Composição de UML *Profiles*

Kleinner Silva Farias de Oliveira[1]
Orientador: Toacy Cavalcante de Oliveira[1]

[1] Faculdade de Informática – Pontifícia Universidade Católica
do Rio Grande do Sul (PUC-RS) – Av. Ipiranga 6681 – Prédio 32 – CEP 90619-900
Porto Alegre – RS – Brasil
{ksoliveira,toacy}@inf.pucrs.br

Nível: Mestrado
Ano de Ingresso: 2006
Previsão de Conclusão: Dezembro de 2007
Aprovação da Proposta: 15/12/2006

Resumo. *Com o sucesso da Model Driven Architecture (MDA) e da Unified Modeling Language (UML), os modelos estão substituindo o código como o principal artefato de desenvolvimento de software. Em MDA, a transformação e composição de modelos são duas atividades essenciais. A transformação de modelos tem sido amplamente pesquisada e documentada, enquanto que a composição de modelos precisa de uma maior investigação. Com a MDA, surgiram três desafios: (i) criar linguagens de modelagem específicas de domínios (DSML); (ii) compor DSML; (iii) compor modelos representados em DSML. A UML permite a construção de DSML através UML Profiles, porém não oferece um mecanismo capaz de compor estes profiles. Neste contexto, este trabalho apresenta uma proposta de mecanismo de composição de UML Profiles fundamentado em regras de transformação de modelos, regras de composição, estratégias de composição, regras de correspondências e composição de modelos baseado na assinatura. Um modelo formal deste mecanismo será construído utilizando a linguagem de modelagem estrutural Alloy o que tornará possível realizar uma análise automática do modelo usando Alloy Analyser. Além disso, será construído uma ferramenta com o objetivo de validar este mecanismo.*

Palavras-chave: *Composição de Modelos, Unified Modeling Language, UML Profiles*

1. Caracterização do problema

A tendência atual da Engenharia de Software consiste em tratar, através do uso de linguagens de modelagem, programas no nível dos seus conceitos, com o objetivo de simplificar o projeto, a evolução e a manutenção. Além disso, deseja-se aumentar a capacidade de adaptação frente às rápidas mudanças das tecnologias e atender às exigências do *time-to-market*. Esta tendência propõe o aumento do nível de abstração no qual os software são projetados e desenvolvidos, mudando o foco do código para os modelos. Segundo Bran Selic em [Selic 2003], estas mudanças são comparadas com o salto da linguagem *assembly* à terceira geração de linguagem de programação. *Model Driven Architecture* (MDA) [OMG 2003] é um exemplo desta tendência, a qual trata-se de uma abordagem *Model Driven Development* (MDD) [Selic 2003] da *Object Management Group* (OMG) [OMG 2007].

No contexto de MDA, ao contrário de se construir software através da integração de componentes de software existentes, ou seja, colocando-os para interoperarem, o objetivo é construir novas aplicações ricas em funcionalidades a partir da transformação e composição dos modelos dos componentes de software existentes. Na MDA, a transformação e composição de modelos são duas atividades essenciais e tem se tornado cada vez mais importante. A transformação de modelos tem sido pesquisada, documentada e tem alcançado avanços importantes, por outro lado, a composição de modelos necessita de maiores investigações e esforços para solucionar significantes problemas. De acordo com Jean Bézivin em [Bézivin et al. 2006], a composição de modelos trata-se de uma nova área de pesquisa e que se encontra na sua "infância". Com a MDA, surgiram três desafios: (i) criar linguagens de modelagem específicas de domínios (DSML); (ii) compor DSML; (iii) compor modelos representados em DSML.

A UML permite a criação da DSML através de UML *Profiles*, porém seu mecanismo de composição (*Package Merge* [OMG 2007]) apresenta inconsistência, ambigüidade e define regras de composição gerais as quais são aplicadas aos *metatypes* do metamodelo que são mais freqüentemente utilizados, por exemplo: Classes, Associations, Properties e entre outros. Não existe um definição em [OMG 2007] de como deve ser realizada a composição de *metatypes*, como: Packages, Profiles, State Machines, Use Case, etc. Neste sentido, surge a questão de pesquisa deste trabalho: "Como realizar a composição de UML *Profiles*?"

