Melhoria de Processos de Software e Evolução de Ambientes de Desenvolvimento de Software com base no Conhecimento do Domínio e na Cultura Organizacional

Karina Villela[1,2], Gleison Santos[1], Guilherme H. Travassos[1], Ana Regina Rocha[1]

[1] Universidade Federal do Rio de Janeiro
Coordenação dos Programas de Pós-Graduação em Engenharia
Caixa Postal: 68511, CEP: 21945-970, Rio de Janeiro, RJ
{kvillela,gleison,ght,darocha}@cos.ufrj.br

[2] Fundação Bahiana de Cardiologia
Rua Augusto Viana s/n, CEP: 40140-060, Salvador, BA

Resumo

Em busca da melhoria da qualidade de seus produtos e do aumento da produtividade de suas equipes, organizações têm prestado mais atenção em seus processos de software. Este esforço tem envolvido a definição de um processo para a organização e o estabelecimento de metas de melhoria, levando em consideração boas práticas de engenharia de software. No entanto, nossa experiência em definir processos de software para várias organizações tem mostrado que o conhecimento sobre o domínio e a cultura organizacional são aspectos importantes para a melhoria dos processos de software. Desta forma, descrevemos como nossa abordagem para a definição de processos de software e para suporte automatizado ao desenvolvimento de software evoluiu para tratar estes aspectos. O resultado foi o surgimento de duas novas famílias de ambientes de desenvolvimento de software: Ambientes de Desenvolvimento de Software Orientados a Domínio (ADSOD) e Ambientes de Desenvolvimento de Software Orientados a Organização (ADSOrg). Este artigo apresenta, ainda, um resumo das lições que temos aprendido na definição e melhoria de processos de software.

Palavras-Chave:

Qualidade de Processo, Melhoria de Processo de Software, Processo Padrão de Software, Ambiente de Desenvolvimento de Software, Orientação a Domínio, Gestão de Conhecimento.

Abstract

The search for quality and productivity improvements has encouraged organizations to focus on their software processes. This effort comprises defining a software process and setting improvement goals by taking into account good software engineering practices. However, our experience on defining software processes for several organizations has shown us that domain knowledge and organizational culture are important aspects to consider in software process improvement. This way we describe how our approach to define software processes and to provide automated support for software development has evolved to deal with such aspects. The result has been the appearance of two new families of software development environments: Domain-Oriented Software Development Environments (DOSDE) and Enterprise-Oriented Software Development Environments (EOSDE). This article also presents a short description of the lessons we have been learning from the definition and improvement of software processes.

Keywords:

Process Quality, Software Process Improvement, Standard Software Process, Software Development Environment, Domain-orientation, Knowledge Management.

1. Introdução

A definição, o uso e a melhoria contínua de processos de software são metas importantes para organizações que desenvolvem software. O esforço realizado para alcançar estas metas, geralmente, contempla os aspectos puramente técnicos relacionados às práticas e métodos de engenharia de software, sem considerar muito as restrições do ambiente de trabalho, a cultura da organização ou o conhecimento e a experiência das equipes de desenvolvimento. Defendemos que as características particulares de cada organização e de seus grupos de trabalho sejam consideradas durante a definição de processos de software, de forma que estes sejam adequados ao contexto em que serão utilizados.

Temos definido, ao longo dos últimos anos, vários processos de software para organizações de diferentes tipos e domínios de atuação. Constatamos, entretanto, que a definição e execução de um processo de software não são suficientes para garantir o sucesso de um projeto. Mesmo considerando os padrões internacionais, os modelos de maturidade, e os modelos de processo, métodos e ferramentas da engenharia de software, a definição de um processo de software pode não estar adequada a um contexto específico. Dificuldades com alta rotatividade de pessoal, equipes geograficamente distribuídas e falta de conhecimento do domínio por parte das equipes de software não são problemas técnicos, mas podem determinar o sucesso ou fracasso de um projeto. Tais dificuldades requerem soluções de engenharia de software para que o processo de software seja adequado, bem aceito e utilizado com sucesso.

Neste artigo, apresentamos, como motivação para as idéias propostas, nossa experiência envolvendo a definição, uso e melhoria de processos de software no domínio de cardiologia. O trabalho neste domínio é parte de um convênio iniciado em 1994 entre a Universidade Federal de Rio de Janeiro (COPPE/UFRJ) e a Fundação Bahiana de Cardiologia (FBC). O principal objetivo do convênio é apoiar a FBC no desenvolvimento de diferentes tipos de software médico.

Inicialmente, um processo de software foi definido para o desenvolvimento de um sistema especialista. Num momento posterior, foi definido o processo padrão da organização, estabelecendo os elementos fundamentais que devem ser incorporados em qualquer processo de software da organização [4]. Este processo padrão foi, então, especializado e instanciado de acordo com as necessidades específicas dos projetos conduzidos na organização. Com esta abordagem, a definição dos diferentes processos de software dentro de uma mesma organização é padronizada para que o desenvolvimento e a manutenção de software se tornem mais gerenciáveis. Consideramos, entretanto, que o processo padrão não deve só considerar boas práticas de desenvolvimento, mas também os problemas cruciais de desenvolvimento de software existentes na organização. Este último aspecto tem se mostrado decisivo para o sucesso de um processo de software.

