FalaOCL: Uma Ferramenta para Parafrasear OCL

Geraldo Zimbrão[1], Rodrigo A. Miranda[1], Jano M. de Souza[2], Francisco P. Neto[3]
[1]DCC/UFRJ – Ilha do Fundão, Rio de Janeiro, Brazil
[2]COPPE /UFRJ – Ilha do Fundão, Rio de Janeiro, Brazil
{zimbrao,estolano,jano}@cos.ufrj.br
[3]Marinha do Brasil – Ilha das Cobras, Rio de Janeiro, Brazil
pneto@dabm.mar.mil.br

Resumo

A Object Constraint Language (OCL) é uma linguagem proposta para ser utilizada em conjunto com outras ferramentas, tais como os diagramas de classe da UML, e que serve para especificar restrições em um modelo de classes. Além disso, existem uma série de abordagens propondo o uso de OCL para a documentação de Regras de Negócio. Embora a OCL seja uma linguagem declarativa e de fácil entendimento, programadores pouco familiarizados com ela podem ter certa dificuldade em entender expressões em OCL. Este artigo apresenta a ferramenta FalaOCL. Esta ferramenta é um módulo parafraseasor de OCL, ou seja, dada uma expressão em OCL, a ferramenta gera um texto que explica, em linguagem natural, a expressão dada. Com isso, é possível a um usuário sem habilidades de programação em OCL entender expressões em OCL, facilitando o trabalho de validação e manutenção do sistema.

Abstract

The Object Constraint Language (OCL) is a language to be used together with other UML tools like the class diagram, and its main purpose is to states constraints in a class model. Moreover, there are many approaches suggesting the use of OCL to states Business Rules. Although OCL is an easy to understand declarative language, non OCL-familiarized programmers can experiment some difficulty in understanding OCL expressions. This paper presents the tool FalaOCL (OCLTalk). FalaOCL is a tool that paraphrases OCL expressions, that is, it generates an equivalent representation of the given expression in natural language. Using this tool, an user that does not know OCL or is not a programmer is able to understand OCL expressions. Then, this tool make easy to validate and maintain an information system.

1. Introdução

Embora exista uma série de propostas para a formalizar a especificação (ou parte dela) de Sistemas de Informação, a maior parte da especificação ainda é feita em Linguagem Natural, ou pelo menos deve ser possível gerar através das ferramentas utilizadas uma documentação em Linguagem Natural. Recentemente, tem sido difundido o uso da linguagem OCL para a representação de restrições nos modelos de Classes, e inclusive, tem sido proposto por alguns autores o uso da OCL para a representação das Regras de Negócio. De qualquer forma, o uso da OCL está sendo difundido, e ainda há uma carência de ferramentas para suportá-lo. Este trabalho apresenta a ferramenta FalaOCL. Esta ferramenta é um módulo parafraseasor de OCL, ou seja, dada uma expressão em OCL, a ferramenta gera um texto que explica, em linguagem natural, a expressão dada. Com isso, é possível a um usuário sem habilidades de programação em OCL entender expressões em OCL, facilitando o trabalho de validação e manutenção do sistema. Além disso, é possível gerar automaticamente documentação sobre as restrições especificadas em OCL.

A ferramenta FalaOCL foi desenvolvida como um módulo de um sistema maior, o Atenas [10]. O Atenas é um sistema que permite a documentação e também a formalização de regras de negócio em OCL, suportando a sua manipulação desde as fases iniciais da análise até as etapas de manutenção. No entanto, por ser desenvolvido como um módulo a parte, a ferramenta FalaOCL pode ser utilizado independentemente do uso do Sistema Atenas.

Este artigo está organizado da seguinte forma: a seção 2 apresenta a motivação, definindo o problema. A seção 3 apresenta o compilador que realiza a tradução de OCL para Linguagem Natural. A seção 4 apresenta alguns exemplos de expressões traduzidas. A seção 5 mostra as conclusões e trabalhos futuros.

2. Motivação

Muitos autores defendem que regras de negócio devem ser estabelecidas declarativamente ([1], [2], [3], [6], [7], [8]). Uma regra de negócio estabelecida declarativamente não especifica os eventos em que a regra deve ser verificada – cabe ao sistema inferir os eventos onde a regra pode ou não ter sido violada.

