Proposta de uma Arquitetura para o Gerenciamento de Regras de Negócio em LPS com Base na MDA

Jaguaraci Batista Silva

Laboratório de Sistemas Distribuídos – Universidade Federal da Bahia – Salvador, BA

jaguarac@ufba.br

Abstract. *Are upheld in the SPL domain all business rules that define semantics links between entities for its completeness. By any change in business needs implies to keep of these rules up-to-date and is necessary to fix several application code to realize manually the necessary repair, making it a hard work for developers. The purpose of this works to create a new boarding to support software products in front of constant changes of business rules in this scenario, with thought at the reference architecture to manage them.*

Resumo. *Fazem parte do domínio de um uma LPS todas as regras de negócio que definem os laços semânticos entre entidades para a sua completude. Qualquer mudança nos requisitos de negócio implica na manutenção dessas regras e é preciso percorrer todo o código para realizar as mudanças necessárias manualmente, tornando esse trabalho árduo para os desenvolvedores. A proposta deste trabalho é criar uma nova forma de manter produtos de software frente às trocas constantes de regras de negócio neste cenário, com o suporte de uma arquitetura para gerenciá-las.*

1. Introdução

Há tempos, pessoas envolvidas com a fabricação de software vêm criando consciência de que para a satisfação dos clientes, é necessário construir produtos com qualidade. Porém não basta ter uma boa qualidade, se não for economicamente viável [Travassos *et al*, 2002]. A engenharia de domínio, bem como, outras técnicas visa à reutilização e está entre as técnicas mais relevantes para que se possa construir um software em menor tempo, maior confiabilidade e tendo como conseqüência um menor custo.

A Arquitetura Orientada a Modelos ou MDA (*Model-Driven Architecture*) [OMG, 2009] é uma abordagem de desenvolvimento de software dirigido por modelos (MDD ou *Model-Driven Development*) [Bragança e Machado, 2007] que utiliza modelos em diferentes níveis de abstração visando separar a arquitetura conceitual do sistema do seu modelo específico de plataforma. A definição de mecanismos de transformação permite a tradução dos modelos em código de linguagem de alto nível que, por sua vez, são compilados para geração do código executável da aplicação, facilitando o reuso e a manutenção das aplicações. Tais vantagens têm propiciado a crescente adoção da MDA como *framework* de desenvolvimento em diversos domínios de aplicação nos últimos anos.

Apesar dos avanços da comunidade científica na área de construção e reuso de modelos MDA [OMG, 2009] as empresas que trabalham utilizando linhas de produtos de software precisam construir um domínio de aplicação contendo as mesmas regras de negócio em várias plataformas, muitas vezes para prover funcionalidades similares em plataformas específicas. Qualquer mudança nos requisitos de negócio implica em esforço e custo de manutenção dos modelos específicos e independentes de plataforma. Para garantir que as regras de negócio ou propriedades específicas do domínio sejam atendidas pelo modelo gerado é comum aos desenvolvedores inserir o código que representa cada regra diretamente no modelo específico de plataforma (PSM) [OMG, 2009], o que tende a ser fastiento e sujeito a erros. Assim, a falta de mecanismos facilitadores que permitam a garantia de atendimento a essas regras, frente às constantes trocas de requisitos de negócio, torna-se um empecilho para a construção de aplicações confiáveis e de fácil reuso e manutenção.

Este trabalho propõe uma arquitetura para o gerenciamento eficiente de regras de negócio a partir de uma abordagem top-down usando a arquitetura orientada a modelos (MDA). A seção 2 e 3 apresenta os trabalhos relacionados, as pesquisas realizadas até o momento e os seus resultados. A seção 4 aduzi a respeito da arquitetura proposta. Por fim, as conclusões, contribuições e referências estão na parte final deste artigo.

2. Trabalhos Relacionados

A aproximação da MDA com uma família de LPS é um tema relevante e emergente onde se busca reduzir os custos de produção de software [Braga et al, 2007]. Em 4SRS ou *Four Step Rule Set*, foi concebido um framework utilizado pela Universidade do Minho. A sua finalidade é integrar partes de diferentes trabalhos na área de MDA e LPS visando à obtenção de regras de negócio a partir dos diagramas de casos de uso e derivando-as para diagramas de classe da UML [Braga et al, 2007].

Da aplicação conjunta da MDA em uma LPS também pode se beneficiar do melhoramento da gerência das variabilidades do domínio, sendo possível uma solução para o problema de sincronia entre os modelos específicos e independentes de plataforma. Um dos possíveis resultados seria a evolução das propriedades do domínio: conceitos, relações e restrições dentro de uma família de LPS de forma independente da arquitetura de implementação [Deelstra et al, 2003]. Em um dos trabalhos pesquisados, foi criada uma ontologia de domínio para modelar variabilidades de um software de apoio à configuração que inclui: componentes e *features* com suporte as estruturas de composição e seus atributos, interfaces, conexões e restrições. A ontologia utilizou a linguagem natural e um perfil UML com o suporte às inferências sobre o domínio para fornecer verificação semântica formal dos componentes [Asikainen et al, 2006].

