Lições Aprendidas sobre Desenvolvimento Dirigido por Modelos a partir da refatoração de uma ferramenta

Valdemar Vicente Graciano Neto[1,*], **Sofia Larissa da Costa**[1], **Juliano Lopes de Oliveira**[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – Campus II – CEP 74001-970 – Goiânia – GO – Brasil
Universidade Federal de Goiás - UFG
{valdemar, sofia, juliano}@inf.ufg.br

Resumo. *Modelos são artefatos utilizados para registrar conhecimento durante a concepção de um software. O processo de desenvolvimento de software consiste em sucessivas transformações que convertem modelos em código-fonte. Ao longo dos últimos anos esforços tem sido conduzidos para automatizar partes da transformação entre modelos. A esta iniciativa deu-se o nome de Desenvolvimento de Software Dirigido por Modelos (DSDM). Uma ferramenta de apoio ao desenvolvimento de Sistemas de Informação Empresariais foi criada baseando-se nos conceitos de DSDM. Porém, limitações foram observadas e uma refatoração foi proposta. Este trabalho detalha o funcionamento da ferramenta, suas limitações e as lições aprendidas ao longo de seu desenvolvimento e refatoração.*

Abstract. *Models are artifacts used to register the domain along the software development. The software development process consists of successive transformations that convert models into source code. Over recent years efforts have been conducted to automate parts of the transformation between models. The initiative was called Model-Driven Development (MDD). A tool to support development of Enterprise Information Systems was created based on the MDD concepts. However, limitations were observed and a refactoring was proposed. This paper details the tool features, its limitations and lessons learned during its development and refactoring.*

1. Introdução

O desenvolvimento de software pode ser visto como um processo de aprendizado e aquisição de conhecimento [von Staa 2010]. Modelos são artefatos utilizados na Engenharia de Software para capturar o conhecimento adquirido durante o processo de desenvolvimento de software. Eles ajudam analistas a entender problemas complexos e suas soluções potenciais através de abstração [Selic 2003].

Vários modelos são usados para tentar expressar os conceitos do domínio de conhecimento para o qual o software é construído. Existem modelos específicos para cada fase do processo de desenvolvimento de software. O modelo de requisitos é usado como entrada para discussão e produção de modelos de projeto e arquitetura. Estes modelos são

*Com auxílio financeiro da CAPES

levados em consideração para a estruturação do código-fonte e o código-fonte é o cenário primário para especificação de casos de teste. O processo de desenvolvimento de software resume-se a uma série de **transformações entre modelos**.

Existem iniciativas para gerar software a partir de transformação automatizada de modelos há algumas décadas [de Oliveira et al. 1995, Puerta 1997] mas recentemente os estudos nesta área tem se intensificado rumo à construção de ferramentas capazes de transformar modelos efetivamente [Selic 2003, Mellor et al. 2003, Miller and Mukerji 2003]. Tal paradigma de construção de software foi denominado Desenvolvimento de Software Dirigido por Modelos (DSDM) (*do inglês, Model-Driven Development - MDD*), e diferencia-se do modo convencional de produzir software por automatizar a transformação entre modelos.

Para garantir alguma padronização na forma como ferramentas deveriam apoiar a automatização de transformações, o OMG (*Object Management Group*) publicou em 2001 e atualizou em 2003 a especificação **MDA** (*Model-Driven Architecture*) [Miller and Mukerji 2003]. Tal norma provê uma descrição genérica sobre os elementos que deveriam existir no Processo de DSDM.

Seguindo alguns aspectos da especificação MDA, foi criada e descrita por [Almeida et al. 2009, Da Silva and de Oliveira 2009, Boff and de Oliveira 2010] uma ferramenta para auxiliar no processo de desenvolvimento de um tipo específico de software: Sistemas de Informação Empresariais (SIE). SIEs são baseados em três aspectos: interfaces de usuário CRUD (*Create, Read, Update, Delete*), informações do domínio do negócio e processos de negócio.

