Uma Máquina de Processos de Desenvolvimento de Software Baseada em Agentes Inteligentes

Leonardo Gresta Paulino Murta
murta@cos.ufrj.br

Orientador: *Cláudia Maria Lima Werner*

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

Abstract

The objective of this work is to provide an architecture based on intelligent agents that allows to execute and control software processes. The use of intelligent agents provides a more pro-active and dynamic architecture, differentiating it from the traditional static approaches, which are mainly based on state transition diagrams. In this approach, a modeled process is represented by *Prolog* predicates stored in a knowledge base, and used throughout process execution and control.

The knowledge base represents the current state of the process and it is explored by intelligent agents which are responsible for maintaining the correct flow of process execution, informing developers about possible activities that can be done, tools that can be used, etc. Moreover, to support process management, measurements such as team productivity can be performed by the agents and informed to the project manager.

Palavras-Chave: Automação de Processos de Software, Agentes Inteligentes, Processo de Software.

1 Introdução

O desenvolvimento de software, como toda atividade criativa, necessita de um processo que o sistematize, envolvendo atividades, pessoas e ferramentas necessárias para a sua execução, assim como artefatos utilizados ou produzidos pelas atividades.

A partir de um processo bem definido, é possível identificar o nível de 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.

Entretanto, 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, controlar o fluxo de trabalho dos engenheiros de software, aprender com as atitudes tomadas por desenvolvedores experientes, fazer sugestões para desenvolvedores inexperientes e aprimorar o próprio processo.

A motivação deste trabalho é poder aumentar a possibilidade do processo ser seguido corretamente pelos desenvolvedores, através do uso de uma ferramenta de acompanhamento da execução de processos que seja pró-ativa e dinâmica, baseada em agentes inteligentes.

2 Trabalhos Similares

Vários trabalhos similares a este existem na literatura. Nesta seção, apresentamos alguns deles, enfatizando suas diferenças e similaridades em relação a abordagem proposta.

Christie (1995) [CHR95] descreve um modelo de processo de software, nomeado ProNet, que é centrado no fluxo de atividades que compõem um projeto de software. O modelo descreve artefatos que devem estar disponíveis, condições que devem ser conhecidas antes que uma atividade comece, agentes participantes da atividade, e restrições que a atividade deve respeitar. Dois elementos complementam o modelo: armazenadores, que controlam a persistência de elementos de projeto ao longo do processo, e combinações, que permitem que vários elementos formem uma dependência composta para uma atividade. Dependências de atividades são representadas por relações entre atividades e outros elementos. Comparando com a proposta deste trabalho, além do uso de dependências para execução de atividades, usamos opções para o fluxo do processo após a execução da atividade.

Vasconcelos et al. (1998) [VAS98] e Araújo (1998) [ARA98] apresentam uma descrição de processos através de padrões, tornando possível a sua reutilização. Os padrões de processos documentam informações referentes a identificação, ao contexto, a solução proposta e aos padrões relacionados. As informações do padrão sobre a solução proposta são descritas através de um diagrama multi-nível, não cíclico, que exibe as atividades e suas dependências. Essas atividades manipulam documentos, recursos, métodos e ferramentas. A execução do processo é baseada em uma máquina de transição de estados. Esta abordagem difere da abordagem proposta por executar o processo de forma reativa, baseada em transição de estados.

3 Abordagem Proposta

A abordagem proposta consiste em mapear o processo modelado em predicados *Prolog* [POO98]. Esses predicados formam uma base de conhecimento do processo, que é utilizada por agentes inteligentes responsáveis pelo controle e execução do processo. A figura 1 descreve a arquitetura proposta. Cada região tracejada na figura 1 é detalhada nas sub-seções a seguir.

3.1 Modelagem do processo

Neste trabalho, não temos como objetivo definir mais uma forma de modelar processos, entretanto, alguns pré-requisitos para esta modelagem precisam ser considerados para que seja possível o mapeamento de processos em predicados *Prolog* (figura 1.a).

