Odyssey-MEC: Uma Ferramenta para o Controle da Evolução no Contexto do Desenvolvimento Dirigido por Modelos

Chessman Corrêa[1], Leonardo Murta[1,2], Claudia M. L. Werner[1]

[1]PESC/COPPE - Universidade Federal do Rio de Janeiro – Rio de Janeiro, RJ – Brasil

[2]Instituto de Computação - Universidade Federal Fluminense – Niterói, RJ – Brasil

{chessman,werner}@cos.ufrj.br

leomurta@ic.uff.br

Abstract. *Models are used as first class artifacts to create software in Model-driven Development. These models may evolve independently due to modifications, making them inconsistent with each other. However, inconsistencies cannot be allowed because these models represent the same software. In this scenario, versioning is not sufficient to control the evolution of these interconnected models. This paper presents Odyssey-MEC, a client/server tool that supports server side transformations, synchronization and versioning to control interconnected models evolution.*

Resumo. *Modelos são usados como artefatos fundamentais para a criação de software no Desenvolvimento Dirigido por Modelos. Esses modelos podem evoluir independentemente em função de modificações, tornando-se inconsistentes. Porém, uma vez que representam o mesmo software, inconsistências não podem ser permitidas. Este trabalho apresenta o Odyssey-MEC, uma ferramenta com arquitetura cliente-servidor que oferece suporte à transformação, sincronização e versionamento de modelos para controlar a evolução consistente de modelos inter-relacionados.*

Palavras-chave. *Model-driven Architecture, Model-driven Development, Evolução de Software, Sincronização de Modelos, Versionamento de Modelos*.

1. Introdução

O **Desenvolvimento Dirigido por Modelos** (*MDD – Model Driven Development*) (Soley, 2000) é caracterizado pelo uso de modelos como principais artefatos para a criação de *software*. A **MDA** (*Model Driven Architecture* – Arquitetura Orientada a Modelos) (Soley, 2000) é o *framework* da OMG (*Object Management Group*) para o MDD. No MDA, um modelo independente de plataforma (PIM – *Platform Independent Model*) é usado para gerar um modelo para uma plataforma específica (PSM – *Platform Specific Model*) através de uma transformação. Exemplos de plataformas para as quais o PSM pode ser gerado são o *Java EE* e o *.NET*. O PSM é usado no final do processo para a geração de código fonte para a respectiva plataforma.

Na abordagem *elaboracionista*[1], modelos MDA em diferentes níveis de abstração podem ser modificados independentemente, principalmente quando existem pessoas com papéis diferentes trabalhando no mesmo projeto ou quando são usadas ferramentas distintas para a criação do PIM e do PSM. Neste cenário, modificações realizadas em um PSM para uma plataforma específica que estão no nível de abstração do PIM fazem com que o modelo fique desatualizado. Nesse caso, além dos modelos tornarem-se inconsistentes, a desatualização do PIM pode dificultar ou inviabilizar a geração do mesmo sistema para outra plataforma tecnológica. Assim, as diferentes representações do mesmo software podem evoluir de forma independente, ou então, algumas partes podem evoluir enquanto outras não.

Para solucionar esse problema, este artigo apresenta o Odyssey-MEC, uma ferramenta para o controle da evolução consistente de modelos no contexto do MDD com MDA. Este artigo está focado principalmente nos aspectos de implementação. Corrêa *et al*. (2008) descrevem maiores detalhes sobre a abordagem que motivou a construção da ferramenta.

O artigo está organizado da seguinte maneira: a Seção 2 apresenta a arquitetura do Odyssey-MEC; a Seção 3 apresenta as principais funcionalidades, enquanto a Seção 4 apresenta alguns aspectos da implementação. A Seção 5 demonstra o uso da ferramenta através de um exemplo. A Seção 6 apresenta alguns trabalhos relacionados. Finalmente a Seção7 descreve as conclusões, limitações e trabalhos futuros.

