Construção de Ambientes de Desenvolvimento de Software Orientados a Domínio na Estação TABA

Káthia M. Oliveira[1,2], Gleison Santos[1], Fábio Zlot[1], Gustavo Guedes[1], Alessandro Cerqueira[1], Catia Gallota[1,3], Luis Filipe Machado[1], Karina Lima[1,2], Francisco Rapcham[4], Ricardo Falbo[4], Guilherme H. Travassos[1], Ana Regina Rocha[1]

[1]COPPE Sistemas - Universidade Federal do Rio de Janeiro – Brasil
[2]UCCV/FBC - Universidade Federal da Bahia – Brasil
[3] IPqM – Instituto de Pesquisas da Marinha – Brasil
[4]Departamento de Informática - Universidade Federal do Espírito Santo – Brasil

e-mail: {kathia, darocha, ght}@cos.ufrj.br

Resumo

Desenvolver software em um domínio em que os desenvolvedores não tem nenhuma experiência não é uma tarefa fácil. Além da própria complexidade da atividade de desenvolvimento de um sistema, os desenvolvedores tem que lidar com a necessidade de conhecer os conceitos do domínio para facilitar a interação com seus usuários e permitir a construção do sistema. Ambientes de Desenvolvimento de Software Orientados a Domínio visam oferecer apoio automatizado aos desenvolvedores de software que não tem familiaridade com o domínio, utilizando o próprio conhecimento do domínio como fator essencial para o desenvolvimento de sistemas. Neste artigo, descrevemos as principais características desses ambientes e como eles são construídos utilizando a infra-estrutura da Estação TABA, um meta-ambiente que gera, através de instanciação, ambientes de desenvolvimento de software adequados às particularidades de processos de desenvolvimento e de projetos específicos. A partir destas definições, apresentamos NETUNO, um ADSOD para o domínio de Acústica Submarina que foi construído utilizando essa infra-estrutura.

1. Introdução

Durante o desenvolvimento de software, os desenvolvedores têm que lidar com diferentes atividades não triviais. A mais crítica dessas atividades é, provavelmente, a identificação correta dos requisitos do sistema e sua descrição. Esta atividade é ainda mais difícil quando os desenvolvedores não conhecem o domínio ou não têm nenhuma experiência em desenvolver software para aquele domínio. Neste caso, a produtividade no desenvolvimento também é prejudicada. Acreditando que o uso do conhecimento do domínio durante o desenvolvimento de software pode tornar esse processo mais fácil e melhorar a produtividade, definimos e estabelecemos um modelo para construção de Ambientes de Desenvolvimento de Software Orientados a Domínio (ADSOD) (Oliveira *et al*. 1999a).

ADSOD podem ser vistos como uma evolução dos ambientes de desenvolvimento de software tradicionais que se propõem a prover apoio para o desenvolvimento, reparo e melhorias em software e para o gerenciamento e controle destas atividades (Brown *et al.* 1992), no sentido de que, além de oferecerem ferramentas de apoio para essas atividades, se propõem a disponibilizar o conhecimento do domínio, de forma organizada e estruturada, para orientar o desenvolvimento de software.

Construir um ADS, no entanto, não é uma atividade simples. Para apoiar esta atividade, o grupo de engenharia de software da COPPE/UFRJ definiu e construiu a Estação TABA, que tem como objetivo auxiliar na definição, implementação e execução de Ambientes de Desenvolvimento de Software (ADS) (Rocha *et al.* 1990). Com uma infra-estrutura bem definida e experiências anteriores de construção de ADS (Werneck *et al.* 1995, Werner *et al.* 1997), decidimos estender a Estação TABA para permitir a definição e construção de ADSOD.

Neste artigo, apresentamos como construímos ADSOD utilizando a infra-estrutura da Estação TABA. Inicialmente (seção 2), apresentamos os ADSOD, no que se refere a seus conceitos principais e características básicas. Na seção 3, descrevemos brevemente os principais aspectos da Estação TABA e as novas funcionalidades acrescentadas para permitir a construção de ADSOD. Na seção 4, apresentamos NETUNO, um ADSOD específico para o domínio de Acústica Submarina, construído a partir dessas funcionalidades. Finalmente, na seção 5, apresentamos as conclusões e os trabalhos futuros previstos para este projeto.

2. Ambientes de Desenvolvimento de Software Orientado a Domínio

Ambientes de Desenvolvimento de Software Orientados a Domínio (ADSOD) são ambientes de desenvolvimento de software que apoiam os engenheiros de software em domínios específicos através do uso do conhecimento deste domínio durante todo o processo de desenvolvimento. Basicamente, ADSs são implementações de processos de desenvolvimento de software, disponibilizando apoio automatizado e integrado para a maioria das atividades previstas no processo de desenvolvimento.

Para permitir o uso do conhecimento durante o desenvolvimento de software, um ADSOD possui os seguintes requisitos mínimos: *(i)* apoiar a investigação do domínio durante o desenvolvimento através de algum nível de formalização, *(ii)* permitir ao desenvolvedor trabalhar diretamente com o domínio do problema, *(iii)* apoiar a identificação, acesso e uso de objetos do conhecimento do domínio, e, *(iv)* apoiar o trabalho colaborativo entre usuários do domínio (ou especialistas do domínio) e desenvolvedores dado que, com o conhecimento do domínio integrado no ambiente, o trabalho com os usuários pode ser mais fácil por se utilizar um vocabulário comum e particular do domínio em questão.