O intuito desta ferramenta, em alinhamento com os objetivos da MDA, é elevar o nível de abstração na produção de software trazendo ganhos como geração automática de código, produtividade, manutenibilidade, portabilidade, reusabilidade e rastreabilidade ([Kleppe et al. 2003, Miller and Mukerji 2003].

Porém, após a construção da ferramenta, alguns problemas foram diagnosticados: grande acoplamento e espalhamento do código referente às regras de transformação de modelos, características de negócio e interação com o usuário altamente acopladas, baixa aceitação dos SIEs gerados devido à pouca manutenibilidade e capacidade de customização das telas geradas e SIEs gerados não orientados a processos (sobrecarregando a memória do usuário mesmo para efetuar tarefas simples na ferramenta) [Loja et al. 2010].

Diante de tais limitações, verificou-se a necessidade de uma refatoração na ferramenta para torná-la mais aderente à MDA com a expectativa de reduzir alguns dos problemas relatados.

Este trabalho tem por intuito relatar lições aprendidas durante a refatoração da ferramenta descrita por [Almeida et al. 2009, Da Silva and de Oliveira 2009, Boff and de Oliveira 2010] que podem beneficiar futuros pesquisadores interessados em avançar o estado da arte em DSDM. Para tanto, a Seção 2 trata mais detalhamente os conceitos envolvidos em DSDM. A Seção 3 apresenta de forma mais clara e minuciosa a ferramenta descrita por [Almeida et al. 2009, Da Silva and de Oliveira 2009, Boff and de Oliveira 2010]. A Seção 4 discute as lições aprendidas pelos autores durante a execução da refatoração da ferramenta. A Seção 5, por fim, apresenta as conclusões do trabalho realizado e os trabalhos futuros.

2. Desenvolvimento de Software Dirigido por Modelos

Por razões históricas modelos são infrequentes dentro da Engenharia de Software e, quando são usados, frequentemente são colocados em segundo plano. DSDM possui seu foco principal nos *modelos* [Selic 2003]. O software é visto como o resultado de uma sequência de transformações entre modelos. DSDM automatiza a transformação entre modelos [Mellor et al. 2003].

Para tornar a automação uma realidade, os modelos precisam ter um significado definido [Mellor et al. 2003]. Uma forma de prover tal significado é criar metamodelos. Metamodelos fazem asserções sobre o que pode ser expresso em modelos [Seidewitz 2003]. Além disso, eles proveem sintaxe e semântica definidas para os modelos produzidos a partir deles.

MDA é uma especificação para desenvolvimento de software definido pelo OMG [Kleppe et al. 2003] que propõe a utilização de metamodelos e modelos. Como mostra a Figura 1, a MDA propõe que um *Modelo Independente de Plataforma (Plataform Independent Model - PIM)* - como um modelo de domínio - seja usado como entrada em uma ferramenta de transformação para produzir automaticamente um *Modelo Específico de Plataforma (Plataform Specific Model - PSM) - como um modelo arquitetural ou de projeto*.

Para transformar modelos utiliza-se uma **definição de transformação**, um conjunto de regras que descrevem como um modelo em uma linguagem origem pode ser transformado em um modelo em uma linguagem destino [Kleppe et al. 2003]. Segundo a MDA, as regras ou definições de transformação são elementos plugáveis à ferramenta de transformação.

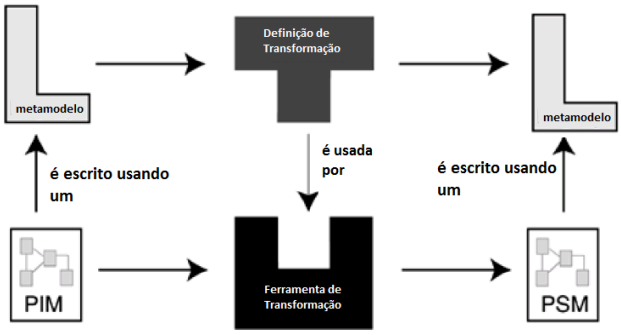

Figure 1. Transformações entre modelos segundo MDA.

