Object Constraint Language (OCL)

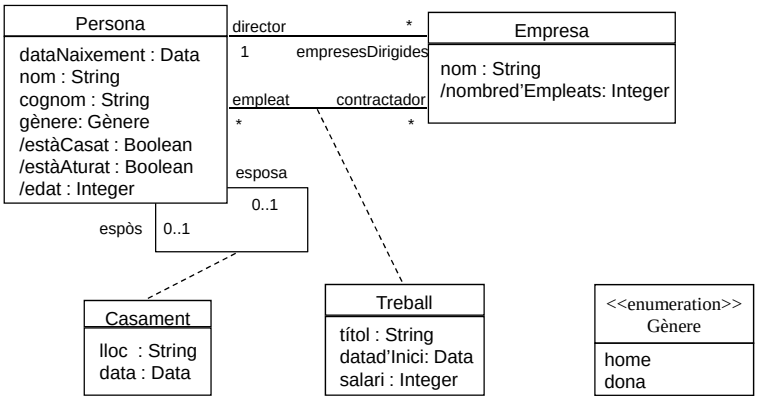
Object Constraint Language (OCL)

- Per a què serveix?
- Propietats del Model Conceptual
- Col·leccions: conjunts, bosses i seqüències
- Navegació per classes associatives, associacions recursives i associacions ternàries
- Generalització / Especialització
- Com especificar en OCL

Per a què serveix OCL?

- Els models gràfics no són suficients per a una especificació precisa i no ambigua
- L'OCL:
  - és un llenguatge formal
  - és un llenguatge d'expressions (no té efectes laterals)
  - no és un llenguatge de programació, sinó d'especificació
  - és un llenguatge tipat
- S'usa per:
  - especificar invariants (restriccions i regles de derivació) del Model Conceptual
  - especificar precondicions, postcondicions i sortides de les operacions

Exemple



Propietats dels objectes

- Cada expressió OCL:
  - s'escriu en el context d'una instància d'un tipus determinat
  - defineix una propietat d'aquesta instància
- Una propietat pot fer referència a:
  - *atributs d'una classe d'objectes*
  - *navegació a través de les associacions*

Propietats dels atributs d'una classe d'objectes:

- Propietat: edat d'una persona -- enter

context Persona **inv:**
l'expressió ha de ser certa per a totes les instàncies de la classe
self.edat
instància contextual de l'expressió (punt de partida)

- Restricció de la propietat: "l'edat de les persones ha de ser superior o igual a zero"

context Persona **inv:**
self.edat >= 0

o, alternativament: **context** p:Persona **inv:**
p.edat >= 0

Propietats dels objectes (II)

Propietats d'una navegació a través d'associacions:

Partint d'un objecte concret, podem navegar a través de les assocciacions del Model Conceptual per referir-nos a d'altres objectes i a les seves propietats

- Com es navega? Objecte.nom-de-rol

objecte de partida *nom de rol d'una associació de l'objecte*

Resultat: conjunt d'objectes de l'altre extrem de nom-de-rol

- si no hi ha nom de rol especificat, es pot usar el nom de la classe d'objectes de l'altre extrem de l'associació (amb minúscules)

- Exemples de navegació:

context Empresa **inv:**

```
self.director          -- director de l'empresa -- Persona
self.director.nom      -- nom del director -- String
self.empleat           -- empleats de l'empresa -- set(Persona)
self.empleat.espòs     -- esposos dels empleats -- set(Persona)
```

Col·leccions: conjunts, bosses i seqüències

Una *col·lecció d'elements* pot ser del tipus:

- *Conjunt:* cada element ocorre una única vegada a la col·lecció
- *Bossa (multiconjunt):* la col·lecció pot contenir elements repetits
- *Seqüència:* bossa on els elements estan ordenats

Exemple:

- "nombre de treballadors diferents que treballen per a una persona"

context Persona::num-treb:Integer **derive:**
self.empresesDirigides.empleat -> size()

Incorrecte: el resultat és una bossa i pot contenir repetits.

context Persona::num-treb:Integer **derive:**
self.empresesDirigides.empleat -> asSet() -> size()

Regles de navegació:

