Um Servidor de Conhecimento de Processo de Software

RICARDO DE ALMEIDA FALBO[1,2]
CREDINÉ SILVA DE MENEZES [2]
ANA REGINA CAVALCANTI DA ROCHA[1]

[1]COPPE SISTEMAS E COMPUTAÇÃO
UFRJ
Caixa Postal 68511, CEP 21945-970
Rio de Janeiro - RJ
{rfalbo, darocha}@cos.ufrj.br

[2]Departamento de Informática
UFES
Av. Fernando Ferrari, s/n
CEP 29000-060, Vitória - ES
{falbo, credine}@inf.ufes.br

Abstract

As the knowledge-based support in a Software Engineering Environment grows, it becomes necessary to integrate knowledge in it. In this paper we propose the use of Knowledge Servers to promote knowledge integration in SEEs. Knowledge Servers can improve integration by offering knowledge components to be reused and shared among tools. A Software Process Knowledge Server was developed to promote knowledge integration in the TABA Workstation and it was used to support the development of an intelligent assistant.

PALAVRAS-CHAVE: Processo de Software, Ambientes de Desenvolvimento de Software, Engenharia de Software Automatizada, Sistemas Baseados em Conhecimento.

1. Introdução

Na tentativa de se aperfeiçoar o desenvolvimento de software, em busca de um meio mais racional de produção, duas abordagens têm sido enfocadas:

- melhoria do processo através do qual o software é desenvolvido, e,
- uso de tecnologia para apoiar, automatizar e, portanto, em alguma extensão, reduzir o nível de habilidade requerido para o desenvolvimento de software.

A primeira abordagem envolve a definição, execução, acompanhamento e melhoria de processos e está fortemente associada a modelos e padrões para a qualidade do processo, tais como ISO 9000-3, ISO 12207, CMM e SPICE. Contudo, à medida que os processos tornam-se mais complexos, o volume de pessoas envolvidas e o número de tarefas a serem realizadas crescem em grandes proporções, e torna-se inviável gerenciar tais processos sem ferramentas de apoio. Neste contexto, a segunda abordagem surge como uma área de pesquisa complementar, onde Ambientes de Desenvolvimento de Software (ADSs) têm sido uma das principais metas.

Ambientes de Desenvolvimento de Software provêem serviços de infra-estrutura, permitindo integrar ferramentas individuais ao longo de três dimensões principais - controle, dados e interface com usuário - de modo que o processo de desenvolvimento possa ser conduzido de modo uniforme e consistente [1].

Além disso, dada a complexidade das tarefas realizadas no desenvolvimento de software, é importante que o ADS ofereça suporte baseado em conhecimento

para o desenvolvedor. Com o aumento do número de assistentes inteligentes em um ADS, passa a ser necessário considerar a integração de ferramentas em um ADS como uma questão de quatro dimensões, incluindo, além da integração de dados, controle e interface com o usuário, a *integração de conhecimento* [1]. Idealmente, o conhecimento não deve ficar embutido em uma ferramenta, mas, ao contrário, deve estar integrado ao ambiente para que possa ser compartilhado e reutilizado por diversas ferramentas [2].

Este artigo discute o uso de Servidores de Conhecimento como um meio de promover a integração de conhecimento em ADSs e apresenta o Servidor de Conhecimento de Processo (SCP) desenvolvido para promover o reuso e compartilhamento de conhecimento de processos de software entre ferramentas da Estação TABA [3]. As seções 2 e 3 apresentam, respectivamente, a arquitetura geral de um Servidor de Conhecimento e o Servidor de Conhecimento de Processo. A seção 4 discute uma abordagem de Engenharia de Conhecimento apoiada em uma infra-estrutura de componentes de conhecimento, como o são os Servidores de Conhecimento. A seção 5 mostra como o SCP deu suporte à construção de um assistente inteligente para apoiar a definição de processos de software. Finalmente, na seção 6, são apresentadas as conclusões deste trabalho.

2. Servidores de Conhecimento - Arquitetura Geral

Integrar conhecimento em um ADS significa, em última instância, compartilhar e reutilizar conhecimento entre suas ferramentas. Assim como qualquer porção de software, o conhecimento tem de ser construído *para* reuso, a fim de poder ser reutilizado em um desenvolvimento *com* reuso.

Para compartilhar conhecimento em um ADS, então, é preciso mudar a forma como são construídas suas ferramentas. Na maioria dos ADSs, tais como em [4,5], o conhecimento do ambiente é dado pelo conhecimento embutido em cada uma de suas ferramentas, isto é, as ferramentas possuem o conhecimento necessário para seu propósito e cabe a elas oferecer o suporte baseado em conhecimento ao ambiente. Entretanto, o que se faz necessário é a construção de um modelo de conhecimento para o ambiente e seu uso por cada uma das ferramentas. Neste caso, cabe ao ambiente oferecer o suporte baseado em conhecimento para suas ferramentas [6].