Por ser um framework conceitual, MDA não apresenta propostas arquiteturais para as ferramentas de transformação dirigidas por modelos. Logo, cada iniciativa implementa a plugabilidade de regras de transformação de uma forma diferente. A falta de consenso sobre como aderir à MDA é considerado um fator que prejudicou o projeto da ferramenta tema desta pesquisa. A próxima seção traz uma descrição mais detalhada desta ferramenta.

3. Descrição da ferramenta

A ferramenta descrita por [Almeida et al. 2009] fornece apoio ao processo de desenvolvimento de SIEs. São fornecidos *templates* de interfaces CRUD que são utilizados tanto

para instanciar o metamodelo da ferramenta (gerando o novo SI) quanto para popular o SI gerado (gerando instâncias do modelo) [Da Silva and de Oliveira 2009].

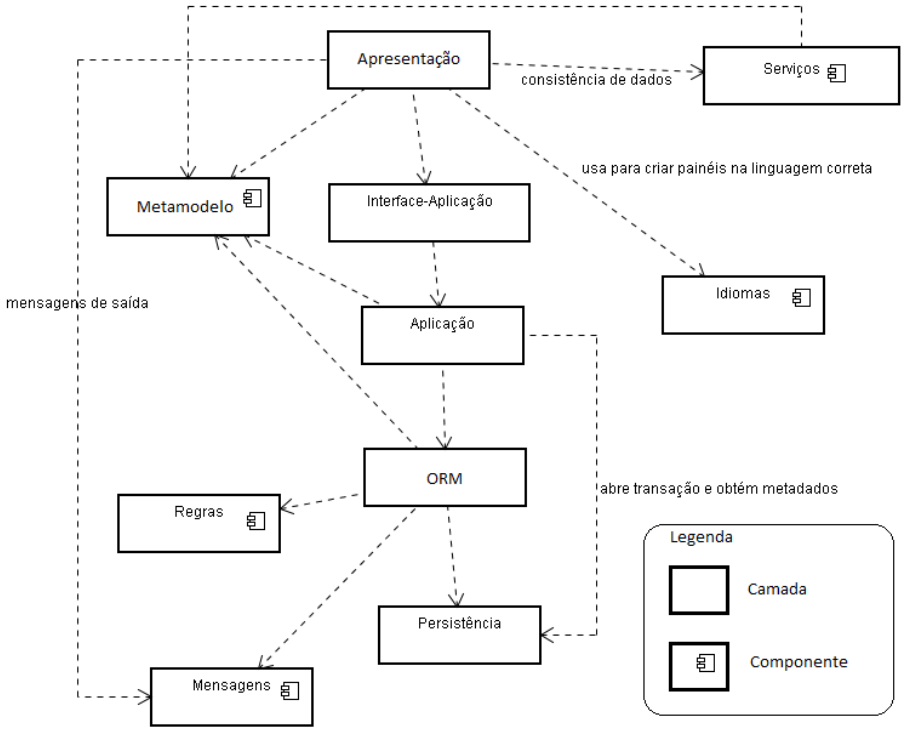

Figure 2. Diagrama de Componentes referente à ferramenta.

A Figura 2 ilustra a arquitetura da ferramenta baseada em camadas e componentes. A camada de Apresentação é responsável por geração de telas e interação com o usuário. O componente de Serviços é utilizado para fins de consistência e validação dos dados inseridos pelo usuário nas telas. O componente de Mensagens fornece mensagens de diálogo com os usuários que são emitidas pelo componente de Serviços caso os dados não sejam consistentes ou válidos. O componente de Idiomas é responsável por aspectos de internacionalização do software. O componente Metamodelo guarda a representação do metamodelo para modelagem do domínio de SIE. A camada Interface-Aplicação encapsula os dados inseridos na camada de Apresentação para transitar objetos ao longo da arquitetura da aplicação gerada.

