Processo de Manutenção Evolutiva de uma Ferramenta de Geração de Código

Túlio Ribeiro Torres[1], Matheus Carvalho Viana[1]

[1]Universidade Federal de São João del-Rei (UFSJ)
PÇA. Fr. Orlando, 170, Centro – São João del-Rei – MG – Brasil
tuliotorres7@hotmail.com, matheuscviana@ufsj.edu.br

Abstract. *JMOGEN is a tool that facilitates the construction of domain-specific frameworks and their use in the development of information systems. However, its initial version generated the code of frameworks and systems with no documentation. The objective of this work was to perform an evolutionary maintenance process in the tool to include the functionality of class diagrams of the frameworks and systems developed through it. From these models, a visualization of the structure of the generated code is obtained, facilitating its understanding and the maintenance.*

Resumo. *A ferramenta JMOGEN facilita a construção de frameworks específicos de domínio e a utilização destes no desenvolvimento de sistemas de informação. Contudo, sua versão inicial gerava o código dos frameworks e sistemas sem documentação alguma. Este trabalho teve como objetivo realizar processo de manutenção evolutiva na ferramenta para incluir a funcionalidade de geração de diagramas de classes dos frameworks e sistemas desenvolvidos por meio dela. A partir desses modelos, obtém-se uma visualização da estrutura do código gerado, facilitando o entendimento e a manutenção do mesmo.*

Introdução

Quando há a necessidade de desenvolver diversos sistemas dentro do mesmo domínio, é inviável construir cada um a partir do zero. Nesse caso, o uso de *frameworks* específicos de domínio pode ser uma solução para reduzir o retrabalho. Contudo, construir um software de natureza abstrata e flexível exige conhecimento sobre o domínio retratado e técnicas avançadas de desenvolvimento. Além disso, desenvolver um sistema utilizando um *framework* de maneira adequada também exige experiência.

A ferramenta Jmogen [Viana et al. 2013] é um *plug-in* do Eclipse IDE desenvolvido para facilitar o desenvolvimento de *frameworks* e a instanciação destes em sistemas de informação. Por meio dessa ferramenta, o desenvolvedor cria um modelo do domínio e obtém o código do *framework* do domínio e uma linguagem de modelagem (*Domain-Specific Language* - DSL). As DSLs são linguagens cujas estruturas representam entidades do domínio que retratam [J. M. Jezequel, 2012], além de permitir modelar e gerar sistemas de informação pertencentes a este domínio.

A versão inicial da Jmogen era capaz de gerar o código dos *frameworks* e dos sistemas, porém não gerava uma documentação que permita uma visualização dos elementos (pacotes, classe, atributos, etc.) que compunham esse código e facilitasse o seu entendimento e a sua manutenção. Este tipo de documentação é essencial pois auxilia na representação do sistema, como um todo, e de possíveis alterações que devem

ser realizadas para auxiliar na manutenção preventiva ou corretiva do mesmo. Portanto, este projeto teve como objetivo a realização de um processo de manutenção evolutiva da ferramenta Jmogen para incluir a geração de diagramas de classes.

Nesse processo de manutenção, foram adicionados à ferramenta dois módulos: o primeiro é responsável pela geração do modelo de classes dos *frameworks*; e o segundo pela geração do modelo de classes dos sistemas. Ambos os módulos foram construídos utilizando a ferramenta Acceleo[1] e os modelos de classes gerados por esses módulos podem ser visualizados no editor UML Papyrus[2].

Fundamentação Teórica

Um domínio consiste em um foco de aplicação de um sistema. Um conjunto de sistemas que pertencem ao mesmo domínio compartilham um núcleo de características semelhantes e algumas variabilidades (Rumpe et al. 2010). Por exemplo, em um domínio de clínicas médicas, algumas das características poderiam ser: Atendimento e Paciente. Atendimento poderia ter duas variabilidades, Gratuito ou Privado, que poderiam ocorrer, ora em alguns sistemas, ora em outros, dentro do mesmo domínio.

Um *framework* é um software que implementa um conjunto de funções de forma genérica e que pode ser configurado para dar origem a sistemas específicos. Alguns frameworks auxiliam na implementação de partes específicas dos sistemas, como persistência de dados e organização da arquitetura em camadas, já outros apoiam o desenvolvimento de sistemas de informação, pois implementam domínios de aplicação específicos da indústria, negócios, etc. [Fayad e Johnson, 1999].

