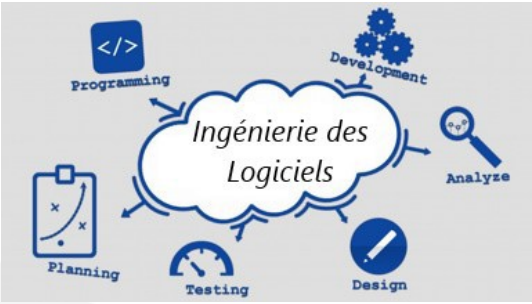
Ingénierie des Logiciels



Dr. Said MEGHZILI

Centre Universitaire de Mila

Département de Mathématique et Informatique

Email: s.meghzili@univ-mila.dz

1.0

Février 2022

Table des matières

I Chapitre 05 : Introduction à OCL

1. Introduction

Définition : Object Constraint Language (OCL)

- OCL est un langage formel pour écrire des expressions de manière précise.
- Il est basé sur la logique des prédicats du premier ordre.
- Les contraintes OCL ont une sémantique formelle, par conséquent, peuvent être utilisées pour réduire l'ambiguïté dans les modèles UML.
  Exemple: L'age d'une personne doit être supérieur ou égal à 18 ans.
- Prend en charge les concepts d'objets.
- Mais - OCL n'est pas un langage de programmation:
  - Aucun flux de contrôle.
  - Pas d'effets secondaires.
- Pourquoi OCL? Parce qu'UML ne suffit pas!

Exemple : UML ne suffit pas!

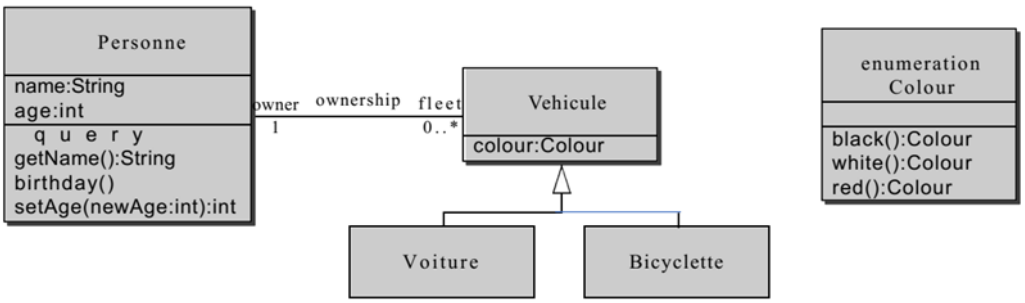


Figure 5.1 : Diagramme de classe d'application de gestion de véhicules

Contraintes !

- Nombre possible de propriétaires qu'une voiture peut avoir
- Âge requis des propriétaires de voitures
- Exigence qu'une personne puisse posséder au plus une voiture noire

Manque de précision: le diagramme de classe ne permet pas d'exprimer ces contraintes.

Attention

- Alors !
  - Nous avons besoin d'un langage pour aider avec la spécification.
  - Nous recherchons un «add-on» au lieu d'un tout nouveau langage avec des capacités de spécifications complètes.
  - Pourquoi pas la logique du premier ordre? - Pas OO.
  - OCL est utilisé pour spécifier les contraintes sur les systèmes OO.
  - OCL n'est pas le seul.
  - Mais OCL est le seul à être standardisé.

Fondamental : Objectifs du langage OCL

- Spécifier des invariants pour les classes et les types.
- Spécifier les conditions préalables et postérieures aux méthodes.
- Comme langage de navigation.
- Pour spécifier des contraintes sur les opérations.
- Tester les Exigences et les spécifications.

2. Typologie des contraintes

Définition : Notion de contexte

Mot-clé ***context*** : une contrainte OCL est toujours définie dans un contexte. Ce contexte est l'instance d'une classe.

Exemple :

- ***context Personne***:
  La contrainte OCL s'applique à la classe Personne, c'est-à-dire à toutes les instances de cette classe.
- ***context Personne::setAge(newAge:int):int***:
  La contrainte OCL s'applique à la méthode setAge(newAge:int).