A camada de Aplicação fornece funcionalidades CRUD para manipulação de dados do domínio. A camada ORM ou *Object-Relational Mapping* é responsável por efetuar o mapeamento objeto-relacional. Os dados inseridos pelo usuário e encapsulados em um objeto na camada Interface-Aplicação são convertidos em registros a serem persistidos pela camada de Persistência.

O componente de Regras é responsável por gerenciar as Regras de Negócio do domínio do SI criado. A ferramenta permite gerenciar regras de dois tipos: derivação e de validação de dados. Elas são escritas em OCL e automaticamente transformadas em *stored procedures* utilizando técnicas baseadas em modelos [Boff and de Oliveira 2010].

As camadas Apresentação, Interface-Aplicação, Aplicação e ORM são dependentes do componente de Metamodelo. As manipulações de objetos, tais como mapea-

mento objeto-relacional, validação de dados, trânsito de informações e geração de regras de validação e derivação são baseadas no formato e ordem dos dados e do metamodelo.

Não há uma etapa de Projeto de Interface explícita. O código da camada de Interface é gerado de modo automático com base na leitura do modelo e do metamodelo. Para cada tipo de atributo específico, é renderizado um tipo de widget pré-definido.

As fases de Projeto, Codificação, Teste e Implantação são apoiadas em partes pela ferramenta pois: 1) a arquitetura e os padrões de projeto aplicados ao SI gerado são inerentes à ferramenta (o SI gerado adere ao padrão arquitetural da ferramenta - camadas - e utiliza controles da ferramenta para efetuar tarefas básicas de um SI como trânsito de dados, mapeamento objeto-relacional, gerência de regras de negócio, internacionalização e geração de interfaces; 2) a codificação ainda acontece pois, para domínios específicos, é necessário programar funcionalidades específicas; 3) Pode-se realizar testes de modo caixa-preta, verificando a conformidade entre as entradas de usuário e as saídas esperadas; 4) O SI gerado é implantado em conjunto com a ferramenta pois ele utiliza-se de várias funcionalidades providas por ela.

Após uma breve descrição das funcionalidades e composição da ferramenta, é possível relatar alguns problemas identificados através de experimentação e aplicação na construção de um SI voltado para manejo de gado de corte e leite no Estado de Goiás:

1. Não é gerado código-fonte de interação com o usuário. As telas são renderizadas em tempo de execução, mapeando atributos de entidades do negócio em widgets específicos. Como o mapeamento é pré-definido, as telas geradas são consistentes mas não é possível customizá-las. O leiaute não é configurável, e aspectos como tamanho de fonte, cores, posição dos elementos, dentre outros atributos visuais são fixos.
2. As transformações de dados e leitura de modelos para geração de interface, validação e derivação de dados, e mapeamento objeto-relacional fazem com que vários componentes da arquitetura sejam muito dependentes da definição do metamodelo. Caso haja uma evolução do metamodelo - uma necessidade genuína [Wachsmuth 2007] - a quantidade de pontos em que é necessário realizar modificações é muito grande.
3. Metadados de interface e de negócio atualmente estão acoplados. O cadastro de ambos os tipos de metadados é feito sem diferenciação.
4. Somente interfaces CRUD são geradas, sendo que estas não são o único tipo existente e requisitado em SIE.

Logo, a geração de interfaces de usuário, regras de negócio armazenadas em *stored procedures* e mapeamento objeto-relacional é baseada em modelos mas os códigos de mapeamento são independentes, replicados, espalhados e não modularizados, o que motivou a refatoração da ferramenta. Algumas alterações já foram realizadas e novos metamodelos foram construídos [Loja et al. 2010, da Costa et al. 2010]. Alterações na arquitetura e projeto da ferramenta foram propostas e a seção seguinte ilustra as principais lições aprendidas a partir da execução de análise e projeto de uma nova edição desta ferramenta com base nas deficiências encontradas na versão anterior.