A semântica do mecanismo de composição da UML não é bem definida, sendo difícil de especificar o que ele realmente faz, e quais as suas propriedades. Além disso, as mais populares ferramentas de modelagem (como IBM Rational Software Modeler) não implementam o mecanismo de composição da UML. De acordo com Rumbaugh, Jacobson and Booch [Rumbaugh et al. 2005]: "o uso do mecanismo de composição da UML pode gerar confusões e deve ser evitado se possível".

1.1. Objetivo

Este trabalho tem como objetivo desenvolver um mecanismo de composição sintática dos UML *Profiles*. Como objetivos específicos, tem-se:

1. Identificar as atividades necessárias e as fases nos mecanismos de composição.
2. Elaborar um modelo formal deste mecanismo. Isto será construído utilizando a linguagem de modelagem estrutural *Alloy* [Jackson 2007] o que tornará possível

realizar uma análise automática do modelo usando o *Alloy Analyser* [Jackson 2007].
3. Especificar uma ferramenta de software e desenvolver um protótipo tendo como base a especificação do mecanismo de composição.
4. Elaborar um estudo de caso utilizando o mecanismo de composição.

2. Fundamentação teórica

A UML é uma linguagem visual para especificação, construção e documentação de artefatos de software. Trata-se de uma linguagem de propósito geral que pode ser usada tanto em orientação a objeto quanto no paradigma orientado a componentes. Além disso, pode ser aplicada para uma grande diversidade de domínios de aplicação (por exemplo, finanças, saúde, telecomunicação, etc) e plataformas de implementação (por exemplo, Java, Struts, .NET, J2ME, etc.). A UML tem sido adotada como linguagem de modelagem padrão sendo usada tanto na indústria quanto na academia. Ela foi definida usando a abordagem de metamodelagem e seu metamodelo possui um arquitetura que leva em consideração: modularidade, extensão, reúso e *layering*.

Em determinadas situações a UML não é apropriada para modelar aplicações para algum domínio específico. Este é o caso, por exemplo, quando a sintaxe ou a semântica dos elementos da UML não são capazes de expressar conceitos específicos de um sistema em particular, ou quando é necessário restringir ou customizar algum elemento da UML que é de caráter geral. Sendo necessário, desse modo, a construção de uma DSML.

A OMG define duas possíveis formas para definir uma linguagem específica de domínio. A primeira é baseada na definição de uma nova linguagem usando os mecanismos fornecidos pela OMG para definir linguagem visual baseadas em objetos. Assim, a sintaxe e a semântica dos elementos da nova linguagem são definidas para se adaptar a características específicas do domínio (este mecanismo é o mesmo usado para construir a UML). A segunda alternativa é baseada na especialização da UML, na qual os elementos da linguagem são especializados, colocando novas restrições, enquanto respeita o metamodelo da UML deixando a semântica original dos elementos inalteradas (por exemplo, as propriedades dos pacotes, classes, atributos, operações, etc. permanecem iguais, porém novas restrições são adicionadas as suas originais definições e relacionamentos)

Na segunda abordagem, a UML fornece mecanismos de extensão (*stereotypes*, *tagged values* e *constraints*) para especializar seus elementos, permitindo customizar extensões da UML para domínios de aplicações específicos. Estas customizações são conjuntos de extensões agrupadas em UML *Profiles*, os quais fornecem um mecanismo de extensão *lightweight* para a UML. Este mecanismo permite especializar as metaclasses do metamodelo da UML através de uma extensão conservativa.

A composição de modelos é definida como um tipo especial de transformação de modelos, que tem dois (ou mais) modelos de entrada M_A e M_B e um conjunto de operações que visão combinar os conteúdos dos modelos obtendo um modelo M_{AB} como resultado da composição (M_A+$M_B \rightarrow M_{AB}$).

Com o objetivo de promover o reúso e realizar um regerenciamento de complexidade, a UML agrupa seus conceitos em pacotes. Estes pacotes podem ser estendidos e ter novos recursos através do processo de composição. Para isto, os pacotes são definidos

como modelos de entradas no mecanismo de composição de modelos definido pela UML, o *Package Merge*.

