Chap 5 Langage de contraintes objet (Object Constraint Langage : OCL)

1. Introduction

UML permet d'exprimer certaines contraintes:
Contraintes structurelles :

- les attributs dans les classes, les différents types de relations entre classes, la cardinalité et la navigabilité des propriétés structurelles, etc. ;

Contraintes de type :

- typage des propriétés, etc. ;

Contraintes diverses :

- les contraintes de visibilité, les méthodes et classes abstraites.

1.2. Écriture des contraintes

Une contrainte constitue une condition ou une restriction sémantique exprimée sous forme d'instruction dans un langage textuel qui peut être naturel ou formel. La chaîne constitue le corps écrit dans un langage de contrainte qui peut être :

- naturel ;
- dédié, comme OCL ;
- ou encore directement issu d'un langage de programmation.

1.3 Représentation des contraintes et contraintes prédéfinies

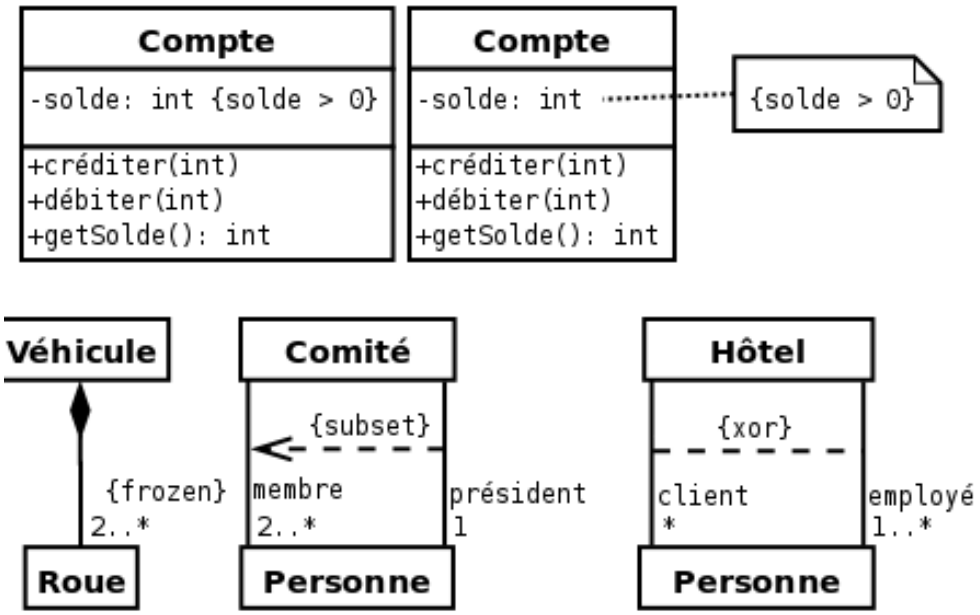


Figure 1 : UML permet d'associer une contrainte à un élément de modèle de plusieurs façons.
Sur les deux diagrammes du haut, la contrainte porte sur un attribut qui doit être positif. En bas à gauche, la contrainte {frozen} précise que le nombre de roues d'un véhicule ne peut pas varier. Au milieu, la contrainte {subset} précise que le président est également un membre du comité. Enfin, en

bas à droite, la contrainte {xor} (ou exclusif) précise que les employés de l'hôtel n'ont pas le droit de prendre une chambre dans ce même hôtel.

Figure 2 : Ce diagramme exprime que : une personne est née dans un pays, et que cette association ne peut être modifiée ; une personne a visité un certain nombre de pays, dans un ordre donné, et que le nombre de pays visités ne peut que croître ; une personne aimerait encore visiter toute une liste de pays, et que cette liste est ordonnée (probablement par ordre de préférence).

Nous venons de voir, quelques contraintes prédéfinies ({frozen}, {subset}, {xor}, {ordered} et {addOnly}). Le langage de contraintes objet OCL apporte une solution élégante à cette insuffisance.

2. OCL ?

C'est avec OCL (Object Constraint Language) qu'UML formalise l'expression des contraintes. Il s'agit donc d'un langage formel d'expression de contraintes bien adapté aux diagrammes d'UML.
OCL peut s'appliquer sur la plupart des diagrammes d'UML et permet de spécifier des contraintes sur l'état d'un objet ou d'un ensemble d'objets.

2.1. Pourquoi OCL ?

OCL est un langage formel volontairement simple d'accès. Il possède une grammaire élémentaire (OCL peut être interprété par des outils).

2.2. Illustration par l'exemple

Soit une application bancaire, Il nous faut donc gérer :

- des comptes bancaires ;
- des clients ;
- et des banques.