A UML tem tido uma vasta aceitação na comunidade de sistemas de informação para a modelagem de *software*. Uma das ferramentas sugeridas pela UML é a OCL [9]. OCL é a linguagem da UML para a especificação de restrições, e pode ser utilizada para especificar também as regras de negócio, tanto as regras derivativas quanto as restritivas ([2], [3]). Uma expressão em OCL deve estar conectada a um objeto de uma classe, chamado de contexto, e pode se referir a este objeto ou a qualquer um de seus atributos ou operações. Uma regra de negócio derivativa pode ser expressa em OCL como um método de alguma classe. Como exemplo, tome a definição de uma operação que calcula a idade de um empregado:

```
Context Empregado::idade(): Integer
result = ano( systemDate() – dataNascimento )
```

Já uma regra de negócio restritiva pode ser expressa em OCL como um invariante, ou seja, uma expressão que deve ser sempre verdadeira. Como exemplo, temos uma regra determinando que o salário de um empregado não pode ser menor que o salário base do cargo.

```
Context Empregado inv:
salario >= cargo.salarioBase
```

A linguagem OCL é bastante abrangente, sendo equivalente ao SQL em poder de computação [5]. Porém, é mais clara e concisa do que SQL, contendo por exemplo diversas operações para a manipulação de conjunto. O exemplo a seguir estabelece que, se um projeto é de manutenção, então o número de funcionários alocados deve ser menor ou igual a vinte, dos quais pelo menos três devem ser do mesmo departamento do projeto.

```
Context Projeto inv:
categoria == 'Manutenção' implies
(funcAlocados->count() <= 20 and
 departamento.funcionarios->intersects( funcAlocados )->count() > 3)
```

Enquanto a maioria das regras derivativas é chamada implicitamente pela aplicação, as regras restritivas estão implicitamente ligadas aos eventos – dizemos que um evento potencialmente viola ou dispara a verificação de uma determinada regra. Desta forma, a regra anterior deve ser verificada em dois eventos de sistema distintos: quando um novo empregado for contratado ou quando o salário de um funcionário for alterado.

2.1 Visão Geral da Arquitetura do Atenas

O Atenas possui uma arquitetura que permite a documentação formal das regras de negócio, bem como mecanismos para ligar esta especificação com a documentação em linguagem natural e com o código fonte. Isto é feito usando OCL para representar formalmente Regras de Negócio.

Para que um sistema funcione adequadamente, todas as regras de negócio devem estar de alguma forma documentadas e implementadas no sistema. Portanto, cada regra de negócio irá passar por um ciclo com várias atividades, mas que podem ser agrupadas em duas: documentação e implantação da regra.

- **Documentação:** Inicialmente, a regra de negócio reside externamente ao sistema, seja em um documento oficial, seja na cabeça de um usuário ou ainda sob a forma de uma tradição. A partir do reconhecimento de que uma regra de negócio não está representada no sistema é necessário adquirir o conhecimento necessário para a sua implementação no sistema. A isto chamamos documentação da regra de negócio. A documentação de uma regra produz como resultado a descrição da regra com detalhamento, clareza e precisão suficiente para a sua efetiva implantação.
- **Implantação:** apenas o que estiver codificado pode ser percebido pelo usuário como funcionalidade da aplicação. Portanto, a partir da documentação da regra deve ser feita a sua codificação em alguma linguagem de programação. A esta atividade chamamos de implantação da regra de negócio.

Tradicionalmente, a documentação da regra é feita em linguagem natural, possivelmente obedecendo a algum padrão de documento, e realizada em conjunto com o usuário (o detentor do conhecimento da regra de negócio) durante a fase de análise ou durante a fase de manutenção do sistema, e depois passada para uma equipe de projetistas e programadores que irão realmente codificar a regra de negócio. A abordagem proposta pelo sistema gerenciador de regras de negócio Atenas difere da abordagem tradicional na forma de documentar cada regra de negócio: além de documentar as regras de negócio em linguagem natural será utilizada também uma linguagem formal, a OCL. Como consequência, será possível gerar automaticamente o código para a implantação da regra. As vantagens desta abordagem são amplamente listadas na literatura correlata.

A arquitetura proposta para o Sistema Gerenciador de Regras de Negócio Atenas se divide nas seguintes fases: na etapa de documentação temos a aquisição, estruturação e formalização da regra de negócio; na etapa de implantação temos a fase de compilação da regra de negócio. A figura 1 ilustra a arquitetura proposta.

Na arquitetura proposta, o código referente às regras de negócio estará residindo em uma camada chamada *camada de regras de negócio*. Todo o acesso ao banco deve ser feito através desta camada, ou seja os aplicativos que compõem o sistema devem acessar exclusivamente os serviços oferecidos pela camada de regras de negócio. Esta camada será mantida automaticamente pelo sistema gerenciador de regras de negócio.

O formato final desejável para uma regra de negócio é a sua especificação em OCL. No entanto, a sua aquisição pode ser feita em linguagem natural, linguagem semi-estruturada ou diretamente em OCL. Se a aquisição da regra de negócio for feita em linguagem semi-estruturada, em um passo posterior ela será mapeada para OCL. Se for feita em linguagem natural, em um passo posterior ela será mapeada para uma linguagem semi-estruturada e depois para OCL.

