, os papéis de conhecimento. Além disso, funções podem referenciar estruturas de conhecimento que são usadas, mas cujo conteúdo não se altera durante a resolução do problema - os *papéis estáticos*. As únicas referências que uma função faz ao conhecimento de domínio são através dos papéis de conhecimento de entrada, de saída e estáticos.

CommonKADS utiliza uma linguagem gráfica para descrever modelos de tarefa, na qual funções são representadas por elipses e papéis de conhecimento por retângulos. Ligações com setas indicam a direção do fluxo de conhecimento, sendo que setas duplas são usadas para indicar papéis estáticos. Na figura 5, os papéis de conhecimento estáticos foram omitidos para tornar o modelo mais claro.

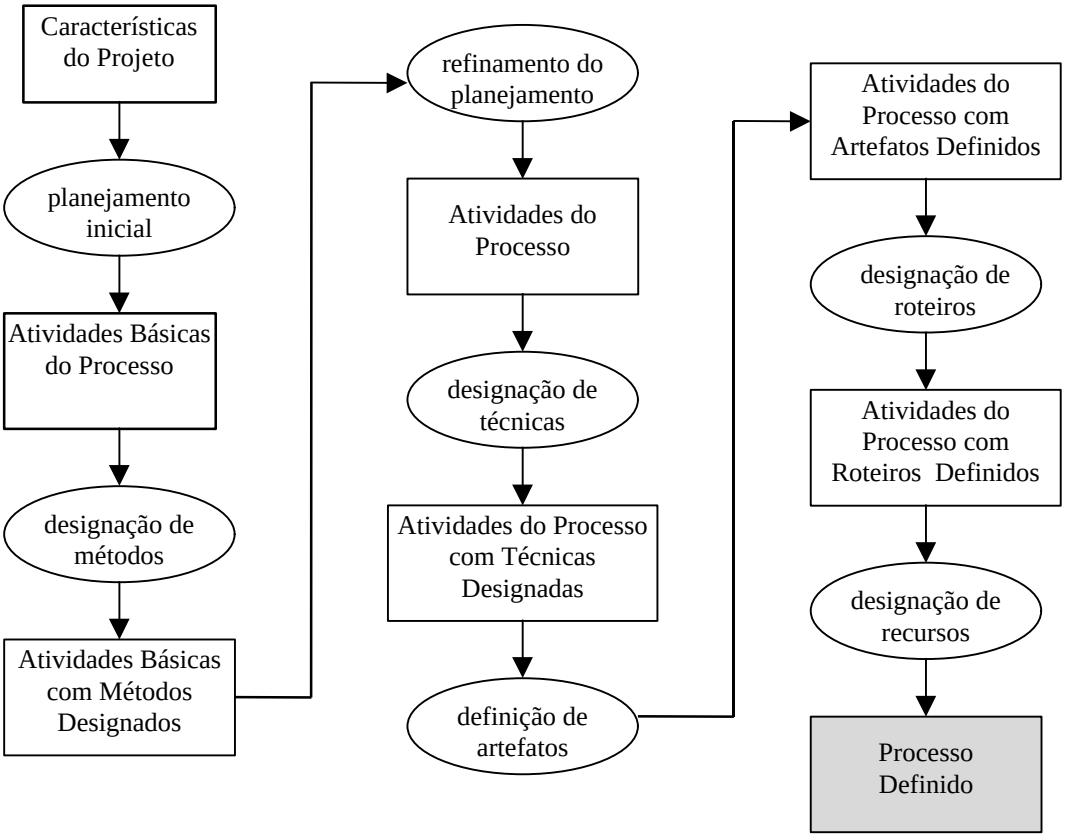

Figura 5 - Modelo de Tarefa Global para *Assist-Pro*.

Os modelos do Servidor de Conhecimento são gerais e, portanto, precisam ser adaptados para representar a definição de processos de software. O modelo adaptado de planejamento é mostrado na figura 7. Uma comparação entre os modelos das figuras 6 e 7 ilustra o processo de adaptação de um modelo de tarefa. No contexto da definição de processos de software, a função *Selecionar Modelo de Plano* corresponde à seleção do modelo de ciclo de vida para o processo. De fato, todos os elementos do modelo genérico (figura 6) referentes a *Modelo de Plano* (*Conhecimento de Avaliação de Plano, Conhecimento sobre Modelos de Plano* e *Modelo de Plano*) dizem respeito, no contexto da definição de processos de software (figura 7), a *Modelos de Ciclo de Vida* e, portanto, tiveram de ser adaptados (respectivamente, para *Conhecimento sobre a Aplicabilidade de Modelos de Ciclo de Vida*, *Conhecimento de Modelos de Ciclo de Vida* e *Modelo de Ciclo de Vida*).