De plus, on aimerait intégrer les contraintes suivantes dans notre modèle :

- un compte doit avoir un solde toujours positif ;
- un client peut posséder plusieurs comptes ;
- une personne peut être cliente de plusieurs banques ;
- un client d'une banque possède au moins un compte dans cette banque ;
- un compte appartient forcément à un client ;
- une banque gère plusieurs comptes ;
- une banque possède plusieurs clients.

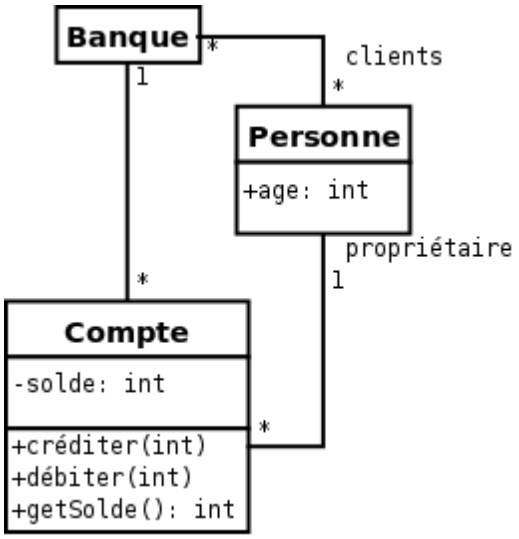


Figure 3 : Diagramme de classes modélisant une banque, ses clients et leurs comptes.
La montre un diagramme de classes correspondant à la problématique que nous venons de décrire.

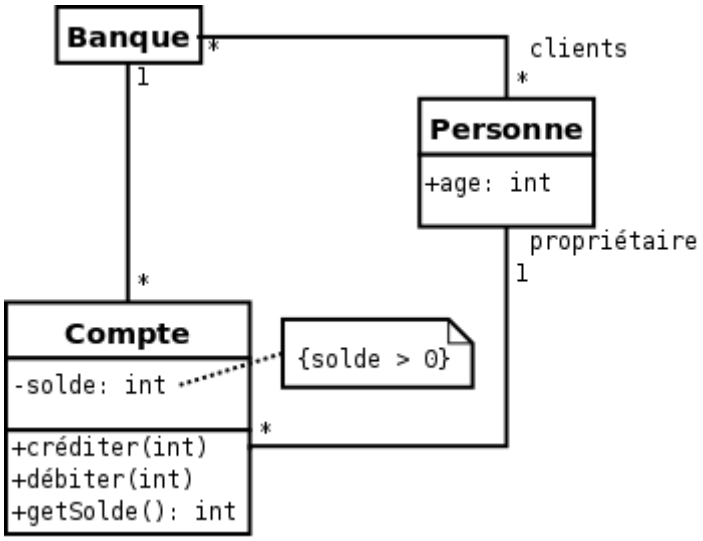


Figure 4 : Ajout d'une contrainte sur le diagramme de la figure 3.
Un premier problème apparaît immédiatement : rien ne spécifie, dans ce diagramme, que le solde du client doit toujours être positif. Pour résoudre le problème, on peut simplement ajouter une note précisant cette contrainte ({solde > 0}), comme le montre la figure 4.

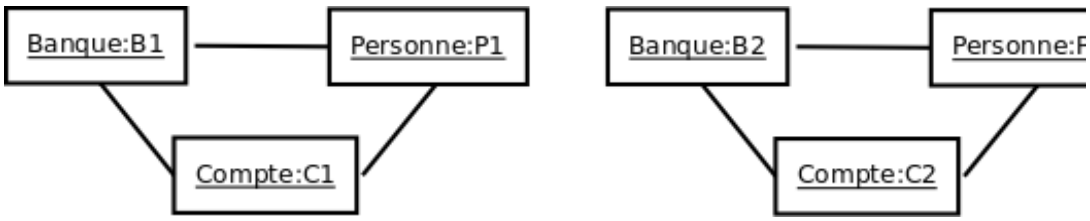


Figure 5 : Diagramme d'objets cohérent avec le diagramme de classes de la figure 4.

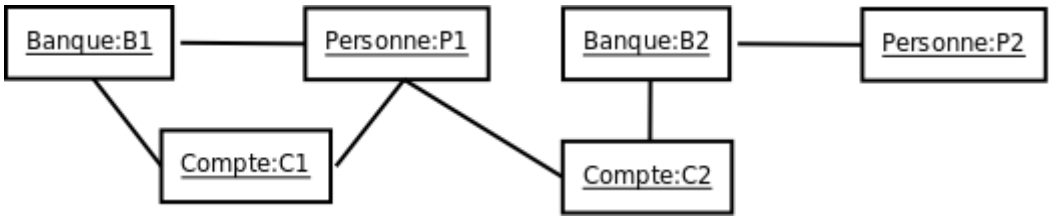


Figure 6 : Diagramme d'objets cohérent avec le diagramme de classes de la figure 4, mais représentant une situation inacceptable.