A ferramenta FalaOCL oferece a capacidade de, partindo de uma regra de negócio escrita em OCL, gerar uma representação em linguagem natural equivalente. O objetivo é facilitar a validação junto ao usuário das regras de negócio já formalizadas: o usuário estabelece uma regra de negócio em linguagem natural ou semi-estruturada, que depois é convertida para a representação em OCL, e depois restaurada para uma linguagem natural padronizada, de forma que seja possível ao próprio usuário validar a regra produzida.

Finalmente, é importante notar que já existe na literatura uma proposta para a implementação de um parafraseador de OCL para o Inglês e Alemão. No entanto, trata-se ainda de um protótipo que não está disponível para avaliação, e o artigo [11] não dá maiores detalhes de sua implementação, de modo que não podemos comparar aquele trabalho com a nossa proposta – exceto pelo fato de que ele serve para validar também que é possível construir um parafraseador de OCL.

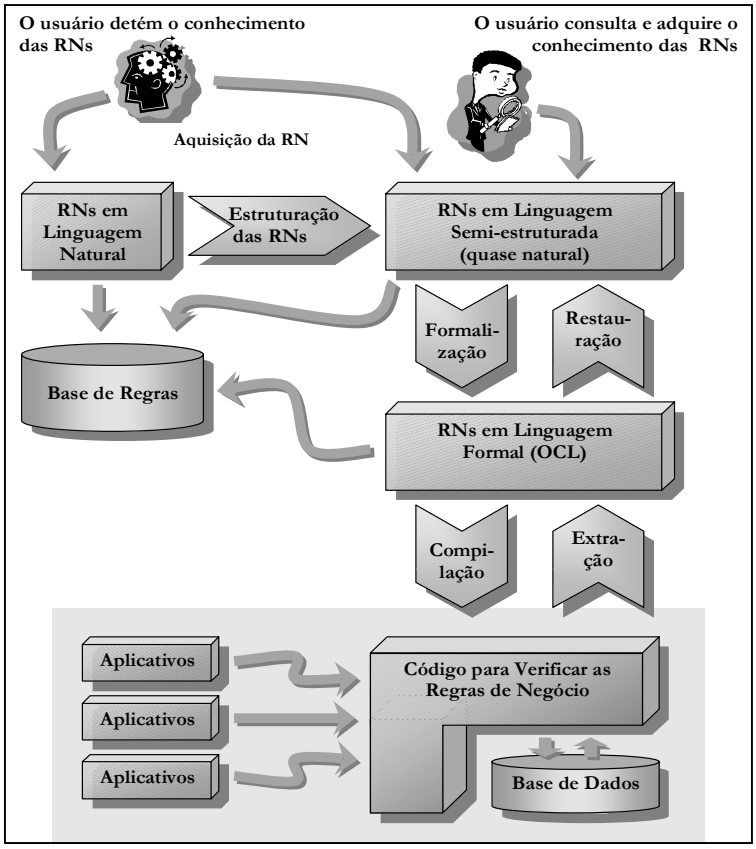

Figura 1 – Arquitetura do Atenas

3. O Compilador OCL

O compilador de OCL para SQL foi implementado usando um porte do Lex e Yacc para o Delphi conhecido como TPLY, e disponível na Internet [4], e utilizando a gramática original do OCL.

3.1 Glossário de Termos

Existe um glossário de termos simples, alimentado pelo sistema Atenas, mas cuja estrutura pode ser facilmente obtida de outras ferramentas tais como dicionários de dados etc. Basicamente, o glossário faz um mapeamento entre nomes de campos, que possuem certas restrições das linguagens de programação (p.e., não contém espaços em branco nem preposições) e nomes familiares ao domínio da aplicação. Além disso, são permitidas traduções específicas para determinadas classes usando a notação "ponto": NomeDaClasse.Atributo. A Tabela 1 mostra alguns exemplos.

Termo Formal	Linguagem Natural
Qtde	Quantidade
data_prevista_termino	data prevista para o término
Cargo.salarioBase	Salário base do cargo

Tabela 1 – Glossário de Termos

Além disso, o glossário contém informação sobre gênero e número de cada termo para a correta colocação dos artigos no texto gerado.