Définition : Invariants

Un invariant exprime une contrainte sur un objet ou un groupe d'objets qui doit être respectée en permanence. Un invariant est une expression OCL booléenne - prend la valeur true / false. Le ***Mot-clé*** d'un invariant est : ***inv***.

Exemple

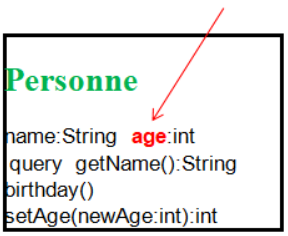


Figure 5.2 : La classe Personne

context Personne

inv: age >= 18

Contrainte :"Pour toutes les instances de la classe Personne (dans la figure 5.2), l'age doit toujours être supérieur ou égal à 18 ans".

Exemple

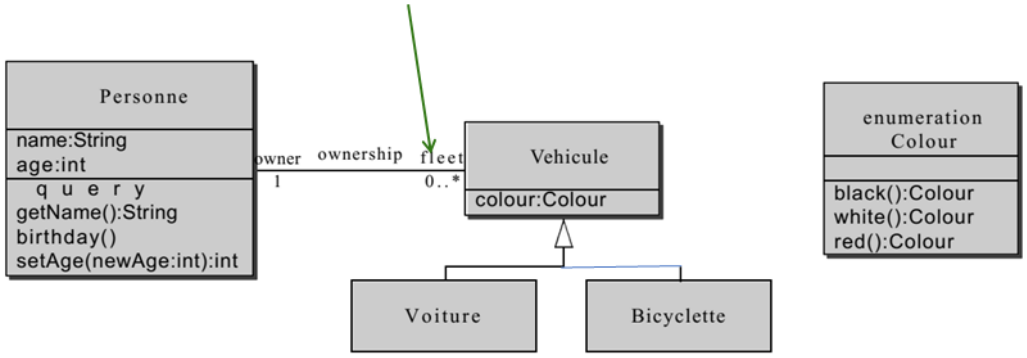


Figure 5.3 : Le rôle fleet : ensemble des véhicules

Les objets de contexte peuvent être désignés dans l'expression à l'aide du mot-clé «***self***».

Context Personne

inv: self.fleet->size <= 5

Contrainte :" Personne ne possède plus de 5 véhicules". La figure 5.3 montre le rôle ***fleet***.

Définition : Pré et Post conditions

OCL peut également spécifier des opérations à l'aide des Pré et Postconditions :

- 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 :

- Attribut ***result*** : référence la valeur retournée par l'opération.
- mon_attribut***@pre*** : référence la valeur de mon_attribut avant l'exécution de l'opération.

Syntaxe pour préciser l'opération : ***context*** ma_classe::mon_op(liste_param) : type_retour.

Exemple : Pré et Postconditions sur l' opération setAge(...) dans la figure 5.4.

Context Personne::setAge(newAge:int)

pre: newAge >= 0

Post: self.age = newAge

Contrainte : "Si setAge (...) est appelée avec un argument non négatif, l'argument devient la nouvelle valeur de l'attribut age".

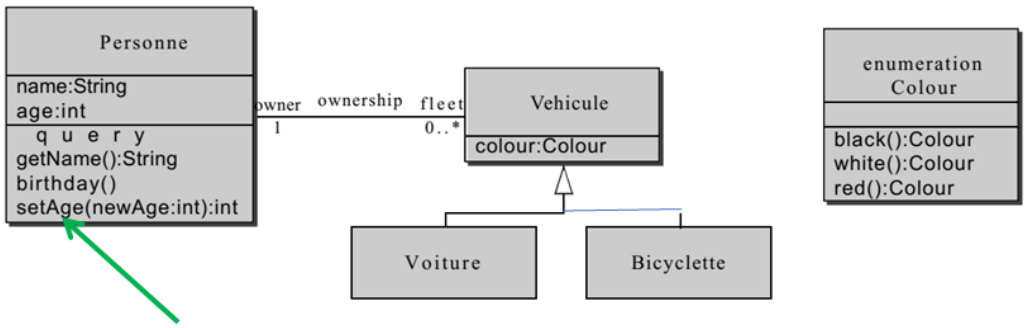


Figure 5.4 : L'opération setAge(...) de la classe personne

Remarque : Pré et Postconditions

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

Définition : Conception par Contrat

Pre et Post conditions permettent de définir une ***conception par contrat*** entre l'appelant d'une opération et l'appelé:

- Si l'appelant respecte les contraintes de la pré-condition alors l'appelé s'engage à respecter la post-condition.
- Si l'appelant ne respecte pas la pré-condition, alors le résultat de l'appel est ***indéfini***.

Exemple

Context Person::setAge(newAge:int)

pre: newAge >= 0

Post: self.age = newAge

- Si l'appelant à ***setAge*** respecte la pré-condition ***newAge >=0***, alors setAge affectera à l'attribut ***age*** la valeur du parametre ***newAge***. Sinon, la valeur de l'attribut age sera ***indéfinie***.

3. Types de base et opérations

Attention

Pour être bien formée, une expression OCL doit être conforme aux règles de conformité de type. Par exemple, vous ne pouvez pas comparer un entier avec une chaîne de caractères. En plus, chaque classificateur dans un modèle UML devient un type OCL.

Définition : Types de base OCL

Les types et opérateurs prédéfinis dans les contraintes OCL [15]*:

- ***Boolean*** -> true, false
  Opérations: and, or, xor, not, implies, if-then-else
- ***Integer*** -> 1, -5, 2, 489, 26524, ...
  Opérations : *, +, -, /, abs()
- ***Real*** -> 1.5, 3.14, ...

Opérations : *, +, -, /, floor()

- ***String*** -> 'Université Mila...'
  Opérations : toUpper(), concat()

Complément : types de collection OCL

Les ensembles, Multi-Ensembles (Bags) et séquences sont prédéfinis dans OCL et sont des sous-types: ***collection***. OCL propose un grand nombre d'opérations prédéfinies sur les collections. Ils sont tous de la forme: ***collection-> opération (arguments)***. La collection ***OCL-Type*** est la superclasse générique d'une collection d'objets de type T. Il existe quatre types de collections [17]*:

- ***Set***: Défini au sens mathématique. Chaque élément ne peut apparaître qu'une seule fois.
- ***OrderedSet*** : idem mais avec ordre (les éléments ont une position dans l'ensemble).
- ***Bag***: une collection, dans laquelle les éléments peuvent apparaître plusieurs fois (multi-Set).
- ***Séquence***: un multi-ensemble, dans lequel les éléments sont ordonnés.
- Exemples :
  – { 1, 4, 3, 5 } : Set(integer)
  – { 1, 2, 3, 5 } : OrderedSet(integer)
  – { 1, 4, 1, 3, 5, 4 } : Bag(integer)
  – { 1, 1, 3, 4, 4, 5 } : Sequence(integer)

4. Opérations pour les collections OCL

Définition

Les collections sont tous de la forme: collection-> opération (arguments). Il existe plusieurs opérations sur les collections :

- size: Integer
  Retourne le nombre d'éléments dans la collection.
- includes(o:OclAny): Boolean
  Retourne True, si l'élément o est dans la collection.
- count(o:OclAny): Integer
  Compte le nombre de fois qu'un élément est contenu dans la collection.
- isEmpty: Boolean
  Retourne True, si la collection est vide
- notEmpty: Boolean
  Retourne True, si la collection n'est pas vide.
- union(c1:Collection) Retourne l'union avec la collection c1.
- intersection(c2:Collection)
  Retourne l'intersection avec Collection c2 (contient uniquement des éléments, qui apparaissent dans la collection ainsi que dans la collection c2).
- including(o:OclAny)
  Collection contenant tous les éléments de la collection et l'élément o.
- select(expr:OclExpression) Sous-ensemble de tous les éléments de la collection, pour lesquels l'expression OCL expr est vraie.