- **context** Compte
- **inv** : solde > 0
- **context** Compte :: débiter(somme : int)
- **pre** : somme > 0
- **post** : solde = solde@pre - somme
- **context** Compte
- **inv** : banque.clients -> includes (propriétaire)

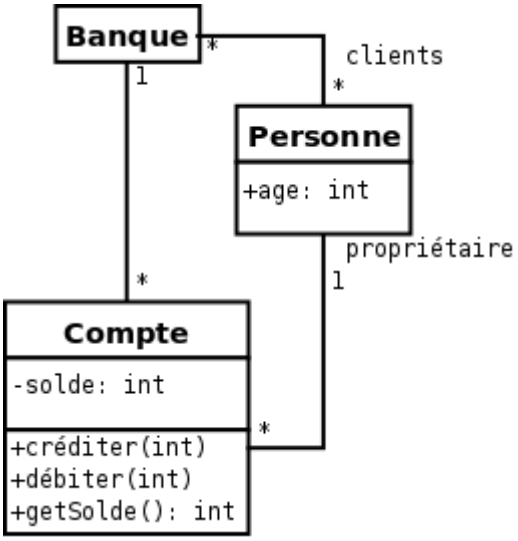


Figure 7 : Exemple d'utilisation du langage de contrainte OCL sur l'exemple bancaire.

Le langage OCL est particulièrement adapté à la spécification de ce type de contrainte. La figure 7 montre le diagramme de classes de notre application bancaire accompagné des contraintes OCL adaptées à la spécification du problème. La contrainte ne décrit absolument pas l'implémentation d'une méthode.

3. Typologie des contraintes OCL

3.1. Contexte (context)

Une contrainte est toujours associée à un élément de modèle. C'est cet élément qui constitue le contexte de la contrainte. Il existe deux manières pour spécifier le contexte d'une contrainte OCL :

- en écrivant la contrainte entre accolades ({}) dans une note. L'élément pointé par la note est alors le contexte de la contrainte ;
- en utilisant le mot-clef context dans un document accompagnant le diagramme

a. Syntaxe

Sélectionnez

context <élément>

<élément> peut être une classe, une opération, etc. Pour faire référence à un élément op (comme un opération), il faut utiliser les :: comme séparateur.

b. Exemple

Le contexte est la classe Compte :

- **context** Compte

Le contexte est l'opération getSolde() de la classe Compte :

- **context** Compte::getSolde()

3.2. Invariants (inv)

Un invariant exprime une contrainte prédicative sur un objet, ou un groupe d'objets, qui doit être respectée en permanence.

a. Syntaxe

Sélectionnez
inv : <expression_logique>
<expression_logique> est une expression logique qui doit toujours être vraie.

b. Exemple

Le solde d'un compte doit toujours être positif.

- **context** Compte
- **inv :** solde > 0

3.3. Pré conditions et pos tconditions (pre, post)

Une pré condition (respectivement une post condition) permet de spécifier une contrainte prédicative qui doit être vérifiée avant l'appel d'une opération.
Dans l'expression de la contrainte de la post condition, deux éléments particuliers sont utilisables :

- l'attribut result qui désigne la valeur retournée par l'opération ;
- et <nom_attribut>@pre qui désigne la valeur de l'attribut <nom_attribut> avant l'appel de l'opération.

a. Syntaxe

- Précondition :
  Sélectionnez
  pre : <expression_logique>
- Post condition :
  Sélectionnez
  post : <expression_logique>

<expression_logique> est une expression logique qui doit toujours être vraie.

b. Exemple

Concernant la méthode débiter de la classe Compte, la somme à débiter doit être positive pour que l'appel de l'opération soit valide et, après l'exécution de l'opération, l'attribut solde doit avoir pour valeur la différence de sa valeur avant l'appel et de la somme passée en paramètre.

