OCL
Object Constraint Language

Zineb BOUGROUN

Plan

- Introduction : pourquoi OCL?

- Les principaux concepts d'OCL.

Introduction

Il existe plusieurs formes pour exprimer les contraintes dans UML:

- **Contraintes structurelles :**

les attributs dans les classes, les différents types de relations entre classes (généralisation, association, agrégation, composition, dépendance), la cardinalité et la navigabilité des propriétés structurelles, etc. ;

- **Contraintes de type :**

typage des propriétés... ;

- **Contraintes diverses :**

les contraintes de visibilité, les méthodes et classes abstraites (contrainte *abstract*), ...

Introduction

Une contrainte c’est :

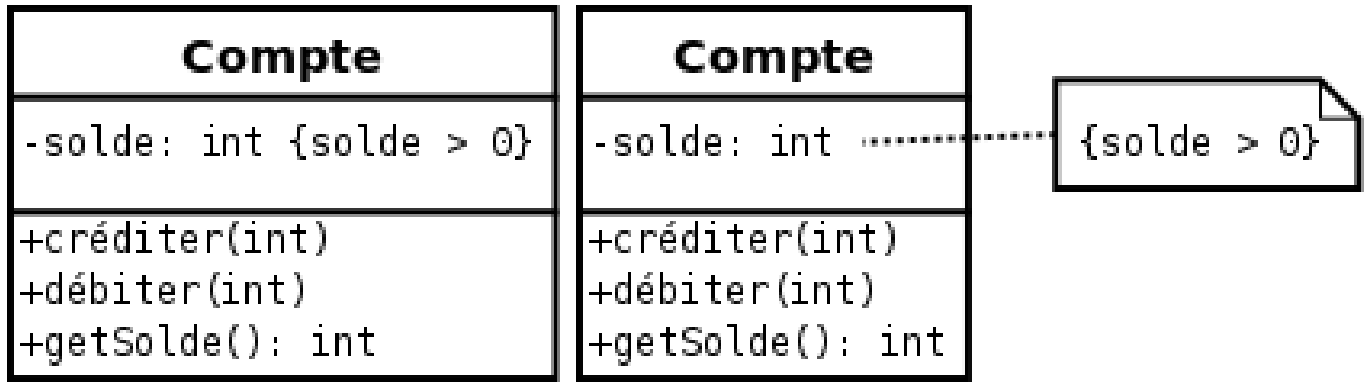
- Une condition ou une restriction sémantique
- Exprimée sous forme d'instructions dans un langage textuel :
  - formel ou naturel
- Peut être attachée à n'importe quel élément de modèle ou liste d'éléments de modèle.