De forma geral, um ADSOD é caracterizado por possuir o conhecimento do domínio e permitir o acesso e uso desse conhecimento durante o desenvolvimento de software. O conhecimento do domínio é organizado em um modelo chamado Teoria do Domínio, cuja característica principal é o uso de ontologia do domínio (Van Heijst *et al.* 1997). Ontologia (Gruber 1995) é uma especificação explícita de uma conceituação, ou seja, uma especificação explícita dos objetos, conceitos e outras entidades que assumimos existir em uma área de interesse, além das relações entre estes conceitos e restrições expressas através de axiomas. Uma ontologia do domínio (Van Heijst *et al.* 1997) expressa uma conceituação para um domínio particular, definindo restrições na estrutura e conteúdo do conhecimento desse domínio. Estas restrições são explicitamente expressas na forma de axiomas formais.

Um Teoria do Domínio é, então, composta de ontologias do domínio, podendo ser dividida em sub-teorias. Cada sub-teoria considera os conceitos que estão semanticamente relacionados e em um mesmo nível de abstração. Faz parte da Teoria do Domínio, ainda, a identificação das tarefas relacionadas àquele domínio e o mapeamento de quais sub-teorias são necessárias para o entendimento da tarefa.

O uso da Teoria do Domínio ao longo do desenvolvimento é estabelecido pela inclusão, no processo de software, de uma sub-atividade denominada "investigação do domínio", que deve estar presente em qualquer atividade do desenvolvimento em que seja importante o entendimento e o uso do domínio. Cada atividade de desenvolvimento pode ter maior ou menor necessidade da investigação do domínio. De forma geral, a maior influência da investigação do domínio refere-se às atividades de entendimento do problema, ou seja na definição e modelagem do sistema, já que o bom entendimento do domínio é crucial para a qualidade do sistema produzido e adequação aos requisitos do usuário. Na análise de requisitos, por exemplo, a sub-atividade de investigação do domínio é de extrema importância. Considerando algumas das principais atividades da análise, como elicitação de requisitos e modelagem do sistema, podemos perceber essa influência.

No que se refere à elicitação de requisitos (ou elicitação do conhecimento para sistemas baseados em conhecimento), a investigação do domínio é importante tanto na preparação para a realização desta atividade quanto na sua efetiva realização. Nos dois momentos, ferramentas específicas para o domínio podem apoiar o desenvolvedor de software facilitando o processo. Ferramentas específicas do domínio diferem das ferramentas tradicionais por utilizarem o vocabulário do domínio. Na preparação para a elicitação de requisitos, ferramentas especialmente construídas para explorar o conhecimento do domínio podem mostrar os conceitos, descrições, suas relações e as características definidas na Teoria do Domínio. A Teoria do Domínio provê, ainda, o mapeamento de quais conceitos são importantes para determinadas tarefas, indicando o que realmente deve ser investigado.

Na modelagem de dados do sistema, a investigação do domínio auxilia a estruturação dos dados de acordo com a organização definida na Teoria do Domínio. O próprio modelo da Teoria do Domínio serve como um modelo inicial que pode ser utilizado pelos engenheiros de software quando estiverem definindo conceitos específicos do domínio. Neste sentido, algumas pesquisas tem proposto uma correlação entre os conceitos ontológicos e os utilizados em modelagem de dados do tipo entidade-relacionamento (Wand, 1996) e para a modelagem de objetos (Parson *et al.* 1997). A investigação do domínio permite, ainda, a identificação de restrições (axiomas) entre os conceitos definidos na teoria do Domínio, que podem funcionar como restrições de integridade dos dados modelados.

3. A Estação TABA e a Construção de ADSOD

A Estação TABA (Rocha *et al*. 1990) é um meta-ambiente capaz de gerar, através de instanciação, ambientes de desenvolvimento de software adequados às particularidades de processos de desenvolvimento e de projetos específicos. Desta forma, a Estação TABA possui facilidades para a definição de processos, métodos e ferramentas CASE a serem utilizadas no processo de desenvolvimento. Estes elementos estão organizados em um modelo de integração do ambiente, permitindo que diferentes ambientes sejam definidos e instanciados.

A estrutura da Estação TABA (Travassos, 1994) foi definida como um conjunto de componentes integrados que possuem controle sobre a sua existência, suas informações, estados e funcionalidades básicas associadas como por exemplo, o controle de processos,

interface com o usuário, reutilização, cooperação, suporte inteligente, entre outros. A utilização, em conjunto, destes componentes permite a integração de ferramentas ao ambiente e provê recursos para a incorporação de ferramentas externas. A utilização desses componentes define, ainda, a filosofia de integração da Estação TABA e dos seus ambientes instanciados através de quatro tipos de integração: *(i)* apresentação e interação, possibilitando a homogeneização das formas de apresentação das informações e das técnicas de interação com o usuário, *(ii)* a integração de gerenciamento e controle, provendo serviços e funcionalidades que permitam o funcionamento de forma organizada ao longo do processo de desenvolvimento, *(iii)* integração de dados, estabelecendo a forma como as ferramentas da Estação realizarão o tratamento das informações, e, *(iv)* integração do conhecimento, que torna possível os serviços básicos de armazenamento, gerenciamento e utilização do conhecimento descrito e adquirido ao longo do processo de desenvolvimento. A integração de conhecimento se dá através da utilização de servidores do conhecimento (Falbo *et al*. 1999a), no qual o conhecimento é modelado para reuso através de ontologias, sendo disponibilizado um conjunto de componentes que podem ser usados por várias ferramentas a serem desenvolvidas.