Depois de alguns anos de parceria, percebemos que a falta de conhecimento sobre o domínio de cardiologia e a falta de conhecimento sobre a própria organização por parte das equipes de software eram os problemas cruciais do desenvolvimento de software na organização. Estes problemas, identificados logo nos primeiros anos de parceria, continuam sendo cruciais para o desenvolvimento de software na organização, devido à alta rotatividade de pessoal, uma característica comum em ambiente acadêmico de pesquisa e desenvolvimento. Em uma abordagem evolutiva, nós tratamos primeiro da falta de conhecimento sobre o domínio. Para isto, foi incluída no processo padrão uma atividade

chamada investigação do domínio e construído o que denominamos de Ambiente de Desenvolvimento de Software Orientado a Domínio (ADSOD) [11,12]. Entretanto, a abordagem ainda não estava completa. O passo seguinte foi a evolução da definição dos processos de software e do ambiente de desenvolvimento de software, de modo a contemplar o conhecimento organizacional, o que deu origem aos Ambientes de Desenvolvimento de Software Orientados à Organização (ADSOrg) [19].

O restante deste artigo é organizado em 5 seções. A seção 2 apresenta a primeira tentativa de definir um processo de software na FBC, enfatizando o problema percebido após a melhoria deste processo. As seções 3 e 4 descrevem propostas evolutivas de melhoria e apoio automatizado de processos de software. Estas propostas lidam com os problemas cruciais do desenvolvimento de software identificados na organização. Na seção 5, é feito um resumo das lições aprendidas que influenciaram a abordagem adotada para definição de processos. Finalmente, na seção 6, são feitas considerações finais sobre as propostas apresentadas e sobre o uso dos resultados em outras organizações. Trabalhos futuros também são mencionados.

2. Um Processo de Software para Sistemas Baseados em Conhecimento

Os projetos de software conduzidos na FBC envolvem pesquisa e normalmente são desenvolvidos por grupos pequenos de pesquisadores/desenvolvedores. As características destes grupos de trabalho têm mudado ao longo do tempo, mas um aspecto importante, a alta rotatividade dos desenvolvedores, está sempre presente, uma vez que os projetos são patrocinados pelo governo através da concessão de bolsas temporárias.

A primeira definição de um processo de software na FBC foi feita em 1994, com o início de um projeto para o desenvolvimento de um sistema especialista para apoio ao diagnóstico de enfarto agudo do miocárdio (SEC) [13]. A primeira atividade, neste projeto, foi elaborar um documento que descrevesse um processo de software baseado na norma ISO 9000-3 [6], nas características de sistemas especialistas e nos aspectos específicos do projeto. Este documento definiu um ciclo de vida evolutivo, os métodos e ferramentas a serem utilizados para a construção do produto, a documentação a ser produzida, os atributos de qualidade do software, e os procedimentos e métodos para garantia da qualidade do software e para gerência do projeto. O método de análise adotado foi o proposto pelo KADS [18].

A garantia da qualidade do software foi a principal preocupação durante o desenvolvimento do SEC, uma vez que o sistema lida com vidas humanas. Desta forma, uma característica muito importante do processo foi os procedimentos de garantia de qualidade utilizados ao longo do desenvolvimento do software. Diferenças entre software convencional e sistemas especialistas determinaram as peculiaridades da validação do sistema. Inspeções foram realizadas desde as fases iniciais, permitindo que erros fossem descobertos na mesma fase em que eram cometidos, evitando, assim, que se propagassem. Isto minimizou a quantidade de re-trabalho ao longo do processo [14].

O processo foi utilizado para o desenvolvimento da primeira versão do SEC, que terminou com uma avaliação do próprio processo de software. Esta avaliação resultou na melhoria do processo de software, o que englobou revisão do ciclo de vida, definição de extensões para o KADS e uso de diferentes técnicas para refinamento e validação do sistema [22]. Porém, a reformulação do processo de software não foi o bastante. A equipe de desenvolvimento não tinha nenhuma experiência em desenvolver software para o domínio de cardiologia e, embora

não tenha sido definido explicitamente no processo de software, foi necessário executar muitas atividades para entender os conceitos necessários de cardiologia e interagir adequadamente com os especialistas do domínio (os cardiologistas). Neste momento, foi identificado que a falta de conhecimento sobre o domínio de cardiologia por parte das equipes de software era um problema crucial para o desenvolvimento de software na FBC. Especialistas do domínio (i.e. cardiologistas) viam o processo de aquisição de conhecimento e levantamento de requisitos como repetitivo e exaustivo, pois precisavam explicar conceitos básicos de assistência médica e de cardiologia para os desenvolvedores. Este problema tornou-se ainda mais grave devido à alta rotatividade de pessoal nas equipes de software.

3. Melhoria e Apoio Automatizado de Processos de Software baseados em Orientação a Domínio

A experiência descrita na seção anterior e outras experiências similares nos permitiram constatar a importância de se considerar, explicitamente no processo de software, atividades que permitam aos desenvolvedores adquirir o conhecimento do domínio necessário à execução de suas atividades. Esta constatação repercutiu no nosso trabalho de pesquisa em Ambientes de Desenvolvimento de Software (ADS) e nos levou a formular a seguinte hipótese:

Ambientes de Desenvolvimento de Software apóiam melhor o desenvolvimento e a manutenção de um produto de software se forem capazes de fornecer conhecimento do domínio aos desenvolvedores.