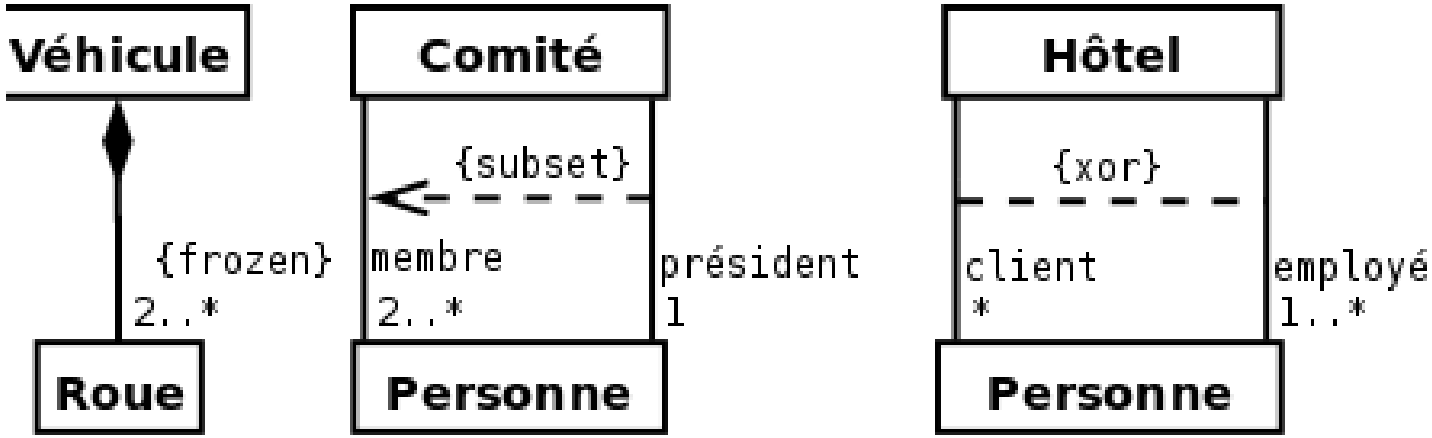
Introduction

- La Contrainte constitue le corps écrit dans un **langage** de contrainte qui peut être :
  - naturel ;
  - dédié, comme OCL ;
  - directement issu d'un langage de programmation.

Introduction

- Exemple de contrainte exprimé avec UML :





Introduction

- Question:
  - Est-ce que UML est suffisant?

- Réponse :
  - Exemple d’application bancaire

Introduction

Un client d'une banque possède au moins un compte dans cette banque

Un client peut être client de plusieurs banques

Une banque gère plusieurs comptes

Un client peut posséder plusieurs comptes

Un compte doit avoir un solde toujours positif

Une banque possède plusieurs clients

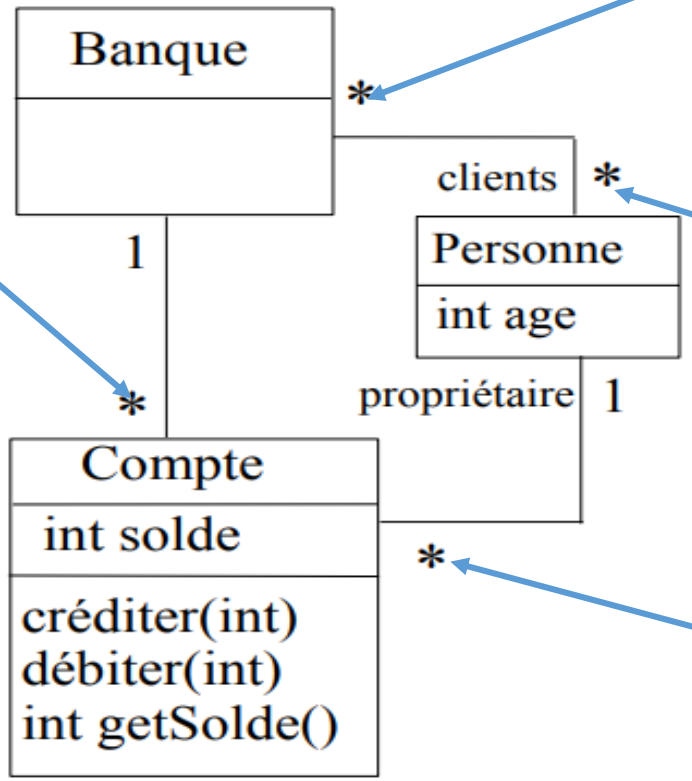
Seul un client de la banque peut avoir un compte