Para ser possível instanciar um ADSOD, alguns requisitos foram adicionados à Estação TABA. Entre eles, destacamos: *(i)* apoiar a definição da Teoria do Domínio; *(ii)* apoiar a descrição de tarefas que serão utilizadas em diferentes Teorias do Domínio; *(iii)* permitir a evolução de conceitos do domínio e tarefas a medida que os ADSOD sejam utilizados e novos ADSOD sejam construídos para um mesmo domínio; *(iv)* armazenar e indexar projetos desenvolvidos no domínio, associando-os aos conceitos da Teoria do Domínio; e, *(v)* permitir a definição de um processo que possua a atividade de investigação do domínio. Assim sendo, com esta extensão da Estação TABA, será possível instanciar ADS a partir do processo de software, de características do domínio ou de ambos.

Para atender a estes requisitos, novas facilidades foram definidas para a estação TABA. Um editor de Teoria do Domínio (EDITED) é utilizado para permitir a introdução do conhecimento do domínio no ambiente. Este editor utiliza um meta-modelo de classes para organizar o conhecimento de forma que os objetos do domínio deste modelo tornam-se classes nos ambientes instanciados. Desta forma, todos os conceitos e relações do domínio definidos são organizados de acordo com a estrutura definida pelos servidores de conhecimento, disponibilizando módulos de conhecimento do domínio passíveis de serem utilizados.

A descrição de tarefas, nesta primeira versão, é feita através de simples documentos que descrevem as principais características da tarefa, fornecendo referências bibliográficas para a mesma. Um trabalho mais completo envolve a descrição através de ontologias de tarefa (Van Heijst *et al.* 1997).

A evolução dos conceitos é apoiada basicamente através da instanciação da Teoria do Domínio a medida que novas aplicações são desenvolvidas. No entanto, novos conceitos podem surgir, devendo ser acrescentados à Teoria do Domínio. Esta atividade é realizada manualmente pelo engenheiro de software e consiste na inclusão de novos conceitos, mas não da alteração de conceitos já existentes. Dessa forma, aplicações já existentes manterão sua integridade.

Finalmente, os processos de desenvolvimento a serem definidos para instanciar ambientes devem considerar a atividade de investigação do domínio nas principais atividades de desenvolvimento (conforme definido na seção 2). Em um mesmo domínio, diferentes tipos de

software podem ser desenvolvidos e, portanto, diferentes processos devem ser definidos, implicando na criação de vários ambientes de acordo com o tipo de software. Ao definirmos os diferentes processos para cada tipo de software, dentro de uma mesma instituição, as atividades de desenvolvimento que o compõem devem ser similares e padronizadas. A padronização do processo de desenvolvimento tem tido bastante ênfase nas recentes pesquisas sobre processo de software (ISO/IEC 12207 1995, Emam *et al.* 1998). Para que seja possível esta padronização, consideramos necessária a definição de um Processo Padrão para a organização, que sirva de base para a definição dos diferentes processos de desenvolvimento. Processo padrão é definido pelo SPICE (*Software Process Improvement and Capability dEtermination*) (Emam *et al.* 1998) como a definição operacional do processo básico que guia o estabelecimento de um processo comum dentro da organização. O processo padrão descreve os elementos fundamentais que devem ser incorporados em qualquer processo definido.

A partir do Processo Padrão definido para a organização, diferentes processos de software podem ser especializados de acordo com as características de cada tipo de software. Um tipo de software refere-se a um conjunto de software que utiliza uma determinada tecnologia de desenvolvimento (como sistemas baseados e conhecimento, sistemas convencionais, etc.) seguindo um paradigma específico (como orientado objeto, estruturado, etc.). Neste momento, novas atividades podem, também, ser definidas e incluídas no processo especializado. Um determinado tipo de software pode ser desenvolvido através de diferentes modelos ciclos de vida e utilizando-se diferentes métodos e ferramentas. A adoção de um determinado método pode, ainda, implicar na inclusão de atividades específicas ao método. No entanto, todos os elementos básicos definidos no Processo Padrão deverão sempre estar presentes nos processos especializados. Para ser utilizado em um projeto específico, o processo especializado mais adequado para um determinado tipo de software deve ser instanciado para atender às características do projeto específico, devendo-se considerar o tamanho e complexidade do produto, as características da equipe de desenvolvimento, a expectativa de vida útil do software e demais características do projeto.

A Figura 1 mostra o esquema global de definição de um processo de software a ser utilizado na Estação TABA. O processo de software instanciado final é, então, utilizado para instanciar o ADS na Estação. Seguindo esse esquema, para a construção de ADSOD, a atividade de investigação do domínio deve ser introduzida como atividade na definição do Processo Padrão. Desta forma, garantimos que todos os processos especializados e, posteriormente, instanciados vão considerar a orientação ao domínio no desenvolvimento do software específico (ROCHA *et al*. 1999).

Três ferramentas apoiam a definição de um processo de software na Estação TABA: EDIT-PRO, para simples edição de um processo já previamente definido pelo engenheiro de software; ASSIST-PRO (Falbo *et al.* 1999b), que fornece assistência inteligente na escolha do ciclo de vida e atividades mais adequadas ao sistema a ser desenvolvido; e, DEF-PRO (Machado *et al.* 1999), uma ferramenta mais completa que apoia desde a definição do processo padrão até a definição do processo instanciado, considerando não apenas a norma ISO/IEC 12207 (1998), modelos de maturidade CMM (Paulk *et al.* 1995) e SPICE (Emam *et al.* 1998), mas também as características da organização, tipo de software, tecnologia de desenvolvimento e o tipo de ambiente que se deseja instanciar (orientados ao domínio ou não). Todas essas ferramentas utilizam o conhecimento sobre processo de software já armazenado na Estação no que se refere a atividades, ferramentas, métodos, procedimentos e ciclos de vida (Falbo *et al.* 1998).