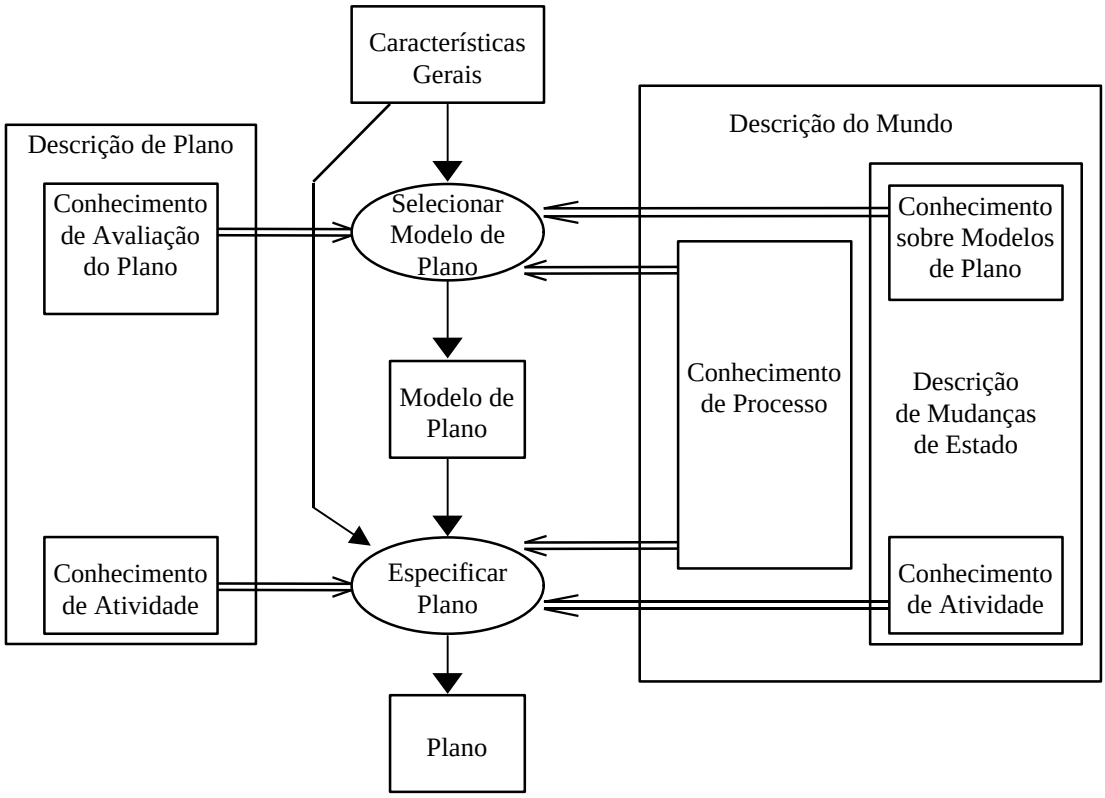

Figura 6 – O Modelo de Tarefa de Planejamento do Servidor de Conhecimento de Processo.