- *si la multiplicitat de l'associació és 1 el resultat és un objecte (o un conjunt d'un únic objecte).*
- *si la multiplicitat de l'associació és >1 el resultat és un conjunt.*
- *si es navega per més d'una associació i la multiplicitat d'alguna d'elles és >1 el resultat és una bossa, encara que de vegades pot ser un conjunt.*

Operacions bàsiques sobre col·leccions (I)

Select: especifica un subconjunt de la col·lecció

- "persones majors de 50 anys que treballen a una empresa"

context Empresa **inv:**
self.empleat -> select(edat>50)

context Empresa **inv:**
self.empleat -> select(p | p.edat>50)

context Empresa **inv:**
self.empleat -> select(p:Persona | p.edat>50)

Collect: especifica una col·lecció que es deriva d'una altra, però que conté objectes diferents

- "edats (amb repetits) dels empleats d'una empresa"

context Empresa **inv:**
self.empleat -> collect(dataNaixement)

versió simplificada: self.empleat.dataNaixement

Operacions bàsiques sobre col·leccions (II)

forAll: expressió que han de satisfer tots els elements

- "tots els empleats de l'empresa s'anomenen Jack"

  context Empresa **inv:**

  self.empleat -> forAll(nom='Jack')

- "la clau externa d'Empresa és el seu nom"

  context Empresa **inv:**

  Empresa.allInstances() -> forAll(e1,e2 | e1<> e2 implies e1.nom<>e2.nom)

Exists: condició que satisfà almenys un element

- "com a mínim un empleat de l'empresa s'ha de dir Jack"

  context Empresa **inv:**

  self.empleat -> exists(nom='Jack')

IsUnique: retorna cert si l'expressió s'avalua a un valor diferent per cada element de la col·lecció

- "la clau externa d'Empresa és el seu nom"

  context Empresa **inv:**

  Empresa.allInstances()-> isUnique(nom)

Ús d'OCL a l'esquema conceptual de dades

- Definició de restriccions textuals d'integritat
  - *"Les persones casades han de ser majors d'edat"*

    context Persona **inv:**

    self.esposa -> notEmpty() implies self.esposa.edat >= 18

    and self.espòs -> notEmpty() implies self.espòs.edat >= 18
  - *"Una empresa té com a màxim 50 empleats"*

    context Empresa **inv:**

    self.empleat -> size() <= 50
  - *"Una persona no pot tenir alhora un espòs i una esposa"*

    context Persona **inv**:

    not ((self.esposa -> notEmpty()) and (self.espòs -> notEmpty())
- Definició d'atributs derivats
  - *"definició de l'atribut derivat nombred'Empleats"*

    context Empresa::nombred'Empleats : Integer

    derive: self.empleat -> size()
  - *"definició de l'atribut derivat estàAturat"*

    context Persona::estàAturat : Boolean

    derive: self.contractador-> isEmpty()

Observació: Noteu la diferència en la sintaxi de la part del context segons si es tracta d'una restricció d'integritat (inv) o d'un atribut derivat (derive)

Navegació per classes associatives

Navegació a una classe associativa:

Es fa servir el nom de la classe associativa (amb minúscula)

- "els sous de les persones que treballen a la UPC han de ser més alts que 1000"

  context Empresa **inv:**

  (self.nom ='UPC' implies self.treball -> forAll (t | t.salari > 1000)

Navegació des d'una classe associativa:

S'usa el nom de rol de l'extrem cap on es vol navegar

Si no s'ha especificat nom de rol, s'usa el nom de la classe.