Outra questão relevante é a necessidade de se prover técnicas e ferramentas para que se possa ter agilidade na tradução da lógica de negócios. Alguns trabalhos permitem a integração com ferramentas existentes no mercado, o que facilita o uso dessas técnicas nas empresas. Joukhadar (2008) criou um framework baseado em perfis da linguagem UML e utiliza diagramas de classes para capturar os conceitos, suas propriedades e relações e provê a especificação de regras de negócios em linguagem natural, de forma automática sem a ajuda do programador. Em uma pesquisa recente foi criada a

plataforma *SMICE* [Wu et al, 2010] baseada na MDA com objetivo de envolver tanto o cliente como os desenvolvedores na construção de softwares para a WEB. A plataforma oferece uma ferramenta para desenho dos componentes básicos dos processos, suas integrações e regras de negócio. O cliente define todo processo utilizando qualquer software específico para BPM [Santos et al, 2011] e a ferramenta converte o processo e suas regras de negócio do formato BPMN para WS-BPEL, simplificando a criação e manutenção de serviços na Web.

3. Resultados

Durante a construção de uma linha de produtos é preciso também ter um processo conciso para construir e verificar os pontos de variabilidade do domínio e a falta de um conjunto detalhado de atividades e ferramentas para apoiar esse processo foi um dos principais problemas enfrentados na pesquisa. Em um trabalho anterior Silva *e Saba* (2008) propuseram a criação de um processo de integração de componentes com base no guia *CMMI [SEI, 2002]* e uma metodologia de *Business Process Management* (BPM), que mescla a padronização de documentos, diagramas e representação de atividades junto com técnicas de modelagem de requisitos baseadas em casos de usos. A técnica de extração de requisitos com base em casos de uso foi utilizada por ser bastante difundida nas empresas desenvolvedoras de software. Aproveitando essa padronização foi criada uma aproximação com ontologias para a concepção de um processo a fim de guiar a construção, verificação e validação de domínios de aplicação [Silva, 2010].

A transformação de modelos MDA com base em ontologias de domínios requereu aprofundar os estudos sobre quais ferramentas e técnicas poderiam ser utilizadas para permitir a sua construção e verificação semântica. Silva e Pezzin (2007) definiram um toolkit utilizando ferramentas do mercado combinando o uso de modelos nas linguagens OWL e UML. Esse trabalho possibilitou a transformação do modelo conceitual ou ontologia de domínio no formato OWL (PIM) em um modelo para banco de dados (PSM). Realizando uma verificação e validação semântica do domínio antes da transformação dos modelos através de raciocínio automático, usando lógica de descrição sobre o arquivo OWL gerado a partir da ontologia [Haarslev, 2004].

Durante as inspeções de código foi constatado que os axiomas ou regras de negócios provocavam um alto grau de acoplamento, dificultando a sincronia, evolução e manutenção dos modelos. Por serem convertidas normalmente para estruturas de controle de uma linguagem de programação (e.g. IF-Then-Else), tornavam-se muitas vezes repetitivas. A proliferação de muitos pontos de controle de regras de negócio, frente às constantes trocas de requisitos, tornou-se um trabalho árduo para os desenvolvedores. Uma solução para este problema foi criar um processo para separação e validação de regras de negócio [Silva e Barreto, 2008] e a ferramenta *OWLtoASPECTJ* para realizar esta tarefa de forma automática [Silva, 2008]. A ferramenta transforma os axiomas, após serem verificados e validados semanticamente, em um novo modelo específico de plataforma (PSM) contendo apenas as regras de negócio. Através da programação orientada a aspectos [Laddad, 2003], as regras encontram dinamicamente e em tempo de execução, os métodos a serem interceptados, tornando possível a verificação da mesma regra em vários pontos da aplicação. Além disso, as regras geradas na linguagem *AspectJ* podem ser reutilizadas em outras plataformas que

utilizam a linguagem Java [Soares *et al*, 2002] agregando valor ao uso da MDA em uma família de LPS.

4. Proposta para o Gerenciamento de Regras de Negócio MDA em LPS

A Figura 1 apresenta a proposta em forma de diagrama de componentes da UML para facilitar a compreensão. A arquitetura ou ambiente de construção e verificação de modelos MDA será composto por 4 componentes: 1 - Ferramenta de Construção de Domínio, 2 - Ferramenta de Verificação Semântica de Regras de Negócio, 3 - Ferramenta de Separação do Domínio da aplicação e 4 - Repositório de Regras de Negócio ou Base de Conhecimento.