Uma idéia chave, introduzida pela comunidade de Inteligência Artificial para aumentar a reusabilidade do conhecimento e facilitar a manutenção dos sistemas resultantes, consiste em separar os conhecimentos de domínio e de tarefa. Esta separação mostra que há dois problemas basicamente distintos, apesar de relacionados, no projeto de um sistema baseado em conhecimento e que estes podem ser, pelo menos até certo ponto, investigados e resolvidos separadamente. Desta forma, podem existir modelos estruturados separados para representar o domínio e a tarefa.

Com base no princípio da distinção entre conhecimento de domínio e de tarefa e na necessidade de se prover componentes de conhecimento reutilizáveis, desenvolvemos o conceito de Servidores de Conhecimento. Um Servidor de Conhecimento é uma infra-estrutura de conhecimento sobre um *universo de discurso*, provendo módulos de conhecimento (conhecimento de domínio), passíveis de serem combinados para atender às especificidades de uma aplicação particular, e máquinas de inferência customizadas para tipos de problema variados (conhecimento de tarefa).

Para permitir o reuso de conhecimento de tarefa é necessário prover

templates de resolvedores genéricos de problema. Ao invés de prover apenas sistemas de representação e suas máquinas genéricas de inferência, um Servidor de Conhecimento deve oferecer uma biblioteca de resolvedores de problemas, passíveis de serem instanciados e adaptados para aplicações particulares.

No que tange ao reuso de conhecimento de domínio, é desejável que a base de conhecimento do Servidor seja modular e baseada em ontologias. Uma ontologia é uma especificação de uma conceituação [7], definindo conceitos e relações, e suas definições, propriedades e restrições, escritas na forma de axiomas. A instanciação de uma ontologia compreende um conjunto de declarações sobre elementos do universo de discurso, usando os conceitos e relações definidos na ontologia. Ontologias e suas instanciações são implementadas em módulos de conhecimento, de modo que a base de conhecimento de uma aplicação que se comprometa com uma ou várias ontologias venha a ser a conjunção dos módulos de conhecimento correspondentes, mais o conhecimento específico da aplicação. Desta forma, como mostra a figura 1, a arquitetura de um Servidor de Conhecimento compreende essencialmente:

- uma base de conhecimento modular, onde cada módulo de conhecimento (MC) contém um corpo de conhecimento reutilizável, construído com base em ontologias, e,
- a máquina de inferência do sistema de representação adotado, associada a um conjunto de *templates* de resolvedores de problemas, para especializá-la para os tipos de problema mais freqüentemente encontrados no universo de discurso.

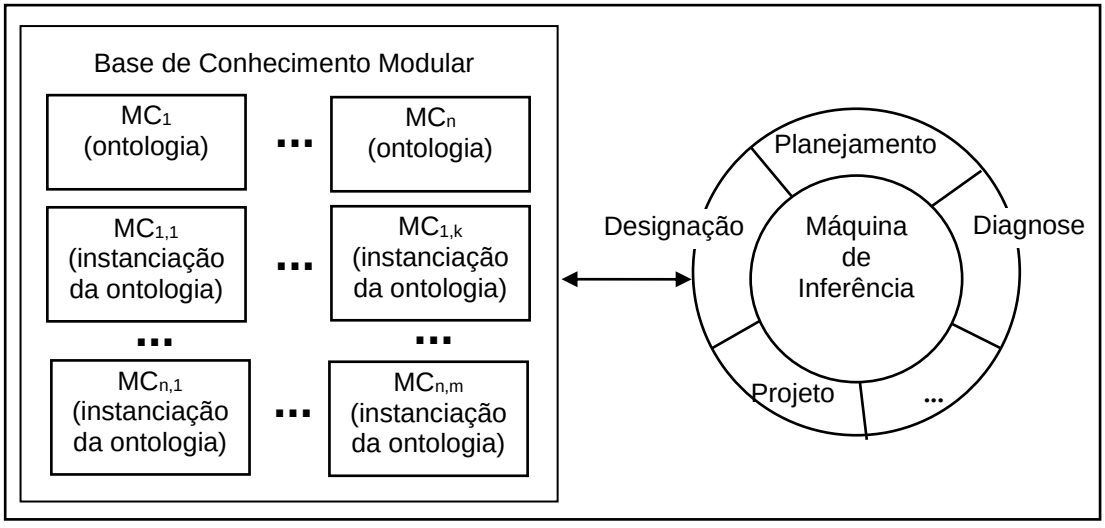

Figura 1 - Arquitetura Geral de Servidores de Conhecimento

A base de conhecimento modular é essencialmente uma biblioteca de ontologias, já que seu critério de modularização é dado pelo uso de ontologias. Cada ontologia é implementada em um módulo de conhecimento na linguagem do sistema de representação, assim como cada uma de suas instanciações.