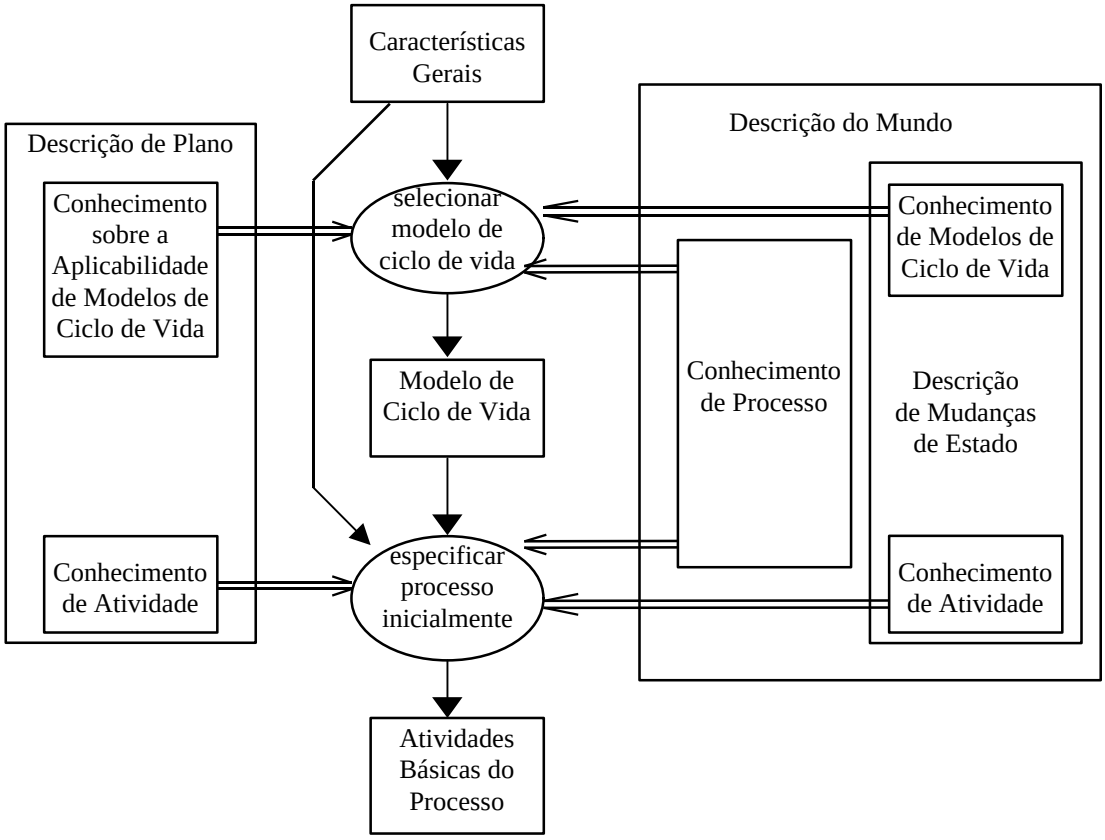

Figura 7 – O Modelo de Tarefa de Planejamento Adaptado à Definição de Processos.

4.4 – Compatibilização das Ontologias com os Modelos de Tarefa

Selecionados os componentes de domínio e de tarefa, é necessário estabelecer as devidas correlações entre eles. Como anteriormente citado, o elo de ligação são os papéis de conhecimento, ou seja, os papéis que o conhecimento de domínio desempenha no processo de resolução de problemas. Assim, nesta etapa, procurou-se estabelecer como os papéis de conhecimento dos modelos de tarefa poderiam ser preenchidos com o conhecimento descrito nas ontologias, e quais destes papéis não encontravam correspondentes componentes de domínio e, portanto, deveriam ser objeto de uma etapa de aquisição de conhecimento.

Analisando os modelos de tarefa de Assist-Pro e os módulos de conhecimento derivados das ontologias, foi possível estabelecer correlações para todos os papéis de conhecimento, exceto para o papel *Conhecimento sobre Aplicabilidade de Modelos de Ciclo de Vida* do modelo de tarefa de planejamento (figura 7). Assim, este conhecimento teve de ser elicitado e modelado.

4.5 – Aquisição e Modelagem do Conhecimento Específico de Assist-Pro

Modelos de ciclo de vida são selecionados em função de fatores como: tecnologia de desenvolvimento e paradigma a serem aplicados no desenvolvimento do software, características do problema a ser resolvido, características do software a ser desenvolvido e características da equipe de desenvolvimento.