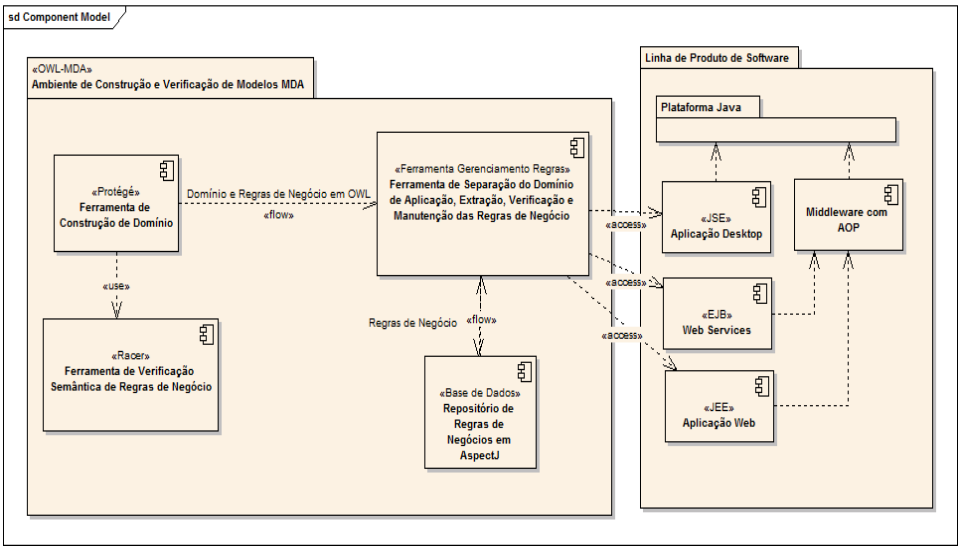

Figura 1 – Arquitetura Proposta para o Gerenciamento das Regras de Negócio em LPS usando MDA.

A ferramenta de edição de ontologias representada pelo componente <Protégé> [Protégé, 2011] será designada para a construção da ontologia de domínios da LPS. O primeiro passo é a criação do domínio conceitual da aplicação contendo: as entidades, suas propriedades e relacionamentos e os axiomas. O artefato base produzido por esse componente será uma ontologia de domínio consonante com os requisitos da aplicação. A construção da ontologia será guiada a partir de casos de uso que identificam os requisitos funcionais de uma família de LPS [Silva e Barreto, 2008].

O domínio conceitual da aplicação ou ontologia de domínio deverá sofrer inferências para a sua verificação e validação através da ferramenta de verificação semântica de regras de negócio, representado na Figura 1 pelo componente <Racer> [Haarslev, 2004]. Após serem validados, os axiomas (regras de negócios) serão transformados do formato OWL (*Ontology Web Language*) (PIM) para uma linguagem de programação orientada a aspectos (PSM) com a ajuda da ferramenta de separação do domínio da aplicação ou componente <Ferramenta de Gerenciamento de Regras> OWLtoAspectJ [Silva, 2008].

As regras de negócio extraídas da ontologia de domínio farão parte de um único modelo (PSM) que servirá para o tratamento da lógica de negócio e serão armazenadas em uma base de conhecimento ou repositório de regras de negócios, reproduzido na gravura pelo componente <Base de Dados>. Os outros modelos serão criados de acordo

com as necessidades não-funcionais dos casos de uso. No exemplo da LPS proposta na Figura 1, seria necessário gerar 3 modelos específicos para a plataforma Java: o primeiro para compor uma aplicação que irá executar em computadores desktops <JSE>, o segundo para uma aplicação Web <JEE> e o último serviços Web <EJB>.

A arquitetura deverá provê o suporte à atualização das regras de negócios de forma centralizada e automática. A ferramenta de separação do domínio da aplicação fará a extração das novas regras da ontologia corrigida (Seção 3), realizará o merge na base de conhecimento das regras antigas e atuais, realizando busca semântica através de taxonomias e finalizará com a inserção das novas regras na LPS.

Com a utilização do paradigma de programação orientada a aspectos [Kiczales, 1996], os modelos de regras e específicos de plataforma podem ser combinados para que a aplicação possa interceptar automaticamente as regras em tempo de execução sem que o desenvolvedor necessite conhecer os pontos de reparação do software (Seção 3). Seja usando diretamente a linguagem AspectJ, componente <JSE>, ou um *Midleware* com o seu suporte, componentes <EJB> e <JEE>.

