Charon: uma Ferramenta para a Modelagem, Simulação, Execução e Acompanhamento de Processos de Software

Leonardo Gresta Paulino Murta
Marcio de Oliveira Barros
Cláudia Maria Lima Werner
{murta, marcio, werner}@cos.ufrj.br
COPPE/UFRJ – Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro
Caixa Postal 68511 – CEP. 21945-970
Rio de Janeiro – Brasil

Resumo

Este artigo apresenta uma ferramenta extensível, baseada em agentes inteligentes, que fornece suporte para a modelagem, simulação, instanciação, execução, acompanhamento, monitoramento e evolução de processos de software. A arquitetura da ferramenta permite a separação entre o controle, representado por agentes, e as informações de execução do processo, representadas por bases de connhecimento. Além disso, a ferramenta pode ser estendida através da construção de novos agentes, que implementam novos requisitos. Essa construção ocorre por meio da instanciação de um *framework* e do uso de uma ontologia que define a *API* para a comunicação com as bases de conhecimento.

Abstract

This paper presents an extensible tool, based on intelligent agents, that provides support for modeling, simulating, instantiating, enacting, accomplishing, monitoring and evolving software processes. The tool architecture allows the separation between control, represented by agents, and executing process information, represented by knowledge bases. Therefore, the tool may be extended through the construction of new agents, which implement new requirements. This construction occurs by the instantiation of a framework and the use of an ontology that defines the API for communicating with the knowledge bases.

1. Introdução

O desenvolvimento de software, como toda atividade de engenharia, necessita de um processo que o sistematize, envolvendo atividades, pessoas e ferramentas necessárias para a sua execução, assim como artefatos consumidos ou produzidos pelas atividades. A partir de um processo bem definido, é possível alavancar a qualidade do produto a ser gerado, pois a qualidade do produto é fortemente dependente da qualidade do processo pelo qual ele é construído e mantido. Além do estabelecimento do processo, é necessário fazer um acompanhamento de sua execução, objetivando, entre outras coisas, fornecer informações que permitam a tomada de decisões gerenciais, de acordo com situações detectadas durante sua execução, e controlar o fluxo de trabalho dos engenheiros de software através da automação do processo. O software responsável por automatizar a execução de processos é conhecido como máquina de processos [1].

Existem várias abordagens para a construção de ambientes de desenvolvimento de software com ênfase na modelagem e execução de processos, como por exemplo: SPADE [2], Memphis [3], ProSoft [4], HyperCode [5], A.I.P.M. Fusion [6], CAGIS [7], EPOS [8], Merlin [9], ProNet/ProSim [10], Dynamite [11], APEL [12] e Mokassin [13]. Entretanto, todas essas abordagens têm deficiências quanto as suas formas de utilização, atuação ou extensão [14].

Ainda não existe uma forte preocupação na forma de utilização das máquinas de processo. As abordagens existentes permitem que processos sejam modelados, mas restringem o seu uso a um determinado nível de abstração. É interessante construir processos através de blocos, ou componentes, e reutilizar esses componentes de processos em outros contextos, facilitando a utilização da máquina de processos como um todo.

A atuação da maioria das abordagens é reativa, consistindo no tratamento de eventos provenientes dos próprios desenvolvedores. Todavia, a máquina de processos pode agir de forma pró-ativa, prevendo certas situações e atuando sem a interferência do desenvolvedor, fazendo com que a execução siga adiante ou notificando um determinado desenvolvedor que algo deve ser feito num dado momento.

Os requisitos de um ambiente centrado em processo podem variar com o tempo, motivando a extensão do ambiente. A facilidade de extensão do ambiente está diretamente relacionada com a forma em que a sua arquitetura foi projetada, facilitando o atendimento de novos requisitos. Uma das motivações deste trabalho está na possibilidade de uma diminuição nos custos financeiros e do tempo necessário para a inclusão de novos requisitos.

Dentre as tecnologias atualmente disponíveis, a tecnologia de agentes inteligentes apresenta a estruturação necessária para a construção de sistemas pró-ativos e de fácil extensão, devido as suas características de decomposição, abstração e organização [15].

O objetivo deste trabalho é fornecer uma ferramenta para a modelagem, simulação, instanciação, execução, acompanhamento, monitoramento e evolução de processos de software reutilizáveis que seja pró-ativa e de fácil extensão, utilizando uma arquitetura para a construção de agentes inteligentes e uma ontologia para a comunicação entre os agentes que define o vocabulário necessário para a representação dos processos modelados.