Na primeira subseção a seguir, descrevemos a abordagem para definição de processos com orientação a domínio que adotamos na FBC. Na segunda subseção, apresentamos o apoio automatizado que definimos para a abordagem através da definição de Ambientes de Desenvolvimento de Software Orientados a Domínio (ADSOD).

3.1 Orientação a Domínio na Definição de Processos de Software

A partir da experiência obtida na definição de processos para projetos específicos da FBC, um processo padrão foi definido para a organização em 1999, levando em consideração os vários processos de software já em uso, a ISO/IEC 12207 [5] e a necessidade de inserir orientação a domínio no processo de software. Para atender este último ponto, uma atividade nova, denominada investigação do domínio, foi definida e incluída no processo padrão [11,12]. Essa atividade também foi incluída como parte de outras atividades nas quais é importante considerar o conhecimento do domínio, tais como: análise de requisitos e projeto. O conhecimento do domínio de cardiologia, utilizado para apoiar os desenvolvedores de software na execução de suas atividades, foi organizado como um conjunto de ontologias [3].

O processo padrão definido para a FBC contém atividades comuns a qualquer processo de desenvolvimento de software e outras que são particulares da FBC. Entre as últimas, temos: pesquisa de sistemas similares (sempre executada devido ao caráter inovador dos projetos), prototipação (realizada como uma forma de auxiliar os cardiologistas no entendimento do sistema que será desenvolvido), testes com dados reais de pacientes e avaliação do sistema em

ambiente real (já integrado com os procedimentos dos cardiologistas, mas antes de ser realmente implantado). Cada atividade do processo padrão tem um conjunto de sub-atividades previamente definidas.

Usando o processo padrão, vários processos de software foram especializados, levando-se em consideração as características de cada tipo de software desenvolvido na organização (sistemas baseados em conhecimento, sistemas para Web etc.). Em cada especialização, as atividades foram organizadas de acordo com um modelo de ciclo de vida adequado para o tipo de software. As atividades do processo padrão foram refinadas em atividades mais detalhadas e novas atividades foram incluídas. Métodos, técnicas e ferramentas foram selecionados para cada atividade. Para ser utilizado em um projeto específico, o processo especializado mais adequado é instanciado para se ajustar às peculiaridades do projeto, tais como: complexidade, tamanho, cronograma e tempo de vida do produto.

A Figura 1 resume a abordagem, mostrando os passos desde a definição do processo padrão até a definição dos processos instanciados, que serão efetivamente utilizados para conduzir um projeto.

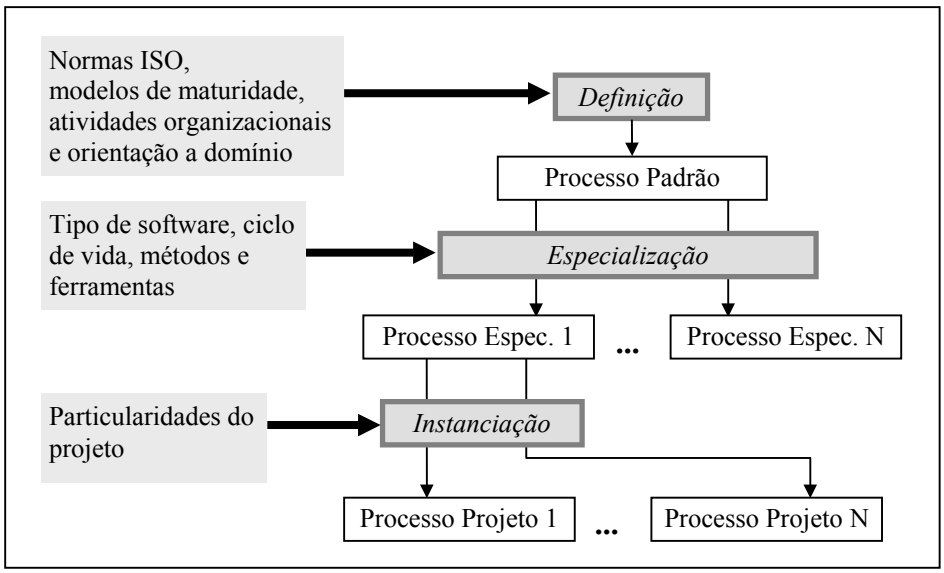

Figura 1: Esquema de Definição de Processos de Software [11]

3.2 Ambientes de Desenvolvimento de Software Orientados a Domínio

Para fornecer suporte automatizado para a abordagem descrita na subseção anterior, foi utilizada a infra-estrutura da Estação TABA [15], um meta-ambiente, desenvolvido na COPPE/UFRJ, capaz de gerar ambientes de desenvolvimento de software centrados em processo [1]. A infra-estrutura fornecida pela Estação TABA foi estendida para armazenar e prover conhecimento do domínio e para permitir a instanciação de processos para projetos específicos a partir de um processo padrão [7,11]. Assim, a Estação TABA tornou-se capaz não só de instanciar ambientes de desenvolvimento de software (ADS) convencionais, mas também Ambientes de Desenvolvimento de Software Orientados a Domínio (ADSOD) [11,12], definidos como ambientes que apóiam o desenvolvimento de software em domínios

específicos, fornecendo o conhecimento do domínio embutido no ambiente para guiar os desenvolvedores de software ao longo das várias fases do processo.