3. Caracterização da contribuição

Nas seções anteriores definimos os desafios e os problemas encontrados na área de composição de modelos e a ausência de uma definição de uma mecanismo de composição de *profiles*. Desse modo, a abordagem desenvolvida apresenta uma solução para estes problemas constituindo a base da contribuição.

A partir da especificação das atividades necessárias para realizar a composição de UML *Profiles* e da definição do fluxo entre elas, busca-se disponibilizar um guia para realizar a composição de modelos. Desta forma, de acordo com os modelos a serem composto (por exemplo, UML *Profiles*, diagram de classes), tem-se um embasamento de como deve se proceder a composição destes modelos. Este guia servirá como base para futuros esforços na área de composição de modelos, assim como, para o desenvolvimento de aplicações neste contexto.

Além disso, com a construção de um metamodelo de composição de modelos que estende o metamodelo da UML é possível especificar o relacionamento de composição entre dois *profiles*. Para isto está sendo desenvolvido: (i) um metamodelo que define a sintaxe e semântica do relacionamento de composição; (ii) regras de boa formação especificando as restrições no relacionamento de composição; (iii) descrição da semântica de composição.

Uma importante contribuição é a construção do mecanismo de composição baseado em regras de composição, regras de transformação de modelos, regras de correspondência, assinatura de modelos e estratégias de composição. Com isso, é possível desenvolver outros trabalhos adaptando estas regras e estratégias para outros contextos. Além disso, a construção de uma ferramenta coloca em prática o mecanismo de composição proposto, fornecendo um ambiente de composição de UML *Profiles*.

4. Etapas e estado atual do trabalho

Para a realização da composição de UML *Profiles* foram definidas as seguintes etapas, a serem realizadas na ordem em que se apresentam.

Etapa 1: realização de levantamento bibliográfico sobre composição de modelos, UML *Profiles*, UML e trabalhos relacionados. Nesta etapa busca-se o embasamento teórico necessário sobre cada item citado. Assim como, coletar informações sobre as formas e técnicas utilizadas para a realização de composição e fazer um levantamento das estratégias usadas para solucionar os conflitos que surgem durante a composição. Esta etapa é continuamente refeita com o objetivo verificar o impacto de outros trabalhos na área na abordagem proposta.

Etapa 2: levantamento e definição das atividades necessárias para realizar a composição dos *profiles*. Além disso, construção do mecanismo de composição e definição da extensão do metamodelo para especificar o mecanismo de composição. Já foram definidas e identificadas algumas atividades que são agrupadas em quatro fases: (i) fase inicial; (ii) fase de comparação; (iii) fase de composição; (iv) fase de pós-composição. O mecanismo de composição baseado em regras de composição,

regras de transformação de modelos, regras de correspondências e estratégia de composição está sendo definido e encontra-se na sua fase final. Além disso, a definição da extensão do metamodelo da UML também encontra-se em seu estado final.

Etapa 3: consistirá em realizar uma especificação formal do mecanismo de composição proposto na Etapa 2. Para isto será utilizado a linguagem de modelagem estrutural *Alloy* [Jackson 2007] o que tornará possível realizar uma análise automática do modelo usando o *Alloy Analyser* [Jackson 2007]. Esta etapa ainda não foi inicializada.

Etapa 4: serão realizadas duas atividades: (i) análise e projeto da ferramenta; (ii) implementação da ferramenta de composição de *profiles*. Esta etapa ainda não foi inicializada.

Etapa 5: após o desenvolvimento da ferramenta, será realizado um estudo de caso com o objetivo de pôr em prática o que foi desenvolvido nas etapas anteriores. Esta etapa ainda não foi inicializada.

5. Trabalhos relacionados

Em [Reddy et al. 2006], os requisitos funcionais de um sistema são modelados separadamente de seus *cross-cutting concerns*, que são vistos como elementos aspectuais. Os modelos dos requisitos funcionais e dos aspectos são combinados através de uma processo chamado *composition* com o objetivo de obter uma visão integrada dos modelos. Sendo a composição dos modelos conduzida por diretivas de composição e baseada na assinatura dos modelos.

