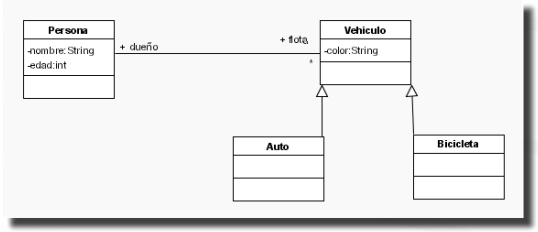
OCL: Object Constraint Language

- Un Diagrama de clases define las características básicas del modelo de objetos
- El diagrama de clases suele ser insuficientemente expresivo:
  - Muchas veces, se cuelan instanciaciones del diagrama que no condicen con un estado deseable del sistema
- OCL brinda
  - mayor expresividad para definir qué estados son válidos
  - mayor expresividad para definir el efecto de una operación en términos del estado del sistema
    - Expresar mas formalmente el modelo de operaciones
    - Expresar mas formalmente operaciones relacionadas con la etapa del diseño
- El vocabulario de OCL depende del modelo de clases (i.e., hay una signatura que se desprende del modelo de clases)

Object Constraint Language

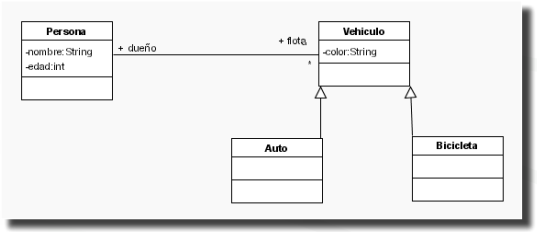


¿Cómo modelar los siguientes requerimientos?

- Posible número de dueños que un vehículo puede tener.
- Edad requerida para los dueños de autos.
- Toda persona deba tener al menos un auto negro.

Object Constraint Language

"El dueño de un vehículo debe tener al menos 18 años"

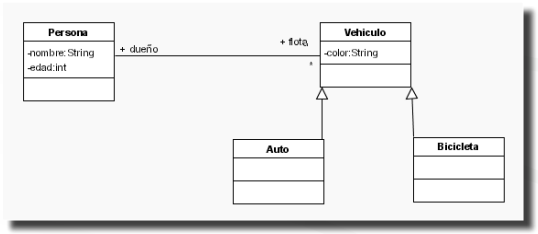


context Vehículo

inv: self.dueño.edad >= 18

Object Constraint Language

¿Qué significa esto?

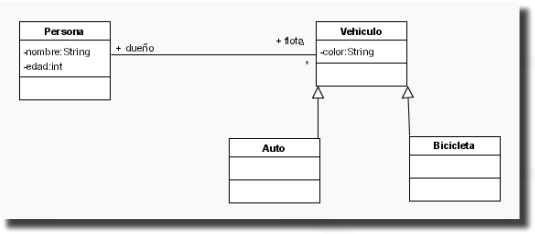


context Persona

inv: self.edad >= 18

Object Constraint Language

"El dueño de un auto debe tener al menos 18 años"

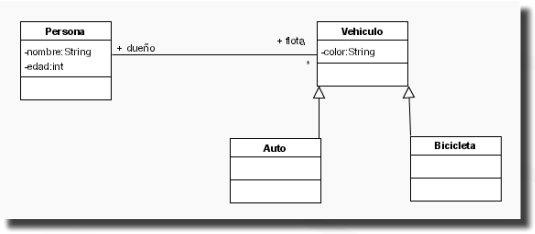


context Auto

inv: self.dueño.edad >= 18

Object Constraint Language

"Nadie tiene más de 3 autos"

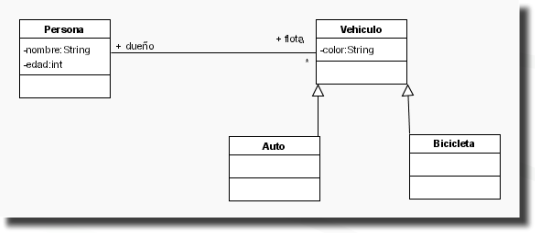


context Persona

inv: self.flota->size <= 3

Object Constraint Language

"Todos los autos de una persona son negros"