O conhecimento do domínio é incluído na Estação TABA pelo engenheiro de conhecimento e é armazenado em um componente chamado Teoria do Domínio, ficando disponível para a instanciação de diferentes ADSOD pertencentes ao mesmo domínio. Uma ferramenta (EdiTeD) [11] é fornecida para auxiliar o engenheiro de conhecimento nesta atividade. Uma Teoria de Domínio contém um conjunto de ontologias para representar o conhecimento sobre o domínio e um conjunto de tarefas pertinentes ao domínio, tais como interpretação e diagnóstico, que se encontram mapeadas com o conhecimento do domínio. No entanto, as tarefas, em si, são independentes de domínio (por exemplo: diagnóstico de doença e diagnóstico de falha mecânica).

Uma ferramenta (Def-Pro) foi incluída na Estação TABA para apoiar o engenheiro de software na definição de processos seguindo o esquema descrito na Figura 1 [7,8].

Nesta ferramenta, o engenheiro de software inicia pela definição do processo padrão. Para isto, ele escolhe a norma ou modelo de maturidade que será utilizado como base para definição do processo. Se for selecionado um modelo de maturidade, o nível de maturidade também precisa ser especificado. A ferramenta, então, solicita que o engenheiro de software especifique os processos de ciclo de vida que devem fazer parte do processo padrão. Quando o processo padrão está baseado em um modelo de maturidade, as atividades do nível de maturidade especificado são incluídas automaticamente. O engenheiro de software pode, ainda, selecionar atividades de outros níveis de maturidade do modelo ou de outras normas, trocar atividades por outras correspondentes definidas em alguma norma, selecionar atividades orientadas a domínio e atividades específicas da organização previamente definidas ou incluir tais atividades. De maneira similar, quando o processo padrão está baseado em uma norma, o engenheiro de software pode selecionar atividades previstas na norma, trocar atividades por outras correspondentes definidas em algum modelo de maturidade, pode também selecionar atividades dos modelos de maturidade sem atividade correspondente na norma, além de selecionar ou incluir atividades orientadas a domínio e atividades específicas da organização.

Definido o processo padrão, o engenheiro de software pode definir os processos especializados de acordo com o tipo de software, o paradigma e/ou o método a ser adotado. O engenheiro de software deve especializar cada processo de ciclo de vida do processo padrão que seja passível de especialização. Se for especificado um método que orienta a execução do processo de ciclo de vida como um todo, as atividades previstas no método são automaticamente inseridas no processo especializado. Caso contrário, o engenheiro de software seleciona atividades dentre as que são próprias dos métodos associados ao paradigma. Ainda, se foi especificado o tipo de software, o engenheiro de software seleciona atividades dentre as tipicamente associadas ao tipo de software. Por fim, o engenheiro de software instancia o processo especializado mais adequado ao seu projeto, estabelecendo o tipo de ciclo de vida e o número de iterações, incluindo novas atividades e detalhando as atividades já previstas. O detalhamento de uma atividade é feito através da especificação de sub-atividades, de procedimentos para execução das atividades, de artefatos requeridos e produzidos, além dos recursos humanos, de hardware e de software que são necessários.

Na instanciação de um ambiente de desenvolvimento de software, o engenheiro de software especifica o processo instanciado para o qual o ambiente será gerado e a Teoria do Domínio que deve ser incluída. As ferramentas especificadas no processo instanciado são

automaticamente incluídas no ambiente gerado. Para avaliação da abordagem foi construído o ADSOD Cordis, utilizando-se o processo de software definido na FBC e a Teoria do Domínio definida para o domínio da cardiologia [11].

4. Melhoria e Apoio Automatizado de Processos de Software baseados em Conhecimento Organizacional

Ao longo dos anos de parceria com a FBC, percebemos que, além do conhecimento do domínio, outro tipo de conhecimento seria de grande interesse para os desenvolvedores, o conhecimento organizacional. O conhecimento organizacional envolve vários tipos de conhecimento, dentre os quais destacam-se o conhecimento sobre a própria organização e dados e experiência obtidos em projetos anteriores. Para o desenvolvimento de alguns tipos de software, o conhecimento sobre os processos organizacionais é muito importante, até mesmo essencial. Sistemas de informação hospitalar são um exemplo, pois acompanham e apóiam os processos de atendimento médico desde a recepção do paciente até o faturamento da conta. Devido à alta rotatividade de pessoal nas equipes de software, profissionais da área de saúde freqüentemente tinham que explicar novamente um processo para um desenvolvedor recém-chegado à FBC. Para minimizar este problema, a organização tem se esforçado em manter profissionais-chave em cada equipe de software, de modo a assegurar a transmissão do conhecimento para os demais desenvolvedores. No entanto, não é possível evitar que estes profissionais-chave deixem a organização em algum momento. Outro aspecto observado foi a importância, principalmente durante as atividades de planejamento, de se identificar quem na organização detém o conhecimento necessário à execução de uma determinada atividade. Esta informação, relativamente simples de ser fornecida, pode assegurar que as pessoas mais adequadas à realização de uma tarefa estão sendo contatadas, ao mesmo tempo em que evita o desperdício de tempo no processo de coleta de informações.