4. Lições Aprendidas

A partir da refatoração que vem sendo conduzida e da experiência acumulada é possível listar algumas lições e recomendações sobre ferramentas para DSDM. Tais recomendações podem ser utilizadas como referência para pesquisadores que vão iniciar seus projetos de desenvolvimento de software dirigido por modelos com apoio de ferramentas especializadas. São elas:

1. **Os modelos devem ser coesos e pouco acoplados**. Ao tentar gerar software a partir de modelos é importante que se utilize um conjunto de modelos e não apenas um modelo único que tente capturar todos os aspectos associados ao domínio. Até agora identificou-se que para o domínio de SIE deveriam ser usados, pelo menos, os modelos de Processo de Negócio [Loja et al. 2010], Interação Homem-Computador e Domínio [da Costa et al. 2010]. Além disso é importante que dois ou mais modelos estabeleçam relações para representar um aspecto de interesse do mundo real, algo conhecido como *correlação (correlability)* [Limbourg and Vanderdonckt 2004].
2. **Ferramentas devem ser específicas de domínio**. É importante que as ferramentas construídas para produção de software dirigida por modelos não tenham ambições genéricas, ao menos inicialmente. O grande sonho da comunidade científica de DSDM é converter as ferramentas de transformação em verdadeiros compiladores de modelos. Mas, a partir do que foi desenvolvido, percebe-se que por enquanto é mais viável e prudente investir esforços em soluções específicas de domínio para resolver pequenos problemas dentro do processo de desenvolvimento de software para a seguir conseguir gerar software completo para aquele domínio e, por fim, gerar uma ferramenta com intuito realmente genérico.
3. **Deve haver um conjunto mínimo de modelos**. À luz da experiência com a ferramenta há indícios de que há um conjunto mínimo de modelos capaz de gerar SIEs. Inicialmente os modelos de Processo de Negócio e de Interação Humano-Computador e Domínio são os primeiros candidatos, mas pode haver outros modelos necessários para a geração efetiva de SIEs e outros domínios devem também ter seus conjuntos mínimos.
4. **Importância da aderência à MDA**. Na ferramenta descrita há uma diversidade de tipos de regras de transformação. Há regras que geram interfaces CRUD baseadas nos dados de negócio, que efetuam o mapeamento objeto-relacional, que geram *stored procedures* de manipulação dos dados e que geram regras de negócio (derivação e validação) implantadas no SI gerado. Todas estas regras dependem em maior ou menor grau do metamodelo vigente. Se o metamodelo precisa mudar, todas estas regras também tem de ser modificadas. Com isso percebe-se que a adoção de MDA pode auxiliar na plugabilidade e modularidade das regras de transformação.
5. **Há categorias de regras de transformação**. No momento em que a ferramenta estiver aderente à MDA e um conjunto de regras for utilizado para gerar o SI, haverá subconjuntos de regras, cada um associado a um aspecto do SI (IHC, Processos e Domínio, neste caso) e outros subconjuntos de regras responsáveis por integrar os aspectos do SI.
6. **MDD formaliza o processo de desenvolvimento de software**. Considerando que modelos são marginalizados dentro do processo de desenvolvimento de software

convencional, quando a produção baseada em modelos é adotada isso deve forçar os engenheiros de software a documentar o conhecimento adquirido, o que minimiza o problema relatado por [Selic 2003]. O foco sai da codificação e vai para a construção de modelos.

5. Conclusões

A experiência proporcionou este conjunto de recomendações que podem ser seguidas para desenvolver ferramentas de geração de SIEs baseados em modelos mais manuteníveis e customizáveis. Tais recomendações estão sendo levadas em consideração para a refatoração que encontra-se em andamento.

Conclui-se que a geração de software baseada em modelos pode auxiliar na automatização do processo de desenvolvimento de software. No entanto, é importante que as ferramentas criadas para apoiar este trabalho sejam aderentes à MDA, provendo maior flexibilidade e manutenibilidade.

Nossas suspeitas são que para cada domínio de conhecimento deve haver um conjunto mínimo de modelos capazes de fornecer informação suficiente para transformadores de modelos gerarem um software completo. Os modelos devem ser monolíticos porém coesos e regras de transformação devem ser especificadas para cada modelo deste conjunto mínimo e para integrar tais modelos.