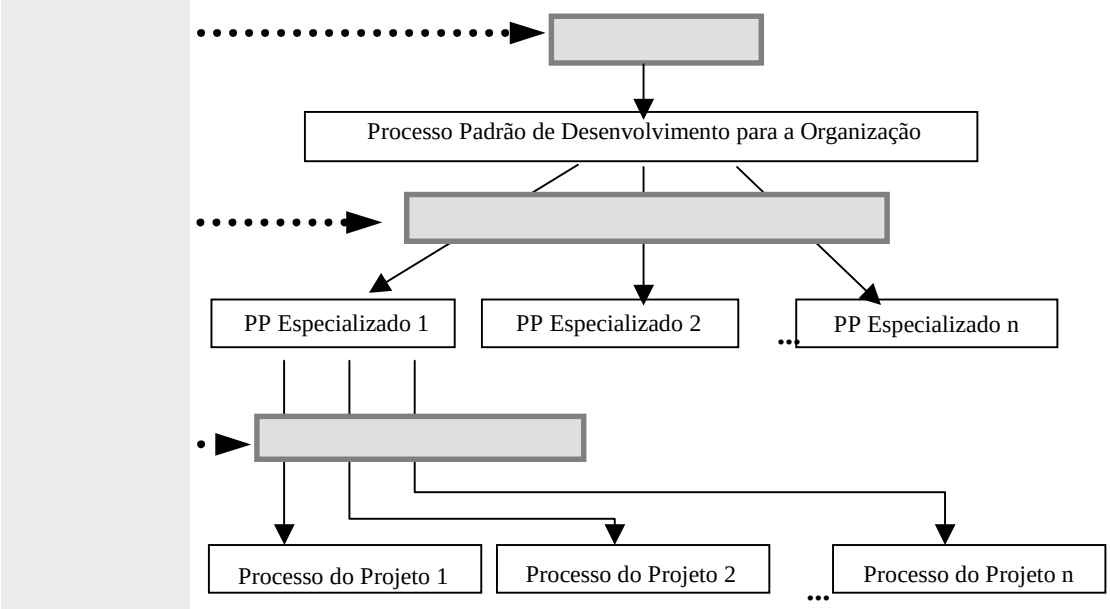

Figura 1 - Esquema de Definição de Processo de Software para um ADS na Estação TABA

Após definido um processo e definida uma Teoria do Domínio, é possível, então, instanciar um ADSOD. A instanciação de ADSOD implica na criação e compilação do código referente à definição das classes que possuem informação sobre o processo de software e as classes do domínio (que compõe o servidor de conhecimento do domínio específico) geradas por EDITED. Para isto, foi definido um *framework* de ambiente que é compilado e copiado para um diretório específico, permitindo que o ADSOD instanciado seja independente da Estação TABA.

A primeira versão da Estação TABA foi implementada em Eiffel na plataforma Unix da Sun Workstation. Com o projeto de ADSOD, foi construída uma nova versão da Estação TABA utilizando C++ e a plataforma Windows, tornando-a mais portável para permitir experimentos de uso dos ambientes gerados.

3. NETUNO: um ADSOD para o domínio de Acústica Submarina

NETUNO é um ADSOD para Acústica Submarina desenvolvido de acordo com as particularidades do GAS/IPqM (Grupo de Acústica Submarina do Instituto de Pesquisas da Marinha). Conforme definido nas seções anteriores, para definir e construir um ADSOD, é necessário definir a Teoria do Domínio e o processo de software específico, o que implica na definição de um processo padrão para a instituição.

A Acústica Submarina (Urick, 1983) é a parte da Física que estuda a propagação do som no meio líquido, mais precisamente na água do mar. Para descrevê-la, usamos duas subteorias: a de Ambiente Acústico e a de Teoria de Propagação. A subteoria de Teoria da Propagação compreende os conceitos de Modelo de Propagação e Perda, que modelam matematicamente a propagação do som e a atenuação que a onda sonora sofre ao se propagar no meio acústico. A Figura 2 representa, de forma gráfica e resumida, a subteoria de Ambiente Acústico. Uma série de axiomas fazem, também, parte da ontologia e foram descritos em linguagem natural, sendo responsáveis pela semântica dos termos neste domínio.

São exemplos desses axiomas: *(i)* todo sonar recebe som; *(ii)* o aquecimento solar provoca o aparecimento de um duto superficial; *(iii)* dentro do duto, a perda na propagação é diminuída