Numa abordagem similar à utilizada para tratar a falta de conhecimento sobre o domínio, constatamos a importância de se considerar, explicitamente no processo de software, atividades que permitam ao desenvolvedores adquirir e registrar conhecimento organizacional relevante para os projetos de software. Isto nos levou a ampliar a nossa abordagem, modificando nossa hipótese inicial para:

Ambientes de Desenvolvimento de Software apóiam melhor o desenvolvimento e a manutenção de produtos de software se forem capazes de fornecer aos desenvolvedores, além do conhecimento do domínio, conhecimento organizacional.

A subseção a seguir apresenta os elementos que representam nossa proposta para captura e fornecimento do conhecimento organizacional. Na segunda subseção, o apoio automatizado é discutido.

4.1 Conhecimento Organizacional na Definição de Processos de Software

Para fornecer conhecimento organizacional aos desenvolvedores da FBC, decidimos ter um modelo da organização, contemplando inicialmente a sua estrutura, os seus processos e a distribuição de conhecimento, habilidades e experiências ao longo desta estrutura e destes processos [21]. Uma ontologia de organização foi definida com o intuito de permitir a

descrição deste modelo. Acreditamos que a disponibilidade do modelo da organização para as equipes de software irá resultar em aumento da produtividade, na medida em que permitirá agilidade na localização de pessoas e servirá de base para levantamentos de requisitos relacionados aos processos organizacionais. Esta última atividade também ganha em qualidade quando pode partir de um modelo continuamente utilizado e validado.

Além de disponibilizar o modelo da organização sempre atualizado para as equipes de software, pretendemos começar a registrar as lições aprendidas ao longo dos processos de software para que seja possível começar a derivar as melhores práticas para a organização. Três tipos de lições aprendidas são mencionados em [10]: informativas, de sucesso e de fracasso. As lições informativas explicam como proceder em uma determinada situação, as lições de sucesso fornecem exemplos de problemas que foram resolvidos de maneira positiva e as lições de fracasso fornecem exemplos de respostas negativas à tentativa de solucionar um problema, além de maneiras potenciais de contornar a situação. Sendo assim, nossa proposta inicial é descrever uma lição aprendida em termos do seu tipo, contexto em que foi gerada, descrição do problema, solução recomendada ou adotada e resultado esperado ou obtido. Esta proposta inicial está sendo detalhada e refinada em outro trabalho de pesquisa que temos conduzido. A relevância de tornar explícito para os desenvolvedores de software as lições aprendidas e melhores práticas da organização tem sido ressaltada na literatura técnica [2,9,10,17] e é também especialmente importante em um ambiente com alta rotatividade de pessoal.

Em função dos propósitos mencionados acima, duas novas atividades surgiram para os processos de software: a atividade de aprendizado sobre a organização e a atividade de registro de lições aprendidas.

A atividade de aprendizado sobre a organização tem como sub-atividades o estudo da finalidade da organização, o estudo dos aspectos estruturais, o estudo dos aspectos comportamentais e o estudo da distribuição de conhecimento ao longo da organização. Esta atividade será incluída nos processos de software especializados e instanciados sempre que necessário. Suas sub-atividades podem aparecer também com sub-atividades de outras atividades do processo. A atividade não foi incluída no processo padrão porque não seria relevante, por exemplo, para o desenvolvimento de um sistema especialista em enfarto agudo do miocárdio.

A atividade de registro de lições aprendidas foi incluída como última atividade do processo padrão e também como parte de outras atividades em que uma oportunidade de aprendizado possa aparecer.

4.2 Ambientes de Desenvolvimento de Software Orientados a Organização

Novamente, a infra-estrutura da Estação TABA foi estendida. Desta vez, o objetivo foi permitir a configuração de um ADS para uma organização específica. O ambiente configurado, um Ambiente de Desenvolvimento de Software Orientado à Organização (ADSOrg) [19], irá conter todo o conhecimento acumulado pela organização que tenha sido capturado explicitamente por sua importância para o desenvolvimento de software, o que inclui o modelo da organização, o processo padrão da organização, os processos especializados para os tipos de software desenvolvidos, além das lições aprendidas e melhores práticas identificadas ao longo do tempo. ADSOrg visam apoiar o aprendizado organizacional sobre desenvolvimento de software e fornecer aos desenvolvedores o conhecimento

acumulado pela organização e relevante para a execução de suas tarefas. A partir dos processos específicos definidos em um ADSOrg, são instanciados os ambientes para os projetos, que podem ser orientados a domínio ou não. Os ambientes instanciados são os ambientes que apóiam o desenvolvimento propriamente dito dos produtos de software. Estes ambientes têm acesso ao conhecimento da organização registrado no ADSOrg. A Figura 2 resume o esquema para geração de ambientes a partir da Estação TABA.

A definição do processo padrão e a sua especialização de acordo com os tipos de software desenvolvidos na organização são realizadas na Estação TABA, como pré-requisito para a configuração do ADSOrg. Tanto a definição do processo padrão quanto a sua especialização são tarefas complexas, que exigem ampla experiência na definição de processos, além de profundo conhecimento da organização. Desta forma, considerou-se conveniente que um ADSOrg fosse configurado já com estes processos definidos. Consultores na área de processo e profissionais experientes da organização trabalharão em conjunto na definição dos processos padrões e especializados, de forma a fornecer uma base confiável para a definição dos processos voltados para projetos específicos. A instanciação dos processos especializados de acordo com as particularidades dos projetos específicos é realizada no ADSOrg, para que a organização tenha liberdade para definir os processos mais adequados a cada um de seus projetos de software. Isto representa uma modificação à abordagem anteriormente apresentada na Figura 1.