Este trabalho está organizado em 3 seções, além desta primeira de introdução, que descreve a sua motivação, o seu objetivo e a sua organização, assim como trabalhos relacionados existentes na literatura. Na segunda seção, é descrita a utilização da ferramenta proposta. Na terceira seção, a ferramenta é detalhada tecnicamente, através de discussões sobre os elementos que a compõem. Finalmente, na quarta seção, são descritas as contribuições e limitações deste trabalho e relacionados os trabalhos futuros.

2. Utilização da ferramenta

A utilização da ferramenta consiste, inicialmente, na modelagem gráfica do processo. Para isso, deve ser utilizado o ambiente de modelagem da ferramenta, que é baseado na notação do diagrama de atividades da UML.

O ambiente de modelagem permite a criação de processos compostos e primitivos, como exibido na Figura 1. Um processo composto contém um workflow que define como os seus sub-processos estão relacionados. Os workflows permitem a utilização de fluxos de retorno, recursões a super-processos, paralelismo, pontos de sincronismo e pontos de decisão. Um processo primitivo define o seu roteiro de execução, as ferramentas que devem ser utilizadas, os papéis de desenvolvedores autorizados, o tempo estimado de execução e os artefatos consumidos e produzidos pela atividade.

A partir de um processo modelado, é possível iniciar a construção de um projeto de desenvolvimento de software através da instanciação do processo. A instanciação do processo é precedida por uma etapa de simulação, onde a ferramenta verifica se existem erros de modelagem e estima os tempos de execução dos processos compostos, que posteriormente servirão como referência na visualização do andamento do projeto. Caso nenhum erro tenha sido encontrado, o processo é instanciado no projeto e o gerente seleciona quais desenvolvedores exercerão os papéis modelados, como exibido na Figura 2.

Deste momento em diante, o processo está em execução. Sempre que um determinado desenvolvedor entrar no ambiente de desenvolvimento, a ferramenta irá informar quais atividades estão pendentes para ele e quais decisões devem ser tomadas, como exibido na Figura 3. Os pontos de

decisão representam possíveis mudanças do fluxo principal de execução do workflow. Eles são representados por uma pergunta associada a um conjunto de respostas. Cada resposta pode mudar o caminho da execução do processo.

Em função do término de alguma atividade pendente, ou da tomada de decisão, novas atividades podem ficar pendentes para diferentes desenvolvedores, e novas decisões podem se tornar necessárias, dando continuidade ao desenvolvimento do projeto.

Durante a execução do processo, o gerente poderá acessar o ambiente de monitoramento do processo para verificar quais atividades já foram executadas, quantas vezes cada atividade foi executada e como foi a execução em relação à previsão gerada na simulação, como exibido na Figura 4. Para fornecer essa visão macro do processo, foi utilizada uma notação de cores e sobreposições. A notação de cores utiliza vermelho para indicar que uma determinada atividade foi executada em tempo pior que o previsto, verde para indicar que uma determinada atividade foi executada com tempo dentro do previsto e amarelo para indicar que a atividade ainda está em execução. A notação de sobreposições indica quantas vezes uma determinada atividade foi executada, repetindo o desenho da atividade de forma sobreposta, com as execuções mais recentes em cima.

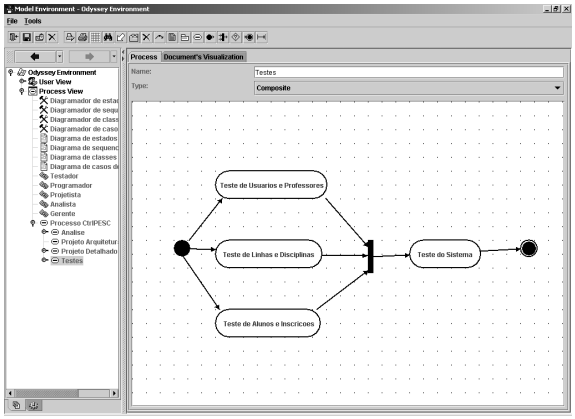

Figura 1: Ambiente de modelagem de processos.

