Um Framework para Especificação Declarativa de Sintaxes Concretas em Linguagens de Modelagem

Félix S. de Souza Neto e Sandro S. Andrade

Grupo de Pesquisa em Sistemas Distribuídos, Otimização, Redes e Tempo-Real (GSORT)
Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBa)
Av. Araújo Pinho, nº 39 - Canela - Salvador - BA - CEP: 40.110-150

{felixneto, sandroandrade}@ifba.edu.br

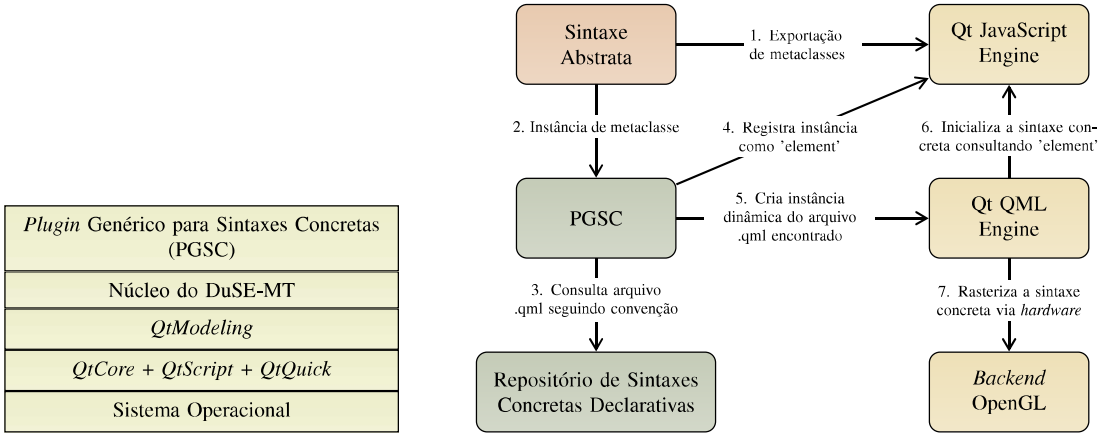
Resumo. *A utilização de modelos em projetos de desenvolvimento de software é atualmente reconhecida como fator importante para a melhoria da produtividade do processo e da qualidade dos artefatos gerados. Tais modelos são geralmente descritos em alguma linguagem de modelagem, que disponibiliza os constructos necessários e viabiliza a manipulação sistemática de modelos. Sintaxes concretas visuais permitem que a criação dos elementos do modelo, configuração dos seus atributos e definição de relacionamentos sejam realizados de uma forma muito mais simples. Este artigo apresenta o projeto e implementação de um framework para implementação de sintaxes concretas visuais de forma declarativa e independente do metamodelo em questão. O framework tem sido utilizado para implementar a sintaxe concreta de constructos básicos da UML em uma ferramenta open-source de modelagem.*

1. Introdução

A definição das tecnologias básicas para criação de modelos de *software* e para integração destes modelos em processos de desenvolvimento tem sido o foco de diversas pesquisas na área de *Model-Driven Software Engineering* (MDSE) [Brambilla et al. 2012]. Modelos de *software* são descritos em alguma linguagem de modelagem. Dentre estas, as linguagens baseadas em metamodelos têm sido amplamente utilizadas como infraestrutura básica para abordagens da MDSE. Uma linguagem de modelagem baseada em metamodelo é constituída por três elementos principais: sintaxe abstrata, sintaxe concreta e semântica [Voelter et al. 2013].

A sintaxe abstrata define as metaclasses que representam os constructos disponibilizados pela linguagem, seus diversos atributos e como estes constructos se relacionam. A sintaxe concreta, por sua vez, viabiliza a criação de um modelo de uma forma mais simples do que utilizar uma linguagem de programação de propósito geral para instanciar diretamente as metaclasses, ajustar atributos e definir relacionamentos. Ela disponibiliza elementos textuais ou visuais de maior abstração, tornando as operações de modelagem mais simples e produtivas, bem como viabilizando o suporte por ferramentas.

Embora iniciativas tais como o *Eclipse Modeling Framework* (EMF) [Steinberg et al. 2009], *Graphical Editing Framework* (GEF) [Rubel et al. 2011] e Xtext/Xtend [Bettini 2013] facilitem consideravelmente a construção de novas linguagens de modelagem, a implementação de sintaxes concretas e de mapeamentos com elementos da sintaxe abstrata é ainda uma tarefa desafiadora. O esforço de codificação necessário para implementação das representações visuais e definição dos mecanismos de *binding*



(a) *Stack* de tecnologias do DuSE-MT (b) Principais componentes do *framework* proposto

Figura 1. Stack de tecnologias (a) e principais componentes (b) utilizados no *framework* proposto.

entre sintaxe concreta e abstrata é geralmente alto. Adicionalmente, a solução deve ser escalável ao manipular grandes modelos.