Introduction

Un client d'une banque possède au moins un compte dans cette banque

Une banque possède plusieurs clients

Une banque gère plusieurs comptes



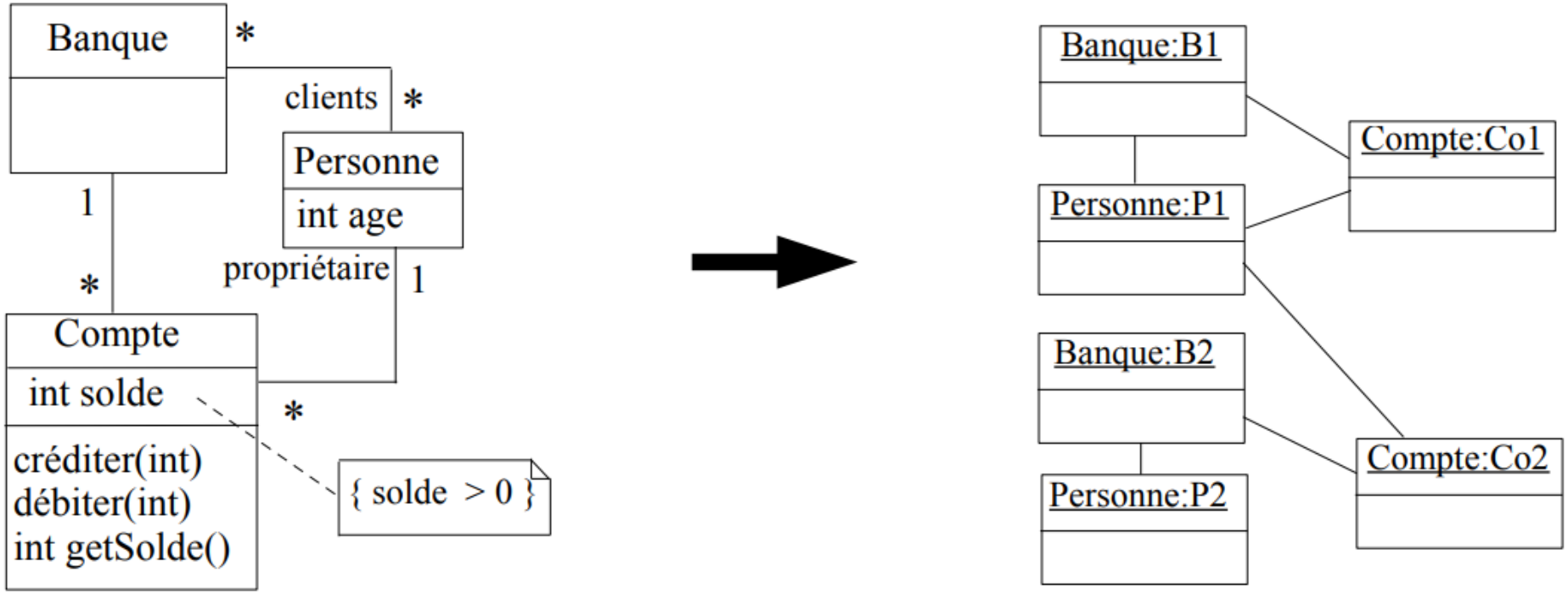
Un client peut être client de plusieurs banques

Un compte doit avoir un solde toujours positif

Seul un client de la banque peut avoir un compte

Un client peut posséder plusieurs comptes

Introduction



- Diagramme d'instances valide vis-à-vis du diagramme de classe mais ne respecte pas la spécification attendue:
  - Une personne a un compte dans une banque où elle n'est pas cliente
  - Une personne est cliente d'une banque mais sans y avoir de compte

Introduction

- Pour spécifier complètement une application
  - Diagrammes UML seuls sont généralement insuffisants
  - Nécessité de rajouter des contraintes

- Comment exprimer ces contraintes ?
  - Langue naturelle mais manque de précision, compréhension pouvant être ambiguë
  - Langage formel avec sémantique précise : par exemple OCL

- OCL : Object Constraint Language
  - Langage de contraintes orienté-objet
  - Langage formel (mais « simple » à utiliser) avec une syntaxe, une grammaire, une sémantique (manipulable par un outil)
  - S'applique entre autres sur les diagrammes UML