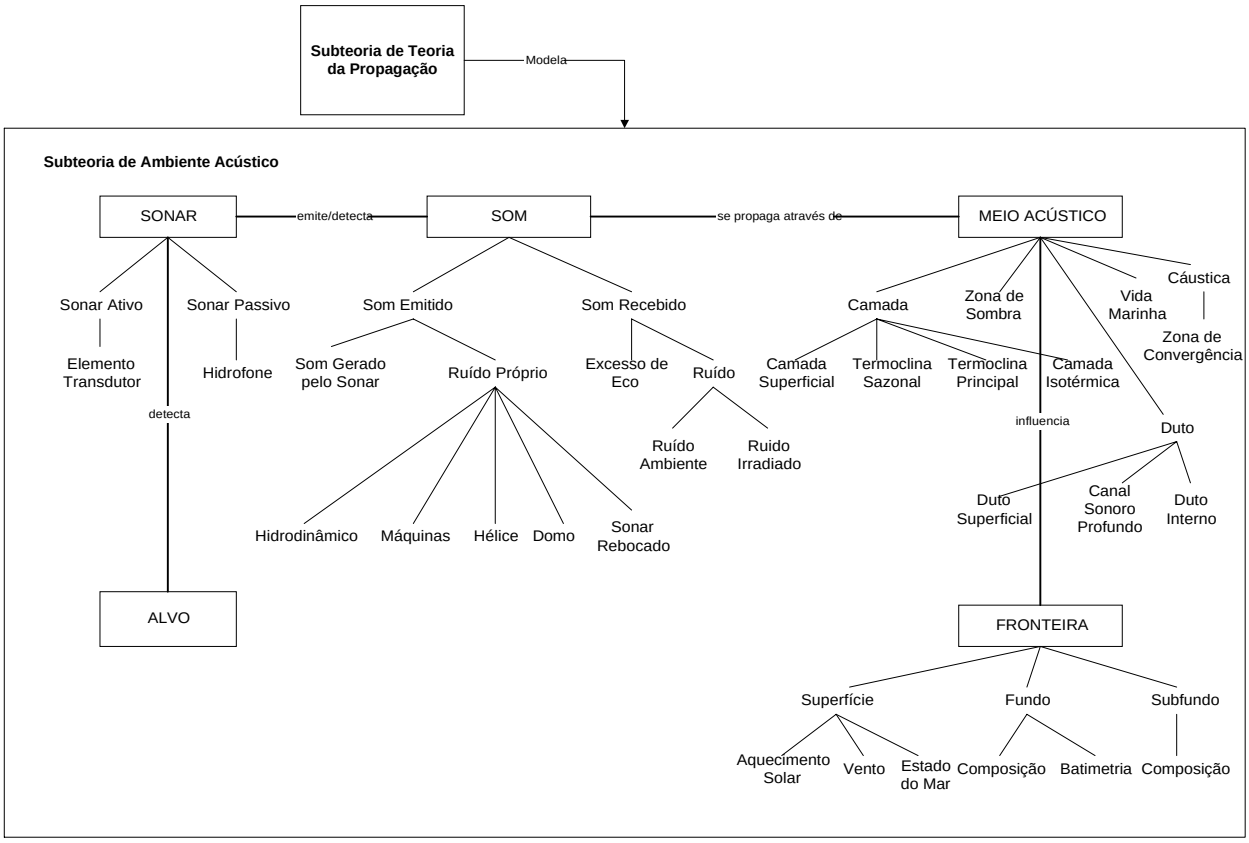

Figura 2: Teoria do domínio – sub-teoria de Ambiente Acústico

Tendo em vista as características particulares GAS/IPqM, organizamos as atividades do processo padrão em quatro etapas principais (Figura 3): *contratação*, com todas as atividades necessárias para estabelecer o contrato; *pesquisa*, que se refere ao estudo do problema e está presente ao longo de todas as outras etapas; *desenvolvimento*, com as atividades necessárias para se definir e construir o software específico; e *manutenção*, com as atividades referentes às modificações que se tornem necessárias. A Tabela 1 apresenta as atividades do processo padrão definido e seus objetivos principais. Cada uma dessas atividades foi descrita de acordo com suas sub-atividades, produtos gerados e pessoal responsável pela realização dessas atividades. Um aspecto importante nessa definição foi a identificação clara dos diferentes grupos que participam do processo de software: pesquisadores que só se envolvem com tarefas relacionadas à pesquisa, pesquisadores que também atuam como analistas/programadores e especialistas em informática.

Especializamos o processo padrão do GAS/IPqM para definir um processo de desenvolvimento para software científico. Esse processo foi instanciado para um projeto específico. Com todas as características definidas, foi, então, utilizada a Estação TABA para construção do ambiente. Desta forma, a Teoria do Domínio foi introduzida na Estação TABA através do EDITED (Figura 4), o processo foi editado utilizando o EDIT-PRO (Figura 5) e, então, o ADSOD específico (NETUNO) foi gerado (Figura 6).