- **context** Compte::débiter(somme : Real)
- **pre :** somme > 0
- **post :** solde = solde**@pre** - somme

3.4. Résultat d'une méthode (body)

Ce type de contrainte permet de définir directement le résultat d'une opération.

a. Syntaxe

Sélectionnez

body : <requête>

<requête> est une expression qui retourne un résultat dont le type doit être compatible avec le type du résultat de l'opération désignée par le contexte.

b. Exemple

- **context** Compte::getSolde() : Real
- **body :** solde

3.5. Définition d'attributs et de méthodes (def et let…in)

def est un type de contrainte qui permet de déclarer et de définir la valeur d'attributs comme la séquence let…in. def permet également de déclarer et de définir la valeur retournée par une opération interne à la contrainte.

a. Syntaxe de let…in

Sélectionnez

let <déclaration> = <requête> in <expression>

Un nouvel attribut déclaré dans <déclaration> aura la valeur retournée par l'expression <requête> dans toute l'expression <expression>.

b. Syntaxe de def

Sélectionnez

def : <déclaration> = <requête>

<déclaration> peut correspondre à la déclaration d'un attribut ou d'une méthode. <requête> est une expression qui retourne un résultat dont le type doit être compatible avec le type de l'attribut, ou de la méthode, déclaré dans <déclaration>. Dans le cas où il s'agit d'une méthode, <requête> peut utiliser les paramètres spécifiés dans la déclaration de la méthode.

c. Exemple

Pour imposer qu'une personne majeure doit avoir de l'argent

context Personne

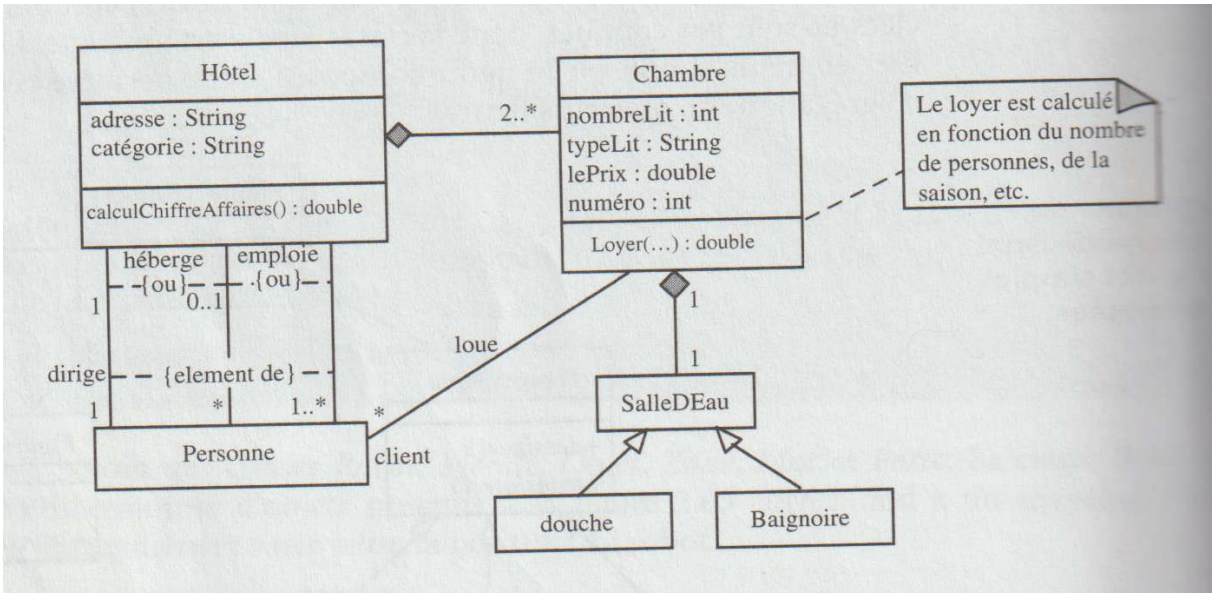
- **def :** argent : int = compte.solde->sum()
- **context** Personne
  inv : age>=18 implies argent>0

Exemple corrigé :

Un hôtel est composé d'au moins deux chambres. Chaque chambre dispose d'une salle d'eau qui peut être une douche ou une salle de bain. L'hôtel héberge des personnes. Il peut employer du personnel et est dirigé par un des employés. L'hôtel a les caractéristiques suivantes : une adresse, le nombre de pièces, la catégorie. Une chambre est caractérisée par le nombre et le type de lits, le prix et le numéro. On peut calculer le chiffre d'affaires, le loyer en fonction des occupants.

1. Donner le diagramme de classe.
2. Utiliser les contraintes pour afficher les relations.

Solution :



Référence :

1. G. Booch, J. Rumbaugh, I. Jacobson, « The Unified Modeling Language (UML) user guide", Addison-Wesley, 1999.
2. Benoit charroux, aomar Osmani, Yann Therry-Mieg,"UML2 synthèse et exercices",Pearson édition france,ISBN2-7440-7124-2,2005.
3. G. Booch et al., « Object-Oriented Analysis and Design, with applications », Addison- Wesley, 2007.
4. Cours UML 2.0 de Laurent Audibert , site http://www.developpez.com