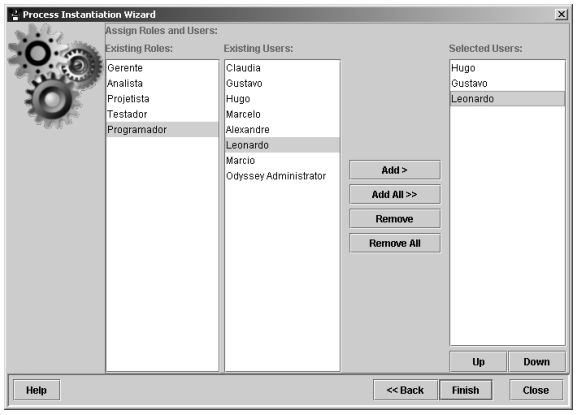

Figura 2: Wizard de instanciação do processo.

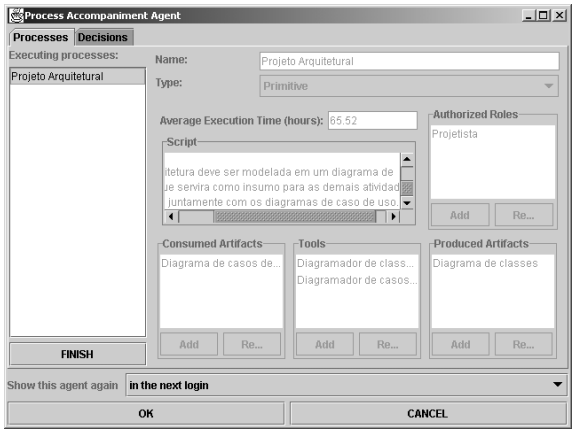

Figura 3: Lista de atividades pendentes.

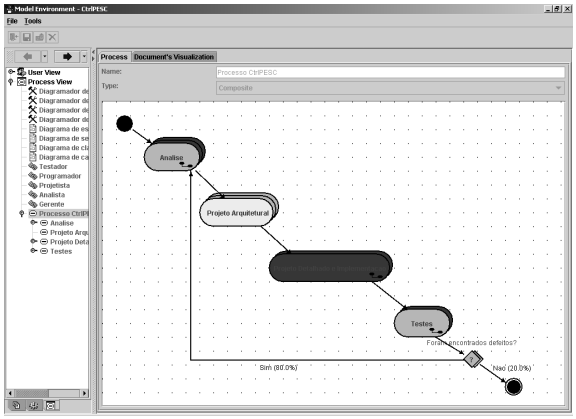

Figura 4: Processos em execução.

Sempre que necessário, qualquer desenvolvedor pode requisitar a sua lista de pendências e notificar à ferramenta que determinadas atividades foram finalizadas. Entretanto, caso algum desenvolvedor tenha solicitado, por engano, para a ferramenta finalizar uma atividade ainda em andamento, o gerente pode fazer uso de um recurso para retroceder o processo para um determinado instante de tempo, fazendo com que todas as ações ocorridas após esse instante sejam desconsideradas.

Outro recurso importante provido pela ferramenta é o suporte a reinstanciação do processo. É notória a necessidade de mudanças na definição do processo depois da sua entrada em execução. Algumas situações somente são descobertas durante a execução real do processo. Essas situações, que

não foram previstas durante a modelagem, devem ser consideradas a partir do momento que foram detectadas. Para isso, a ferramenta permite que o processo volte para a etapa de modelagem, seja adaptado e reinstanciado, sem que as informações atuais sobre a execução sejam perdidas. Este recurso também é útil para suportar a otimização de processos em execução.

3. Detalhamento Técnico

As principais atividades envolvidas no uso da ferramenta, descritas na Figura 5, demandam mecanismos específicos para a sua execução. Nesta seção, esses mecanismos são brevemente discutidos. Outros trabalhos [14; 16] descrevem de forma mais detalhada a abordagem que está por trás da implementação da ferramenta.