A máquina de inferência é aquela provida pelo sistema de representação de conhecimento adotado. Os *templates* de resolvedores de problemas, por sua vez, implementam modelos de tarefa genéricos para os tipos de problemas que ocorrem com freqüência no universo de discurso do Servidor de Conhecimento.

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 suportado

por ele. 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. Assim, com estes componentes disponíveis, é possível uma mudança de enfoque na construção de assistentes inteligentes para um ADS.

3. O Servidor de Conhecimento de Processo

Para materializar a idéia de Servidores de Conhecimento, foi desenvolvido, para a Estação TABA, um Servidor de Conhecimento para o universo de discurso de processos de desenvolvimento de software, o Servidor de Conhecimento de Processo (SCP). A motivação para a construção deste servidor advém do reconhecimento da importância de se ter conhecimento sobre processos de software explicitamente representado e passível de compartilhamento por várias ferramentas em um ADS.

A Estação TABA [3] 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 de software [8]. Entretanto, tais ferramentas não oferecem nenhum tipo de suporte baseado em conhecimento para o engenheiro de software nestas tarefas que, sem dúvida, são bastante complexas e requerem uma gama variada de conhecimento. Assim, escolhemos o universo de discurso de processos de desenvolvimento de software para materializar as discussões sobre integração de conhecimento através de Servidores de Conhecimento. O objetivo do Servidor de Conhecimento de Processo é apoiar a construção de ferramentas baseadas em conhecimento de processo na Estação TABA.

O universo de discurso de processos de software, contudo, é bastante amplo, envolvendo um grande volume de conhecimento. Assim, para efeito da construção do protótipo do Servidor de Conhecimento de Processo, optamos por considerar, como ponto de partida, apenas o conhecimento envolvido na definição de um processo de software – o primeiro passo no tratamento de processos de software. Este conhecimento é a base para o apoio à descrição de processos de software no meta-ambiente, mas é, também, extremamente útil para as outras ferramentas de acompanhamento, execução e adaptação de processo. A partir deste núcleo, o Servidor de Conhecimento poderá ser estendido para contemplar uma competência mais abrangente.

Levando em consideração a competência do Servidor de Conhecimento de Processo, a arquitetura geral de Servidores de Conhecimento, mostrada na figura 1, foi instanciada na arquitetura mostrada na figura 2.

A base de conhecimento modular deve ser baseada em uma ontologia de processo de software e em outras ontologias, desenvolvidas como base para a ontologia de processo de software, a saber ontologias de atividade, recurso e procedimento [2,6]. As instanciações das ontologias derivam, também, módulos de conhecimento. Uma vez que a Estação TABA possui uma máquina de inferência Prolog integrada a ela [9], os módulos de conhecimento foram escritos nesta linguagem [2].

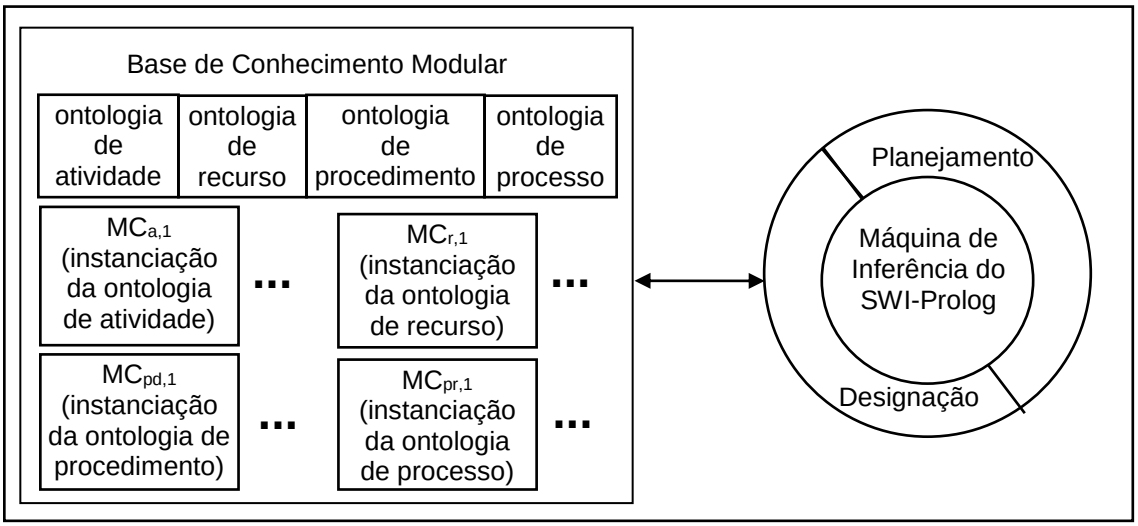

Figura 2 - O Servidor de Conhecimento de Processo.