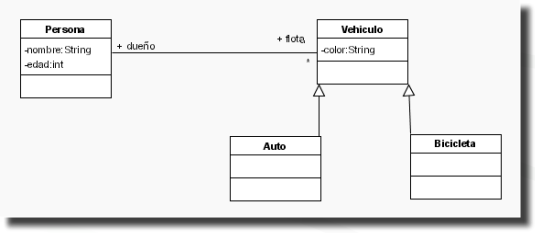
context Persona

inv: self.flota->forAll(v | v.color="negro")

Conjunción

Object Constraint Language

"Nadie tiene más de 3 vehículos negros"



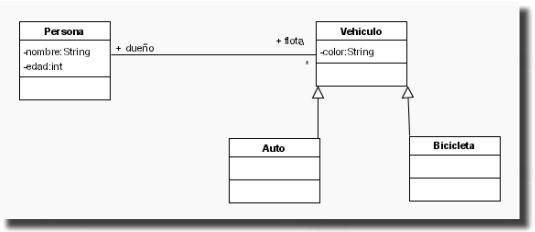
Retorna un conjunto de vehículos

context Persona

inv: self.flota->select(v | v.color="negro")->size <= 3

Object Constraint Language

"Ninguna persona menor a 18 años, posee un auto"



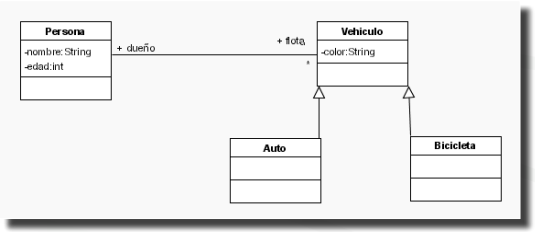
Retorna un conjunto de vehículos que no son autos

context Persona

inv: self.edad<18 implies self.flota–>forAll(v | not v.oclIsKindOf(Auto))

Object Constraint Language

"Existe un auto rojo"

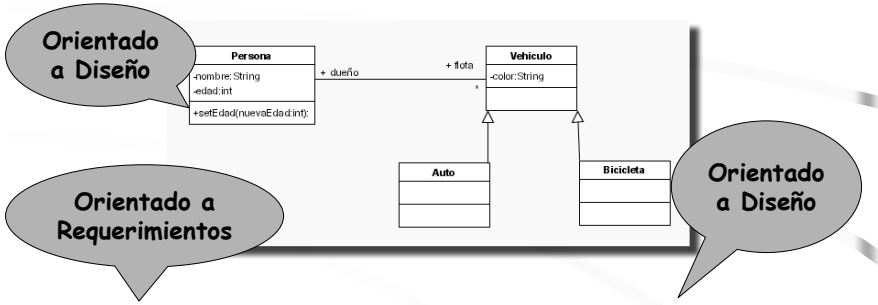


context Auto

inv: Auto.allInstances()->exists(c | c.color="rojo")

Object Constraint Language

Puede utilizarse para definir precondiciones y postcondiciones de operaciones (o casos de uso!)



setEdad(p:Persona, nuevaEdad:int)

pre: p.nuevaEdad >= 0

post: p.edad = nuevaEdad

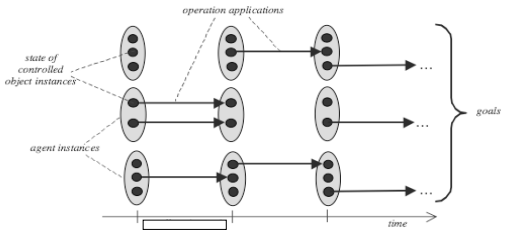
context Persona::setEdad(nuevaEdad:int)

pre: nuevaEdad >= 0

post: self.edad = nuevaEdad

Semántica

- La semántica de una descripción OCL está dada en téminos de secuencias alternantes modelos de objetos y operaciones
  - Cada "foto" es una instanciación válida del diagrama de clases
    - Es decir, respeta relaciones entre clases, atributos, etc.
  - Cada "foto" cumple los invariantes OCL
  - La foto inmediatamente anterior (posterior) de la aplicación de una operación cumple su precondición (poscondición)
    - Se asume una "frame axiom" (condición marco) que dice que nada que no sea nombrado explícitamente en la poscondición puede cambiar



Al igual que el modelo de objetivos y de operaciones tiene una semántica "**pruning**"