A Jmogen é uma ferramenta que auxilia o desenvolvedor na criação de um modelo de um domínio [Viana et al. 2013]. A partir desse modelo são gerados o código do *framework* e a DSL do domínio modelado. A partir da DSL, o desenvolvedor pode modelar sistemas de informação de forma mais simples quando comparado a linguagens de modelagem de uso geral, como a UML, uma vez que a DSL retrata os elementos e restrições do domínio retratado. A Jmogen utiliza esse modelo para gerar o código do sistema, que por sua vez, faz uso do *framework* do domínio.

Metodologia de Pesquisa

O processo de manutenção da Jmogen iniciou com a construção do módulo de geração de modelos de classes UML dos *frameworks* (domínio). Primeiramente, foi feita uma análise da estrutura do código dos *frameworks* gerados. Os dois primeiros pacotes do código: *dcore.model*, que contém classes abstratas com atributos e funções comuns a todas as classes da camada de modelo dos *framework* e dos sistemas; *dcore.persistence*, que contém classes abstratas com a funcionalidade básica de persistência de dados. Esses dois primeiros pacotes estão presentes em todos os *frameworks* gerados pela Jmogen. Os outros dois pacotes, *nome_do_domínio.model* e *nome_do_domínio.persistence*, também possuem classes abstratas que implementam as funcionalidades da camada de modelo e de persistência, porém os atributos, métodos e relacionamentos dessas classes correspondem ao que foi definido no modelo do domínio. Por exemplo, no domínio de clínicas de saúde, que possui a característica *Paciente*, o pacote *clinicas.model* contém a classe *Paciente*, com os atributos e métodos, e o pacote *clinicas.persistence* contém a classe *PacienteDAO*, com os métodos que garantem a persistência da classe Paciente.

1 https://www.eclipse.org/acceleo/
2 https://www.eclipse.org/papyrus/

Como os pacotes *dcore.model* e *dcore.persistence* são fixos e comuns a todos os modelos de sistemas gerados, independentemente do domínio do *framework*, o modelo de classe (arquivo .uml) desses pacotes foi construído manualmente. O passo seguinte foi construir um gerador de arquivos .uml (que internamente são arquivos XMI), que contém um modelo das classes dos pacotes específicos de cada *framework*. Para a construção desse gerador, foi utilizado o Acceleo, que permite construir geradores *model-to-text* com base em *templates*. Com isso, foram criados os diversos *templates*, cada um relacionado com uma estrutura do diagrama de classes da UML, como pacotes, classes, atributos, métodos, relacionamentos de associação, herança e dependência. Os *templates* criados foram compilados pelo Acceleo, dando origem ao módulo gerador capaz de ler um modelo de domínio (arquivo .dcore criado com a Jmogen) e gerar um arquivo .uml que descreve o código do *framework*. Com isso, em sua nova versão, a Jmogen passou a gerar o modelo de classes dos *frameworks* junto com o código deste último e a DSL do domínio. A Figura 1 mostra um trecho do código do *template* responsável por gerar as classes nos arquivos .uml.

```
[template public gFeature(aFeature : Feature)]
<packagedElement xmi:type="uml:Class" xmi:id="[aFeature.name/]Class" name="[aFeature.name/]"
    [for (aAttribute : Attribute | aFeature.attributes)]
        [gAttribute(aAttribute)/]
    [/for]
    [for(aAssociation : Association | aFeature.associations)]
        [gAssociationProperty(aAssociation)/]
    [/for]
    [for (aOperation : Operation | aFeature.operations)]
        [gOperation(aOperation)/]
    [/for]
</packagedElement>
[/template]
```

Figura 1. Template responsável por gerar as classes nos modelos UML.

A construção do módulo de geração de modelos de classes dos sistemas também iniciou com uma análise da estrutura do código dos sistemas gerados pela Jmogen. Os sistemas gerados por ela também são organizados em pacotes referentes às camadas de modelo e persistência. Suas classes estendem as classes do *framework* do domínio. Por exemplo, em um sistema para clínicas veterinárias, pertencente ao domínio de clínicas de saúde, no pacote da camada de modelo, a classe *Animal* estende a classe Paciente do *framework*. Da mesma forma, no pacote da camada de persistência do sistema, a classe *AnimalDAO* estende a classe *PacienteDAO* do *framework*.