A camada de resolvedores de problemas deve customizar a máquina de inferência para os tipos de problemas inerentes à competência do Servidor de Conhecimento, ou seja, a definição de processos de software.

A definição de um processo envolve, tipicamente, dois tipos de problema genéricos: o planejamento e a designação. Tarefas de planejamento têm por objetivo construir um plano com base no estado inicial do mundo e no estado que se deseja atingir. O plano resultante é representado, de modo geral, como uma seqüência de ações que, quando executada partindo do estado inicial, deve conduzir ao estado meta [10]. Tarefas de designação visam preencher uma estrutura com elementos, de modo que determinados requisitos e restrições sejam satisfeitos. Quando a estrutura a ser preenchida é um plano, os elementos a serem designados às atividades do plano são geralmente recursos e o problema é então chamado de *programação* ou *escalonamento* [10].

Modelos de tarefa para estes dois tipos de problemas foram adaptados para o contexto da definição de processos de software, derivando *templates* de resolvedores de problemas [2]. Estes templates foram implementados em Eiffel, a linguagem de programação utilizada na implementação do ambiente.

Desta forma, o SCP torna disponíveis dois tipos de componentes reutilizáveis de conhecimento: módulos de conhecimento, correspondendo ao conhecimento de domínio, derivados a partir das ontologias e suas instanciações, e templates de resolvedores de problemas, correspondendo ao conhecimento de tarefa, que podem ser especializados para derivar resolvedores de problemas para aplicações específicas. A figura 3 mostra o diagrama de classes[1] do modelo de conhecimento da Estação TABA.

A classe *Conhecimento Domínio* pode, ainda, ser especializada para contemplar outros domínios de aplicação, como por exemplo Direito e Medicina [11], permitindo, assim, a criação de Servidores de Conhecimento para estes domínios.

[1] usando a notação da UML.

