IX Brazilian Symposium on GeoInformatics, Campos do Jordão, Brazil, November 25-28, 2007, INPE, p. 233-238.

Desenvolvimento de SIG para Web utilizando MDA

Carlos Eduardo R. de Mello, Geraldo Zimbrão da Silva, Jano M. de Souza

Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68.511 – Zip Code: 21945-970 – Rio de Janeiro – RJ – Brazil

{carlosmello,zimbrao,jano}@cos.ufrj.br

Abstract. *This work aims at increasing the development productivity of the Geographic Information Systems through the MDA. We defined an UML extension which adds geographic information to the class diagram. From this extension, we have been implementing a cartridge in the AndroMDA tool which allows the automatic generation of the MapServer configuration files.*

Resumo. *O objetivo deste trabalho é aumentar a produtividade no desenvolvimento de SIG para Web através do uso do padrão MDA. Definimos uma extensão da UML que agrega ao modelo de classes informações geográficas utilizadas nos SIG. A partir desta extensão, estamos implementando um cartucho na ferramenta AndroMDA que permite a geração automática dos arquivos de configuração do MapServer.*

1. Introdução

No desenvolvimento de Sistemas de Informação Geográfica (SIG), ferramentas como servidores de mapas são utilizadas para dar suporte à busca, recuperação e visualização de mapas. Uma ferramenta de código-aberto muito utilizada é o MapServer, um ambiente de desenvolvimento de SIG para Web desenvolvido pela Universidade de Minnesota [Carvalho, 2004]. Para utilizar o MapServer é necessário que o desenvolvedor conheça todas as suas funcionalidades, como arquivos de configuração, etiquetas utilizadas e a arquitetura do ambiente. O desenvolvimento de SIG utilizando esse ambiente pode se tornar um processo longo e repetitivo. Dependendo do tamanho e da complexidade do SIG, os arquivos de configuração e trechos de código-fonte podem ser replicados inúmeras vezes pelo desenvolvedor, gerando retrabalho.

Portanto, propomos a utilização do padrão de Arquitetura Orientada a Modelo (*Model Driven Architecture* – MDA) [Mellor, 2004] no desenvolvimento de SIG para Web, com o objetivo de melhorar a produtividade dos desenvolvedores. Esse padrão apóia todo o ciclo de vida de desenvolvimento de aplicações, através da utilização de modelos. No MDA, a especificação das funcionalidades do sistema é isolada da especificação da implementação das funcionalidades para uma plataforma ou tecnologia específica. Com isso, além da possibilidade de construir modelos de uma maneira formal, o MDA também permite que transformações automáticas sejam realizadas entre modelos, com o objetivo de se obter um produto final de software [Mellor, 2004]. Essas transformações automáticas diminuem o tempo, o esforço e o retrabalho no desenvolvimento, melhorando a produtividade. Existem várias ferramentas que implementam o padrão MDA, uma delas é o AndroMDA [Bohlen, 2003]. Esta é uma ferramenta de código-aberto para geração de código através de transformações

automáticas. Essas transformações são realizadas por cartuchos, capazes de gerar novos modelos ou código.

O objetivo deste trabalho é tentar aumentar a produtividade do desenvolvimento de SIG para Web, através da geração automática de código a partir de modelos de Diagrama de Classes. Como meio para alcançar este objetivo, propomos uma extensão da Linguagem de Modelagem Unificada (*Unified Modeling Langague* – UML) [Booch, 1999], de forma que esta dê suporte ao desenvolvimento de aplicações de SIG para Web. Além disso, estamos desenvolvendo um cartucho para ferramenta AndroMDA, para que a geração automática do código da estrutura básica dos arquivos de configuração do MapServer possa ser realizada.

Na seção 2 deste trabalho, apresentamos uma visão geral de MDA, seus modelos e seu funcionamento. Na seção 3, apresentamos nossa proposta de extensão da UML para apoiar o desenvolvimento de SIG para Web. Em seguida (seção 4), apresentamos as transformações implementadas no cartucho do AndroMDA para gerar a estrutura básica dos arquivos de configuração do MapServer. Finalmente, na seção 5, descrevemos as considerações finais e o andamento deste trabalho.

2. Visão geral do MDA

Nesta seção, apresentamos uma visão geral do MDA e da ferramenta AndroMDA. Mais detalhes sobre o MDA podem ser encontrados em [Mellor, 2004][Kleppe, 2003][Frankel, 2003].