5. Conclusão

Gerenciar as regras de negócio para um produto de software não é trivial e tal dificuldade aumenta substancialmente em uma linha de produtos de software. Neste âmbito, pensar nos modelos MDA para apoiar a gerência de tais regras torna-se uma alternativa relevante. O artigo propõe uma arquitetura para melhorar o gerenciamento de regras de negócio no contexto de linhas de produtos de software, utilizando uma abordagem MDA. Foram definidos os componentes que integram essa arquitetura, bem como, os resultados alcançados até o momento. Espera-se em uma próxima etapa validar a arquitetura em um estudo de caso real de um sistema para uma grande instituição do mercado financeiro usando as tecnologias Java (e.g. JSE, EJB e JEE) [Sun, 2011] por já ser bastante utilizada no mercado.

Referências

Asikainen,T.,Mannisto, T.,Soininen, T.. (2006) “Kumbang: A domain ontology for modelling variability in software product families”. Advanced Engineering Informatics, Vol. 21 (2006) 23–40, Springer.

Bragança, A. Machado, R. J. (2007) “Model Driven Development of Software Product Lines”. IEEE 6th International Conference on The Quality of Information and Communication Technology (QUATIC), 2007, Lisboa, Portugal.

Desltra, S. Sinnema, M., Gurp, J. V., Bosch, J.. (2003) “Model Driven Architecture as Approach to Manage Variability in Software Product Families”. Workshop on Model Driven Architecture: Foundations and applications, 2003, Enschede, Holanda.

Gimenes, I. M., Travassos, G. H.. (2002) O enfoque de Linha de Produto para Desenvolvimento de Software. In: Sociedade Brasileira de Computação; Ingrid Jansch Porto. (Org.). XXI JAI - Livro Texto. Florianopolis: Sociedade Brasileira de Computação, v. 2, p. 1-32.

Haarslev, V., Muller, R.. (2004) “Racer’s User Guide and Reference Manual”. Versão 1.7.19.

Joukhadar, A., Al-Maghout, H. (2008) "Improving agility in business applications using ontology based multilingual understanding of natural business rules", International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA 2008). Damascus, Siria.

Kiczales, G. (1996). "Aspect-oriented programming". ACM Computing Surveys, 28A(4), 1996.

OMG (2009). MDA Guide version 1.0.1. Formal Document: 03-06-01. Disponível em: http://www.omg.org/cgi-bin/apps/doc?omg/03-06-01.pdf. Acesso: Maio/2011.

Protégé. "Ontology Editor and knowledge-base framework". http://protege.stanford.edu/. Acesso: Maio/2011.

Laddad, R.. "AspectJ in Action, Practical Aspect-Oriented Programming". Manning,

ISBN 1-930110-93-6. 2003.

Santos, A.G., Santos F. G., Mendes F. A. T., Cruz G. M., Silva J. B., Freitas J. V. V. B., Santana M. R., Pastor S. O. (2006) "Metodologia de Processos de Negócios", Universidade Federal da Bahia, Programa de Residência em Software com foco em e-government, disponível em http://twiki.im.ufba.br/bin/view/Residencia/Trabalhos. Acesso: Maio/2011.

SEI CMMI (2002) "The Capability Maturity Model for Software". Version 1.1-CMU/SEI-2002-TR-012, March.

Silva, J. B., Saba, H. (2008). "Modelagem das áreas de Processo do CMMI usando uma metodologia de BPM e notações do SPEM". 34º Congresso Infobrasil TI & Telecom 2008, Fortaleza.

Silva, J. B., Barreto, L. P.. (2008). "Separação e Validação de Regras de Negócio MDA Através de Ontologias e Orientação a Aspectos". Simpósio Brasileiro de Componentes, Arquitetura e Reuso de Software 2008, Porto Alegre.

Silva, J. B.. (2010). "PROCEDA – Um Processo para Construção e Verificação Semântica de Domínios em uma Linha de Produtos de Software". Universidade Federal da Bahia. Relatório de Pesquisa, Agosto/2010.

Silva, J. B., Pezzin J. "The Formal Verification of an Application Conceptual Model Using MDA and OWL". World Congress on Engineering and Computer Science (WCECS 2007), San Francisco, 2007.

Silva, J. B. "OWLtoAspectJ: A Tool for Transformation from Conceptual Rules of Domain to Aspects". I Seminário de Pesquisa em Ontologia no Brasil, Niterói, 2008.

Soares, S. Borba, P. "Programação Orientada a Aspectos em Java". VI Simpósio Brasileiro de Linguagens de Programação, Rio de Janeiro, 2002.

Sun (2011). Sun Microsystems (2011). "Java Technology". http://www.java.sun.com. Acesso: Maio/2011.

Wu, M., Jin, C., Ying, J.. (2010). "SMICE: A Platform Supports Business Process Modeling and Integration". 2nd IEEE International Conference on Information Management and Engineering (ICIME2010). Chengdu, China.