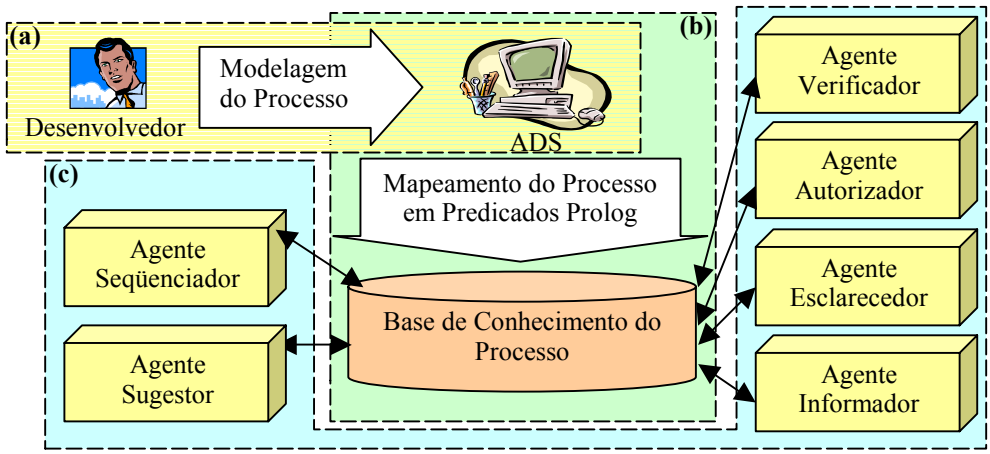
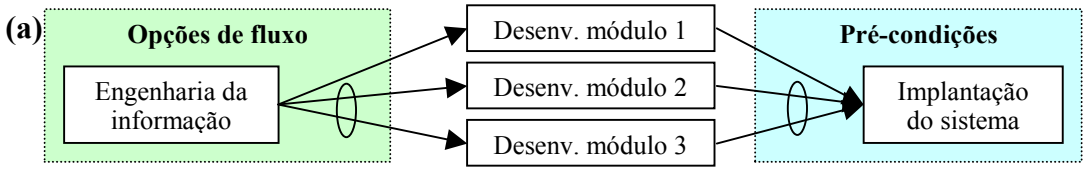


Figura 1: Arquitetura proposta

Para que o processo possa ser reutilizável, não podemos utilizar nenhuma definição que restrinja o número de camadas entre um processo abstrato (i.e., composto por outros processos) e um processo concreto (i.e., executável), pois um mesmo processo pode ser reutilizado em diferentes níveis de abstração. Desta forma, chamamos todos esses elementos de "componentes de processo", ou simplesmente "processos".

Para que a representação de processos e de seus sub-processos seja flexível, pode-se utilizar, por exemplo, grafos E/OU[1] [LIU89], tanto para oferecer opções de fluxo após a execução de um processo, quanto para definir pré-condições necessárias para sua execução. No processo descrito na figura 2.a, é possível a execução dos módulos 2 e 3 em paralelo ou do módulo 1[2].



(b)

```
precondicoes('Engenharia da informação', []).
precondicoes('Desenv. Módulo 1', ['Engenharia da informação']).
precondicoes('Desenv. Módulo 2', ['Engenharia da informação']).
precondicoes('Desenv. Módulo 3', ['Engenharia da informação']).
precondicoes('Implantação do sistema', ['Desenv. módulo 1',
                                        ['Desenv. módulo 2', 'Desenv. módulo 3']]).
opcoes('Engenharia da informação', ['Desenv. módulo 1',
                                   ['Desenv. módulo 2', 'Desenv. módulo 3']]).
opcoes('Desenv. Módulo 1', ['Implantação do sistema']).
opcoes('Desenv. Módulo 2', ['Implantação do sistema']).
opcoes('Desenv. Módulo 3', ['Implantação do sistema']).
opcoes('Implantação do sistema', []).
```

Figura 2: Representação de um processo exemplo

3.2 Representação do Processo

Para que a execução do processo assistida por computador seja viável, é necessário um mapeamento das informações do processo para uma representação interna. A representação através de predicados *Prolog* possibilita o uso de uma máquina de inferência, facilitando a obtenção de informações sobre a execução do processo. Desta forma, a base de conhecimento armazena as informações sobre o estado atual do desenvolvimento mapeadas em *fatos Prolog*

[1] Grafos E/OU são usados em inteligência artificial para modelar uma tarefa em termos de uma série de objetivos e sub-objetivos. Cada objetivo é representado por um nó e seus nós sucessores são objetivos mais primitivos. Os objetivos que só são satisfeitos quando todos os sub-objetivos imediatos forem satisfeitos são representados por arestas E. Os objetivos que são satisfeitos quando qualquer sub-objetivo imediato for satisfeito, são representados por arestas OU.