Parte deste conhecimento está descrito na ontologia de processo. Entretanto, aspectos como características do problema e da equipe de desenvolvimento não foram capturados. De fato, muito deste conhecimento é inerentemente heurístico e, portanto, não deve ser descrito em uma ontologia. Assim, uma nova etapa de aquisição de conhecimento teve de ser realizada. Esta etapa foi ainda, fortemente guiada pelas ontologias. O conhecimento foi elicitado para alguns modelos de ciclo de vida, sendo que o vocabulário básico definido pela ontologia de processo foi usado para descrever este conhecimento. Obviamente, foi necessário estender este vocabulário para considerar termos específicos, não previstos na ontologia. A figura 8 mostra parte da base de conhecimento correspondente. Além disso, novas instanciações das ontologias foram realizadas para incorporar o conhecimento sobre atividades, procedimentos, recursos, artefatos e modelos de ciclo de vida ainda não considerados.

```
muitoindicado(P, cascata):- tamanho(P, pequeno), complexidade(P, baixa).
muitoindicado(P, incremental):- tamanho(P, medio), complexidade(P,baixa),
                          necessidadeuso(P, sim).
muitoindicado(P, incremental):- tamanho(P,grande), complexidade(P,baixa),
                          necessidadeuso(P, sim), modularidade(P, alta).
muitoindicado(P, incremental):- tamanho(P,grande), complexidade(P,baixa),
                          necessidadeuso(P, nao), modularidade(P,media).
muitoindicado(P, incremental):- tamanho(P,grande), complexidade(P,baixa),
                          necessidadeuso(P, nao), modularidade(P, alta).
muitoindicado(P, rad) :- tamanho(P, pequeno), complexidade(P, baixa),
                          necessidadeuso(P, sim).
muitoindicado(P, rad) :- tamanho(P, medio), complexidade(P, baixa),
                          necessidadeuso(P, sim), modularidade(P, alta).
```

Figura 8 –Parte da Base de Conhecimento Específica de Assist-Pro.

5. Assist-Pro: Funcionalidades e Funcionamento

Assist-Pro está disponível como uma opção do menu *Utilitários* do meta-ambiente da Estação TABA, como mostra a figura 9. Uma vez selecionada esta opção, uma sessão de Assist-Pro é iniciada.

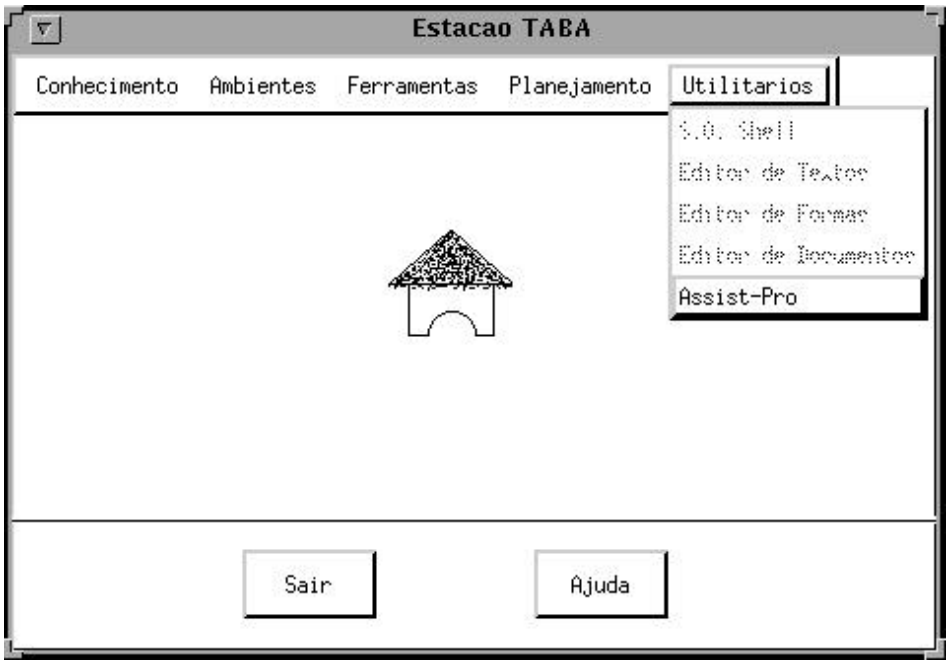

Figura 9 – Menu Utilitários da Estação TABA e a Opção Assist-Pro.

Esta primeira versão de Assist-Pro tem abrangência limitada à definição de processos de software para projetos de Sistemas de Informação e desconsidera a possível existência de elementos pré-definidos para o processo, tais como a obrigatoriedade de se empregar um método específico ou uma linguagem de programação particular. Além disso, esta é uma ferramenta de apoio à definição de processos no contexto do ambiente TABA e, portanto, o produto de sua utilização é a instanciação de objetos no modelo TABA de controle de processos, mostrado na figura 10. A partir da instanciação destes objetos, um ADS adequado pode ser gerado, com o objetivo de executar e acompanhar o processo definido, usando as ferramentas existentes na Estação TABA para execução, alteração e acompanhamento de processos [8]. Nenhuma saída impressa ou documento de projeto é gerado.