Este artigo apresenta a proposta e implementação de um *framework* para definição facilitada de sintaxes concretas visuais para linguagens de modelagem. O *framework* é independente do metamodelo utilizado na definição da linguagem e combina uma série de tecnologias que viabilizam: *i)* a utilização de uma linguagem declarativa para definição da sintaxe concreta; e *ii)* renderização suportada por *hardware* das representações visuais utilizadas. Tais características contribuem significativamente para a produtividade e escalabilidade, respectivamente, da solução aqui proposta.

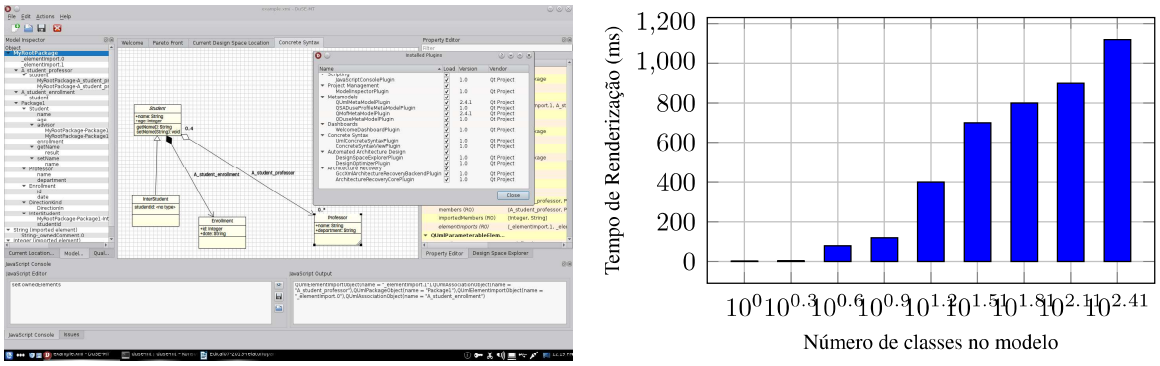
O *framework* proposto foi implementado utilizando a linguagem C++ e o *toolkit* Qt [Qt Community 2014], como parte integrante do DuSE-MT[1] [Andrade and Macêdo 2013] – ferramenta *open-source* para criação e manipulação de modelos de *software*. O *framework* implementa a exportação dos objetos da sintaxe concreta para o interpretador do QML – linguagem declarativa para criação de interfaces gráficas, disponibilizada pelo Qt. Adicionalmente, um mecanismo genérico para mapeamento entre sintaxe abstrata e sintaxe concreta é também disponibilizado pela solução proposta, fazendo com que a implementação de sintaxes concretas se resuma à escrita de código declarativo em conformidade com algumas convenções definidas pelo *framework*.

2. O Framework Proposto

O *framework* proposto neste artigo foi implementado utilizando a linguagem C++ e o *toolkit* Qt, como parte integrante da ferramenta DuSE-MT. Conforme apresentado na Figura 1(a), o DuSE-MT é uma aplicação *cross-platform* que utiliza os módulos *QtCore*, *QtScript* e *QtQuick* para implementar os serviços básicos de manipulação de modelos via *scripts* e viabilizar a utilização da linguagem declarativa QML para renderização das sintaxes concretas.

Todas as funcionalidades para criação, manipulação e serialização de modelos de

[1]http://duse.sf.net



(a) (b)

Figura 2. Exemplo de sintaxe concreta produzida (a) e avaliação de desempenho do framework implementado (b).

software estão presentes no módulo *QtModeling*. O núcleo do DuSE-MT define uma infraestrutura que, via mecanismos de reflexão computacional, permite a manipulação de modelos de forma independente do metamodelo utilizado. Uma arquitetura baseada em *plugins* permite a extensão facilitada da ferramenta e foi o mecanismo básico para implementação do *framework* aqui descrito.