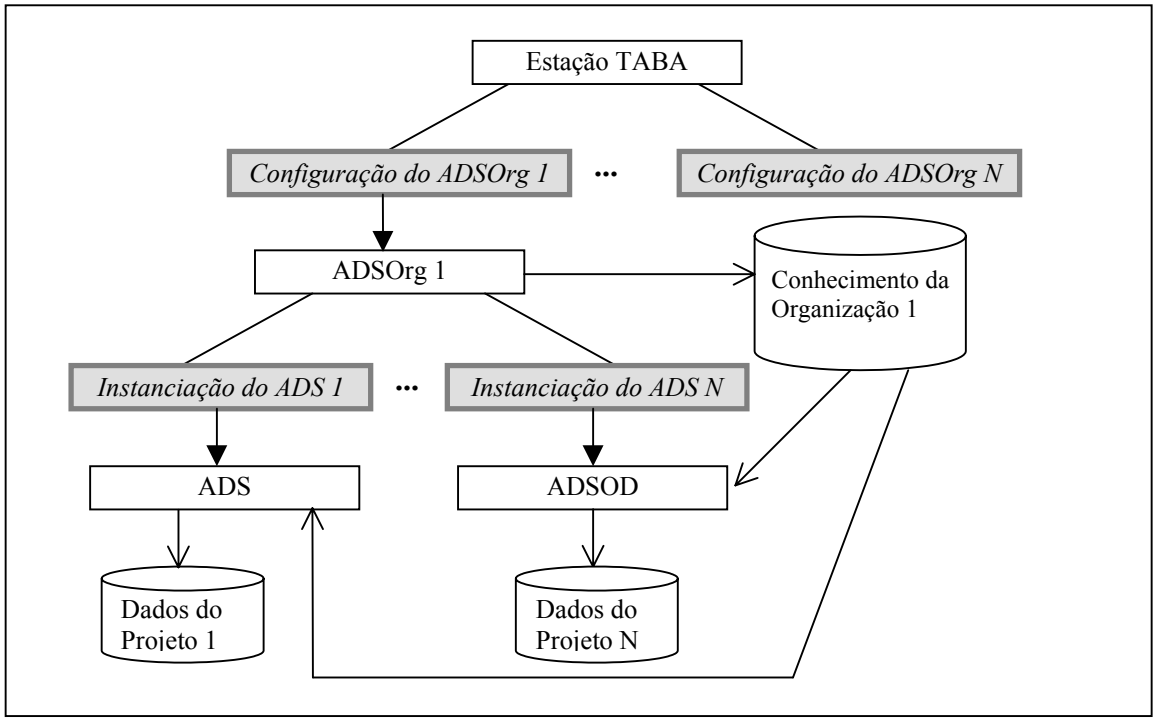

Figura 2: Esquema para geração de ambientes a partir da Estação TABA

A ferramenta para definição de processo anteriormente existente (Def-Pro) foi modificada para refletir o novo esquema. Além disto, uma outra ferramenta, que permitia a edição de processos para projetos específicos previamente definidos (Edit-Pro) [16], teve sua funcionalidade ampliada para permitir também a edição do processo padrão e de processos

especializados. A ferramenta de definição de processos guia o engenheiro de software através dos passos necessários à definição de um processo, além de utilizar atividades de uma norma ou de um modelo de maturidade como base para definição das atividades do processo padrão. Já a ferramenta de edição de processos parte do pressuposto que a organização já tem um processo padrão definido, havendo apenas a necessidade de registrá-lo na Estação TABA para que um ADSOrg para a organização seja configurado.

Na configuração de um ADSOrg, o engenheiro de software define as características do ambiente e solicita a sua criação. Estas características incluem a organização para a qual o ambiente será gerado, seu processo padrão e as Teorias de Domínio a serem utilizadas.

Configurado o ADSOrg, o modelo da organização é descrito pelo gerente de conhecimento através do uso de duas ferramentas que permitem a representação e a visualização da distribuição de conhecimento, habilidades e experiências ao longo da estrutura e dos processos organizacionais [21].

A primeira ferramenta fornece para os desenvolvedores de software a representação da estrutura organizacional e a distribuição de conhecimento, habilidades e experiências ao longo desta estrutura. O objetivo principal é possibilitar que desenvolvedores de software rapidamente encontrem, dentro da estrutura organizacional, os profissionais mais adequados à realização de uma atividade ou à solução de um problema. Para isto, a ferramenta permite a representação da estrutura organizacional; a descrição de cada unidade organizacional com a especificação de quais são as posições existentes dentro da unidade organizacional e quais conhecimentos, habilidades e experiências são necessários para o bom desempenho de suas atividades; a associação dos profissionais às posições que ocupam na organização com a especificação dos conhecimentos, habilidades e experiências que possuem; a visualização da estrutura organizacional e a navegação através da mesma; a pesquisa direta de quem na organização possui determinado conhecimento, habilidade ou experiência; e a pesquisa indireta através dos conhecimentos, habilidades ou experiências disponíveis na organização em busca de quem possui conhecimento, habilidade ou experiência mais próximo do desejado.