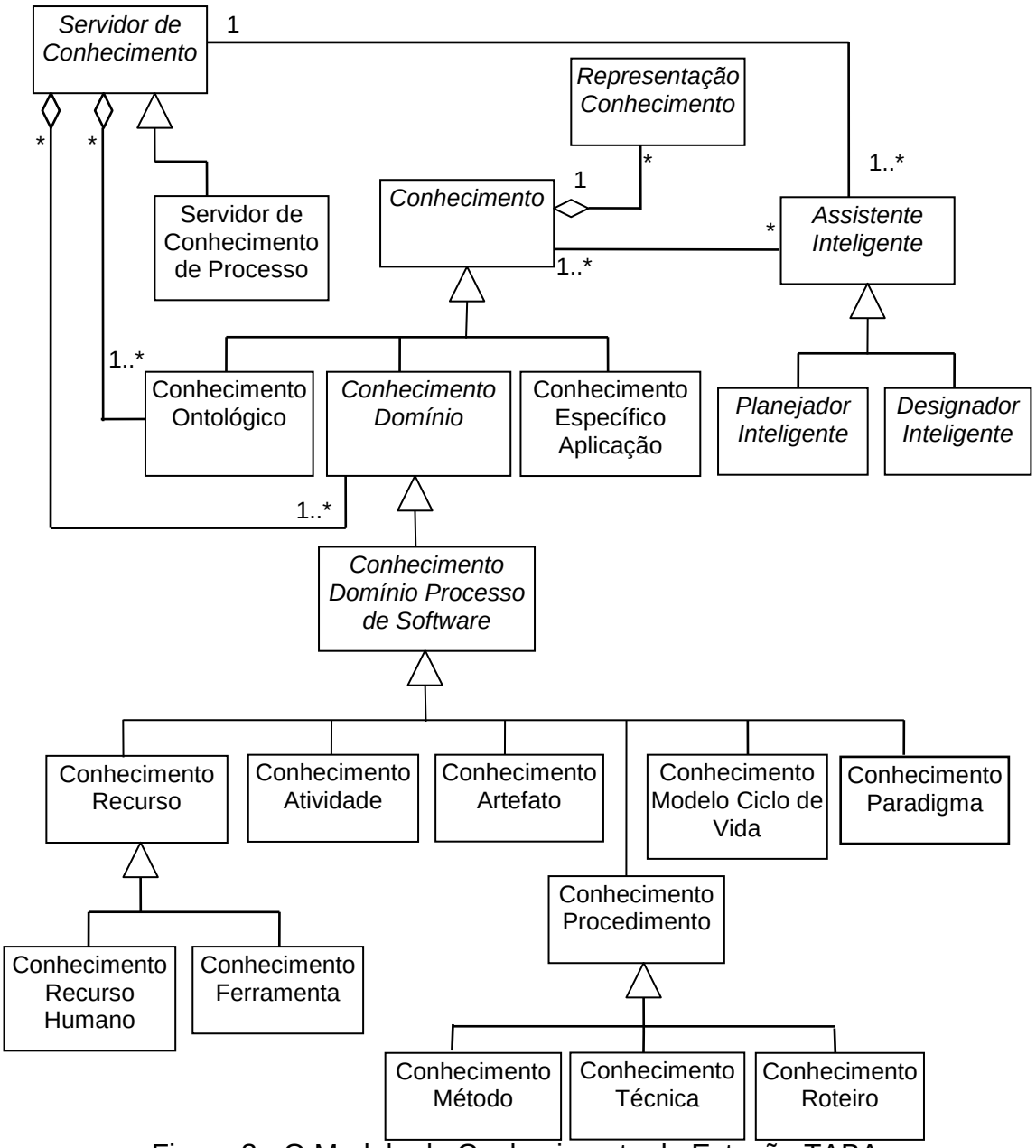

Figura 3 - O Modelo de Conhecimento da Estação TABA.

4. Uma Abordagem de Engenharia de Conhecimento Baseada em Reutilização

Tendo em vista que um Servidor de Conhecimento é, de fato, uma infra-estrutura para apoiar a construção de ferramentas baseadas em conhecimento no ADS, faz-se necessário estabelecer como utilizar esta infra-estrutura no desenvolvimento de assistentes inteligentes. Assim, propusemos uma abordagem de Engenharia de Conhecimento baseada em Reutilização, a ser utilizada na construção de sistemas baseados em conhecimento (SBCs), sempre que houver uma infra-estrutura de componentes reutilizáveis de conhecimento, como são os Servidores de Conhecimento. Esta abordagem compreende os seguintes passos [2]:

1. *Análise de Requisitos*: O desenvolvimento de um SBC, assim como qualquer software, começa pela especificação de requisitos, onde se procura obter uma visão inicial do tipo de aplicação requerida. Esta fase tem o intuito de gerar uma descrição informal do domínio e da tarefa da aplicação e requer, tipicamente,

entrevistas com gerentes e especialistas do domínio, leituras sobre a área, etc.

2. *Seleção/Construção de Ontologias*: Uma vez identificado o domínio da aplicação, deve-se selecionar a(s) ontologia(s) apropriada(s), adequando-a(s) ao desenvolvimento em questão. Entretanto, uma vez que a maioria dos engenheiros de conhecimento ainda não tem à sua disposição uma biblioteca de ontologias, é necessário considerar a construção de novas ontologias. Para tal, deve ser utilizado um método para construção de ontologias, tal como apresentado em [12].
3. *Seleção/Construção de Modelos de Tarefa*: De forma análoga à perspectiva de domínio, uma análise da perspectiva de tarefa deve ser realizada. O propósito da análise de tarefa é decompor a tarefa real da aplicação em um número de tarefas genéricas. Uma vez que já existem muitos trabalhos sobre modelos de tarefa, uma abordagem pragmaticamente eficiente consiste em utilizar modelos de tarefa, como proposto em CommonKADS [10], na realização desta etapa.
4. *Compatibilização da Ontologia com os Modelos de Tarefa*: Nesta etapa, especifica-se a interação entre as ontologias e os modelos de tarefa. Ao utilizarmos duas estratégias separadas para desenvolver modelos de resolução de problema, assumimos que é desejável que as modelagens de tarefa e de domínio tenham um acoplamento suave. O ponto de ligação entre esses dois modelos é exatamente o papel que o conhecimento descrito no modelo de domínio deverá desempenhar no processo de resolução de problema, descrito nos modelos de tarefa. Assim, os *papéis de conhecimento*[2] são o ponto-chave no controle da interação, e como tal, precisam ser descritos claramente. Ao se mapear conceitos da ontologia em papéis dos modelos de tarefa, estabelece-se claramente a relação entre os papéis de conhecimento e as categorias ontológicas que podem preenchê-los. Com este mapeamento, tem-se um modelo completo do esqueleto da aplicação.
5. *Aquisição e Modelagem do Conhecimento Específico da Aplicação*: Uma vez delineado o esqueleto da aplicação, é preciso dar-lhe corpo. Assim, é necessário instanciar a ontologia com o conhecimento necessário para a aplicação e associar métodos de resolução de problemas às tarefas do modelo de tarefas. Enquanto a ontologia define os conceitos usados no universo de discurso, a instanciação de uma ontologia compreende um conjunto de declarações sobre elementos do universo de discurso, usando os conceitos e relações definidos na ontologia. Além das instanciações das ontologias, alguns tipos de conhecimento, específicos da aplicação, tais como heurísticas e conhecimento compilado, devem ser elicitados e instanciados. É importante frisar, contudo, que as ontologias guiam fortemente a realização desta tarefa.
6. *Projeto*: Assim como em qualquer desenvolvimento de software, a fase de Projeto diz respeito a considerações de aspectos tecnológicos impostos pela plataforma de implementação.
7. *Implementação e Testes*: Na implementação, as especificações de projeto são transformadas no sistema final, levando-se em consideração requisitos não funcionais, de caráter simbólico e tecnológico. Finalmente, o sistema deve ser testado para verificar se atende aos propósitos para os quais foi projetado.