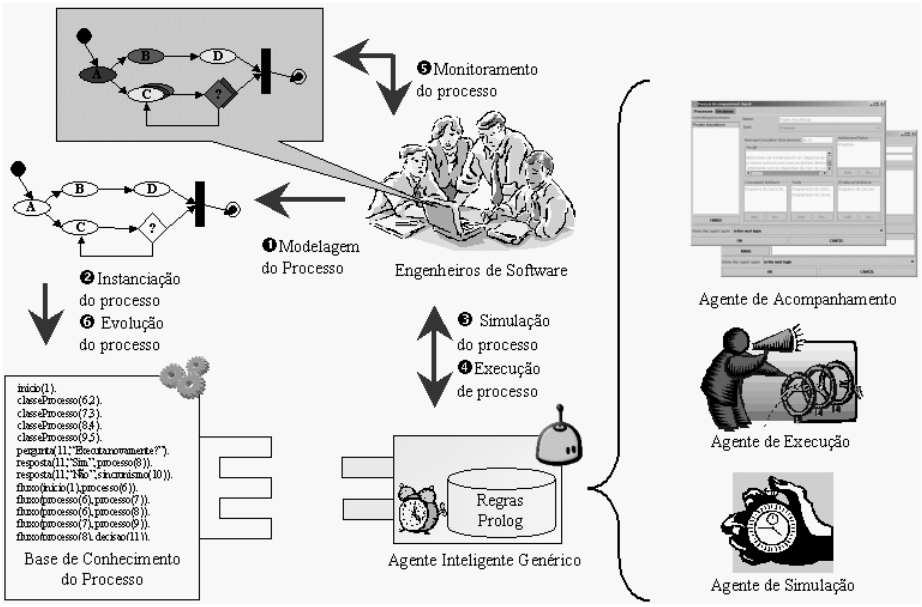

Figura 5: Principais atividades da ferramenta.

Para que a ferramenta fosse de fácil extensão, foi definida uma arquitetura baseada em agentes inteligentes que permite a separação entre os dados que representam a execução do processo e a máquina que efetua a sua execução. A atividade de instanciação do processo consiste em um mapeamento da notação gráfica, utilizada na modelagem, para uma representação Prolog. Essa representação povoa a base de conhecimento do projeto. Essa base de conhecimento contém todas as informações sobre a execução do processo e, conseqüentemente, sobre o andamento do projeto.

Desta forma, os dados do processo são representados por bases de conhecimento, e a máquina de processo é representada por agentes inteligentes. Esses agentes fazem consultas e modificações nas bases de conhecimento, permitindo que o processo seja executado e que novos agentes sejam construídos para atender novos requisitos.

A flexibilidade da máquina de processos é obtida através da sua arquitetura baseada no framework para a construção de agentes inteligentes. Inicialmente, quatro agentes inteligentes foram construídos:

- **Agente de Simulação:** Responsável por verificar o processo modelado quanto a sua corretude, permitindo ou não a sua entrada em execução, e calcular o seu tempo previsto de execução;
- **Agente de Execução:** Responsável por verificar o estado da base de conhecimento, procurando por atividades finalizadas ou decisões que foram tomadas, permitindo o andamento do processo;

- **Agente de Acompanhamento:** Responsável por interagir com o desenvolvedor, indicando as atividades que estão pendentes e as decisões que devem ser tomadas. Também permite a finalização dessas atividades ou a tomada dessas decisões;
- **Agente de Retrocesso:** Responsável por permitir o retorno da base de conhecimento para um determinado instante de tempo no passado, para que seja possível contornar erros na utilização da ferramenta.

Desta forma, novos requisitos à máquina de processos podem ser modelados em novos agentes, que consultam e modificam a base de conhecimento em função dos seus objetivos. Cada agente, construído segundo o framework, pode agir de forma reativa, motivado por eventos vindos do ambiente ou de outros agentes, ou de forma pró-ativa, buscando o seu objetivo mesmo que nenhum evento externo tenha ocorrido.

4. Conclusão

As abordagens tradicionais para a automação de processos de software agem de forma reativa, respondendo a eventos gerados pelos desenvolvedores. Essas abordagens geralmente não têm preocupações referentes à extensão da própria máquina de processos, o que possibilitaria o atendimento a novos requisitos com facilidade.

Para suprir essa necessidade, especificamos e construímos uma máquina de processos, que apresenta como principais características:

- Modelagem gráfica de processos, que permite a utilização de ciclos ou recursões, seguindo uma extensão do diagrama de atividades da UML como notação;
- Simulação do processo modelado antes do início da execução, permitindo a detecção de erros sintáticos de modelagem e indicando possíveis caminhos para a correção;
- Instanciação do processo através de um mapeamento automático da notação gráfica para uma base de conhecimento Prolog, segundo uma ontologia definida;
- Acompanhamento do processo através de worklists que descrevem quais atividades ou decisões estão pendentes para um determinado desenvolvedor;
- Monitoramento da execução do processo através de uma notação de cores e sobreposição;
- Evolução de processos via um mecanismo de reinstanciação;
- Suporte a pró-atividade e extensão da máquina de processos, através da arquitetura baseada em um framework para a construção de agentes inteligentes;

Entretanto, algumas limitações puderam ser detectadas nesse trabalho [14], que motivam a elaboração de trabalhos futuros, possibilitando a evolução da abordagem proposta em relação ao suporte à reutilização de processos, à instanciação de processos, ao controle e monitoramento da execução das ferramentas, à especificação de relatórios gerenciais e verificação de consistência das bases de conhecimento. Outro trabalho futuro é a verificação empírica dos resultados obtidos pela utilização da ferramenta.