Selecionado um projeto, o primeiro passo consiste em fornecer suas características gerais, como mostra a figura 11. Uma vez fornecidas as características do projeto, a seqüência de funções do modelo de tarefa global de Assist-Pro, mostrado na figura 5, pode ser iniciada.

O planejamento inicial corresponde à seleção de um modelo de ciclo de vida e utiliza a janela mostrada na figura 12. Nesta janela, são apresentados para seleção, apenas os modelos de ciclo de vida adequados às características do projeto. Este filtro é feito através da base de conhecimento de Assist-Pro que indica a adequabilidade dos modelos de ciclo de vida.

Nesta etapa, o engenheiro de software pode, ainda, alterar a estrutura básica do modelo de ciclo de vida selecionado, inserindo novas atividades ou removendo atividades propostas pelo modelo escolhido.

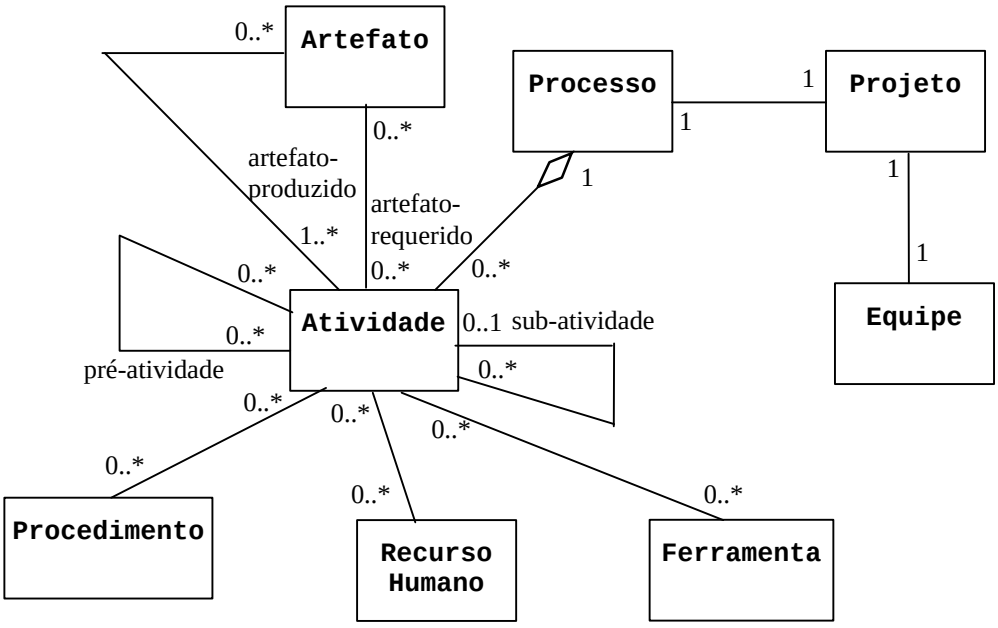

Figura 10 – O Modelo de Classes de Controle de Processos da Estação TABA.