A segunda ferramenta, além de fornecer para os desenvolvedores de software a representação dos processos executados na organização, tem como objetivo fornecer o contexto em que uma certa habilidade, experiência ou conhecimento é utilizado, de forma a facilitar o entendimento tanto da atividade quanto dos conhecimentos necessários para realizá-la. Para isto, a ferramenta permite a representação gráfica dos processos organizacionais, possibilitando a definição de quais são as atividades que compõem um processo, como elas estão relacionadas quanto à ordem de execução e quais são os atores envolvidos. A cada atividade podem ser associados os artefatos que servem de insumos e os que são gerados como produtos, como também quais são os conhecimentos, habilidades e experiências requeridos para a sua realização. A ferramenta também permite a descrição dos elementos que compõem os processos organizacionais, a visualização destes processos e navegação através dos seus diferentes níveis de abstração e a pesquisa de onde uma determinado conhecimento habilidade, experiência é necessário ao longo dos processos organizacionais.

A ferramenta para apoio à atividade de registro de lições aprendidas ainda precisa ser definida e será utilizada nos ambientes instanciados.

5. Lições Aprendidas

A motivação para as propostas de melhoria de processo e evolução dos ambientes de desenvolvimento de software foi fruto da parceria com a FBC, ao longo de 8 anos, para pesquisa e transferência de tecnologia relativa à definição, utilização, avaliação e melhoria dos seus processos de software. Durante este tempo, aprendemos que:

- definir um processo de software e utilizá-lo na condução de um projeto não é suficiente para garantir o sucesso do projeto. É necessário definir um processo que seja adequado ao produto de software específico e à cultura de desenvolvimento da organização;
- a existência de um processo padrão para a organização torna mais fácil a definição e o gerenciamento de diferentes processos de software dentro de uma mesma organização;
- deve-se evitar incluir no processo padrão atividades específicas do primeiro projeto a ter o seu processo definido a partir da abordagem de processo padrão. Além disto, é necessário a revisão periódica do processo padrão a fim de detectar possíveis problemas de concepção;
- boas práticas de engenharia de software não são suficientes para a definição e melhoria dos processos de software. O processo padrão deve considerar os problemas cruciais que são enfrentados pela organização no desenvolvimento do software, e;
- a falta de conhecimento sobre o domínio e sobre a organização representam problemas cruciais para o desenvolvimento de software em organizações com alta rotatividade de pessoal. Mesmo ampla experiência em engenharia de software e conhecimento profundo sobre o contexto do projeto não garantem a previsão de todos os problemas que podem surgir em um projeto.

A abordagem para definição de processos de software, descrita neste artigo, leva em consideração essas lições aprendidas. Normas internacionais, modelos de maturidade, modelos de processo, métodos e ferramentas representam as boas práticas de engenharia de software. O esquema de especialização e instanciação de processos a partir do processo padrão permite a adequação do processo ao produto específico a ser desenvolvido e ao ambiente de desenvolvimento, ao mesmo tempo em que facilita e padroniza a definição de processos de software na organização. A falta de conhecimento sobre o domínio foi tratada através da definição e inclusão da atividade de investigação do domínio nos processos de software. Além disto, duas outras atividades para os processos de software foram definidas com o intuito de tratar o problema de falta de conhecimento sobre a organização: as atividades de aprendizado sobre a organização e de registro de lições aprendidas.

6. Conclusão

Neste artigo, apresentamos as soluções de engenharia de software que temos proposto para lidar com dois problemas enfrentados por desenvolvedores de software, principalmente os novatos em uma organização: a falta de conhecimento sobre o domínio e sobre a organização. Estes problemas são especialmente importantes em organizações com alta rotatividade de pessoal.

A inclusão de atividades anteriormente não previstas, mas freqüentemente realizadas, nos processos de software já representa uma melhoria importante. No entanto, nós acreditamos

que os benefícios são realmente significativos quando modelos do conhecimento do domínio e do conhecimento organizacional são construídos e disponibilizados para os desenvolvedores. Nós acreditamos, ainda, que o apoio automatizado à execução de tais atividades, fornecido através de ambientes de desenvolvimento de software (ADS), pode maximizar estes benefícios. Para lidar com a complexidade de definir e construir ADS capazes de apoiar as novas atividades propostas, primeiro foram definidos e construídos ambientes de desenvolvimento de software capazes de fornecer o conhecimento sobre o domínio para os desenvolvedores (ADSOD). A seguir, a abordagem foi ampliada para que o conhecimento sobre a organização e a experiência acumulada pelos seus desenvolvedores ao longo do tempo também pudessem ser capturados e disponibilizados em ambientes de desenvolvimento de software (ADSOrg).

Entretanto, é necessário esforço para que o conhecimento do domínio seja modelado e representado em um ambiente de desenvolvimento de software, mas, uma vez modelado, pode ser reutilizado na construção de outros ambientes para o mesmo domínio e estar sempre disponível, evitando perda de tempo e insatisfação dos especialistas do domínio. A elaboração do modelo organizacional também requer tempo e necessita de constante atualização. No entanto, além de beneficiar as atividades da organização relacionadas a software, a elaboração do modelo organizacional traz benefícios diretos para a organização como um todo, pois permite que problemas na estrutura, nos processos e na distribuição de conhecimento, habilidades e experiências sejam detectados e corrigidos. Por fim, os benefícios de se manter registros das lições aprendidas e das melhores práticas organizacionais são indiscutíveis, mas necessitam de uma política bem elaborada de incentivo, pois os desenvolvedores tendem a se sentir ameaçados.