A Arquitetura Orientada a Modelo (*Model Driven Architecture* – MDA) é um padrão do Grupo de Gerenciamento de Objetos (*Object Management Group* – OMG) [Mellor, 2004]. Esse padrão compreende todo o ciclo de vida de desenvolvimento de aplicações, através de modelos de desenvolvimento de software. No MDA, as especificações das funcionalidades do sistema estão isoladas das especificações de implementação para uma plataforma ou tecnologia específica. Portanto, o MDA encoraja a especificação de um Modelo Independente de Plataforma (*Platform Independent Model* – PIM), isto é, que não contém informações de nenhuma plataforma ou tecnologia específicas no modelo. O PIM é transformado em um Modelo Específico de Plataforma (*Platform Specific Model* – PSM), onde informações específicas da tecnologia de implementação são acrescentadas ao modelo. O PSM gerado a partir do PIM pode ser transformado em código da aplicação para ser executado na plataforma tecnológica escolhida.

O modelo de nível mais alto desses modelos descritos anteriormente é o Modelo Independente de Computação (*Computation Independent Model* – CIM). Esse modelo descreve o sistema dentro do seu ambiente (domínio de negócio), apresentando o que é esperado que o sistema faça, sem detalhes de como isto será feito. Portanto, os requisitos do sistema são modelados através de um CIM.

O CIM pode ser modelado utilizando a UML ou outras linguagens, de acordo com os requisitos de análise. O PIM e o PSM podem ser modelados utilizando qualquer linguagem de especificação. Tipicamente, a UML é utilizada, pois é um padrão para especificação de sistemas para domínios genéricos e ainda pode ser estendida para apoiar linguagens para domínios específicos.

Na figura 1, podemos observar como os modelos estão relacionados. Os requisitos do sistema são modelados no CIM. A partir modelo do CIM, são realizados refinamentos dos requisitos no PIM. A transformação do CIM para o PIM é feita manualmente, portanto, até esse ponto não é realizada nenhuma transformação automática. A transformação do PIM para o PSM de uma plataforma específica é feita de forma automática por ferramentas de MDA. O código para a plataforma específica é então gerado a partir das transformações realizadas no PSM.

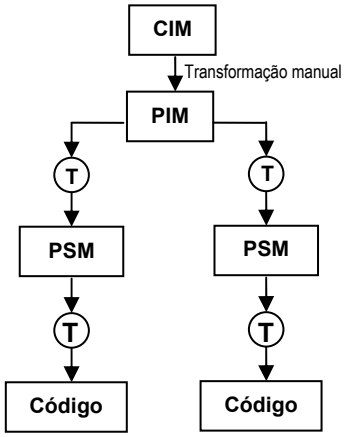

Figura 1 – Arquitetura Orientada a Modelo [Mazón, 2005]

Na ferramenta AndroMDA, as transformações de modelo para modelo e de modelo para código são realizadas pelos cartuchos. O AndroMDA possui cartuchos padrões que permitem o desenvolvimento de sistemas genéricos. Entretanto, esses cartuchos podem ser estendidos, para que sejam suportadas novas transformações para plataformas específicas ou para modelos específicos [Bohlen, 2003]. Também é possível a criação de novos cartuchos, onde novas transformações são definidas com base em novos modelos ou tecnologias.

3. Extensão da UML

Nesta seção, apresentamos uma proposta de extensão da UML, de modo que esta ofereça suporte adequado para a geração dos arquivos de configuração do MapServer.

Os SIG geralmente utilizam o conceito de camadas para a apresentação de mapas [Camara, 1996]. Cada camada é formada por objetos espaciais e essas são sobrepostas formando as imagens dos mapas. O ambiente MapServer também utiliza o conceito de camadas, estas são descritas por um arquivo de configuração chamado *mapfile* [MapServer, 2004]. Além da descrição das camadas, o *mapfile* também possui todas as informações necessárias para que o MapServer reproduza a imagem do mapa referente a esse *mapfile*. Cada imagem reproduzida de um mapa possui um *mapfile* correspondente e cada *mapfile* descreve apenas um mapa. Portanto, a geração correta dos *mapfiles* é fundamental para o desenvolvimento de um SIG.