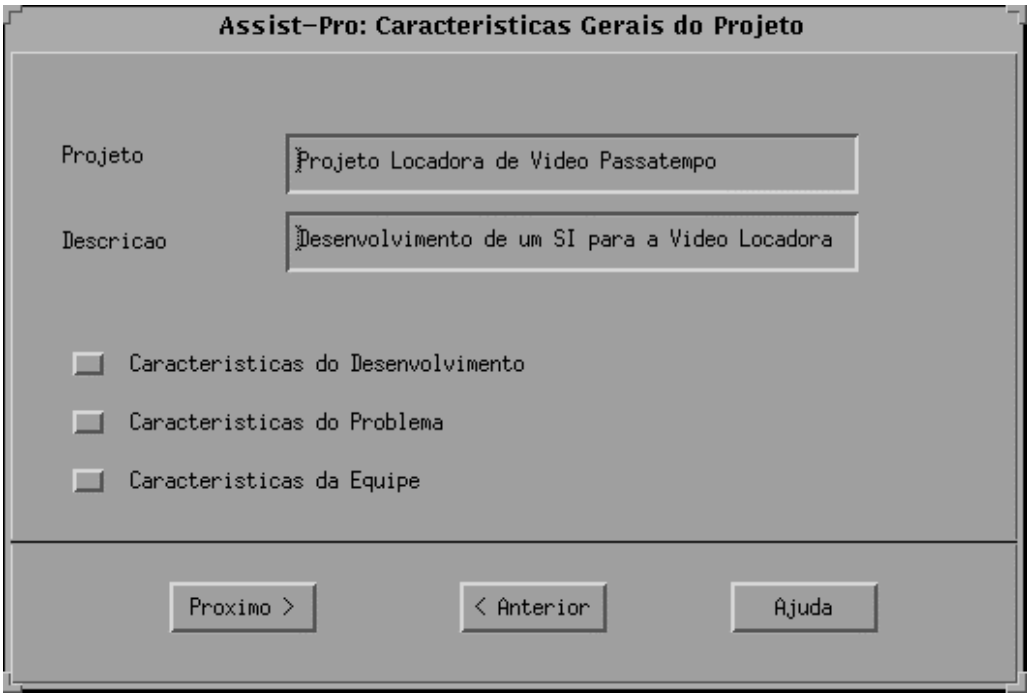

Figura 11 – Janela de Características Gerais do Projeto.

Uma vez selecionado e adaptado o modelo de ciclo de vida, o assistente solicita que sejam designados métodos para algumas de suas atividades. Esta tarefa é também guiada por uma base de conhecimento, esta construída utilizando o conhecimento da ontologia de processo e suas instanciações. A seguir, procede-se o refinamento do planejamento, onde as atividades básicas do processo são refinadas em sub-atividades, usando a janela da figura 13. Neste passo, caso ainda não tenha sido feito um detalhamento para a atividade, a lista de sub-atividades selecionadas é preenchida com a indicação provida pela base de conhecimento correspondente às instanciações das ontologias.

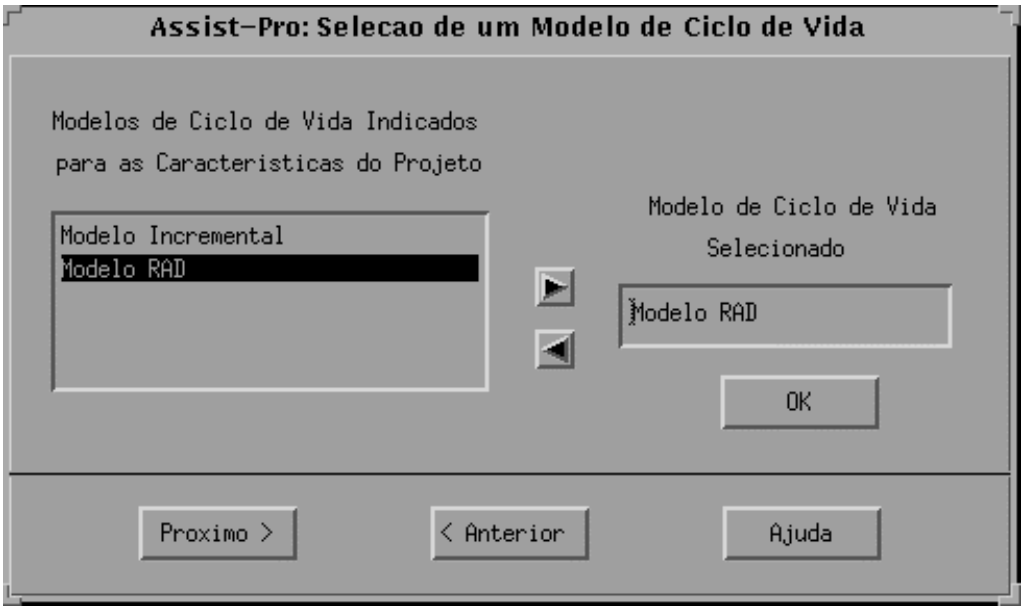

Figura 12 – Janela de Seleção de um Modelo de Ciclo de Vida para o Processo.

Figura 13 – Janela de Detalhamento das Atividades.

Na seqüência, são designados técnicas, artefatos, roteiros e recursos às atividades. Em todos estes passos são utilizadas janelas similares à janela de designação de técnicas a atividades, mostrada na figura 14.

O último passo da definição de processos em Assist-Pro corresponde à definição de precedência entre atividades. Este passo é feito de forma parcialmente automática, com base nas informações dos artefatos requeridos e produzidos pelas atividades. Estas informações dão origem a bases de conhecimento temporárias criadas para realizar este passo, como a mostrada na figura 15. Utilizando estas bases e o módulo de conhecimento da figura 4, Assist-Pro infere algumas precedências. Quando uma atividade depende simplesmente da execução de outra e não de um artefato produzido, esta precedência tem de ser informada pelo projetista de processo. Este passo completa o modelo de classes descrito na figura 10.