- "les persones que treballen no poden estar aturades"

  context Treball **inv:**

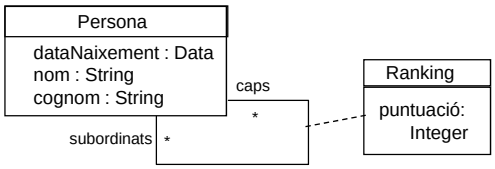
  self.empleat.estàAturat = false

- "un casament ha de ser entre una dona i un home"

  context Casament **inv:**

  self.esposa.sexe = Gènere::dona and self.espòs.sexe = Gènere::home

Navegació a classes associatives d'associacions recursives



- Quan es navega cap a una classe associativa d'una associació recursiva hi ha dues possibilitats depenent de la direcció

  Ex: es pot navegar de Persona a Ranking en direcció a *caps* o en direcció a *subordinats*

- Per distingir s'usa el nom de rol cap on es vol navegar entre claudàtors
  - "la suma de puntuacions que pertanyen a la col·lecció de *subordinats* ha de ser positiva"

    context Persona **inv:**

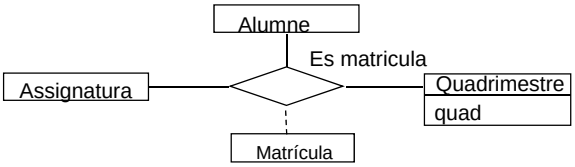
    self.ranking[subordinats] -> sum() > 0

  - "la suma de puntuacions que pertanyen a la col.lecció de *caps* ha de ser positiva"

    context Persona **inv:**

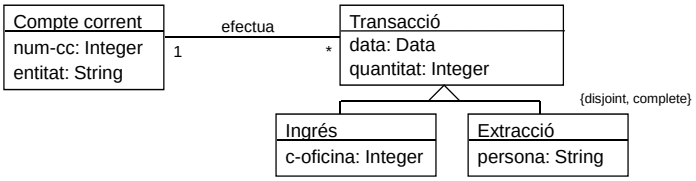
    self.ranking[caps] -> sum() > 0

Navegació per associacions ternàries (amb associativa)



- Navegació cap a la classe associativa o des de la classe associativa: Similar al cas de les associacions binàries.
  context Alumne **inv:**
  self.matrícula --matrícules d'un alumne
  context Matrícula **inv:**
  self.alumne -- alumne d'una matrícula
- Navegació d'una classe a una altra classe de l'associació.
  context Alumne **inv:**
  self.matrícula.assignatura -- assignatures de les que s'ha matriculat un alumne (el resultat és una bossa)
- Navegació d'una classe a una altra classe de l'associació amb una condició de selecció que involucra la classe restant.
  context Alumne **inv:**
  self.matrícula->select(m | m.quadrimestre.quad='1Q-04-05').assignatura
  -- assignatures de les que s'ha matriculat un alumne el quadrimestre '1Q-04-05'

Generalització / Especialització (herència)



Navegació:

- *Accés directe a les propietats de la superclasse*
  context Ingrés **inv:**
  self.compte-corrent.num-cc -- número de compte d'un ingrés
- *Accés a les propietats definides al nivell de subclasse*
  context CompteCorrent **inv:**
  self.transacció->select(oclIsTypeOf(Extracció)).oclAsType(Extracció).persona -> asSet()
  -- persones que han tret diners d'un compte corrent

Aspecte addicional:

- *Selecció d'objectes que pertanyen a la subclasse*
  context CompteCorrent **inv:**
  self.transacció -> select(oclIsTypeOf(Ingrés)) -- ingressos d'un compte

Definició de variables: “let” i “def”

- **Let:** Es fa servir en expressions que tenen subexpressions que s’usen més d’una vegada
  - *“La suma de salaris és menor que 1000 per les persones de menys de 40 anys i més gran o igual que 1000 per la resta”*

```
context Persona inv:
  let ingressos : Integer = self.treball.salari -> sum() in
  if self.edat < 40 then ingressos < 1000 else ingressos >= 1000
  endif
```

- **Def:** Es pot reutilitzar en altres expressions ocl (com si fos un atribut de la classe)

```
context Persona
  def: ingressos : Integer = self.treball.salari -> sum()
context Persona inv:
  if self.edat < 40 then ingressos < 1000 else ingressos >= 1000
  endif
```


Com especificar en O.C.L.

- Una expressió O.C.L s’especifica sempre començant en una classe d’objectes determinada: *instància contextual*
- Una *expressió* es pot especificar de diverses maneres, segons la instància contextual de partida.