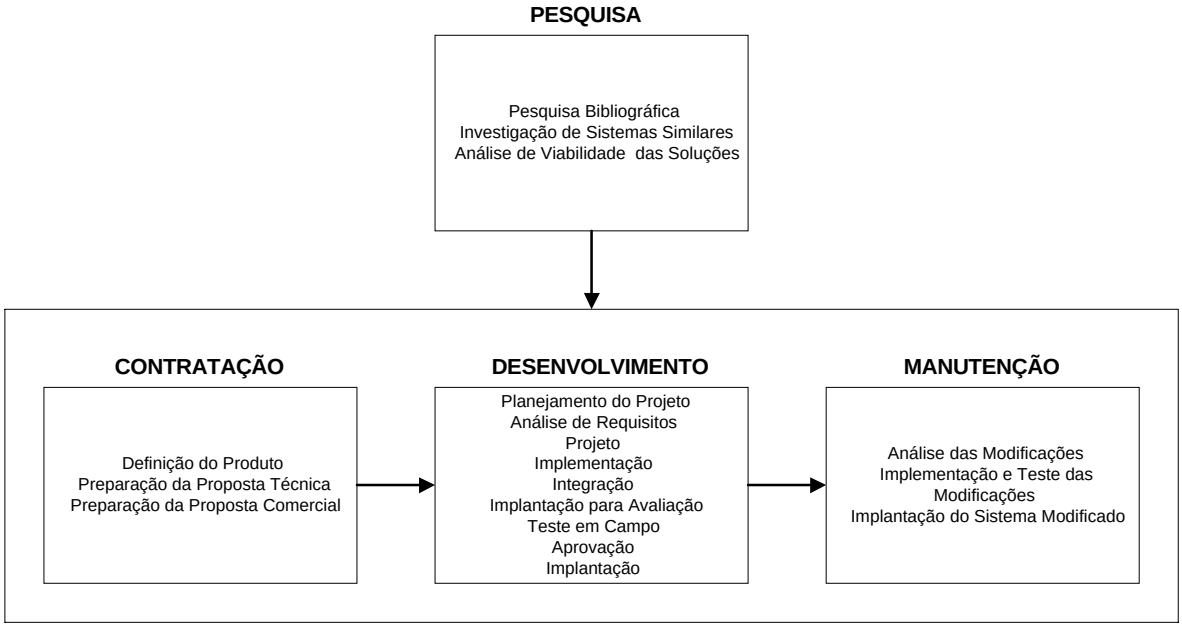

Figura 3: Etapas do processo padrão para o Grupo de Sonar

ATIVIDADE	OBJETIVO
Definição do Produto	Identificar o problema e definir o sistema que seja a solução para o problema
Preparação da Proposta Técnica	Preparar uma proposta técnica que será estudada pelo comprador
Preparação da Propota Comercial	Preparar uma proposta técnico-comercial que será aprovada e assinada
Pesquisa Bibliográfica	Estudar modelos físicos e matemáticos de interesse
Investigação de Sistemas Similares	Estudar sistemas similares com o objetivo de identificar modelos e algoritmos
Análise de Viabilidade das Soluções	Analisar a viabilidade de cada solução proposta
Planejamento do Projeto	Elaborar a primeira versão do Plano de Projeto
Análise de Requisitos	Analisar, modelar e avaliar os requisitos do sistema a ser desenvolvido
Projeto	Elaborar um modelo físico do sistema a partir da modelagem conceitual
Implementação	Implementar os itens da arquitetura na linguagem de programação escolhida
Integração	Integrar módulos e sistemas para se obter o produto de software final, já testado
Implantação para Avaliação	Implantar o sistema no ambiente real para que seja avaliado
Teste em Campo	Colocar o sistema em uso no ambiente real com acompanhamento especializado
Aprovação	Aprovar os resultados obtidos pelo sistema durante seu teste em campo
Análise de Modificações	Analisar os problemas ocorridos durante a operação e uso do sistema
Implementação e Teste das Modificações	Codificar a alteração proposta
Implantação do Sistema Modificado	Implantar o sistema modificado no ambiente de uso

Tabela 1: Atividades do processo Padrão

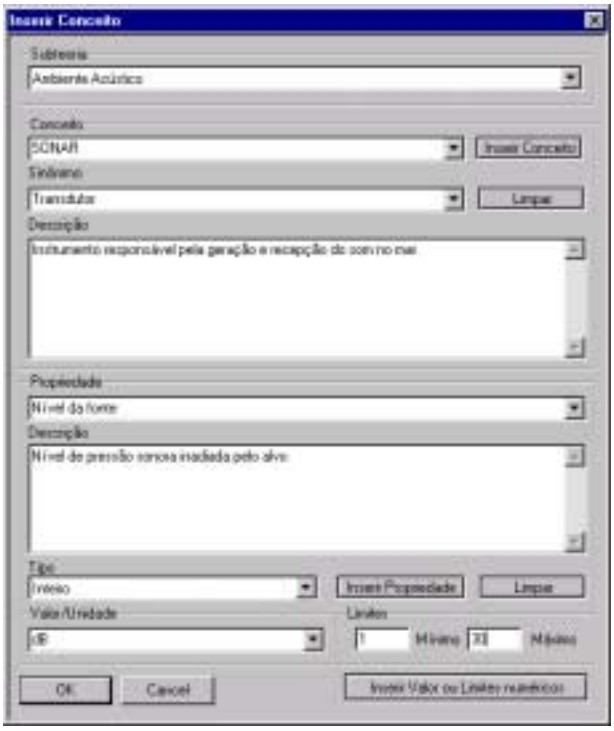

Figura 4 – Definição da Teoria do Domínio

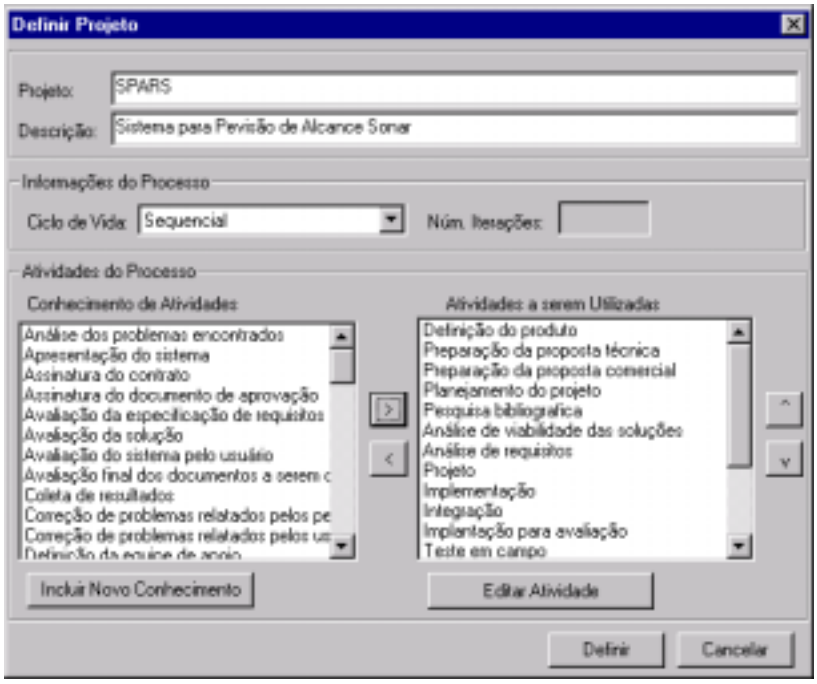

Figura 5 – Definição do Processo de Software

Para apoiar a atividade de investigação do domínio, foi definida uma ferramenta específica para o domínio denominada NAVEGUE. Esta ferramenta é um *browser* específico para o domínio, que orienta a pesquisa a partir do enfoque nos modelos de propagação da onda. Grande parte do trabalho realizado no GAS/IPqM gira em torno da resolução da equação da onda sonora (uma onda mecânica), utilizando-se, para tal, diversos modelos

matemáticos. A ferramenta orienta o desenvolvedor através dos modelos, associando-os aos seus principais conceitos definidos na teoria do domínio.