2. Arquitetura do Odyssey-MEC

O controle da evolução de modelos inter-relacionados requer um mecanismo que realize a sincronização automática e o versionamento de todos os modelos sempre que um modelo é modificado. Além disso, é necessário permitir que os profissionais envolvidos possam realizar modificações nos diferentes modelos de forma independente. Para atender a esses requisitos, o Odyssey-MEC utiliza uma arquitetura cliente-servidor em que o processo de transformação e sincronização de modelos é realizado no lado servidor. A arquitetura do Odyssey-MEC é apresentada na Figura 1.

O componente *Odyssey-VCS* (Murta *et al.*, 2008) é usado para fazer o versionamento de modelos. O componente *Odyssey-MDA* (Maia, 2006) faz a transformação de modelos. A *Máquina de Sincronização (MS)* controla o processo de transformação e sincronização de modelos inter-relacionados. O *Gerenciador de Transações* garante que todo o processo de versionamento, transformação e sincronização ocorra dentro do contexto de uma única transação.

O Odyssey-MEC possui três tipos de repositórios: modelo, mapas de transformação e registros de transformação (rastros). O *repositório de modelo* mantém o histórico da evolução de um modelo específico. Desse modo, para cada modelo é criado um repositório separado. No contexto da MDA, são criados repositórios para o PIM e para o PSM de cada plataforma. O *repositório de mapas de transformação* mantém os mapas que são usados pelo Odyssey-MDA para fazer a transformação de modelos.

---

[1] A abordagem *elaboracionista* considera a geração do PSM a partir do PIM e do código fonte a partir do PSM.

Finalmente, o *repositório de registros de transformação* mantém os rastros gerados durante o processo de transformação.

O processo de transformação e sincronização de modelos é disparado a partir de *hooks* do Odyssey-VCS. O *hook pós-login* prepara o Odyssey-VCS para fazer o versionamento dos modelos de um projeto específico e registra os *hooks* que disparam o processo de transformação e sincronização. O hook *pré-checkin* é usado para iniciar a transação assim que um desenvolvedor faz o *check-in* de um modelo. O *hook pós-merge* faz a recuperação de informações relacionadas aos perfis aplicados ao modelo. O *hook pós-checkin* aciona a máquina de sincronização, que executa o Odyssey-MDA para fazer transformações e sincroniza os modelos do projeto.

