Rapport projet SmallUML

Clément BAUCHET - Jasone LENORMAND

Encadré par Gerson SUNYE

Décembre 2016

Introduction

Dans ce rapport, nous présenterons notre travail pour le projet de Model Driven Engineering, qui consistait en la réalisation d'un métamodèle ainsi qu'une grammaire pour l'écrire, puis le transformer vers un autre métamodèle *(UML)*.

1 Modèle SmallUML

La première partie du projet est de concevoir un métamodèle *SmallUML*, qui est une version simplifiée d'UML où on ne spécifie que le côté métier du modèle. Cela inclut donc les éléments suivants : *Classes, Types, Énumérations, Attributs, Méthodes, Associations et Héritages.*

Nous avons d'abord défini une classe abstraite *NamedElement* avec un attribut *name*, dont héritent toutes les classes, ce qui leur permet d'avoir un nom en définissant cet attribut une seule fois. De plus, nous avons un *SuperType*, dont héritent les classess *Class*, *Type* et *Enumeration*, pour regrouper en un type les différents composants possibles d'un paquetage.

Nous avons utilisé EMF Ecore pour produire facilement un diagramme de classe et générer le code du modèle à partir de celui-ci.

2 Grammaire Xtext

Xtext est un outil permettant de définir un langage à l'aide de grammaires. C'est ce que nous utilisons pour la partie suivante, où il faut définir la syntaxe concrète du modèle *SmallUML*. Nous sommes partis d'une grammaire Xtext générée automatiquement depuis notre modèle Ecore, pour la modifier et surtout l'alléger, afin de la rendre plus intuitive à écrire.

Parmi les modifications que nous avons apportées au modèle généré pour l'améliorer, nous allons détailler la modification de Class:

```
Class returns Class:
    isAbstract?='isAbstract'
    'Class'
    name=EString
    '{'
            ('super' super=[Class|EString])?
            ('attribute' '{' attribute+=Attribute ( "," attribute+=Attribute)* '}' )?
            ('operation' '{' operation+=Operation ( "," operation+=Operation)* '}' )?
    '}';
```

Nous ne voulions pas que le mot-clé `super` figure à l'intérieur du corps de la classe. C'est pourquoi nous avons sorti le code correspondant à super pour le mettre après le nom de la classe et renommé `super` en `extends` pour se rapprocher de la syntaxe objet. Cette modification donne le code suivant pour le début du code de la classe :

```
(isAbstract?='abstract')?
    'class'
    name=ID
    ('extends' super=[Class|EString])?
```

Vous trouverez le Xtext obtenu à la génération dans l'archive du projet à côté de notre Xtext final (sous le nom de SmallUML_original.xtext).

3 Transformation ATL

Pour finir, nous spécifions la sémantique du langage : nous utilisons le langage **ATL** pour transformer nous concepts *SmallUML* en concepts *UML*.

Pour la transformation de SmallUML vers UML, nous avons choisi d'utiliser une version d'UML différente de celle fournie, cette dernière présentant trop d'incompatibilités avec SmallUML. L'URI de cette version est `http://www.eclipse.org/uml2/5.0.0/UML`.

Voici un exemple de règles ATL pour transformer certains concepts d'un métamodèle à l'autre :

```
rule Package {
    from
            f : MM!Package
    to
            t : MM1!Package (
```

```
                    name <- f.name,
                    packagedElement <- f.elements->union(f.association)
            )
    }

    rule Class {
        from
                f : MM!Class
        to
                t : MM1!Class (
                        name <- f.name,
                        isAbstract <- f.isAbstract,
                        superClass <- f.super,
                        ownedAttribute <- f.attribute,
                        ownedOperation <- f.operation
                )
    }
```

La transformation se lance sur un fichier au format .xmi, qui représente une instance de notre métamodèle *SmallUML*, autrement dit un modèle dont la définition est conforme à *SmallUML*. Le résultat est un autre fichier .xmi, celui-ci conforme au modèle *UML* spécifié.

Dans ce fichier résultat, on trouve tous les composants du paquetage à la bonne place, excepté les rôles des associations, sous forme de propriétés en-dehors du paquetage. Mais en regardant en détail les *Associations*, on remarque que leur propriété *memberEnd* fait bien référence à ces propriétés.