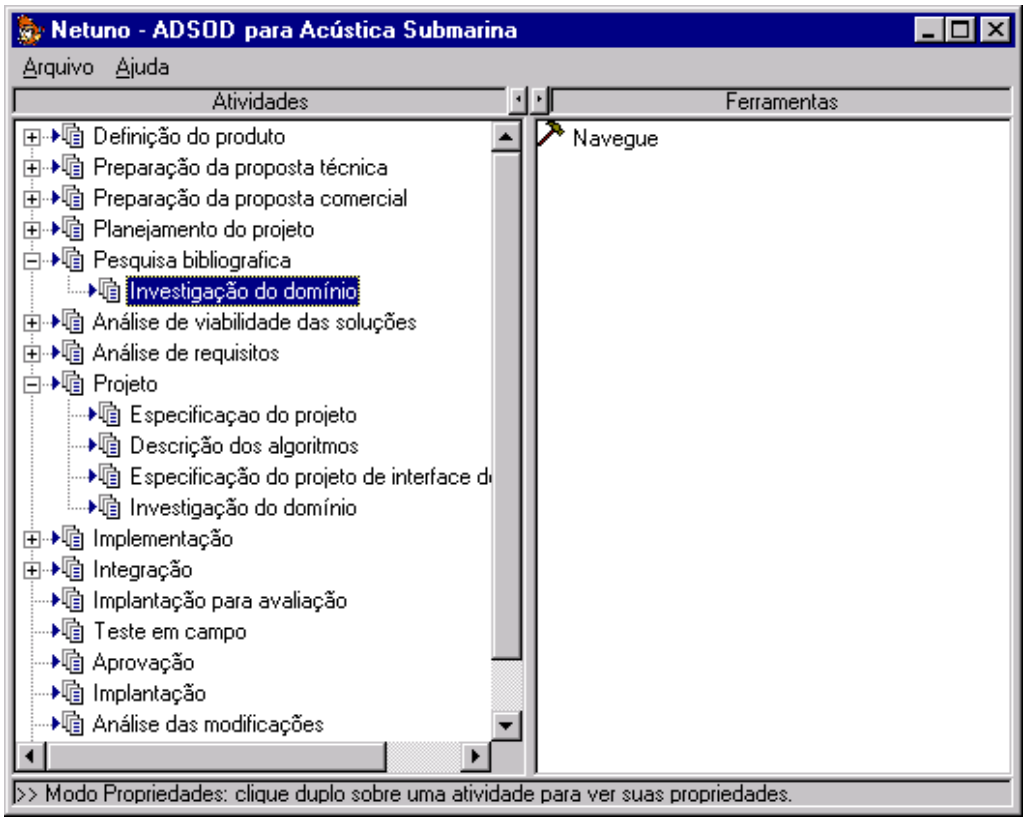

Figura 6 – NETUNO: um ADSOD para Acústica Submarina

5. Conclusão e Trabalhos Futuros

O uso do conhecimento do domínio durante o processo de desenvolvimento de sistemas é um fator diferencial para desenvolvedores que não conhecem o domínio ou não têm experiência em desenvolver software para aquele domínio. Baseados nessa idéia, definimos Ambientes de Desenvolvimento de Software Orientados a Domínio para apoiar engenheiros de software no desenvolvimento de sistemas orientados por um processo de desenvolvimento que enfoca o entendimento do domínio através de uma atividade específica que utiliza o conhecimento do domínio introduzido no ambiente.

Para apoiar a construção de ADSOD utilizamos a infra-estrutura da Estação TABA, um meta-ambiente amplamente utilizado nos projetos de pesquisa na COPPE/UFRJ. Neste artigo, descrevemos como ADSOD são construídos utilizando a Estação TABA e apresentamos NETUNO, um ADSOD específico para o domínio de Acústica Submarina. Além do ADSOD NETUNO, a Estação TABA também foi utilizada para construir CORDIS, um ADSOD para Cardiologia (Oliveira *et al.* 1999b).

Nosso trabalho atual tem se concentrado em definir um gerenciamento completo do conhecimento, já que é reconhecido que as organizações que desenvolvem software lidam de forma intensa com diversos tipos de conhecimento. Nos ADSOD instanciados a partir do modelo apresentado neste artigo, o conhecimento do domínio já estará disponível para a organização em uma representação explícita e sem ambigüidades. No entanto, o conhecimento necessário a uma organização de desenvolvimento de software extrapola o

conhecimento do domínio, envolvendo normas organizacionais, melhores práticas, relatos de experiências, entre outros. Desta forma, o objetivo é dar continuidade ao trabalho realizado com ADSOD, inserindo em tais ambientes outras formas de conhecimento essenciais à organização, dando atenção especial às experiências de desenvolvimentos anteriores. Através da gerência do conhecimento (O´Leary, 1998, Skyrme, 1998, Kouwenhoven, 1998, Abecker et al. 1998, Markkula, 1999), pretendemos instanciar ambientes de desenvolvimento de software orientados ao domínio e, também, a uma organização.

Estamos, portanto, abrindo uma ampla linha de pesquisa em ambientes de desenvolvimento de software com enfoque central no conhecimento. O objetivo é aumentar a eficiência da organização no desenvolvimento de software, melhorando a maneira como ela administra seu conhecimento, ou seja, capacitando-a a reter conhecimento especializado, aumentar a disponibilidade de especialistas e melhorar a qualidade das decisões e das soluções de problemas.