Entretanto, dependendo do tipo de mapa a ser reproduzido, o seu respectivo *mapfile* pode ficar com um número grande de linhas. Esse excesso de linhas no *mapfile* pode levar o desenvolvedor a esquecer de colocar camadas ou os nomes corretos dessas camadas no *mapfile*. Outro problema é a associação dos objetos espaciais contidos nos

mapas (rios, cidades, estados *etc.*) com as classes de negócio do próprio SIG (rios, cidades, estado *etc.*).

Portanto, para tentar amenizar esses problemas, propomos uma extensão do Diagrama de Classes da UML. Incluímos no Diagrama de Classes dois estereótipos e dois valores etiquetados.

O primeiro estereótipo cujo nome foi definido por *Layer* só pode ser aplicado a uma classe que represente uma camada em um mapa no sistema. Este estereótipo possui um valor etiquetado chamado *geometry* cujos valores possíveis são: *LINE*, *POLYGON*, *POINT*. Esses possíveis valores definem o tipo geométrico do objeto espacial representado pela classe, isto é, linha, polígono ou ponto [Güting, 1994].

O segundo estereótipo que definimos é o *ItemClass*. Este só pode ser associado a um atributo de uma classe que possua o estereótipo *Layer*. O atributo com o estereótipo *ItemClass* define como dividir os objetos espaciais de uma camada em vários grupos. Por exemplo, uma camada onde temos várias rodovias e cada rodovia possui o atributo *principal*, definindo o tipo da rodovia. Se quisermos reproduzir um mapa com as rodovias principais coloridas de azul e as rodovias secundárias de vermelho, precisamos indicar para o MapServer qual o atributo que contém esta informação. O estereótipo *ItemClass* possui o valor etiquetado *NumClasses* que contém o número de grupos formados dentro da camada. No exemplo anterior, podemos ter dois grupos de rodovias, principais e secundárias, portanto o valor etiquetado de *NumClasses* é 2.

Na figura 2, podemos observar um exemplo de uma aplicação modelada na extensão do diagrama de classes proposto. As classes **UnidadeFederacao**, **Municipio**, **Sede** e **Rodovia** são classes de negócio do sistema e estão associadas a objetos espaciais contidos em um mapa. Note que a classe **Concessionaria** é apenas uma classe de negócio, portanto, não possui nenhum estereótipo ou valor etiquetado.

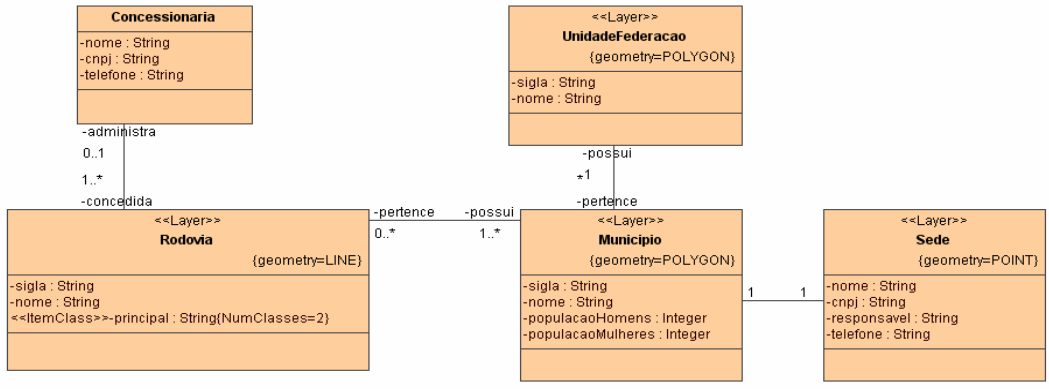

Figura 2 – Aplicação modelada na extensão do modelo de classes

4. Transformações

Nesta seção apresentamos as transformações realizadas pelo cartucho do AndroMDA que estamos implementando para a geração dos arquivos de configuração do MapServer. A seguir descrevemos a seqüência das operações realizadas pelo AndroMDA e pelo cartucho no processo de geração do *mapfile*.

O AndroMDA carrega o modelo do diagrama de classes a partir de um XML padrão exportado por uma ferramenta de modelagem. Em seguida, o AndroMDA gera

um modelo de Metafacades que realiza transformações nos dados de maneira que o cartucho possa trabalhar e tratar esses dados do modelo.