A arquitetura anterior não foi aderente à MDA por dificuldades em encontrar soluções arquiteturais que viabilizassem a criação de uma ferramenta com regras de transformação entre modelos plugáveis. Como trabalhos futuros vislumbra-se o estudo e levantamento de soluções arquiteturais que permitam tornar as regras de transformação elementos plugáveis, além de proporcionar uma arquitetura que torne a manutenção dos metamodelos simples e menos onerosa.

References

Almeida, A. C., Boff, G., and Oliveira, J. L. (2009). A framework for modeling, building and maintaining enterprise information systems software. In *Anais do XXIII Simpósio Brasileiro de Engenharia de Software*, pages 115–125. Fortaleza, Brasil.

Boff, G. and de Oliveira, J. L. (2010). Modeling, implementation and management of business rules in information systems. *INFOCOMP Journal of Computer Science*, pages 17 – 28.

da Costa, S. L., Graciano Neto, V. V., Loja, L. F. B., and de Oliveira, J. L. (2010). A metamodel for automatic generation of enterprise information systems. In *Anais do I Congresso Brasileiro de Software: Teoria e Prática - I Workshop Brasileiro de Desenvolvimento de Software Dirigido por Modelos*, volume 8, pages 45–52. UFBA. Salvador, BA, Brasil.

Da Silva, W. C. and de Oliveira, J. L. (2009). Gerência de interface homem-computador para sistemas de informação empresariais: uma abordagem baseada em modelos. *iSys - Revista Brasileira de Sistemas de Informação, Vol. 2, 2009*.

de Oliveira, J. L., Cunha, C. Q., and Magalhães, G. C. (1995). Modelo de objetos para construção de interfaces visuais dinâmicas. In *Anais do 9o. Simpósio Brasileiro de Engenharia de Software*, pages 143 – 158. Recife, PE, Brasil.

Kleppe, A., Warmer, J., and Bast, W. (2003). *MDA Explained: The Model Driven Architecture - Practice and Promise*.

Limbourg, Q. and Vanderdonckt, J. (2004). Addressing the mapping problem in user interface design with usixml. In *In Proc. of the 3rd Int. Workshop on Task Models and Diagrams for User Interface Design TAMODIA2004*, pages 15–16. ACM Press.

Loja, L. F. B., Graciano Neto, V. V., da Costa, S. L., and de Oliveira, J. L. (2010). A business process metamodel for enterprise information systems automatic generation. In *Anais do I Congresso Brasileiro de Software: Teoria e Prática - I Workshop Brasileiro de Desenvolvimento de Software Dirigido por Modelos*, volume 8, pages 37–44. UFBA. Salvador, BA, Brazil.

Mellor, S. J., Clark, A. N., and Futagami, T. (2003). Guest editors' introduction: Model-driven development. *IEEE Software*, 20(5):14–18.

Miller, J. and Mukerji, J. (2003). Mda guide version 1.0.1. Technical report, Object Management Group (OMG).

Puerta, A. R. (1997). A model-based interface development environment. *IEEE Software 14,4 (July/August*, pages 41–47.

Seidewitz, E. (2003). What models mean. *IEEE Software*, 20(5):26–32.

Selic, B. (2003). The pragmatics of model-driven development. *IEEE Software*, 20(5):19–25.

von Staa, A. (2010). Em busca de um meta-ambiente de engenharia de software assistido por computador. In *Anais do I Congresso Brasileiro de Software: Teoria e Prática - I Workshop Brasileiro de Desenvolvimento de Software Dirigido por Modelos*, volume 8, pages 5–12. UFBA. Salvador, BA, Brazil.

Wachsmuth, G. (2007). Metamodel adaptation and model co-adaptation. In Ernst, E., editor, *Proceedings of the 21st European Conference on Object-Oriented Programming (ECOOP'07)*, volume 4609 of *Lecture Notes in Computer Science*, pages 600–624. Springer-Verlag.