3.2 Árvore de Derivação

O compilador foi desenvolvido utilizando-se uma ferramenta LALR, que é um método ascendente, porém o processo de geração do texto em linguagem natural necessita de informações que caminham na árvore ora de forma herdada, ora de forma sintetizada. Para simplificar o esquema de tradução, e também a depuração do tradutor, optou-se por construir uma árvore de sintaxe abstrata e a partir desta árvore realizar o parafraseamento. A árvore reflete a estrutura das construções da linguagem OCL. A Figura 2 ilustra a árvore de sintaxe abstrata para a seguinte expressão em OCL:

```
context Produto inv:
Let desconto = 0.1 * TipoProduto.precoBase in
descontoTotal <= desconto
```

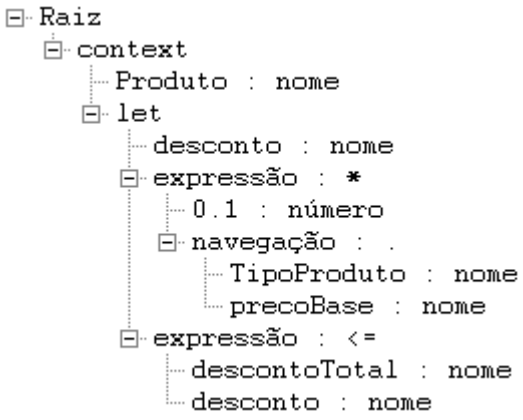

Figura 2 – Árvore de sintaxe abstrata para uma expressão OCL.

A árvore de sintaxe abstrata é então percorrida recursivamente, e o texto final é produzido. As expressões de caminho (navegação por ponto) são traduzidas consultando-se o glossário.

4. Exemplos

A seguir, na tabela 2, temos alguns exemplos de expressões em OCL e a respectiva tradução para linguagem natural.

Expressão em OCL	**Texto Gerado em Linguagem Natural**
context Produto inv: Let desconto = 0.1 * TipoProduto. precoBase in descontoTotal <= desconto	Regra para um Produto : O desconto total deve ser menor ou igual ao desconto, onde o desconto vale 0.1 vezes o preço base do tipo do produto.
context Departamento inv: divisao.QuadroFunc -> includes(projetoPiloto.chefe)	Regra para um Departamento : O chefe do projeto piloto deve ser um elemento do quadro de funcionários da divisão
context Projeto inv: contasCorrente->forAll(conta | conta.saldo > 0)	Regra para um Projeto : Para todos os elementos conta no conjunto de contas correntes, o saldo da conta deve ser maior do que 0
context Carnet inv: vencimento > hoje implies multa > 0	Regra para um Carnet : se a data de vencimento for maior do que a data de hoje então a multa deve ser maior do que 0

Tabela 2 – Exemplos de Tradução de OCL para Linguagem Natural.

5. Conclusões e Trabalhos Futuros

O uso de OCL, seja para representar regras de negócio como no sistema Atenas, seja para representar restrições no modelo UML, pode ser facilitado com a ferramenta FalaOCL, especialmente nas tarefas de validação junto ao usuário ou mesmo geração automática de documentação. Embora muitas vezes possa ocorrer a introdução de alguma ambigüidade no texto gerado, sempre é possível consultar o OCL original para eliminar eventuais dúvidas – tarefa que será sem dúvida facilitada após a leitura do texto gerado em linguagem natural.

Como trabalhos futuros, pretendemos portar a ferramenta FalaOCL para a Web de forma a permitir o uso da mesma por qualquer usuário independentemente de quaisquer instalação. Além disso, planejamos deixar que o usuário especifique o grau de formalismo para geração de texto, de forma a reduzir ou eliminar eventuais ambigüidades.

Bibliografia:

[1] Date,C. J. What not How: The Business Rules Approach to Application Development. Addison-Wesley longman Inc, 2000.

[2] Demuth, B. and Hussmann, H. Using OCL Constraints for Relational Database Design. UML'99 The Unified Modeling Language, 2nd Intl. Conf. Fort Collins, CO, USA, October 1999.

[3] Demuth, B. Hussmann, H. and Loecher, S. OCL as a Specification Language For business Rules in Database Applications. UML'01 The Unified Modeling Language, 4th Intl. Conf. Toronto, Ontario, Canada, October 2001.

[4] Graef, Albert. TPLY: Turbo Pascal Lex/Yacc. http://www.musikwissenschaft.uni-mainz.de/~ag/tply.

[5] GUIDE Business Rules project. Defining Business Rules - What Are They Really? Business Rule Group, final Report, July 2000.

[6] Ross, Ronald G. Business Rule Concepts. Business Rule Solutions Inc, 1998.

[7] Ross, Ronald G. The Business Rule Book: Classifying, Defining and Modeling Rules. 1997.

[8] von Halle, Barbara. Building a Business Rule System, Part 1. DM Review, Faulkner & Gray, January 2001.

[9] Warmer, J. B. and Kleppe, A. G. The object Constraint Language. Addison-Wesley, 1999.

[10] Zimbrão, G., e outros. "ATENAS: Um Sistema Gerenciador de Regras de Negócio", Publicado na Seção Técnica de Ferramentas do XV Simpósio Brasileiro de Engenharia de Software, Rio de Janeiro, Brasil, outubro de 2001.

[11] .Hähnle, R., Ranta, A. "Connecting OCL with the Rest of the World". ETAPS 2001 Workshop on Transformations in UML (WTUML), Genova, Italy, 2001.