Os *templates* do módulo de geração de modelos de classes dos sistemas foram construídos de forma semelhante aos do módulo do domínio. Inclusive, foi possível reutilizar uma parte considerável dos *templates* do módulo anterior, devido ao fato de que as classes dos sistemas são especializações das classes dos *frameworks*. A partir dos *templates*, foi obtido o módulo gerador capaz de ler modelos de sistemas, criados com as DSLs dos domínios, para dar origem aos modelos de classes desses sistemas. Com isso, na nova versão da Jmogen, ao gerar o código de um sistema a partir do seu modelo, o modelo de classes desse sistema também é gerado.

Na Figura 2(a) é mostrado o arquivo .uml do *framework* de clínicas de saúde. Esta figura apresenta diferentes Pacotes, Classes, Propriedades, Operações e Relacionamentos que foram gerados neste trabalho. Arquivos .uml podem ser abertos pelo *plug-in* Papyrus do Eclipse IDE, que proporciona uma visualização gráfica dos modelos UML, como mostrado, em parte, na Figura 2(b), auxiliando no entendimento do sistema e a identificação de peculiaridades que possam ser alteradas ou corrigidas .

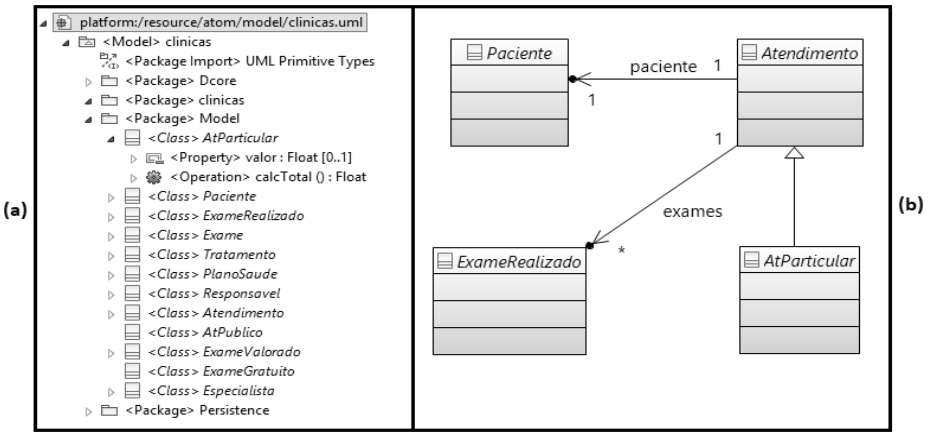

Figura 2. Modelo de Classes gerado pela ferramenta para o domínio de clínicas de saúde.

Resultados

Os resultados conquistados com esta nova versão da ferramenta são: auxílio no desenvolvimento de sistemas baseados em *frameworks*, proporcionando maior eficiência no processo de desenvolvimento e maior qualidade nos produtos. Torna-se possível fornecer uma documentação dos *frameworks* e sistemas gerados de forma automática e eficiente, permitindo uma visualização e entendimento de toda a sua estrutura. Além de facilitar a manutenção dos artefatos gerados, auxiliando na identificação de características que devem ser alteradas/corrigidas. A ferramenta permite corrigir o modelo da aplicação e gerar novamente o código, inclusive mantendo alterações manuais feitas diretamente no código.

Conclusão

As novas funcionalidades tornam a Jmogen uma ferramenta mais completa, cobrindo todos os aspectos do processo de desenvolvimento de software. Este upgrade permite a qualquer pessoa que queira estudar sobre o sistema tenha o seu modelo de classes, auxiliando a suas manutenções e evoluções futuras. A Jmogen pode ser utilizada em atividades relacionadas com o desenvolvimento de sistemas por alunos e empresas. Seu processo de manutenção integrou diversas áreas de estudo da Engenharia de Software, como linguagens de programação, desenvolvimento e manutenção de software, desenvolvimento dirigido por modelo e geração de código.

Referências

Fayad, M. E.; Johnson, R. E. (1999) “Domain-Specific Application Frameworks: Frameworks Experience by Industry”, Wiley, New York.

Rumpe, B. e Schindler, M. e Volkel, S e Weisemoller, I. (2010) “Generative Software Development”, ACM/IEEE International Conference on Software Engineering, v. 2, p. 473–474, maio.

Viana, M. e Penteado, R. A. D. e do Prado, A. F. e Durelli, R. (2013) “F3T: From Features to Frameworks Tool”, 27th Brazilian Symposium on Software Engineering, Brasilia, Brasil.

J. M. Jezequel. Model-Driven Engineering for Software Product Lines. ISRN Software Engineering, 2012.