A infra-estrutura da Estação TABA foi modificada ao longo do tempo para permitir a definição e geração dos ambientes propostos. Ferramentas tiveram sua estrutura modificada, como o Edit-Pro [16] e o Def-Pro [7,8], e outras foram definidas [21]. A Estação TABA possui uma estrutura genérica que permite a construção de ambientes para diferentes organizações, incluindo aquelas cuja área de negócio é o desenvolvimento de software e que, conseqüentemente, têm diferentes tipos de clientes envolvidos em seus processos de software.

As hipóteses identificadas neste artigo ainda precisam ser validadas. Com este intuito, estamos construindo um ADSOrg para a FBC [20]. A partir do uso deste ambiente e dos ambientes a serem instanciados para projetos específicos da organização é que será possível avaliar o papel do conhecimento sobre o domínio e do conhecimento sobre a organização no desenvolvimento de software. Em paralelo, estamos investigando outros tipos de conhecimento organizacional que possam ser importantes no contexto de projetos de software, em adição aos tipos já mencionados no decorrer deste artigo.

7. Agradecimentos

Nossos agradecimentos ao CNPq, pelo apoio financeiro concedido ao projeto Ambientes de Desenvolvimento de Software Orientados a Organização e à UCCV/FBC, por nos permitir discutir neste artigo os problemas enfrentados pela organização no desenvolvimento de software.

8. Referências Bibliográficas

[1] Ambriola, V., Conradi, R., Fuggetta, A., Assessing Process-Centered Software Engineering Environments. ACM Transactions on Software Engineering and Methodology (6-3, 1997), 283-328.

[2] Basili, V., 1989, The Experience Factory: Packaging Software Experience. 14th Annual Software Engineering Workshop, SEL, NASA, USA.

[3] Chandrasekaran, B., Johsephson, R., Bejamins, V., What are Ontologies, and why do we need them? IEEE Intelligent Systems (Jan/Feb, 1999).

[4] Emam, K., Drouin, J. and Melo, W., SPICE – The Theory and Practice of Software Process Improvement and Capability Determination. IEEE Computer Society Press, 1998.

[5] ISO/IEC 12207*:* Information technology - Software Life Cycle Processes, International Standard Organization, 1995.

[6] 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, International Standard Organization, 1990.

[7] Machado, L.F., Modelo para Definição de Processos de Software na Estação TABA.Tese de MSc. da COPPE/UFRJ (Rio de Janeiro, Brasil, Mar 2000).

[8] Machado, L.F., Oliveira, K., Rocha, A.R., Using Standards And Maturity Models For The Software Process Definition. Quality Week (Bruxelas, Bélgica, Nov 2000).

[9] Markkula, M., Knowledge Management in Software Engineering Projects. SEKE'99 (Kaiserlauthern, Germany, Jun 1999), 20-27.

[10] O'Leary, D., Enterprise Knowledge Management. *IEEE Computer*, 31(3) (Mar), 54-61.

[11] Oliveira, K., Modelo para Construção de Ambientes de Desenvolvimento Orientados a Domínio, Tese de DSc. da COPPE/UFRJ (Rio de Janeiro, Brasil, Out 1999).

[12] Oliveira, K., Rocha, A.R., Travassos, G. et al., Using Domain-Knowledge in Software Development Environments. SEKE'99 (Kaiserlauthern, Germany, Jun 1999), 180-187.

[13] Rabelo Jr, A., Rocha, A.R., Oliveira, K. et al., An Expert System for Diagnosis of Acute Myocardial Infarction with ECG Analysis. Artificial Intelligence in Medicine, 10 (1997), 75-92.

[14] Rabelo Jr, A., Rocha, A.R., Oliveira, K. et al., An Expert System for Diagnosis of Acute Myocardial Infarction: Software Quality Assurance Procedures. European Symposium on the Validation and Verification of Knowledge Based-Systems (1995), 117-128.

[15] Rocha, A.R., Souza, J., Aguiar, T., TABA: A Heuristic Workstation for Software Development. *COMPEURO'90* (1990), 126-129.

[16] Santos, G., Zlot, F., Definição e instanciação de ambientes na Estação TABA. Projeto de Final de Curso da UFRJ (Rio de Janeiro, Brasil, Out 1999).

[17] Seaman, C. *et al.*, 1999, An Experience Management System for a Software Consulting Organization. 24th Annual Software Engineering Workshop, SEL, NASA, USA.

[18] Schreiber, G. (ed.), Knowledge Acquisition: The KADS Approach for Knowledge Engineering. Academic Press, 1992.

[19] Villela, K., Travassos, G., Rocha, A.R., Toward Enterprise Oriented Software Development Enviroments. XIV SBES - V Workshop of Theses and Dissertations in Software Engineering (João Pessoa, Brasil, Oct 2000), 379-384.

[20] Villela, K., Oliveira, K., Santos, G et al., CORDIS-FBC: Um Ambiente de Desenvolvimento de Software para Cardiologia. XV SBES - I Workshop de Informática Médica (Rio de Janeiro, Brasil, Out 2001).

[21] Villela, K.; Zlot, F.; Santos, G. et al., Knowledge Management in Software Development Environments. ICSSEA'01 (Paris, França, Dec 2001).

[22] Werneck, V. et al., A Software Development Process for Expert Systems. 10th International Symposium on Methodologies for Intelligent Systems (North Caroline, US, Oct 1997), 209-220.