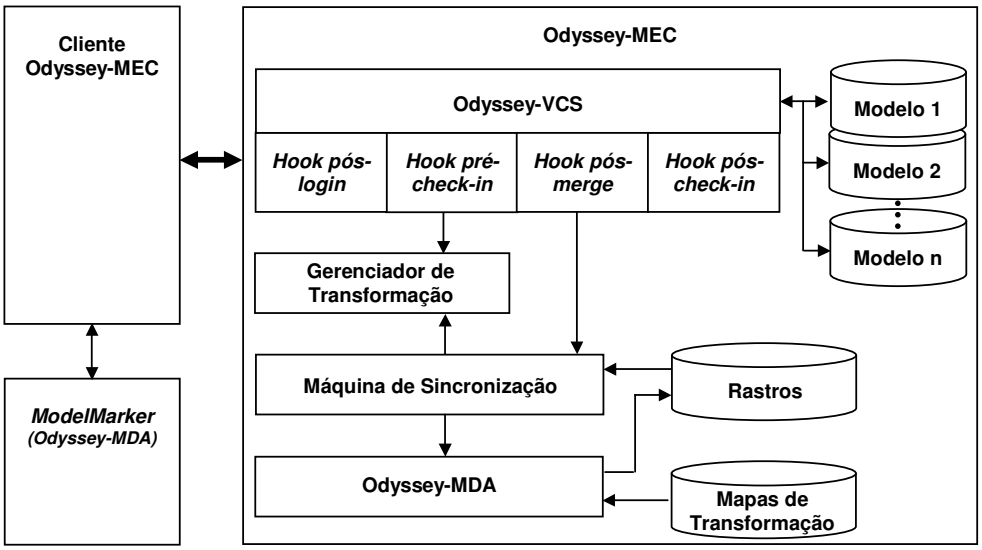

Figura 1. Arquitetura do Odyssey-MEC

O aplicativo cliente do Odyssey-MEC é usado para a realização das tarefas relacionadas com o controle de versão, como *check-in* e *check-out* de modelos, entre outras operações. O programa utiliza o *ModelMarker* (Maia, 2006) como componente para permitir ao engenheiro de software inserir as marcações necessárias aos modelos de modo que o Odyssey-MDA possa executar a transformação.

3. Principais Funcionalidades

3.1. Criação de um Projeto MDD

Para que o Odyssey-MEC possa controlar a evolução dos modelos de um projeto MDD, o primeiro passo é a criação do projeto. Nesta tarefa, o desenvolvedor informa o nome do projeto, a localização do repositório do projeto e os respectivos modelos. Para um projeto usando a MDA, além do PIM, o desenvolvedor deve especificar o PSM de cada plataforma do projeto.

Quando um desenvolvedor efetua o *login* pela primeira vez no repositório do projeto, o Odyssey-MEC gera um repositório para cada modelo, assim como o repositório de registros de transformação.

3.2. Marcação de Modelos

O modelo-fonte deve receber as marcações necessárias para que o componente de transformação Odyssey-MDA possa gerar os PSMs para as plataformas do projeto. Essas marcações são estereótipos e valores etiquetados que são aplicados aos elementos do modelo. A partir delas, a máquina de transformação identifica quais elementos devem ser gerados para cada PSM.

O Odyssey-MEC disponibiliza o *ModelMarker* (Maia, 2006) para que o desenvolvedor insira as marcações no modelo caso a ferramenta CASE usada não forneça o suporte necessário.

3.3. Versionamento de Modelos

O Odyssey-MEC usa o Odyssey-VCS para fazer o versionamento de todos os modelos de um projeto MDD. O processo é iniciado quando um desenvolvedor faz o *check-in*[2] de um modelo. Se não houver conflitos durante o versionamento do modelo, a máquina de sincronização aciona o Odyssey-MDA para executar a transformação de modo a obter o modelo-alvo (um PSM, por exemplo). Após o processo de sincronização, o Odyssey-MEC faz o *check-in* do modelo-alvo, fazendo com que o Odyssey-VCS gere uma nova versão do modelo, se necessário. Se existir outro modelo relacionado ao que acabou de ser versionado, o processo se repete para esse modelo.

3.4. Transformação e Sincronização de Modelos

Transformação e sincronização de modelos são realizadas quando o *hook pós-checkin* do Odyssey-VCS aciona a MS. A primeira tarefa realizada pela MS é acionar o Odyssey-MDA para gerar o modelo-alvo e os rastros através de uma transformação. Os rastros são gravados no repositório de registros de transformação. Em seguida, a MS verifica se o modelo já existe no respectivo repositório. Se não existir, a MS usa o Odyssey-VCS para fazer o *check-in* e, conseqüentemente, o versionamento do modelo. Se existir, a MS usa o Odyssey-VCS para fazer o *check-out*[3] da última versão do modelo existente no repositório. Nesse caso, com base nos rastros gerados durante a criação da versão existente no repositório e nos rastros gerados na última transformação, a MS faz a recuperação das informações de versionamento do modelo versionado e as coloca no modelo que acabou de ser criado. Em seguida a MS faz o *check-in* do modelo, que pode receber, ou não uma nova versão.

4. Implementação

O Odyssey-MEC e os seus componentes[4] usados foram implementados em Java. O EMF (*Eclipse Modeling Framework*) e o UML2 (Budinsky *et al.*, 2003) foram utilizados para a manipulação, persistência e interoperabilidade de modelos. A comunicação entre o aplicativo cliente e o servidor é realizada através de Serviços Web. A ferramenta Odyssey-MEC está disponível em http://reuse.cos.ufrj.br/odysseymec.