Introduction

- Il est issu de la norme de l'OMG
- Version courante : 2.4 (2014)
- Peut s'appliquer sur tout type de modèle, indépendant d'un langage de modélisation donné
- OCL permet principalement d'exprimer deux types de contraintes sur l'état d'un objet ou d'un ensemble d'objets
- Des invariants qui doivent être respectés en permanence
- Des **pré** et **post-conditions** pour une opération
  - Précondition : doit être vérifiée avant l'exécution
  - Postcondition : doit être vérifiée après l'exécution
- Attention : Une expression OCL décrit une contrainte à respecter et non pas le « code » d'une méthode

Les principaux concepts d'OCL

- Une expression OCL est toujours définie dans un **contexte**
  - Ce contexte est une classe
- Mot-clé : context
- Exemple
  - context Compte
- L'expression OCL s'applique à la classe Compte, c'est-à-dire à toutes les instances de cette classe

Les principaux concepts d'OCL

Invariants

- Un invariant exprime une contrainte sur un objet ou un groupe d'objets qui doit être respectée en permanence
- Mot-clé : inv
- Exemple
  - context Compte
  - inv: solde > 0
  - Pour toutes les instances de la classe Compte, l'attribut solde doit toujours être positif

Les principaux concepts d'OCL

- Pour spécifier une opération
  - Précondition : état qui doit être respecté avant l'appel de l'opération
  - Postcondition : état qui doit être respecté après l'appel de l'opération

- Mots-clés : pre et post

- Dans la postcondition, deux éléments particuliers sont utilisables
  - Pseudo-attribut « **result »** : référence la valeur retournée par l'opération
  - **« mon_attribut@pre »** : référence la valeur de mon_attribut avant l'appel de l'opération

- Syntaxe pour préciser la signature de l'opération
  - **context ma_classe::mon_op(liste_param) : type_retour**

Les principaux concepts d'OCL

Exemples

context Compte::débiter(somme : Integer)

pre: somme > 0

post: solde = solde@pre – somme

- La somme à débiter doit être positive pour que l'appel de l'opération soit valide
- Après l'exécution de l'opération, l'attribut solde doit avoir pour valeur sa valeur avant l'appel à laquelle a été soustrait la somme passée en paramètre

context Compte::getSolde() : Integer

post: result = solde

- Le résultat retourné doit être le solde courant

- **Attention** : On ne décrit pas comment l'opération est réalisée mais des contraintes sur l'état avant et après son exécution

Les principaux concepts d'OCL

Types de base et exemples d'opérations associées

- Integer
  - 1, -2, 145
  - *, +, -, /, abs()
- Real
  - 1.5, -123.4
  - *, +, -, /, floor()
- String
  - 'bonjour'
  - concat(), size(), substring()
- Boolean
  - true, false
  - and, or, not, xor, implies, if-then-else
- La plupart des expressions OCL sont de types Boolean
- Notamment les expressions formant les inv, pre et post

Les principaux concepts d'OCL

Opérations prédéfinies sur tous les objets

- oclIsTypeOf (t : OclType) : Boolean
  - retourne vrai si le type de l'objet au titre duquel cette opération est invoquée est de même que le type t passé en paramètre.
- oclIsKindOf (t : OclType) : Boolean
  - permet de déterminer si le type t passé en paramètre correspond exactement au type ou à un type parent du type de l'objet au titre duquel cette opération est invoquée.
- oclInState (s : OclState) : Boolean
  - utilisée dans un diagramme d'états-transitions. Elle est vraie si l'objet décrit par le diagramme d'états-transitions est dans l'état s passé en paramètre.
- oclIsNew () : Boolean
  - utilisée dans une postcondition. Elle est vraie quand l'objet au titre duquel elle est invoquée est créé pendant l'opération

Les principaux concepts d'OCL

Les opérations

OCL définit des opérations qui peuvent être appliquées a tous les objets.