A primeira transformação que o cartucho realiza é geração do esqueleto básico do *mapfile*. Este esqueleto possui o modelo básico e o nome do *mapfile*. O nome do *mapfile* gerado é o mesmo nome do pacote no qual às classes com estereótipo *Layer* estão. Portanto, para a geração de todos os *mapfiles*, o cartucho procura os pacotes que contenham classes com estereótipo *Layer* e gera o esqueleto básico para todos os *mapfiles* definidos de acordo modelo.

A segunda transformação implementada é a geração das camadas. Para todas as classes com o estereótipo *Layer* são geradas camadas com o mesmo nome. Portanto, no exemplo da Figura 2, as classes **Sede**, **Municipio**, **UnidadeFederacao** e **Rodovia** geram as camadas **Sede**, **Municipio**, **UnidadeFederacao** e **Rodovia**, respectivamente, no *mapfile* correspondente ao pacote onde estão essas classes.

A última transformação é a geração dos grupos de objetos espaciais dentro de uma mesma camada. Estes grupos de objetos são chamados de classes dentro do *mapfile*. Toda camada no *mapfile* possui pelo menos uma classe, isto é, pelo menos um grupo de objetos espaciais. Portanto, o cartucho procura nos dados carregados do modelo as classes que possuem o estereótipo *Layer* e que não possuem nenhum atributo com esteriótipo *ItemClass*. Para essas classes é gerada no *mapfile* correspondente apenas uma classe (grupo) dentro da camada com o mesmo nome da classe do modelo. Entretanto, para as classes com estereótipo *Layer* que possuem algum atributo com o estereótipo *ItemClass*, são geradas classes (grupos) dentro da camada com o mesmo nome da classe do modelo. O nome destas classes (grupos) é o mesmo nome da camada concatenada por um número inteiro. O número de classes (grupos) geradas no *mapfile* será igual ao valor do número contido no valor etiquetado *NumClasses* que o atributo possui.

5. Considerações finais

Neste artigo, apresentamos a implementação de um cartucho para o AndroMDA que realiza a geração automática dos arquivos de configuração do MapServer (*mapfiles*).

As transformações realizadas pelo cartucho não geram todo o conteúdo do *mapfile*, mas sim a sua estrutura básica, através das informações que estão disponíveis no modelo. Para a geração completa do *mapfile* é necessário que a extensão da UML utilizada ofereça mais informações sobre os mapas a serem gerados. Entretanto, o objetivo do modelo é ser independente de plataforma, por isso a agregação de mais informações ao modelo deve ser feita com cautela.

Estamos desenvolvendo o cartucho de maneira que novas extensões da UML possam ser implementadas. Atrelado a isso, também estamos trabalhando em propostas de estereótipos e valores etiquetados para diagramas de atividades. Com isso, esperamos que mais informações úteis para o desenvolvimento de SIG possam ser incorporadas ao modelo, sem que ocorra dependência de plataforma.

6. Agradecimentos

Agradecemos ao CNPQ pelo apoio financeiro.

Referências

Bohlen, M. (2003) "AndroMDA", www.andromda.org.

Booch, G.; Rumbaugh, J. and Jacobson, I. (1999) "The Unified Modeling Language User Guide", Massachusetts.

Camara, G. et al. (1996) "Anatomia de Sistemas de Informação Geográfica", Instituto de Computação, UNICAMP, Campinas.

Carvalho, C. A. (2004) "Desenvolvimento de Aplicações WebGIS em MapServer.", EMBRAPA, Campinas.

Frankel, D. S. (2003) "Model Driven Architecture. Applying MDA to Enterprise Computing. Indianapolis", Wiley, Indiana.

Güting, R.H. (1994) "An Introduction to Spatial Database Systems", Special Issue on Spatial Database Systems of the VLDB Journal, Vol.3, No.4, October.

Kleppe, A., Warmer, J. and Bast, W. (2003) "MDA Explained. The Practice and Promise of The Model Driven Architecture.", Addison Wesley.

MapServer, (2004) "MapServer Documentation Project.", http://mapserver.cttmar.univali.br.

Mazón, J. R., Trujillo, J., Serrano, M. and Piattini, M. (2005) "Applying MDA to the development of data warehouses", Proceedings of the 8th ACM international workshop on Data warehousing and OLAP, Bremen, Germany, November.

Mellor, S., Scott, K., Uhl, A. and Weise, D. (2004) "MDA distilled: principles of Model-Driven Architecture", Addison-Wesley.