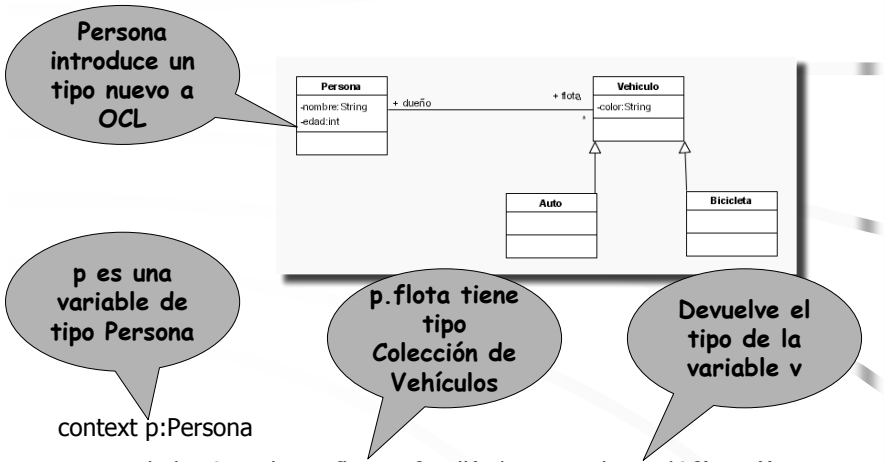
OCL: Resumen

- Especificación formal de
  - invariantes de los objetos
  - pre y post condiciones de operaciones y casos de uso
- Expresiones OCL utilizan vocabulario del diagrama de clases.
- Predicados "navegan" el diagrama de clases.
- context especifica el elemento (clase, operación) del cual se está hablando.
- self indica el objeto actual.
- Soporte para predicar sobre colecciones ("->")
  - select, exists, size, forAll, etc…

Sintaxis

- Definida recursivamente a partir del modelo conceptual
  - Cada clase induce un tipo en el lenguaje
- Expresiones se construyen a partir de:
  - Constantes. Ej. '5', 'Rojo'
  - Variables tipadas. Ej. x:Estudiante, y:Integer
  - Operaciones. Ej. EmitirFactura()
    - (se asumen sin efectos colaterales)
  - Atributos. Ej. Estudiante.nombre
  - Roles. Ej. Piloto.conduce
  - Tipos predefinidos y sus operaciones
    - Tipos básicos:
      - Integer: *, +, -, /, div(), abs(), mod(), max(), min(), sum(), sin(), cos()
      - Real: *, +, -, /, floor(), sum(), sin(), cos()
      - Boolean: and, or, xor, not, implies, if-then-else
      - String:concat(), size(), substring(), toInteger() and toReal()
    - Comparadores de tipos básicos: <, >, =, =>, =<
    - Colecciones: Set, Bag, Sequence
      - …

Clases introducen Tipos en OCL



context p:Persona

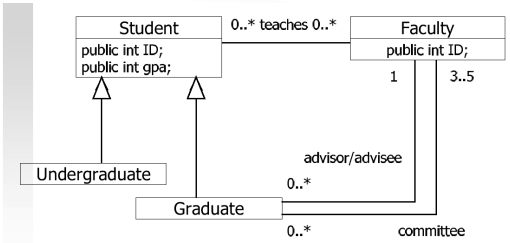
inv: p.edad<18 implies p.flota–>forAll(v | not v.oclIsKindOf(Auto))

Colecciones

- OCL viene con tipos básicos paramétricos para modelar colecciones de objetos
  - Collection(T): Colección genérica de elementos de tipo T
  - Set(T): Colección de elementos de T, no ordenados, sin repetidos
  - OrderedSet(T): Colección de elementos de T, ordenados, sin repetidos
  - Bag(T): Colección de elementos, no ordenados, con repetidos
  - Sequence(T): Colección de elementos, ordenados, con repetidos
- Supongamos X e Y de tipo Collection(T), t de tipo T y p(x): T-->Boolean Estas son operaciones validas:
  - X->size()
  - X->intersection(Y), X->union(Y), …
  - X->isEmpty(), X->notEmpty()
  - X->includes(t), X->excludes(t), X->count(t)
  - X->includesAll(Y), X->excludesAll(Y)
  - X->select(p(x)), X->forAll(p(x)), X->sum()
- Set, prderedSet Bag y Sequence son subtipos de Collection pero tienen especializaciones propias
  - Sequence y OrderedSet proveen first(), last()…