No contexto de programação orientado a objeto (POO) são implementados mecanismos de composição com o objetivo de promover a herança. Uma classe filha é resultado da combinação dos atributos, relacionamentos e comportamentos com os da sua superclasse. A forma de implementar este mecanismo é diferente dependendo se a linguagem suporta herança múltipla ou não. Por exemplo, C++ suporta herança múltipla enquanto Java não suporta, logo os mecanismos de composição nestas linguagens são implementados de forma diferente.

Em [Ledeczi et al. 2001], é apresentado uma extensão da UML para a composição de metamodelos com o objetivo de solucionar limitações da UML. Foi desenvolvido três operadores UML para uso na composição de metamodelos: (i) operador de herança de implementação; (ii) operador de herança de interface e (iii) um operador de equivalência.

Com o objetivo de melhorar a definição da semântica de composição, solucionar a ambigüidade e inconsistência apresentada no mecanismo de composição da UML (*Package Merge*), Alanna Zito em [Zito 2006] propõe uma nova definição e formalização do *Package Merge*.

6. Avaliação dos resultados

Pretende-se avaliar este trabalho realizando um estudo comparativo entre os resultados obtidos pela implementação do mecanismo de composição na ferramenta desenvolvida e os resultados obtidos por simulação. A partir destes resultados serão realizados eventuais ajustes e correções na ferramenta.

Espera-se validar este trabalho a partir da construção de um estudo de caso que vise compor os *profiles* definidos pela OMG ou os disponíveis na literatura. Desta forma, os resultados obtidos com o uso da abordagem poderão ser avaliados e a abordagem refinada. Além disso, o mecanismo de composição proposto terá uma validação formal com *Alloy Analyser*.

7. Considerações finais

A UML permite a construção de DSML através UML *Profiles*, porém não oferece um mecanismo capaz de compor estes *profiles*, como já mencionado. Com isso, este trabalho representa um esforço na relacionamento entre DSML e composição de modelos. Foram identificados alguns trabalhos que apresentam propostas para composição de modelo, porém são trabalhos ainda em estado inicial e que não representam esforços direcionados para a composição de DSML.

Referências

Bézivin, J., Bouzitouna, S., Del Fabro, M. D., Gervais, M. P., Jouault, F., Kolovos, D., Kurtev, I., and Paige, R. (2006). A Canonical Scheme for Model Composition. In *Proceedings of the European Conference on Model Driven Architecture - Foundations and Applications- (ECMDA-FA'06)*. LNCS, Springer-Verlag.

Jackson, D. (2007). The Alloy Analyzer. http://alloy.mit.edu/, Último acesso em 22 abril de 2007.

Ledeczi, A., Nordstrom, G., Gabor Karsai, and Volgyesi, P. (2001). On Metamodel Composition. In *IEEE International Conference on Control Applications*, Cidade do México, México.

OMG (2003). *MDA Guide Version 1.0.1, Object Management Group (OMG)*. http://www.omg.org/docs/omg/03-06-01.pdf.

OMG (2007). Object Management Group (OMG). http://www.omg.org/, Último acesso em 22 abril de 2007.

OMG (2007). *Unified Modeling Language: Infrastructure version 2.0*. Object Management Group. http://www.omg.org/cgi-bin/apps/doc?formal/07-02-03.pdf.

Reddy, Y., France, R., Straw, G., J. Bieman, N. M., Song, E., and Georg, G. (2006). Directives for Composing Aspect-Oriented Design Class Models. In *Transactions of Aspect-Oriented Software Development*, volume 1.

Rumbaugh, J., Jacobson, I., and Booch, G. (2005). *The Unified Modeling Language Reference Manual*. Object Technology Series, Addison-Wesley, Second edition.

Selic, B. (2003). The Pragmatics of Model-Driven Development. *IEEE Software*, 20(5):19–25.

Zito, A. P. (2006). UML's Package Extension Mechanism: Taking a Closer Look at Package Merge. Master's thesis, School of Computing, Quenn's University Kingston, Ontario, Canadá.