[2] Um *papel de conhecimento* em um modelo de tarefas descreve o papel que uma porção de conhecimento desempenha na resolução do problema descrito por este modelo.

5. Usando o Servidor de Conhecimento de Processo para Apoiar a Construção de um Assistente Inteligente

Seguindo a abordagem descrita na seção anterior, desenvolvemos um assistente inteligente para apoiar a definição de processos de software na Estação TABA, usando os componentes de conhecimento do SCP. O escopo de assistência desta ferramenta, contudo, é, neste primeiro momento, limitado: contempla apenas processos de desenvolvimento 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.

A definição de processos de software é uma tarefa que requer conhecimento intenso, sendo realizada por especialistas experientes. Para engenheiros de software menos experientes, esta é uma tarefa que potencialmente requer alguma forma de assistência inteligente. Diferentes aplicações possuem diferentes perfis e estas diferenças têm influência sobre seus processos de desenvolvimento. Para se definir um processo de software é imprescindível que algumas características gerais relacionadas ao projeto sejam observadas, tais como:

- *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 da equipe e da gerência*: a formação, atualização e experiência da equipe são fatores importantes a serem considerados na definição do processo. Além disso, o grau de inovação que a gerência está disposta a enfrentar também tem impacto direto na definição do processo.
- *Responsabilidade pelo desenvolvimento*: deve-se definir quem desenvolverá o software. As seguintes possibilidades devem ser consideradas: o software será desenvolvido com equipe da própria empresa, ou com contratação de terceiros (no todo ou em parte), ou com equipe mista (consultores externos e desenvolvedores da própria empresa). Este aspecto tem impacto direto na definição de atividades de gerência e de controle da qualidade.

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

- Definição de um ciclo de vida para o processo: 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. Um modelo de ciclo de vida deve ser selecionado com base nas características do projeto em questão e adaptado a ele.
- 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 de como as atividades devem ser realizadas: para cada atividade do processo, deve-se definir como esta será realizada, isto é, que procedimentos (métodos, técnicas, roteiros, etc.) 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.

- Definição dos artefatos consumidos e produzidos por uma atividade: para cada atividade do processo é necessário definir que artefatos são consumidos e que artefatos são produzidos por ela
- Definição dos recursos para as atividades: finalmente, uma vez definidas todas as atividades que compõem o processo, deve-se determinar que recursos serão necessários para a realização das mesmas.

Claramente, a definição de um processo de software é uma tarefa que mescla problemas do tipo planejamento e designação. Além disso, grande parte do conhecimento de domínio envolvido nesta tarefa já está disponível no Servidor de Conhecimento de Processo. É claro que algum conhecimento específico desta aplicação teve de ser elicitado e modelado, mas, ao invés de construir este assistente elaborando uma nova conceituação a partir do nada, o conhecimento disponível no Servidor de Conhecimento de Processo foi reutilizado e, mais importante, a ontologia de processo de software forneceu um vocabulário básico e guiou a aquisição do conhecimento mais específico.

Adaptação dos Modelos de Tarefa do Servidor de Conhecimento

Analisando a descrição anterior de como proceder para definir um processo, é possível observar que, basicamente, a definição de processos de software envolve duas grandes tarefas: planejamento do processo e designação de procedimentos, artefatos e recursos para suas atividades, como mostra a figura 4. Assim, os modelos de tarefa para estes dois tipos de problemas foram adaptados para o problema em questão [2]. Na figura 4, não mostramos os papéis de conhecimento estáticos para tornar o modelo mais legível ao leitor.