Navegación por roles

- <Clase>.<rol> retorna una colección de objetos que están del otro lado de la asociación
  - Smith.teaches = {co842, sr771}
- Si multiplicidad = 1 entonces retorna un objeto
  - sr771.advisor = Smith

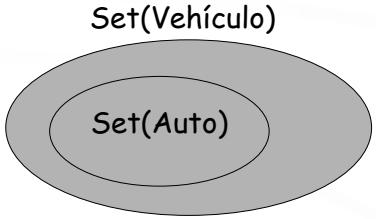


Reglas de Subtipificación

- Sean T1, T2 tipos correspondientes a clases T1 y T2
  - T1 < T2 sii T1 es una subclase de T2
  - Eg. Auto < Vehículo
- Para toda expresión de tipo T
  - Set(T) < Collection(T)
  - Sequence(T) < Collection(T)
  - Bag(T) < Collection(T)
- Si T es un tipo entonces T < OclAny
- La relación < es transitiva
- Si T1 < T2 y C es Collection, Set, Bag o Sequence entonces C(T1) < C(T2)

Subtipado de Colecciones

- Si T1 < T2 y C es Collection, Set, Bag o Sequence entonces C(T1) < C(T2)



Sean Set(Auto):X y Set(Vehículo):Y

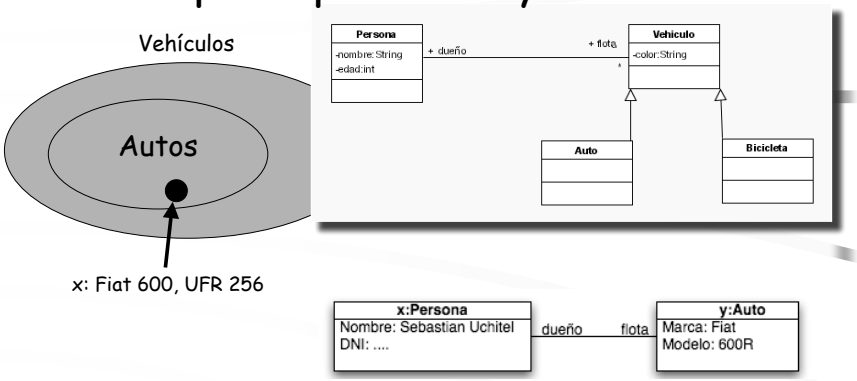
X->union(Y)

¿es una expresión permitida?

¿qué tipo tiene?

¿está bien pensar que X provee una operación unión?

Tipos Aparentes y Reales



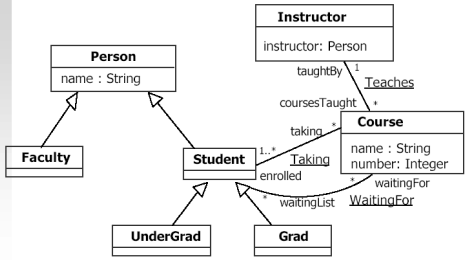
¿De qué tipo es x.dueño?

- El tipo aparente de una expresión es la que puede deducir del la signatura del diagrama de clases. Deducible estáticamente
- El tipo real se deduce del objeto mismo. Deducible dinámicamente

IsKindOf y AsType

- x.IsKindOf(t) retorna si t es el tipo real de x
- x.AsType(t) retorna una referencia (o variable anónima) denotando lo mismo que x pero con tipo aparente t

El procesos de cambiarle el tipo aparente a una variable (en OCL usando AsType) se denomina casting

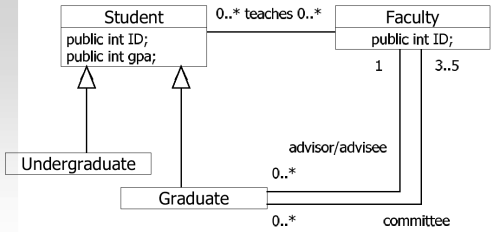


Context i:instructor
inv NoDictaSiCursa:
i.instructor.oclIsKindOf(Grad) implies
i.coursesTaught>insersection (i.instructor.oclAsType (grad). taking)->isEmpty

Ejemplo

Context s: estudiante
inv PrerequisitosRequeridos:
s.acta->includesAll (s.Cursando.prerequsitos
->union (s.anotadoparacursar.prerequisitos))