Figura 14 – Janela de Designação de Técnicas a Atividades.

atividade (planejamento, atividadeGerência).
atividade (definiçãoEscopoSoftware, atividadeGerência).
atividade (estimativaRecursos, atividadeGerência).
atividade (estimativaEsforço, atividadeGerência).
atividade (determinaçãoCronograma, atividadeGerência).
insumo (declaraçãoEscopo, estimativaRecursos).
insumo (declaraçãoEscopo, estimativaEsforço).
insumo (esfoçoEstimado, determinaçãoCronograma).
produto (planoProjeto, planejamento).
produto (declaraçãoEscopo, definiçãoEscopoSoftware).
produto (recursoEstimado, estimativaRecursos).
produto (esfoçoEstimado, estimativaEsforço).
produto (cronograma, determinaçãoCronograma).
subatividade (definiçãoEscopoSoftware, planejamento).
subatividade (recursoEstimado, planejamento).
subatividade (esfoçoEstimado, planejamento).
subatividade (determinaçãoCronograma, planejamento).

Precedências Inferidas

- definiçãoEscopoSoftware precede estimativaRecursos.
- definiçãoEscopoSoftware precede estimativaEsforço.
- estimativaEsforço precede determinaçãoCronograma

Figura 15 – A Inferência da Precedência entre Atividades.

Concluídos estes passos, tem-se um processo definido no modelo de classes de controle de processos da Estação TABA e um ambiente adequado a ele pode ser instanciado pelo instanciador de ambientes do meta-ambiente.

6. Trabalhos Correlatos

Alguns ADSs Centrados em Processo provêem suporte baseado em conhecimento à modelagem de processos, entre eles EPOS e Smart. EPOS [18] possui um planejador inteligente que é responsável pelo detalhamento de tarefas, gerando, automaticamente uma nova rede de subtarefas para cada tarefa composta. O planejador começa com uma tarefa composta e uma declaração da meta a ser atingida e aplica técnicas de encadeamento

progressivo e decomposição hierárquica, juntamente com conhecimento específico do domínio para construir uma rede de subtarefas apropriada.

Smart [19], por sua vez, utiliza uma ferramenta baseada em conhecimento, Articulator, para modelar, analisar e simular processos complexos. Smart possui uma biblioteca baseada em conhecimento, que apoia a organização, acesso e reuso de componentes de processo de software. Possui, ainda, uma base de conhecimento sobre os componentes de processo e provê um conjunto de operações que suportam navegação, busca, composição e abstração de modelos de processo.

Ambos os trabalhos estão bastante em linha com o trabalho aqui descrito. De fato, a motivação principal para o nosso trabalho advém da observação de que o conhecimento nestes ADSs está embutido em ferramentas específicas e, portanto, não está disponível para o ambiente como um todo. Isto é, há uma falta de integração do conhecimento sobre processos no ADS. Sem um modelo de conhecimento fundamentando a construção de bases de conhecimento, perde-se a possibilidade de relacionar conceitos, e as regras de inferência tornam-se elementos simbólicos, dissociados uns dos outros. Idealmente, o conhecimento deveria estar disponível para uso em outras ferramentas e para servir de base para um entendimento sobre o que é um processo de software. Construindo um modelo de conhecimento para o ADS, um vocabulário básico é estabelecido e, portanto, todas as ferramentas empregam os mesmos termos para referenciar os conceitos correspondentes do mundo real. Uma vez que estes conceitos estão amparados em um modelo de conhecimento global para o ADS, suas relações ganham semântica, deixando de ser puramente simbólicas. Assim, o aspecto mais importante de nossa proposta é o uso de ontologias para a modelagem do conhecimento de processos de software.

A própria construção da ontologia de processo de software permitiu que diferentes projetistas de processo atingissem um consenso sobre aspectos desta área de conhecimento. Por conseguinte, esta ontologia pode ser usada, também, para facilitar o entendimento de outras pessoas sobre as várias facetas de um processo de software. Finalmente, outras ferramentas podem ser construídas utilizando este conhecimento, compartilhando um mesmo vocabulário e, permitindo, desta forma, uma integração de ferramentas mais suave na Estação TABA.