Agradecimentos

Agradecemos ao ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela apoio financeiro ao projeto Ambientes de Desenvolvimento de Software Orientado ao Domínio.

Referências Bibliográficas

ABECKER, A., BERNADI, A., HINKELMANN, K. et al, 1998, “Torward a Technology for Organizational Memories”, *IEEE Intelligent Systems*, v.13, n.3, pp.40-48.

BROWN, A., EARL, A., MCDERMID, J., 1992, *Software Engineering Environments: Automated Support for Software Engineering*, England, McGraw-Hill Book Company.

EMAM, K. E., DROUIN, J. AND MELO W., 1998, SPICE – The Theory and Practice of Software Process Improvement and Capability Determination, IEEE Computer Society Press.

FALBO, R. A.; MENEZES, C., and ROCHA, A. R. C., 1998, “Integração do Conhecimento Sobre Processos de Software em um Ambiente de Desenvolvimento”; *In: proceedings of IX Conferência Internacional de Tecnologia de Software*, Paraná, Brasil, Junho, 1998.

FALBO, R., MENEZES, C., ROCHA, A. R., 1999b, “Assist-Pro: Um Assistente Inteligente para Apoiar a Definição de Processos de Software”, In: *Anais do XIII Simpósio Brasileiro de Engenharia de Software*, pp. 147-162, Florianópolis, Brasil, Out.

FALBO, R., MENEZES, C., ROCHA, C., 1999a, “Using Knowledge Servers to Promote Knowledge Integration in Software Engineering Environments”. In: *Proceedings of the 11th Software Engineering an Knowledge Engineering Conference*, pp. 170-175, Kaiserslautern, Alemanha, Jun.

GRUBER, T. R., 1995; “Toward Principles for the Design of Ontologies used for Knowledge Sharing”, *International Journal Human-Computer Studies*, No 43, pp 907-928.

ISO/IEC 12207. International Standard – Information Technology - Software Process Life Cycle, 1995.

KOUWENHOVEN, T., 1998, “Reengineering for Learning”, *SIGGROUP Bulletin*, v.19, n.1, pp.39-45.

MACHADO, L.F.C., ROCHA, A.R., 1999, “Modelo para Definição, Especialização e Instanciação de Processos de Software”, In: *Anais do Workshop de Teses em*

Engenharia de Software - XIII Simpósio Brasileiro de Engenharia de Software, pp. 43-47, Florianópolis, Brasil, Out.

MARKKULA, M., 1999, "Knowledge Management in Software Engineering Projects", In: *Proceedings of the 11th Software Engineering an Knowledge Engineering Conference*, pp. 16-19, Kaiserslautern, Alemanha, Jun.

O´LEARY, D. E., 1998, "Enterprise Knowledge Management", *IEEE Intelligent Systems*, v.13, n.3, pp.54-61.

OLIVEIRA, K. M., ROCHA, A. R., TRAVASSOS, G. H. *et al.*, 1999a, "Using Domain-Knowledge in Software Development Environments", In: *Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering*, pp. 180-187, Kaiserlauther, Alemanha, Jun.

OLIVEIRA, K. M., ROCHA, A. R., TRAVASSOS, G. H. *et al.*, 1999b, "CORDIS: Assistência Automatizada no Desenvolvimento de Software em Cardiologia", In: *Simposio en Informática y Salud - 28 Jornadas Argentinas de Informática e Investigación Operativa, pp 34-48*, Buenos Aires, Argentina, Set.

PARSONS, J. E WAND, Y., 1997, "Using Objects for System Analysis", *Communications of the ACM*, v. 40, n. 12 (Dec), pp. 104-110

PAULK, M. C., WEBER, C. V., CURTIS, B., CHRISSIS, M. B. (eds), 1995, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Carnegie Mellon University, Software Engineering Institute, Addison-Wesley Longman Inc.

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

ROCHA, A. R., MAIDANTCHIK, *C.*, OLIVEIRA *et al.*, 1999, *Experience in Defining, Using and Improving Software Process*, Publicações Técnicas, COPPE/UFRJ, ES 507/99, Rio de Janeito, Brasil.

SKYRME, D. J., 1998, "Knowledge Management Solutions - The IT Contribution", *SIGGROUP Bulletin*, v.19, n.1, pp.34-39.

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

TRAVASSOS, G.H., ROCHA, A.R.C.da, 1994, "O Modelo de Integração de Ferramentas da Estação TABA"; In: *Proceedings of JAIIO*, Buenos Aires, Argentina.

URICK, R.J. , 1983, *Principles of Underwater Sound*., 3 ed., New York, McGaw-Hill.

VAN HEIJST, G. VAN, SCHREIBER, A TH., WIELENGA, B. J., 1997, "Using Explicit Ontologies in KBS Development", *International Journal of Human-Computer Studies*, vol 45, No 2/3; pp 183-292.

WAND, Y, 1996, "Ontology as a foundation for meta-modelling and method engineering", *Information and Sofware Tecnhology*, No 38, 281-287.

WERNECK, V. M., ROCHA, A. R. C. DA, RABELO, A. *et al.*, 1995, "Ambiente para Desenvolvimento de Sistemas Baseados em Conhecimento"; In: *Proceedings of XV Congresso da Sociedade Brasileira de Computação/XXI Congresso Latinoamericano de Informática*, Canela, Brasil, Jul.

WERNER, C. M. L., TRAVASSOS, G. H., DA ROCHA, A. C. *et al.*, 1997, "Memphis: A Reuse Based O. O. Software Development Environment", In: *Proceedings of TOOLS*, Beijing, China, Sep.