- Select
  - retourne le sous-ensemble de col dont les éléments respectent la contrainte spécifiée
- Reject
  - idem mais ne garde que les éléments ne respectant pas la contrainte
- Collect
  - retourne une collection (de taille identique) construite à partir des éléments de col. Ses éléments peuvent être différents
- Exists
  - retourne vrai si au moins un élément de col respecte la contrainte, faux sinon
- forAll
  - retourne vrai si tous les éléments de col respectent la contrainte spécifiée
- AllInstances
  - Retourne toutes les instances de la classe référencée.

Les principaux concepts d'OCL

Les contrats

- Pre et postconditions permettent de faire une conception par contrat
  - Contrat passé entre l'appelant d'une opération et l'appelé (celui qui exécute l'opération)
  - Si l'appelant respecte les contraintes de la précondition alors l'appelé s'engage à respecter la post-condition
  - Si l'appelant ne respecte pas la précondition, alors le résultat de l'appel est indéfini

- Pour exemple précédent
  - Si l'appelant de l'opération débiter passe une somme positive en paramètre, alors le compte est bien débité de cette somme

Les principaux concepts d'OCL

navigation

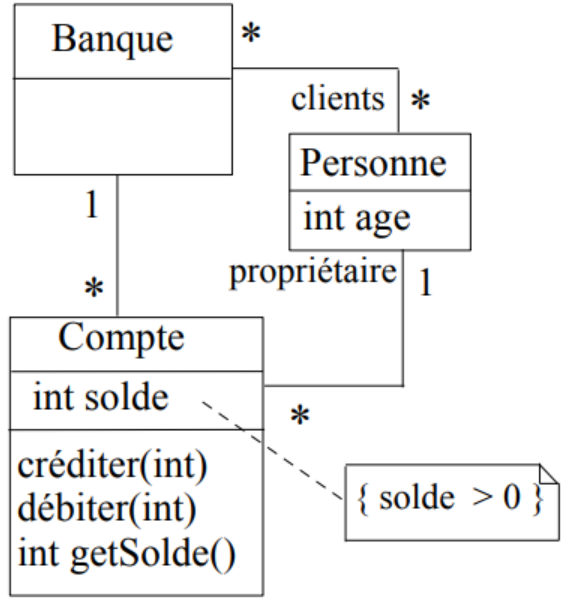
- Dans une contrainte OCL associée à un objet, on peut
  - Accéder à l'état interne de cet objet (ses attributs)
  - Naviguer dans le diagramme : accéder de manière transitive à tous les objets (et leur état) avec qui il est en relation
- Nommage des éléments pour y accéder
  - Attributs ou paramètres d'une opération : utilise leur nom directement
  - Objet(s) en association : on utilise au choix
    - Le nom de la classe associée (avec la première lettre en minuscule)
    - Le nom de l'association si elle nommée
    - Le nom du rôle d'association du coté de la classe vers laquelle on navigue s'il est nommé
- La navigation retourne
  - Si cardinalité de 1 pour une association : un objet
  - Si cardinalité > 1 : une collection d'objets
- Pseudo-attribut particulier
  - self : référence l'objet de départ, d'où part la navigation

Les principaux concepts d'OCL

Exemples

context **Compte**

solde : attribut référencé directement
banque : objet de la classe Banque (référence via le nom de la classe) associé au compte
propriétaire : objet de la classe Personne (référence via le nom de rôle d'association) associée au compte
banque.clients : ensemble des clients de la banque associée au compte (référence par transitivité)
banque.clients.age : ensemble des âges de tous les clients de la banque associée au compte



self.solde ;

self.getSolde() ;

Inv : Self.propriétaire.age > 18

self.débiter(1000).

Les principaux concepts d'OCL

body

- Résultat d’une méthode (body)

- Permet de définir directement le résultat d'une opération
- **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.

