Modalidade do trabalho: Relatório técnico-científico
Evento: XXII Seminário de Iniciação Científica

UMA FERRAMENTA DE APOIO À EDIÇÃO E VALIDAÇÃO DE OVMS TEXTUAIS PARA DAR SUPORTE AO PROCESSO DE ANÁLISE AUTOMÁTICA[1]

Cristiano Politowski[2], Fabrícia Roos-Frantz[3], Rafael Z. Frantz[4], Sandro Sawiki[5].

[1] Pesquisa de Iniciação Científica desenvolvida no Grupo de Pesquisa em Computação Aplicada (GCA) da UNIJUI
[2] Bolsista PIBITI/CNPq, Ciência da Computação, crispolitowski@gmail.com.
[3] Professora Orientadora, frfrantz@unijui.edu.br.
[4] Pesquisador Colaborador, rzfrantz@unijui.edu.br.
[5] Pesquisador Colaborador, sawicki@unijui.edu.br

Introdução

Em 1913, Henry Ford com sua Linha de Produção revolucionou a maneira como eram fabricados os carros e assim reduziu os custos de produção e venda para o consumidor final. O problema, no entanto, era a falta de diversidade dos produtos devido a padronização da produção. Não demorou muito para que os clientes exigissem produtos customizados que fossem ao encontro de suas necessidades individuais.

A customização em massa veio para suprir essa demanda, proporcionando uma técnica de produção de larga escala de bens sob medida para necessidades individuais [DAVIS, 1987]. No entanto, esta técnica agregava custos para as empresas onde, plataformas comuns para diferentes tipos de produtos foram utilizadas, planejando previamente quais partes seriam usadas na construção de produtos posteriores. A junção de customização em massa e plataformas comuns deram origem ao paradigma de Engenharia de Linhas de Produtos de Software (Software Product Line Engineering - SPLE) [POHL, 2005].

Para trabalhar com a SPLE é necessário gerir a variabilidade através de um modelo a parte. Na literatura existem diferentes modelos para representar esta variabilidade, dentre os quais, Feature Model é o mais popular [Kang, 1990]. Pohl et al. 2005, propuseram uma modelagem ortogonal da variabilidade, conhecido como Modelo de Variabilidade Ortogonal (Orthogonal Variability Model - OVM)[Pohl, 2005].

OVM é uma linguagem de domínio específico, que possui notação gráfica e sintaxe abstrata definidas em Pohl (2005). Apesar da notação gráfica, a ferramenta de análise automática existente na literatura, denominada FaMa-OVM, trabalha com OVMs no formato textual [ROOS-FRANTZ, 2012]. Esta prática de usar a versão textual dos modelos como entrada para ferramentas, seja em formato XML ou em formato específico, é comum na literatura [ROOS-FRANTZ, 2012; CLASSEN; BOUCHER; HEYMANS, 2011; MERKLE, 2010].

Modalidade do trabalho: Relatório técnico-científico
Evento: XXII Seminário de Iniciação Científica

A ferramenta FaMa-OVM utiliza como entrada um modelo OVM em formato textual, o qual foi definido informalmente em Roos-Frantz (2012). Esta ferramenta não oferece nenhum suporte à edição de OVMs, e se este modelo de entrada não estiver no formato correto, antes da fase da análise, a ferramenta acusará um erro no carregamento do modelo. A edição de modelos OVM textuais corretos, sem o auxílio de uma ferramenta de edição e correção automática de erros de sintaxe, é uma tarefa trabalhosa e muito propensa a erros.
Desenvolver uma ferramenta de edição de OVM que facilite a tarefa do usuário, isto é, com recurso de complementação de código (code completion), coloração de fonte (coloring), análise estática (static analysis), etc, não é uma tarefa trivial. Primeiro, é necessário definir a gramática da linguagem, logo, criar uma ferramenta de edição que seja capaz de fazer a análise sintática automaticamente, criar meios de validação semântica, criar uma formatação textual a fim de deixar o código uniforme e possibilitar recursos de coloração de sintaxe a fim de aumentar a legibilidade do código.

Metodologia

Revisou-se a literatura técnica e científica relacionada com o tema da pesquisa. A bibliografia, indicada pela professor orientador, constituiu-se de livros e artigos científicos retirados de bases internacionais.
Após o estudo bibliográfico, fez-se um estudo de ferramentas open sources, através de buscas na web e artigos científicos de comparação entre as ferramentas, para auxiliar na criação da linguagem textual e do editor.
Após a identificação da ferramenta mais adequada para o desenvolvimento do editor, passou-se à fase de criação deste. Esta fase constituiu-se de estudo aprofundado da documentação da ferramenta, utilizando, sempre que necessário, pesquisas em fóruns e blogs relacionados. Finalmente, passou-se à fase de validação do editor, a qual deu-se através de testes, utilizando como base os trabalhos de Myers et al (2004), Pressman (1986) e Segura et al (2010).