Méthode : Comment obtenir les collections OCL?

- Une collection peut être générée en énumérant explicitement les éléments.
- Une collection peut être générée en naviguant le long d'une ou plusieurs associations 1:N
    - La navigation le long d'une seule association 1: n donne un ensemble (set)
    - La navigation le long de quelques associations 1: n donne un Bag (multi-ensemble)
    - La navigation le long d'une seule association 1: n avec la contrainte {ordonnée} donne une séquence

5. Accès aux objets et navigation

Fondamental

Dans une contrainte OCL associée à un objet, on peut:

- ***Accéder*** à l'état interne de cet objet (ses attributs)
- ***Accéder*** aux opérations de cet objet : self.operation.
- ***Naviguer*** dans le diagramme : accéder de manière ***transitive*** à tous les objets avec qui il est en relation.

Exemple : Navigation Locale et Directe

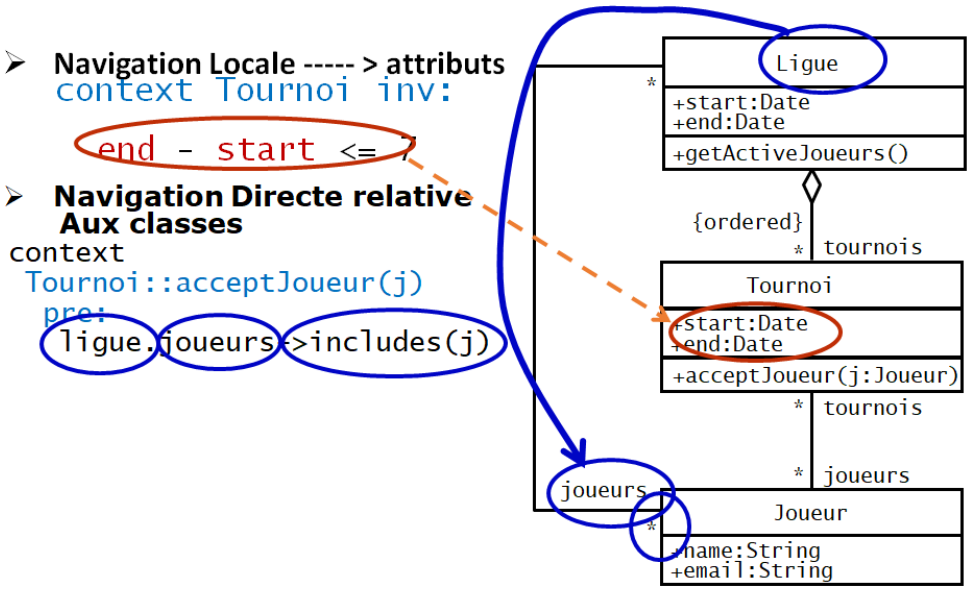


Figure 5.5 : Exemple d'accès aux objets et navigation

- Navigation Locale ----- > attributs

  context Tournoi

  inv: end - start <= 7

  Contrainte :"la durée d'un tournoi doit être inférieur à 7"

- Navigation Directe relative Aux classes

  context Tournoi::acceptJoueur(j)

  pre: ligue.joueurs->includes(j)

  Contrainte : "Si acceptJoueur(j) est appelée avec un argument j, ce joueur (j) doit appartenir à la liste des joueurs de la ligue."

Exemple : Navigation à travers une association : 1..n

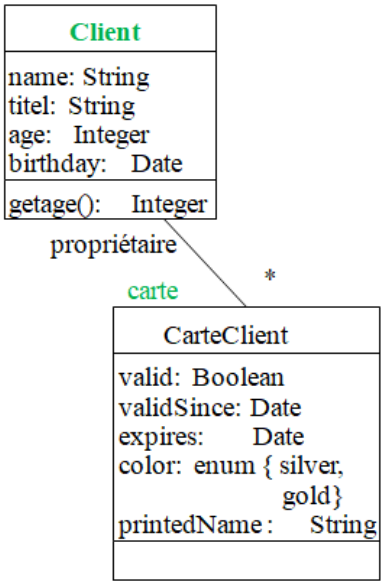


Figure 5.6 : Exemple de navigation : association en cardinalité 1..n

Contrainte :"un client ne doit pas avoir plus de 3 cartes".

context Client

inv: carte->size <= 3

carte désigne un ensemble de cartes de clients.

