Agente para Detecção de Potencial Perda de Sincronismo entre Artefatos de Modelagem e de Implementação

Roger Anderson Schmidt, Claudia Zambon, Anita Fernandes, Rudimar Dazzi

Mestrado em Computação Aplicada – Universidade do Vale do Itajaí (UNIVALI)
Caixa Postal 360 – 88.302-202 – Itajaí – SC – Brazil

rasflp@gmail.com,claudia.zambon@hotmail.com {rudimar, anita.fernandes, cheemann}@univali.br

Abstract. *This paper presents an overview of a software agent capable of preventing source code files from breaking the synchronization between its constructions and the modelling artifacts related to them. The purposed agent monitors a set of files, checks every change in their contents and sends notifications every time any potential impact is identified..*

1. Introdução

Modelos representam ideias em algum nível de abstração. Neste aspecto, constituem eficiente ferramenta de comunicação entre os diversos papéis atuantes em projetos de software. Contudo, para que preservem o seu valor ao longo do ciclo de vida do produto, devem continuar refletindo uma abstração da realidade de forma coerente. O gerenciamento de mudanças figura como um dos maiores desafios na abordagem da Model Driven Engineering (MDE), cujos métodos são encontrados na literatura em variados graus de automação [Paige et al. 2016].

O adequado gerenciamento depende de que relações de rastreabilidade (trace-links) capturem as dependências entre os artefatos criados, o que naturalmente tende a aumentar os esforços de desenvolvimento do software [Mäder and Egyed 2011]. Em contrapartida, pesquisas têm apontado efeitos positivos da manutenção destes links em vários aspectos [Delater and Paech 2013].

Porém, ainda que a rastreabilidade esteja disponível, a evolução dos artefatos de modelagem pode vir a ser comprometida durante o desenvolvimento dos projetos, por conta de vários fatores. Um cenário clássico se apresenta quando o tratamento para um novo requisito é implementado no código-fonte pelo desenvolvedor, sem que um ou mais diagramas afetados sejam também atualizados. No sentido de mitigar o risco exposto, o presente trabalho propõe o desenvolvimento de um agente de software capaz de identificar eventos de alteração no código-fonte e tomar ações apropriadas.

2. Proposta de Desenvolvimento

No contexto da Inteligência Artificial, um agente consiste em uma entidade que percebe seu ambiente através de sensores e age sobre ele através de atuadores. A descrição PEAS da Tabela 1 caracteriza a entidade em questão como um efetivo agente.

Tabela 1. Descrição PEAS para o agente

Desempenho	Ambiente	Atuadores	Sensores
Verificar todos os arquivos de código-fonte, localizar construções mapeadas por *trace-links*, baixo consumo de recursos	Estação de trabalho do desenvolvedor, servidor de integração, repositório do SCV	Mecanismo de varredura dos arquivos de código-fonte, acesso ao repositório do modelo, gravador de marcas para invalidação do código, envio de e-mails de notificação dos eventos	Identificador de mudança do arquivo de código-fonte, identificador de alteração da construção mapeada por *trace-link*

Como ilustra a Figura 1, a conexão entre os elementos de modelagem e as estruturas de código-fonte (trace-links) será estabelecida através de uma tabela de mapeamento de metadados (1) armazenada em banco de dados relacional (2) a ser utilizado como repositório. O mecanismo de *scanning* dos arquivos de código deve adotar o *framework* ANTLR (3) para o *parser* da gramática de diferentes linguagens.

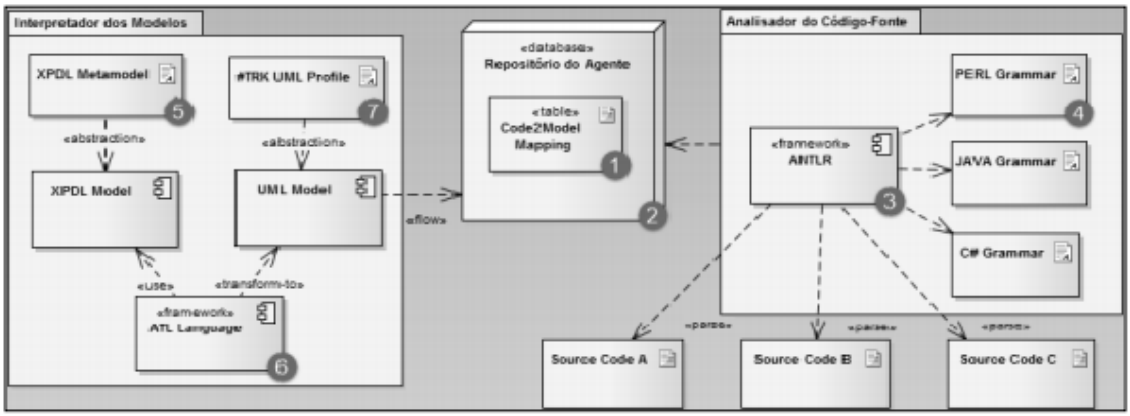

Figura 1. Componentes do agente de detecção

Quanto às alternativas de modelagem, o estudo deve fornecer suporte inicial aos diagramas comportamentais UML de Máquina de Estados e de Atividade, além do diagrama de Fluxo de Processos expresso em XPDL (5). Para uniformizar o modelo de dados do repositório, os diagramas em formato XPDL serão transformados através da linguagem ATL (6) para diagramas UML em conformidade com um Perfil (7) a ser especificado. Um conjunto de convenções de modelagem também deve ser adotado para minimizar as anotações no código-fonte necessárias à manutenção dos *trace-links*.

3. Considerações

Apesar de adotar UML como linguagem de carga do repositório, a proposta de pesquisa apresenta flexibilidade adicional por prever o suporte a linguagens de modelagem de processos no formato XPDL. O detalhamento de alguns desafios ainda se faz necessário, dentre eles: acesso ao repositório de Sistema de Controle de Versões e identificação das diferenças entre as revisões do arquivo. A avaliação dos resultados do trabalho será realizada na forma de um estudo de caso conduzido em uma organização de desenvolvimento de Linhas de Produtos de Software.

Referências

Delater, A. and Paech, B. (2013). Tracing requirements and source code during software development: An empirical study. International Symposium on Empirical Software Engineering and Measurement, p. 25–34.

Mäder, P. and Egyed, A. (2011). Do software engineers benefit from source code navigation with traceability? An experiment in software change management. 2011 26th IEEE/ACM International Conference on Automated Software Engineering, ASE 2011, Proceedings, p. 444–447.

Paige, R. F., Matragkas, N. and Rose, L. M. (2016). Evolving models in Model-Driven Engineering: State-of-the-art and future challenges. Journal of Systems and Software, v. 111, p. 272–280.