Resultados e discussão

Com base na revisão da literatura, exploramos o uso dos chamados Language Workbenchs, os quais são considerados Interface de Desenvolvimento Integrado (IDE's) para auxiliar na construção de linguagens de Propósito Geral e de Domínio Específico [Fowler, 2005]. Essas ferramentas, além de propiciar a criação da gramática da linguagem (normalmente na notação
BNF), também oferecem outros recursos como validação, formatação, coloração de sintaxe,
etc. A partir desta revisão, o primeiro passo para a criação da gramática foi definir qual Language Workbench usar. Para a escolha da ferramenta foram analisados alguns artigos como Markle (2010) e Stoffel (2010), além da avaliação de funcionalidades das mesmas. Xtext [XTEXT, 2013] foi a ferramenta escolhida por ser uma das mais conhecidas na comunidade de desenvolvedores, além de proporcionar uma grande gama de recursos adicionais, que não apenas a criação da gramática.

Modalidade do trabalho: Relatório técnico-científico
Evento: XXII Seminário de Iniciação Científica

No Xtext a gramática é criada, basicamente, fazendo o uso de Parser Rules e Terminal Rules. A primeira é composta por uma sequência de Terminal Rules e não produz tokens terminais. A última faz parte da fase de análise léxica e representa um símbolo atômico.
Um modelo OVM possui dois elementos principais: Variation points e Variants. Além disso, possui um conjunto de restrições entre estes elementos. A parte principal da gramática de OVM textual, criada durante o desenvolvimento deste trabalho, pode ser vista na Figura 1. A Parse Rule "Model" define toda a estrutura da linguagem. São criados rótulos, e apenas um, para cada bloco da linguagem. Cada um desses blocos pode conter N elementos, sejam eles quais forem.

```
5  Model:
6      relationships=Relationships          (variationPoint+=VariationPoint)*
7      attributes=Attributes                (attribute+=Attribute)*
8      globalAttributes=GlobalAttributes    (globalAttribute+=GlobalAttribute)*
9      constraints=Constraints              (constraint+=Constraint)*;
10
11 Relationships:
12     '%Relationships';
13
14 VariationPoint:
15     (name=QualifiedName | '[' name=QualifiedName ']') ':'
16     (cardinality=Cardinality)?
17     (variant+=Variant)* ';';
18
19 Cardinality:
20     '[' min=INT ',' max=INT ']' '{' (variant+=Variant)* '}';
21
22 Variant:
23     name=QualifiedName | '[' name=QualifiedName ']';
24
25 QualifiedName:
26     ID;
```

Figura 1. Gramática da linguagem OVM.

Testes

Para validar o editor criado é necessário criar um conjunto de casos de testes que validem a funcionalidade da ferramenta de edição criada. Testes de software tem o objetivo de revelar falhas no software testado. Isto é feito checando o comportamento do programa com diferentes entradas e saídas [MYERS, 2004; PRESSMAN, 1986].
Cada teste foi pensado para ter como saída somente um tipo de falha, semelhante a Segura (2010). Criar testes que cubram 100% das possibilidades é uma tarefa impossível [MYERS, 2004]. No entanto, Pressman (1986) diz que devemos criar testes que tenham mais chances de achar erros com o menor esforço possível.

Modalidade do trabalho: Relatório técnico-científico
Evento: XXII Seminário de Iniciação Científica

Os testes foram projetados tendo como entrada um modelo OVM tendo como entrada um modelo OVM e como saída um valor verdadeiro ou falso. Primeiramente foram criados testes simples, chamados de básicos, que possuem somente elementos dentro da tag %Relationships. Testes envolvendo Constraints e Attributes também foram feitos. Todos esses testes necessitam uma saída verdadeira para estarem corretos, pois, segundo a sintaxe da linguagem, são modelos válidos. A Tabela 1 mostra os resultados dos testes realizados.

Teste	Descrição	Entrada	Saída
Basic1	Modelo vazio, somente tags.	OVM	Aceito
Basic2	Modelo com *Variation Point* vazio.	OVM	Aceito
Basic3	Mais de um *Variation Point* com diferentes *Variants*.	OVM	Aceito
Constraint1	Todas as *Constraints* possíveis da sintaxe.	OVM	Aceito
Attribute1	Todos os *Attributes* possíveis na sintaxe.	OVM	Aceito
Attribute2	Todos os *Global Attributes* possíveis na sintaxe.	OVM	Aceito

Tabela 1. Testes que esperam um resultado de "Aceito".