Context d: Departamento
Inv ProfesoresDictanObligatorias
d.obligatorias.dicatdapor->forAll(i: Instructor |
i.nstructor.oclIsKindof(Profesor))



Modelado con USE

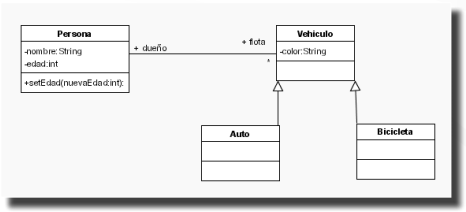
```
model Academia
class Student attributes
  ID : Integer
  gpa : Integer
End

class Undergraduate < Student
class Graduate < Student

class Faculty attributes
  ID : Integer
End

association teaches between
  Student[*] Faculty[*]
End

association advisor between
  Graduate[*] Faculty[1]
end

association committee between
  Graduate[*] Faculty[3-5]
end

constraints

context Student
 inv gpaBound: self.gpa >= 0 &
                self.gpa <= 4
 inv uniqueID:
  Student.allInstances->forAll(
   s1, s2 |
    s1!=s1 implies s1.ID != s2.ID)

context Graduate
  inv advOnCommittee:
   self.committee->includesAll(
     self.advisee)
```

Precondiones y Postcondiciones

Sintaxis:

Context NombreTipo:NombreOpreación(Param1: tipo,…): TipoRetorno
pre nombreprecond1: … param1… …self…
pre nombreprecond2: … param1… …self…

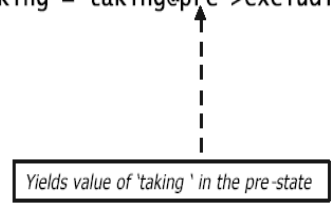
…
post nombrepostcond1: …result… …param1… …self…
post nombrepostcond2: …result… …param1… …self…



context Persona::CumpleAños()
pre: true
post: self.edad = self.edad@pre + 1

Student::dropCourse(c: Course)

```
context Student::dropCourse(c: Course)
  pre  NowTaking:   taking->includes(c)
  post NotTaking:   taking = taking@pre->excluding(c)
```



Sorting sequences of integers

```
context A::sort(s : Sequence(Integer)) : Sequence(Integer)

  post SameSize:
    result->size = s->size

  post SameElements:
    result->forAll(i | result->count(i) = s->count(i))

  post IsSorted:
    Sequence{1..(result->size-1)}->
      forAll(i | result.at(i) <= result.at(i+1))
```


Operadores Para Post-Condiciones

```
model Graph

class Node
operations
  newTarget()
end

association Edge between
  Node[*] role source
  Node[*] role target
end

constraints

context Node::newTarget()
  -- the operation must link exactly one target node
  post oneNewTarget:
    (target - target@pre)->size() = 1

  -- the target node must not exist before
  post targetNodeIsNew:
    (target - target@pre)->forAll(n | n.oclIsNew())
```

- o.a@: Valor de o.a en el estado anterior a la aplicación de la operación
- o.OCLIsNew: True sii o no existía en el estado anterior a la aplicación de la operación

Bibliografía

- Richters, Gogolla 2001. OCL: Syntax, Semantics and Tools
- Especificación de UML-OMG (Capítulo 7, versión 1.5-2003)
  http://www.cs.chalmers.se/Cs/Grundutb/Kurser/form/Papers/OCL_1.5.pdf
- Curso de Dwyer, Hatcliff, Howel (KSU)
  http://www.cis.ksu.edu/santos/771-Distribution/syllabus.html
- Ojo: OCL está en evolución

Relación con Otros Modelos

- OCL y el Modelo Conceptual (Clases y Objetos)
  - Están intrínsicamente relacionados
  - OCL permite mayor expresividad respecto a los estado válidos del mundo
- OCL y Objetivos
  - OCL puede utilizarse para formalizar los objetivos
  - Context t:Trains
    inv PuertaSoloSiVelocidad0yEnPlataforma
    t.velocidad=0 AND
    Platform->allInstances->select(p | p=t.position) -> notEmpty)
- OCL y el modelo de Operaciones
  - OCL puede ser el lenguaje para formalizar operaciones provistas por la maquina y otros agentes
- OCL y el modelo de Contexto
  - Descripción más refinada del estado de los agentes