Conforme ilustrado na Figura 1(b), o componente central do *framework* é o Plugin Genérico para Sintaxes Concretas (PGSC). O fluxo básico de renderização de sintaxes concretas é também apresentado na Figura 1(b). No passo 1, todas as metaclasses do metamodelo em questão são exportadas para a *engine* de JavaScript do Qt, o que permite a consulta e manipulação facilitada de modelos. Ao solicitar, no passo 2, a renderização da sintaxe concreta de uma determina instância de metaclasse (por exemplo, `QUmlClass`), o PGSC consulta um repositório de sintaxes concretas declarativas, buscando por um arquivo-fonte cujo nome segue a convenção `<namespace-do-metamodelo><metaclasse-do-metamodelo>.qml` (`UmlClass.qml` para o exemplo acima).

As informações sobre o *namespace* e metaclasse em questão são obtidas via reflexão computacional, de modo a manter a solução independente de metamodelo. Encontrada a implementação declarativa da sintaxe concreta referente à metaclasse em questão, o PGSC registra, a instância da metaclasse fornecida, no *engine* de JavaScript do Qt, com o identificador `element`. Este identificador é utilizado pelas implementações declarativas para obter as informações necessárias à renderização da sintaxe concreta. A figura 2(a) apresenta um exemplo de sintaxe concreta resultante da implementação declarativa realizada.

3. Validação

O *framework* proposto neste trabalho é caracterizado pela viabilização da implementação produtiva de sintaxes concretas em função do uso de linguagens declarativas. Adicionalmente, o *backend* OpenGL para renderização de arquivos .qml, disponibilizado pelo Qt, garante excelente desempenho, mesmo na renderização de sintaxes concretas formadas por um número grande de elementos.

Com o objetivo de avaliar a escalabilidade das atividades de renderização de mo-

delos UML pelo *framework* proposto, verificou-se como o tempo de renderização varia à medida em que novas classes são introduzidas no modelo. Conforme observado na Figura 2(b), embora aparentemente o tempo de renderização inicialmente cresca exponencialmente, este tempo passa a ter um comportamento polinomial para um número alto de classes no modelo. Acredita-se que os mecanismos introduzidos na versão 5.2 do QML tenha trazidos benefícios substanciais no desempenho de renderizações.

4. Conclusões e Trabalhos Futuros

Este artigo apresentou o projeto e implementação de um *framework* que viabiliza a implementação de sintaxes concretas de forma independente de metamodelo e através da utilização de linguagens declarativas. Foram apresentados os fundamentos sobre as atividades de metamodelagem e definição de sintaxes concretas.

A arquitetura definida para o *framework* permite a extensão facilitada de novas sintaxes concretas sem impactos no núcleo da ferramenta de modelagem. Adicionalmente, o registro e acesso automático às propriedades das metaclasses através de *scripts* alavanca um desenvolvimento mais produtivo de sintaxes concretas.

Como trabalhos futuros pode-se citar o desenvolvimento de mecanismos para alteração de atributos das instâncias de metaclasses diretamente através da sintaxe concreta, a definição de novos elementos QML para facilitar o reuso de elementos gráficos comuns a várias sintaxes concretas e a definição de mecanismos de conversão entre metamodelos de representação de sintaxes concretas.

Referências

[Andrade and Macêdo 2013] Andrade, S. S. and Macêdo, R. J. d. A. (2013). A search-based approach for architectural design of feedback control concerns in self-adaptive systems. In *Self-Adaptive and Self-Organizing Systems (SASO), 2013 IEEE 7th International Conference on*, pages 61–70. IEEE.

[Bettini 2013] Bettini, L. (2013). *Implementing Domain-Specific Languages with Xtext and Xtend*. Packt Publishing Ltd.

[Brambilla et al. 2012] Brambilla, M., Cabot, J., and Wimmer, M. (2012). *Model-Driven Software Engineering in Practice*. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers.

[Qt Community 2014] Qt Community (2014). Qt project. http://qt-project.org/. Acesso: 28/03/2014.

[Rubel et al. 2011] Rubel, D., Wren, J., and Clayberg, E. (2011). *The Eclipse Graphical Editing Framework (GEF)*. Eclipse (Addison-Wesley). Addison-Wesley.

[Steinberg et al. 2009] Steinberg, D., Budinsky, F., Paternostro, M., and Merks, E. (2009). *EMF: Eclipse Modeling Framework 2.0*. Addison-Wesley Professional, 2nd edition.

[Voelter et al. 2013] Voelter, M., Benz, S., Dietrich, C., Engelmann, B., Helander, M., Kats, L. C. L., Visser, E., and Wachsmuth, G. (2013). *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. dslbook.org.