  “els dos membres d’un matrimoni no poden treballar a la mateixa empresa”

```
context Empresa inv:
  self.empleat.esposa -> intersection(self.empleat) -> isEmpty()
context Persona inv:
  self.esposa.contractador -> intersection(self.contractador) -> isEmpty()
```

- Indicacions per escollir la instància contextual
  - si la restricció restringeix el valor de l’atribut d’una classe, aquesta és la classe candidata
  - si la restricció restringeix el valor dels atributs de més d’una classe, qualsevol d’aquestes n’és la candidata
  - normalment, qualsevol restricció hauria de navegar a través del menor nombre possible d’associacions

Com especificar en OCL: exemples

- "L'espòs i l'esposa d'un matrimoni han de ser majors d'edat"

  context Casament **inv:**
  self.esposa.edat >= 18 and self.espòs.edat >= 18 -- *és preferible a*

  context Persona **inv:**
  self.esposa -> notEmpty() implies self.esposa.edat >= 18 and
  self.espòs -> notEmpty() implies self.espòs.edat >= 18

- "Totes les persones han de ser majors d'edat"

  context Persona **inv:**
  self.edat >= 18 -- *no és equivalent a*

  context Empresa **inv:**
  self.empleat -> forAll (edat >= 18)

- "Ningú no pot ser director i empleat d'una empresa"

  context Empresa **inv:**
  not(self.empleat -> includes(self.director))

  context Persona **inv:**
  not(self.empresesDirigides.empleat -> includes(self))

  -- *quina és preferible en aquest cas?*

Operacions estàndard de tipus booleà

Operació	Notació	Resultat
or	a or b	booleà
and	a and b	booleà
or exclusiu	a xor b	booleà
negació	not a	booleà
igualtat	a = b	booleà
desigualtat	a <> b	booleà
implicació	a implies b	booleà
if-then-else-endif	if a then b else b' endif	tipus de b o b'

Operacions estàndard de tipus string

Operació	Notació	Resultat
concatenació	string.concat(string)	string
tamany	string.size()	integer
substring	string.substring(int,int)	string
igualtat	string1 = string2	booleà
desigualtat	string1 <> string2	booleà

Operacions estàndard d'una classe d'objectes

Operació	Resultat
allInstances	retorna el conjunt de totes els elements de la classe d'objectes

Operacions estàndard de tipus col·lecció

Operació	Resultat
size()	nombre d'elements de la col·lecció
count(object)	nombre d'ocurrències de l'objecte
includes(object)	cert si l'objecte pertany a la col·lecció
includesAll(collection)	cert si els elements del paràmetre *collection* són a la col·lecció actual
excludes(object)	cert si l'objecte no pertany a la col·lecció
excludesAll(collection)	cert si els elements del paràmetre *collection* no són a la col·lecció actual
isEmpty()	cert si la col·lecció és buida
notEmpty()	cert si la col·lecció no és buida
sum()	suma de tots els elements
exists(expression)	*expression* és cert per algun element?
forAll(expression)	*expression* és cert per tots els elements?
isUnique(expression)	cert si *expression* avalua a un valor diferent per cada element de la col·lecció actual

Operacions estàndard (específiques) de tipus conjunt i bossa

Operació	Resultat
select(expression)	selecciona el subconjunt d'elements del conjunt o bossa actual per als quals *expression* és cert
reject(expression)	elimina el subconjunt d'elements del conjunt o bossa actual per als quals *expression* és cert
union(set)	resultat d'unir el conjunt o bossa actual amb el *set*
intersection(set)	resultat de la intersecció del conjunt o bossa actual amb el *bag*
union(bag)	resultat d'unir el conjunt o bossa actual amb el *bag*
intersection(bag)	resultat de la intersecció del conjunt o bossa actual amb el *bag*
asSet()	resultat d'eliminar repetits del conjunt o bossa actual

Bibliografia

- OMG - Unified Modeling Language
  Object Constraint Language Specification, v. 2.2.
  Febrer 2010. (Especialment el capítol 7: OCL Language Description)

- J.Warmer; A.Kleppe
  The Object Constraint Language: precise modeling with UML
  Addison-Wesley, 1999.

Pàgines web amb informació d'OCL:

- http://www.omg.org/spec/OCL
- http://www.software.ibm.com/ad/ocl