Também foram feitos testes que esperam resultados errôneos. Estes testes tem o intuito de verificar a própria validação que o editor possui. Semelhante aos testes feitos anteriormente, estes também possuem um nome, descrição, entrada e saída. No entanto, a saída que se espera é um erro do editor, ou seja, caso um erro ocorrer, o teste estará correto. A Tabela 2 mostra os testes realizados. Os arquivos testados podem ser encontrados em http://www.gca.unijui.edu.br/cpolitowski/Models.

Teste	Descrição	Entrada	Saída
Error1	Erro em *Constraints*. Um elemento não pode se auto referenciar.	OVM	Erro
Error2	Erro em *Cardinality*. Cardinalidade mínima não respeitada.	OVM	Erro
Error3	Erro em *Variation Points* e *Variants*. Elementos não podem ter mesmo nome.	OVM	Erro
Error4	Erro em *Constraints*. Ponto de Variação não pode referenciar uma Variante.	OVM	Erro
Error5	Erro em *Attributes*. Atributos devem ter nomes únicos.	OVM	Erro
Error6	Erro em *Global Attributes*. Global Atributos devem ter nomes únicos.	OVM	Erro

Tabela 2. Testes que esperam um resultado de "Erro".

Conclusões

Trabalhar com questões como gramática para construção de linguagens pode se tornar algo complexo devido a falta de padrão. Mesmo com uma ferramenta que traga algumas definições pré-estabelecidas, ainda tem-se um quadro em branco onde se pode criar qualquer coisa de qualquer jeito. No entanto, neste trabalho, o cenário é diferente, pois a linguagem já estava definida, restando a criação da gramática para validá-la. Porém, somente a criação da gramática não é o suficiente para

Modalidade do trabalho: Relatório técnico-científico
Evento: XXII Seminário de Iniciação Científica

a geração de um ambiente de criação e edição de OVMs que dê suporte ao desenvolvedor, é necessário oferecer uma ferramenta.
Este trabalho oferece uma ferramenta standalone de criação e edição de modelos OVM textuais, com recursos de validação e formatação. Esta ferramenta melhora a experiência de uso de modelos OVM e dá suporte ao processo de análise daquelas linhas de produtos onde modelos OVM documentam a variabilidade. A ferramenta foi testada em várias plataformas e seu funcionamento ocorreu sem problemas, ou seja, está pronta para ser usada.

Palavras-Chave: Linha de Produtos de Software; Modelos de Variabilidade; Análise Automática de Modelos de Variabilidade; Linguagens de Domínio Específico; Gramática.

Agradecimentos
Ao CNPq pela concessão da Bolsa de Iniciação em Desenvolvimento Tecnológico e Inovação (PIBITI).

Referências Bibliográficas
CLASSEN, A.; BOUCHER, Q.; HEYMANS, P. A text-based approach to feature modelling: Syntax and semantics of TVL. Science of Computer Programming, v. 76, n. 12, p. 130–1143, dez. 2011.
DAVIS, S. M. From future perfect: Mass customizing. Strategy and Leadership, v. 17, n. 2, p. 16–21, 1989.
FOWLER, M. Domain-Specific Languages. [S.l.: s.n.], 2010. 0–321 p. ISBN 9780321712943.
KANG, K. C. et al. Feature-Oriented Domain Analysis (FODA) Feasibility Study. [S.l.], 1990.
MERKLE, B. Textual modeling tools. New York, New York, USA, p. 139, 2010.
MYERS, G. et al. The Art of Software Testing. [S.l.]: John Wiley & Sons, 2004. (Business Data Processing: a Wiley Series). ISBN 9780471469124.
PRESSMAN, R. S. Software Engineering: A Practitioner's Approach (2Nd Ed.). New York, NY, USA: McGraw-Hill, Inc., 1986. ISBN 0-070-50783-X.
SEGURA, S.; BENAVIDES, D.; RUIZ-CORTéS, A. FaMa Test Suite v1.2. [S.l.], 2010 .
POHL, K.; BöCKLE, G.; LINDEN, F. J. v. d. Software Product Line Engineering: Foundations, Principles and Techniques. [S.l.]: Springer-Verlag, 2005.
ROOS-FRANTZ, F. et al. FaMa-OVM: a tool for the automated analysis of OVMs. In: Proceedings of the 16th International Software Product Line Conference - Volume 2. ACM, 2012. (SPLC '12), p. 250–254.
STOFFEL, R. Comparing Language Workbenches. 2010.
XTEXT. Xtext Documentation. Disponível em: http://www.eclipse.org/Xtext/documentation.html. Acesso em: 04 junho. 2014.

Modalidade do trabalho: Relatório técnico-científico
Evento: XXII Seminário de Iniciação Científica