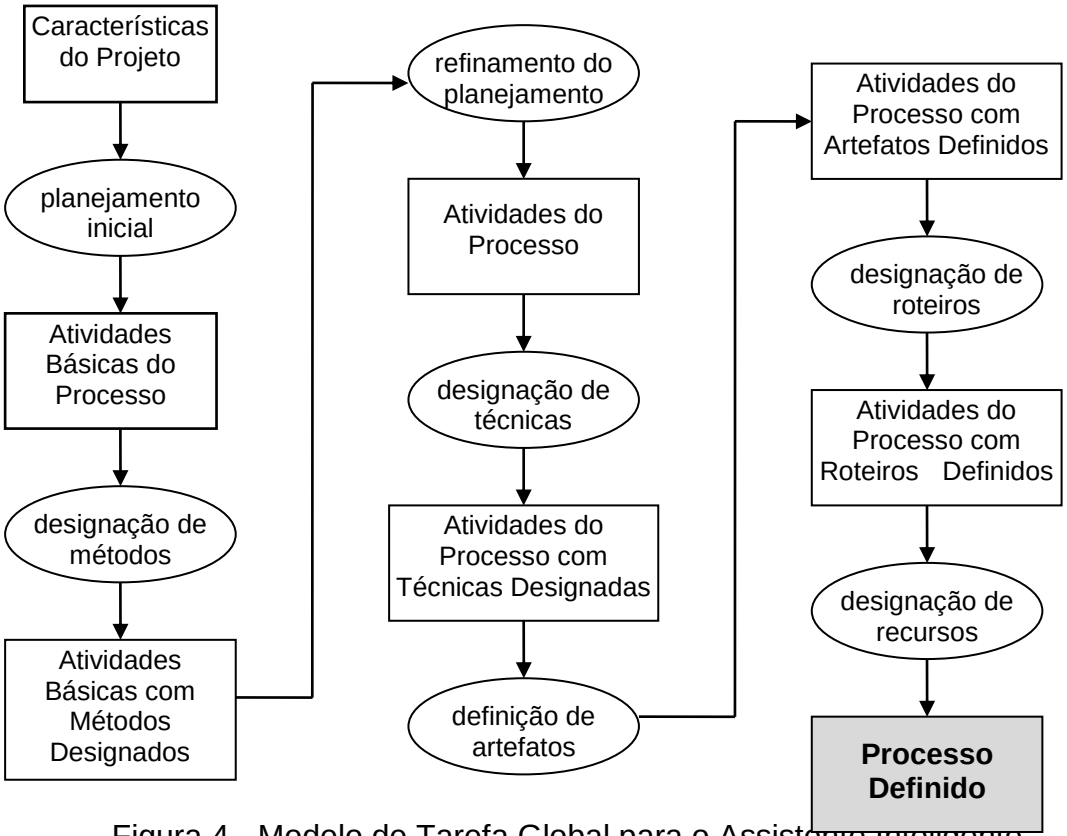

Figura 4 - Modelo de Tarefa Global para o Assistente Inteligente.

O Conhecimento Necessário

Grande parte do conhecimento necessário para a definição de um processo de software já estava disponível no Servidor de Conhecimento de Processo, na forma de módulos de conhecimento oriundos das ontologias e suas instanciações, e, portanto, foram utilizados diretamente. De fato, foi necessário capturar apenas parte do conhecimento sobre aplicabilidade de modelos de ciclo de vida.

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 [2]. Entretanto, aspectos como características do problema e da equipe de desenvolvimento não foram capturados. Assim, uma nova etapa de aquisição de conhecimento teve de ser realizada. Esta etapa foi, fortemente, guiada pela ontologia. Primeiramente, o conhecimento foi elicitado apenas para modelos de ciclo de vida já descritos como instanciações da ontologia, a saber: em cascata, incremental, RAD, evolutivo básico, prototipagem operacional e paralelo/recursivo. Em segundo lugar, o vocabulário básico definido pela ontologia de processo foi usado para descrever o conhecimento, sendo necessário, obviamente, estendê-lo para considerar termos específicos, não previstos na ontologia. As tabelas 1 e 2 apresentam, de forma compacta, o conhecimento elicitado.

Modelo de Ciclo de Vida	**Aplicabilidade**				
	Características do Desenvolvimento		**Características do Problema**		
	Paradigma	Software pode ser colocado em uso rapidamente? (com funcionalidade total / parcial)	Grau de Definição do Problema	Tamanho	Modularidade
Cascata	Estrutural / OO	não	bem-definido	pequeno	qualquer
Incremental	Estrutural / OO	sim	bem-definido	médio a grande	média a alta
RAD	Estrutural / OO	sim	bem-definido	pequeno a médio	alta
Evolutivo Básico	OO	sim	bem ou mal-definido	qualquer	qualquer
Prototipagem Operacional	OO	não	bem ou mal-definido	pequeno a médio	qualquer
Paralelo / Recursivo	OO	sim	bem ou mal-definido	médio a grande	qualquer

Tabela 1 - Características do Desenvolvimento e do Problema a ser resolvido e a adequabilidade de Modelos de Ciclo de Vida.

Além de ser útil para a seleção de um modelo de ciclo de vida para o processo, o conhecimento sobre as características da equipe de desenvolvimento e da gerência, mostrado na tabela 2, é também importante para a definição de

atividades de treinamento a serem incorporadas no processo.