Exemple

context Employee::getWage() : Integer

body: self.wage

Les principaux concepts d'OCL

Conditionnelles

- Certaines contraintes sont dépendantes d'autres contraintes
- Deux formes pour gérer cela
  - **if expr1 then expr2 else expr3 endif**
    - Si l'expression expr1 est vraie alors expr2 doit être vraie sinon expr3 doit être vraie
  - **expr1 implies expr2**
    - Si l'expression expr1 est vraie, alors expr2 doit être vraie également.
    - Si expr1 est fausse, alors l'expression complète est vraie
- Il n'existe pas de if … then sans la branche else
  - Il faut utiliser le implies pour cela

Les principaux concepts d'OCL

Exemples

- **context** Personne

  inv: if age < 18 **then** compte -> isEmpty()

  else compte -> notEmpty() **endif**

  Une personne de moins de 18 ans n'a pas de compte bancaire alors qu'une personne de plus de 18 ans possède au moins un compte

Les principaux concepts d'OCL

collections

- **4 types** de collections (collection d'objets)
  - **Set** : ensemble au sens mathématique, pas de doublons, pas d'ordre
  - **OrderedSet** : idem mais avec ordre (les éléments ont une position dans l'ensemble)
  - **Bag** : comme un Set mais avec possibilité de doublons
  - **Sequence** : un Bag dont les éléments sont ordonnés
- Exemples :
  - { 1, 4, 3, 5 } : Set(Integer)
  - { 1, 4, 1, 3, 5, 4 } : Bag(Integer)
- Notes
  - Un collect renvoie un Bag, un sortebBy un OrderedSet
  - Possibilité de transformer un type de collection en un autre type de collection avec opérations OCL dédiées

Les principaux concepts d'OCL

propriété

- De manière générale en OCL, une propriété est un élément pouvant être
- Un attribut
  - Un bout d'association
  - Une opération ou méthode de type requête
- Accède à la propriété d'un objet avec « . »
  - Exemples
  - **context** Compte **inv**: self.solde > 0
  - **context** Compte **inv**: self.getSolde() > 0
- Accède à la propriété d'une collection avec « -> »
- On peut utiliser « -> » également dans le cas d'un objet (= collection d'1 objet)

Les principaux concepts d'OCL

Primitives

- OCL propose un ensemble de primitives utilisables sur les collections
- **size()** : retourne le nombre d'éléments de la collection
- **isEmpty()** : retourne vrai si la collection est vide
- **notEmpty()** : retourne vrai si la collection n'est pas vide
- **count(obj)** : le nombre d'occurrences de l'objet obj dans la collection
- **includes(obj)** : vrai si la collection inclut l'objet obj
- **excludes(obj)** : vrai si la collection n'inclut pas l'objet obj
- **includesAll(col)** : la collection contient tous les éléments de la collection col
- **excludesAll(col)** : la collection ne contient aucun des éléments de la collection col

- Syntaxe d'utilisation : **objetOuCollection -> primitive**

Les principaux concepts d'OCL

Exemples

- invariants dans le contexte de la classe Compte
- propriétaire -> size() = 1 : le nombre d'objets Personne associés à un compte est de 1
  - Vrai par principe à cause de la cardinalité de 1 qui doit être respectée
  - On manipule ici un objet (cardinalité de 1) comme une collection contenant l'objet
- banque.clients -> size() >= 1 : une banque a au moins un client
- banque.clients -> includes(self.propriétaire) : l'ensemble des clients de la banque associée au compte contient le propriétaire du compte
- banque.clients.compte -> includes(self) : le compte appartient à un des clients de sa banque

Les principaux concepts d'OCL

- OCL ne remplace pas les explications en langage naturel.
  - Les deux sont complémentaires !
  - Comprendre (informel)
  - Lever les ambigüités (OCL)
- Eviter les expressions OCL trop compliquées
  - éviter les navigations complexes
  - bien choisir le contexte (associer l'invariant au bon type !)
  - décomposer une conjonction de contraintes en plusieurs (inv, post, pre)
  - Toujours nommer les extrémités des associations (indiquer le rôle des objets)