EISTI 2008-2009 – Analyse et conception

Object Constraint Language

Pour vérifier vos contraintes OCL, vous devez installer le plugin rational rose Oclarity :

- télécharger le sur : http://www.empowertec.de/downloads/oclarity_rose.htm
- avec la licence gratuite : http://www.empowertec.de/downloads/trial_key.htm

Les contraintes doivent être écrites dans des notes attachées au diagramme de classe. La vérification se fait par le menu Tools/Oclarity/Check all OCL expressions .

Person & Company

Téléchargez sur AREL le modèle **person.mdl** fourni en annexe et écrivez en OCL les contraintes suivantes :

1) Initialement une personne (*Person*) n'est pas mariée (*isMarried*).

Cf. 2)

2) Si on est marié (*isMarried*), c'est que l'on a une épouse (*wife*) ou un mari (*husband*).

```
context Person::isMarried : Boolean
init: false
derive: wife->notEmpty() or husband->notEmpty()
```

3) Toutes les personnes doivent être des femmes (*female*).

```
context p : Person inv:
p.gender = Gender::female
```

4) Toutes les personnes doivent être des hommes (*male*).
(concluez sur les vérifications effectuées par Oclarity)

```
context Person inv:
gender = Gender::male
```

5) Pour être marié, il faut avoir plus de 18 ans.

```
context Person inv:
isMarried implies age > 18
```

6) Le directeur (*manager*) d'une société (*Company*) doit avoir plus de 40 ans.

```
context Company inv:
manager.age > 40
```

7) Tous les employés d'une société doivent avoir plus de 18 ans.

```
context Job inv:
employee.age > 18

context Company inv:
employee->forAll(age > 18)
```

8) Si on a une épouse alors c'est une femme et si on a un mari alors c'est un homme.

```
context Person inv:
wife->notEmpty() implies wife.gender = Gender::female
and husband-> notEmpty() implies husband.gender= Gender::male
```

9) Une personne ne peut être employée par plus de 2 sociétés.

```
context Person inv:
employer->size() < 3
```

10) Le nombre total d'employé (*numberOfEmployees*) d'une société est égal au nombre d'employés simple plus le directeur.

```
context Company::numberOfEmployees : Integer
derive : employee.size() + 1
```

11) Le revenu d'une personne majeure est la somme des salaires des emplois qu'elle occupe.

```
context Person::income() : Integer
pre: age > 18
body: if isUnemployed
         then 0
         else job.salary->sum()
endif
```

12) Une société doit avoir des employés de plus de 50 ans.

```
context Company inv:
employee->exists(age > 50)
```

13) Une société ne peut pas employer d'homonymes.

```
context Company inv:
employee->forAll(p1, p2 | p1 <> p2 implies (
 p1.firstName <> p2.firstName and p1.lastName <> p2.lastName))
```

14) Si une personne possède deux parents (référencés), l'un est une femme et l'autre un homme.

```
context Person inv:
parent->size() = 2 implies (
parent->exists(gender = Gender::female) and
parent->exists(gender = Gender::male))
```

15) Tous les enfants d'une personne ont bien cette personne comme parent et inversement.

```
context Person
inv: child->notEmpty() implies
child->forAll(p: Person | p.parent->includes(self))
inv: parent->notEmpty() implies
parent->forAll(p: Person | p.child->includes(self))
```