Agradecimentos

Os autores gostariam de agradecer ao CNPq pelo investimento financeiro neste trabalho e aos demais integrantes do Projeto Odyssey pelo apoio durante o seu desenvolvimento.

Referências Bibliográficas

[1] SILVA, S. A. D., NUNES, D. J., 1999, *Proposta de uma Ferramenta de Simulação de Processos de Software Baseada em Conhecimento para o Ambiente PROSOFT*. IV Semana Acadêmica do PPGC - UFRGS.

[2] BANDINELLI, S., FUGGETTA, A., GHEZZI, C., et al., 1994, *SPADE: An Environment for Software Process Analysis, Design and Enactment*. Research Studies Press.

[3] VASCONCELOS, F. M., WERNER, C. M. L., 1998, "Organizing the Software Development Process Knowledge: An Approach Based on Patterns", *International Journal of Software Engineering and Knowledge Engineering*, v. 8, n. 4, pp. 461-482.

[4] REIS, R. Q., REIS, C. A. N. D. J., 1999, "Ambiente de Desenvolvimento de Software PROSOFT: Evolução e Estágio Atual". In: *Semana de Informática (SEMINF'99)*, Universidade Federal do Pará.

[5] PERRY, D. E., PORTER, A., VOTTA, L. G., et al., 1996, "Evaluating workflow and process automation in wide-area software development". In: *European Workshop on Software Process Technology*, pp. 188-193, Berlin, Germany.

[6] SANTANDER, V. F. A., GIMENES, I. M. S., MASIERO, P. C., 1997, "Assistência Inteligente ao Processo de Engenharia de Software". In: *XI Simpósio Brasileiro de Engenharia de Software*, pp. 147-161, Fortaleza, Brasil.

[7] WANG, A. I., HANSSEN, A. A., NYMOEN, B. S., 2001, "Design Principles For A Mobile, Multi-Agent Architecture For Cooperative Software Engineering". In: http://citeseer.nj.nec.com/342416.html, Accessed in 20/11/2001.

[8] JACCHERI, L., LARSEN, J., CONRADI, R., 1992, "Software Process modeling and Evolution in EPOS". In: *4th International Conference on Software Engineering and Knowledge Engineering (SEKE'92)*, pp. 574-581, Capri, Italy.

[9] JUNKERMANN, G., PEUSCHEL, B., SCHÄFER, W., et al., 1994, "Merlin: Supporting Cooperation in Software Development through a Knowledge-based Environment ". In Nuseibeh, B., Finkelstein, A., and Kramer, J., Taunton, Inglaterra, John Wiley and Sons.

[10] CHRISTIE, A., 1995, *Software Process Automation: The Technology and Its Adoption*, Berlin, Springer-Verlag Publishing.

[11] HEIMANN, P., JOERIS, G., KRAPP, C., et al., 1996, "DYNAMITE: Dynamic Task Nets for Software Process Management". In: *18th International Conference on Software Engineering*, pp. 331-341, Berlin, Germany.

[12] DAMI, S., ESTUBLIER, J., AMIOUR, M., 1998, "APEL: A graphical yet executable formalism for process modelling", *Automated Software Engineering: An International Journal*, v. 5, n. 1, pp. 61-96.

[13] JOERIS, G., 2001, "Mokassin". In: http://www.informatik.uni-bremen.de/grp/mokassin, Accessed in 18/11/2001.

[14] MURTA, L. G. P., 2002, *Charon: Uma Máquina de Processos Extensível Baseada em Agentes Inteligentes*, Dissertação de M.Sc., COPPE, UFRJ, Rio de Janeiro, Brasil.

[15] JENNINGS, N. R., 2000, "On agent-based software engineering", *Artificial Intelligence*, v. 177, n. 2, pp. 277-296.

[16] MURTA, L. G. P., BARROS, M. O., WERNER, C. M. L., 2002, "Charon: Uma Máquina de Processos Extensível Baseada em Agentes Inteligentes". In: *IDEAS 2002*, Havana, Cuba.