Exemple : Navigation à travers plusieurs associations : 1..n

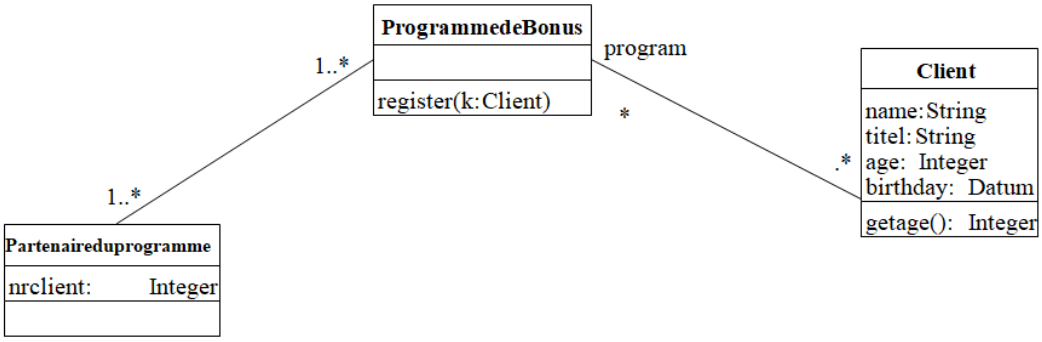


Figure 5.7 : Exemple de navigation : plusieurs associations en cardinalité 1..n

Context Partenaireduprogramme

inv: nrclient = programmedebonus.client->size

Client denote un multi-ensemble de client.

programmedebonus denote un ensemble de programmedebonus.

Exemple : Navigation à travers une association avec contrainte

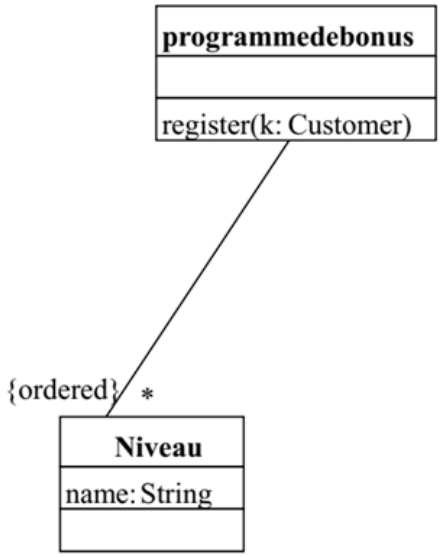


Figure 5.8 : Exemple de navigation : association avec contrainte

La navigation à travers une association avec la contrainte {ordered} produit une séquence.

context programmedebonus

inv: niveau->size = 2

niveau dénote une séquence de niveaux.

6. Conclusion

OCL est un Langage de contraintes orienté-objet. C'est un Langage formel (mais «simple» à utiliser) avec une syntaxe, une grammaire, une sémantique et s'applique entre autres sur les diagrammes UML [14]*. Dans ce chapitre, nous avons abordé les concepts de base du langage OCL : la typologie des contraintes, les types de base, les opérations sur les collections et l'accès aux objet et navigation. Nous avons montré l'efficacité du langage OCL à travers plusieurs exemples.

Références

14 The Object Constraint Language: Precise Modeling with UML, by Jos Warmer and Anneke Kleppe.

15 Marianne Huchard, "Object Constraint Language (OCL) Une introduction", 2007.

17 Cours “OCL”, Laetitia Matignon, Université Claude Bernard Lyon 1, 2012- 2013.