---

[2] Operação realizada para versionar e persistir modelos em um repositório.

[3] Operação realizada para se obter uma a cópia de uma versão de um modelo existente em um repositório.

[4] Odyssey-MEC não foi projetado para permitir o uso de outros componentes no lugar dos atuais.

5. Exemplo de Uso do Odyssey-MEC

Esta seção apresenta como exemplo o controle da evolução de modelos do projeto de um sistema de controle de reservas de um hotel. A Figura 2.a apresenta o PIM de um após o *check-in*, enquanto a Figura 2.b apresenta o PSM para a plataforma Java EE. Esse modelo foi criado e versionado automaticamente. A Figura 2.c apresenta o mesmo modelo versionado após a inclusão do método *listReservations* na classe *ReservationController* do PIM. A Figura 2.d apresenta o PSM-JEE após a sincronização e versionamento. Como pode ser observado, a ferramenta possibilita a evolução consistente de modelos inter-relacionados.

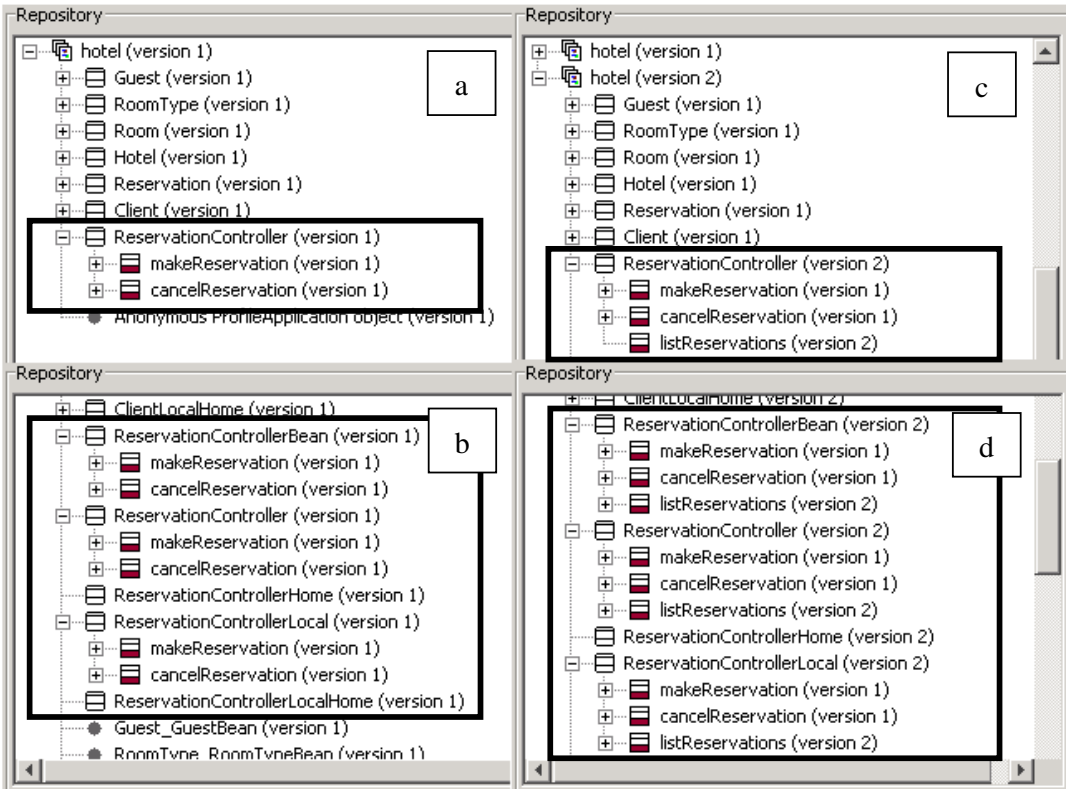

Figura 2. Modelos inter-relacionados versionados e sincronizados.