7. Conclusões

A definição de processos é uma tarefa complexa que pode ser facilitada com o apoio de ferramentas baseadas em conhecimento. Neste trabalho, apresentamos Assist-Pro, um assistente baseado em conhecimento para apoiar esta atividade e seu processo de construção usando a infra-estrutura de conhecimento disponível na Estação TABA.

O aspecto central de nossa abordagem diz respeito ao uso de ontologias para modelar o conhecimento sobre processos de software. Usando ontologias, estabelece-se um vocabulário básico de consenso para o domínio em questão, facilitando a comunicação entre ferramentas e até mesmo entre especialistas. Uma vez que este conhecimento está disponível para o ADS, outras ferramentas podem ser construídas utilizando-o, permitindo, assim, reuso e compartilhamento de conhecimento.

Agradecimentos

Os autores agradecem a Kathia Oliveira por seus comentários e ao CNPq e à CAPES pelo apoio financeiro que permitiu a realização deste trabalho.

Referências[1]

[1] ISO 9000-3; *Quality management and quality assurance standards - Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software*, 1991.

[2] Paulk, M.C., Curtis, B., Chrissis, M.B.; "Capability Maturity Model, Version 1.1", IEEE Software, July 1993.

[3] Dorling, A; "SPICE: Software Process Improvement and Capability dEtermination", Information and Software Technology, v.35, n. 6/7, June/July 1993.

[4] Falbo, R.A. *A Integração de Conhecimento em um Ambiente de Desenvolvimento de Software*, Tese de Doutorado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, Dezembro 1998.

[5] Rocha, A.R.C., et alli; "TABA: a heuristic workstation for software development", COMPEURO'90, Israel, May 1990.

[6] G.H. Travassos; *O Modelo de Integração de Ferramentas da Estação TABA*, Tese de Doutorado, Engenharia de Sistemas e Computação, COPPE/UFRJ, Março 1994.

[7] Falbo, R.A., Menezes, C.S., Rocha, A.R.C.; "Um Servidor de Conhecimento de Processos de Software", Anais da X CITS-QS, Curitiba, Paraná, Maio 1999.

[8] Araújo, M.A. *Automatização do Processo de Desenvolvimento de Software nos Ambientes Instanciados pela Estação TABA*. Tese de Mestrado, Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ, 1998.

[9] Gruber, T.R.; "Towards principles for the design of ontologies used for knowledge sharing", Int. J. Human-Computer Studies, 43(5/6), 1995.

[10] Guarino, N.; "Understanding, building and using ontologies", Int. Journal Human-Computer Studies, 46(2/3), February/March 1997.

[11] Falbo, R.A., Menezes, C.S., Rocha, A.R.; "A Systematic Approach for Building Ontologies", Proceedings of the 6th Ibero-American Conference on Artificial Intelligence, IBERAMIA'98, Lecture Notes in Computer Science, vol 1484, October 1998.

[12] Wielemaker, J.; *SWI-Prolog 2.9 - Reference Manual*, 1998.

[13] Falbo, R.A., Travassos, G.H.; "Improving Tool's Integration on Software Engineering Environments Using Objects and Knowledge". Proc. 3th Int. Conference on Information Systems Analysis and Synthesis (SCI'97/ISAS'97), Caracas, Venezuela, July 1997.

[14] Breuker, J., Van de Velde, W.; *CommonKADS Library for Expertise Modelling*, IOS Press, 1994.

[15] Meyer, B., et alli; *ISE Eiffel: The Environment*. ISE Technical Report TR-EI-39/IE.

[16] ISO-IEC 12207; *Information tecnology - Software - Part 1: Software life-cycle process*, ISO/IEC DIS 12207-1, 1994.

[17] Pressman, R.S.; *Software Engineering: A Practiotioner's Approach*. 4th edition, Mc Graw Hill, 1997.

[18] Jaccheri, M., Conradi, R.; "Techniques for Process Model Evolution in EPOS", IEEE Transactions on Software Engineering, v. 19, n. 12, December 1993.

[19] Garg, P.K., et al.; "The SMART Approach for Software Process Engineering", Proceedings of the 16th Int. Conference on Software Engineering, Italy, May 1994.

[1] Os trabalhos [4, 7,11,13] podem ser encontrados em http://www.inf.ufes.br/~falbo.