[2] O símbolo oval ligando as setas dos módulos 2 e 3 representa E no grafo E/OU.

(figura 1.b). A figura 2.b exibe o mapeamento em fatos *Prolog* do grafo E/OU do exemplo da figura 2.a.

3.3 Execução do Processo

A base de conhecimento definida anteriormente deve conter todas as informações referentes ao processo em execução. Essas informações são manipuladas de forma a possibilitar o acompanhamento da execução do processo. Os elementos responsáveis pela manipulação das informações situadas na base de conhecimento são os "agentes inteligentes", ou simplesmente "agentes".

Cada agente contém uma base de regras *Prolog* própria e faz a manipulação do conhecimento do processo unindo sua base de regras com a base de conhecimento do processo. O objetivo da manipulação varia de agente para agente (figura 3).

Agente	Objetivo
Verificador	Verificar, antes da execução, se a modelagem do processo foi feita corretamente e se é possível sua execução.
Seqüenciador	Permitir o término da execução de um processo e indicar quais processos entrarão em execução.
Esclarecedor	Explicar por que uma atitude não pode ser tomada pelo desenvolvedor.
Sugestor	Sugerir ao desenvolvedor uma atitude substituta no caso de não ser possível executar a atitude desejada.
Autorizador	Autorizar ou não o uso de algum recurso para um determinado desenvolvedor.
Informador	Fornecer informações gerenciais sobre o desenvolvimento do processo.

Figura 3: Exemplos de Agentes Inteligentes

4 Considerações Finais

A abordagem tradicional, baseada em transição de estados, age de forma reativa no acompanhamento do processo. Para contornar este problema, optamos pela utilização de agentes inteligentes, gerando uma arquitetura pró-ativa e dinâmica para o problema em questão.

Este trabalho está sendo desenvolvido no contexto do Projeto Odyssey [WER00], cujo objetivo é a construção de uma infra-estrutura de desenvolvimento de software baseada na reutilização de modelos de domínio, em desenvolvimento pelo grupo de reutilização de software da COPPE/UFRJ. Todas as ferramentas da Infra-estrutura Odyssey serão inicializadas e controladas pela máquina de processos a ser implementada. Está prevista a utilização da Infra-estrutura Odyssey como ferramenta de desenvolvimento de uma biblioteca de componentes reutilizáveis no domínio de processamento legislativo.

Neste momento, estamos detalhando a arquitetura e a previsão de defesa é para junho de 2001.

5 Referências Bibliográficas

[ARA98] Araújo, M.A.P; "Automatização do Processo de Desenvolvimento de Software nos Ambientes Instanciados pela Estação TABA"; Tese de Mestrado, Engenharia de Sistemas e Computação, COPPE/UFRJ, 1998.

[CHR95] Christie, A.; "Software Process Automation: The Technology and Its Adoption"; Springer-Verlag Publishing, Berlin, 1995.

[LIU89] Liu, L.; Horowitz, E.; "A Formal Model for Software Project Management"; IEEE Transactions on Software Engineering, Vol. 15, No. 10, Outubro 1989.

[POO98] Poole, D.; Mackworth, A.; Goebel, R.; "Computational Intelligence: A Logical Approach"; Oxford University Press, Nova York, 1998.

[VAS98] Vasconcelos, F.M.; Werner, C.M.L; "Organizing the Software Development Process Knowledge: An Approach Based on Patterns"; International Journal of Software Engineering and Knowledge Engineering, vol. 8 no. 4 1998, pp. 461-482.

[WER00] Werner, C. et. al; "Infra-estrutura Odyssey: Estágio Atual"; XIV SBES, Caderno de Ferramentas, João Pessoa, outubro 2000; (aceito para publicação).