Finalmente, foi necessário capturar algum conhecimento sobre a responsabilidade pelo desenvolvimento. Basicamente, foram consideradas as seguintes possibilidades:

- software desenvolvido na empresa, sendo a equipe composta apenas por membros da empresa;
- software desenvolvido com contratação de terceiros, no todo ou em parte;
- software desenvolvido por equipes mistas.

Estas diferentes possibilidades de responsabilidades pelo desenvolvimento têm impacto na definição de atividades de gerência e controle da qualidade, além de revelar novos papéis de recursos humanos no contexto do desenvolvimento.

Modelos de Ciclo de Vida	**Aplicabilidade** **Características da Equipe de Desenvolvimento e da Gerência**		
	Nível de Formação	**Nível de Atualização**	**Nível de Experiência**
Cascata	baixo	baixo	baixo
Incremental	baixo	médio	médio
RAD	médio	médio	alto
Evolutivo Básico	médio	médio	médio
Prototipagem Operacional	médio	médio	médio
Paralelo / Recursivo	alto	alto	alto

Tabela 2 - Características da Equipe de Desenvolvimento e da Gerência e a Adequabilidade de Modelos de Ciclo de Vida.

6. Conclusões

Desenvolvimento e manutenção de software são atividades que requerem inteligência, conhecimento e disciplina. Com o aumento da complexidade dos processos de desenvolvimento, faz-se necessário oferecer algum suporte baseado em conhecimento às suas várias atividades. Neste contexto, cada vez mais, técnicas de IA têm sido aplicadas na construção de assistentes inteligentes para apoiar engenheiros de software na realização de suas tarefas. Entretanto, de modo geral, cada uma destas aplicações traz o conhecimento embutido e, portanto, não disponível para o ambiente de desenvolvimento como um todo.

Neste trabalho discutimos o uso de Servidores de Conhecimento como um meio de integrar conhecimento em ADSs. Servidores de Conhecimento permitem que a construção de um assistente inteligente possa ser feita usando toda uma infra-estrutura de conhecimento pré-existente. Modelos de tarefa são adaptados para a aplicação específica; módulos de conhecimento são associados aos papéis de conhecimento dos modelos de tarefa adaptados; e aqueles papéis de conhecimento que não forem completamente preenchidos por módulos de conhecimento do Servidor, deverão ser o alvo da segunda etapa da aquisição do conhecimento.

Para materializar as idéias discutidas, apresentamos o Servidor de Conhecimento de Processo desenvolvido para a Estação TABA, um meta-ambiente de desenvolvimento de software. Este Servidor permite integrar conhecimento de

processos de software ao ADS, fornecendo um conjunto de componentes reutilizáveis de conhecimento. Além disso, comentamos o processo de construção de um assistente inteligente para a definição de processos de software, usando a infra-estrutura do SCP.

Referências

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

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

[3] A. R. C. da Rocha, et al. *TABA: a heuristic workstation for software development*, *COMPEURO'90*, Israel, May 1990.

[4] H. GOMAA, et al. *A Knowledge-Based Software Engineering for Reusable Software Requirements and Architectures*. *Automated Software Engineering*, Vol.3, N.3/4 (August), 1996.

[5] P.K. GARG, et al. *The SMART Approach for Software Process Engineering*. *Proceedings of the 16th Intern. Conf. on Software Engineering*, Italy, May, 1994.

[6] R. A. Falbo; A. R. C. da Rocha; C. S. Menezes. *Integração de Conhecimento sobre Processos de Software em um Ambiente de Desenvolvimento. Anais da IX CITS*, Curitiba, Paraná, 243-254, Junho 1998.

[7] T.R. GRUBER. *Towards principles for the design of ontologies used for knowledge sharing*. *Int. J. Human-Computer Studies*, Vol. 43, N.5/6, 1995.

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

[9] R.A. FALBO, G.H. TRAVASSOS. *Improving Tool's Integration on Software Engineering Environments Using Objects and Knowledge*. *Proceedings of SCI'97/ISAS'97*, Caracas, Venezuela, July 1997.

[10] J. BREUKER, VAN DE VELDE. *CommonKADS Library for Expertise Modelling*, IOS Press, 1994.

[11] K.M. OLIVEIRA, A.R. ROCHA, G.H. TRAVASSOS, S. MATWIN. *Towards a Domain-Oriented Software Development Environment for Cardiology*. *CAiSE'98, 5th Doctoral Consortium*, Pisa, Italy, June 1998.

[12] R.A. FALBO, C.S.MENEZES, A.R.C.ROCHA. *A Systematic Approach for Building Ontologies*. *Proceedings of the IBERAMIA'98*, Lisbon, Portugal, 1998.