4 Annexes

4.1 Diagramme de classes

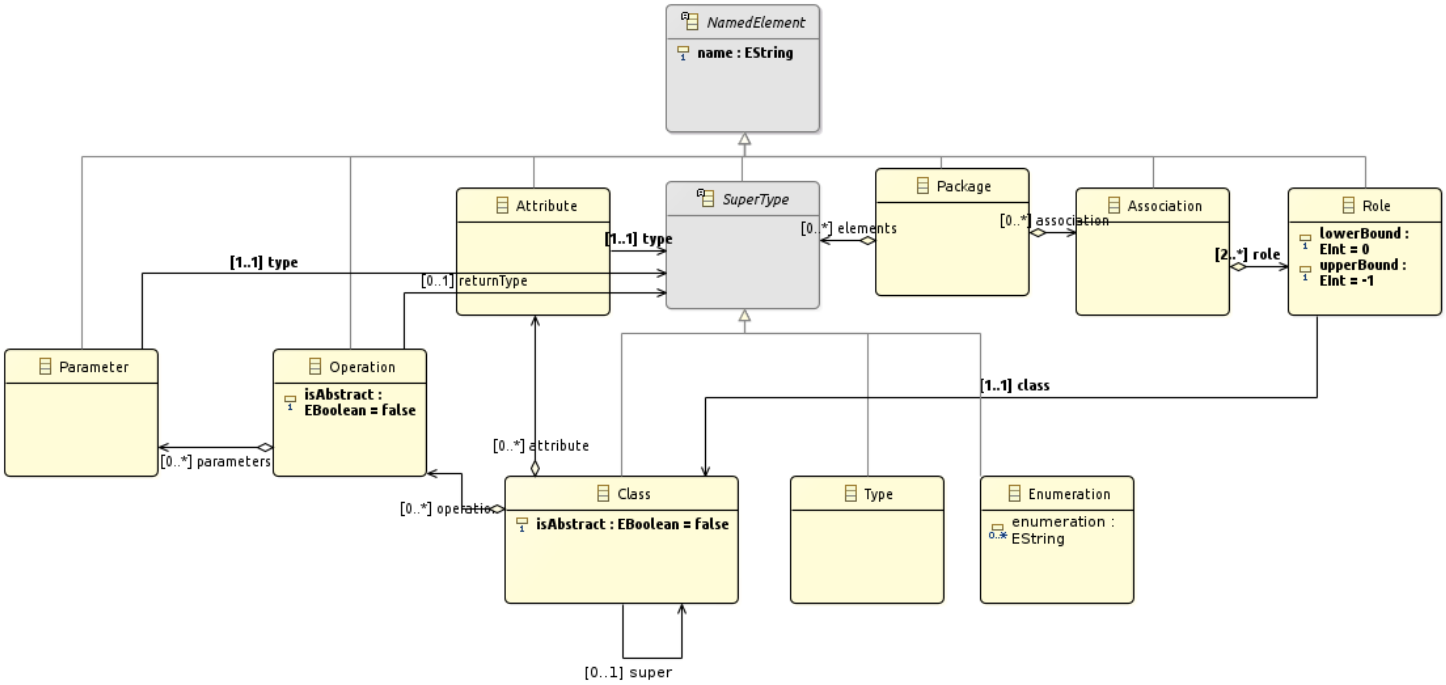

Figure 1: Diagramme de classes du métamodèle **SmallUML**

4.2 Exemple de modèle

Voici un exemple de modèle écrit dans notre langage *SmallUML* :

```
package kfc {
        elements {
                type Integer;
                type Float;
                type String;
                type Boolean;

                enumeration Taille {
                        PETIT, MOYEN, GRAND
                }

                abstract class MenuElement {
                        attributes {
                                attribute taille:Taille,
                                attribute prix:Float
                        }
                }

                class Bucket extends MenuElement {

                        operations {
                                operation remplir {quantite:Integer}
                        }
                }

                class Frites extends MenuElement {
                        attributes {
                                attribute sel:Boolean
                        }
                }

                class Boisson extends MenuElement {
                        attributes {
                                attribute glace:Boolean
                        }
                }

                class Commande {
                        attributes {
                                attribute prix:Float
                                ,
                                attribute a_emporter : Boolean
                        }
                        operations {
                                operation calculerPrix returns Float
                        }
                }
        }

        associations {
                association commande {
                        roles {
                                role articles[1,-1] with MenuElement,
                                role dans_menu[0,-1] with Commande
                        }
                }
        }


}
```