6. Trabalhos Relacionados

Xiong *et al.* (2007) criaram uma abordagem para fazer a sincronização de modelos inter-relacionados durante a execução de transformações, gerando um novo modelo-fonte e um novo modelo-alvo sincronizados. Contudo, a abordagem não realiza o controle de versão dos modelos.

Matheson *et al.* (2004) propuseram uma arquitetura para controlar a evolução de modelos no MDD. É sugerida uma solução independente de ferramentas CASE. Apesar das similaridades com o Odyssey-MEC, nada é mencionado a respeito de como a sincronização e versionamento de modelos são realizadas. Além disso, a ferramenta para a realização da abordagem ainda não foi implementada.

7. Considerações Finais

Este artigo apresentou o Odyssey-MEC como uma ferramenta para controlar a evolução de modelos inter-relacionados criados dentro do contexto do MDD usando o *framework* MDA. A ferramenta utiliza uma arquitetura cliente/servidor em que a transformação, a sincronização e o versionamento são realizados no lado servidor. Desse modo, qualquer ferramenta que tenha a capacidade de exportar e importar modelos usando o formato XMI (*XML Metadata Interchange*) é um cliente em potencial do Odyssey-MEC. Além disso, a arquitetura também permite a implantação de MDD distribuído, possibilitando que pessoas de lugares diferentes possam trabalhar no mesmo projeto.

O componente Odyssey-MDA gera apenas modelos de classes e componentes. De qualquer modo, o Odyssey-MEC, através do componente Odyssey-VCS, ainda pode ser usado para fazer o controle de versão dos demais modelos UML, porém sem a sincronização.

Como trabalhos futuros, pretendemos: (1) criar uma ferramenta para a visualização dos modelos inter-relacionados; (2) controlar a evolução de mapas de transformação e colocar nos registros de transformação (rastros) a versão do mapa que foi usada; (3) expandir o suporte para outros tipos de modelos UML; e (4) usar regras para controlar as modificações que podem ser aceitas pelo Odyssey-MEC. Também pretendemos realizar avaliações em relação à escalabilidade da ferramenta.

Referências

Budinsky, F., Steiberg, D., Merks, E.*, et al.*, 2003, *Eclipse Modeling Framework: A Developer's Guide*, Addison Wesley.

Correa, C.K.F., Murta, L., Werner, C.M.L., 2008, "Odyssey-MEC: Model Evolution Control in the Context of Model-Driven Architecture ". In: *Twentienth International Conference on Software Engineering and Knowledge Engineering*, pp. 67-72, Redwood City, CA, USA, July, 2008.

Maia, N.E.N., 2006, *Odyssey-MDA: Uma Abordagem para Transformação de Modelos*, Tese de Mestrado, COPPE, UFRJ, Rio de Janeiro - RJ.

Matheson, D., France, R., Bieman, J.*, et al.*, 2004, "Managed Evolution of a Model Driven Development Approach to Software-based Solutions". In: *Workshop on Best Practices for Model Driven Development*, Vancouver, Canada, October, 2004.

Murta, L., Corrêa, C.K.F., Prudêncio, J.G.*, et al.*, 2008, "Towards Odyssey-VCS 2: Improvements over a UML-based Version Control System". In: *Proceeedings of the International Workshop on Comparison and Versioning of Software Models (CVSM08)*, pp. 25-30, Leipzig, September, 2008.

Soley, R., 2000, *Model Driven Architecture*, OMG document omg/00-05-05, Object Management Group.

Xiong, Y., Liu, D., Hu, Z.*, et al.*, 2007, "Towards Automatic Model Synchronization from Model Transformations". In: *Proceedings of the 22th IEEE/ACM International Conference on Automated Software Engineering*, pp. 164-173, Atlanta, Georgia, USA.