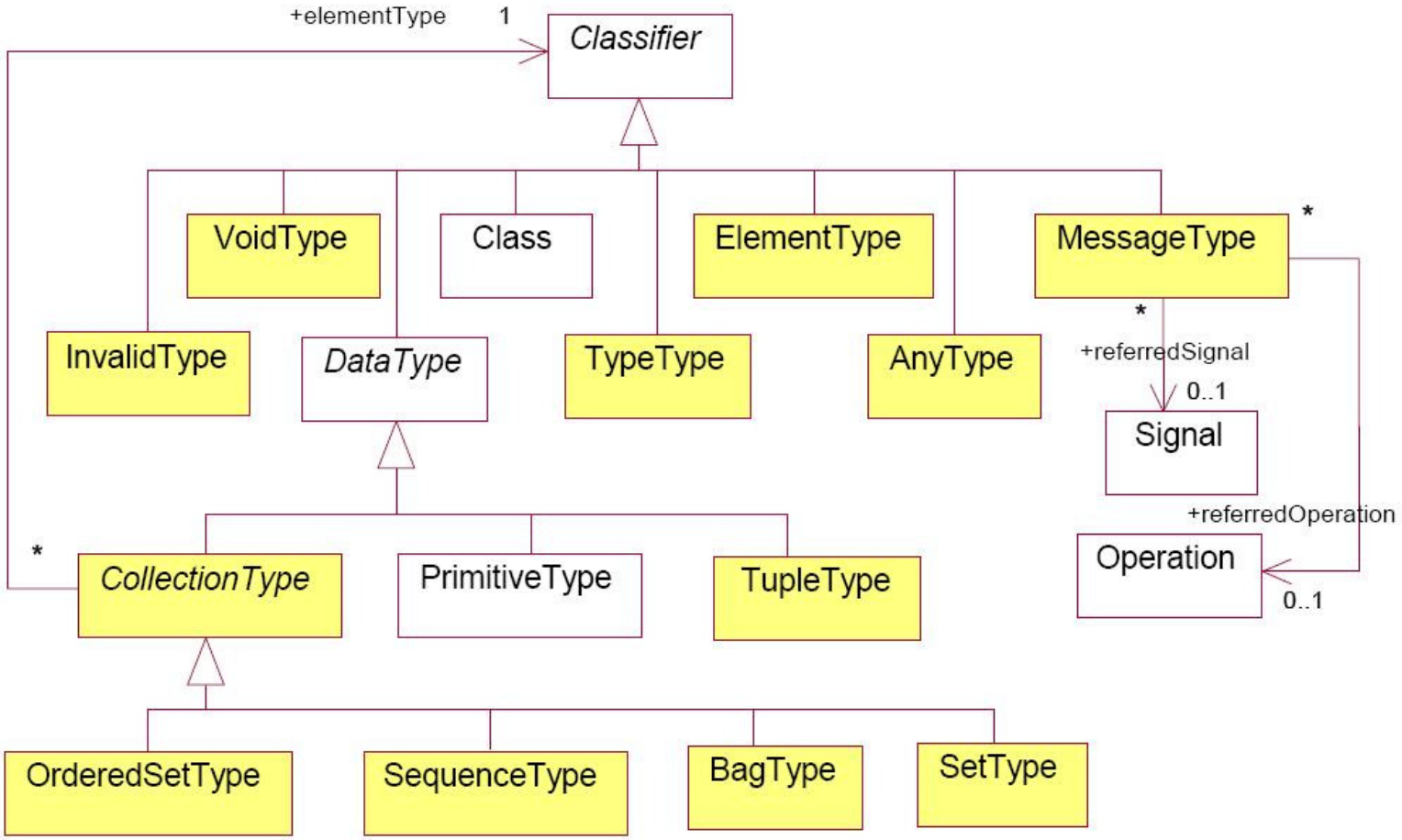
Metamodellek a szoftverfejlesztésben

Object Constraint Language 2

OCL típusok


+elementType
1
Classifier
VoidType
Class
ElementType
MessageType
*
*
+referredSignal
0..1
Signal
InvalidType
DataType
TypeType
AnyType
+referredOperation
Operation
0..1
*
CollectionType
PrimitiveType
TupleType
OrderedSetType
SequenceType
BagType
SetType

Üzenetküldés - ismétlés

- `hasSent` operátor
  - utófeltételben, üzenetet küldünk
  - az eredmény lényegtelen

```
context Subject::hasChanged()
post: observer^update(5, 24)

post: observer^update(?: Integer, ?: Integer)
```

Message operátor

- Minden elküldött üzenet típusa:

  `OclMessage`

- üzenet-szekvenciák

```
context Subject
        observer^^update(5, 24)
```

a `Subject`-től az `observer`nek küldött `update(12, 14)` üzenetek szekvenciája

Message operátor

- különböző célhoz küldött üzenetek:

```
context Subject::hasChanged()
post: let messages: Sequence(OclMessage) =
            observer->collect(o |
                    o^^update(?: Integer, ?: Integer))

       in messages->notEmpty()
```

Message operátor

- szemantikailag egyenértékűek

```
context Subject::hasChanged()
post: observer^update(5, 24)
```

```
context Subject::hasChanged()
post: observer^^update(5, 24)->notEmpty()
```

Message operátor

- paraméterek is referálhatók

```
context Subject::hasChanged()
post: let messages: Sequence(OclMessage) =
                observer^^update(?: Integer, ?: Integer))
       in
         messages->notEmpty() and
         messages->exists(m | m.i > 0 and m.j>= m.i)
```

`OclMessage` műveletei

isSignalSent()	Boolean	szignál volt ?
isOperationCall()	Boolean	operáció hívás volt ?
hasReturned()	Boolean	igaz, ha a hívás befejeződött és van visszatérési érték
result()	a hívott oper. típusa	a visszatérési érték

Package

- A névtér probléma szokásos megoldása

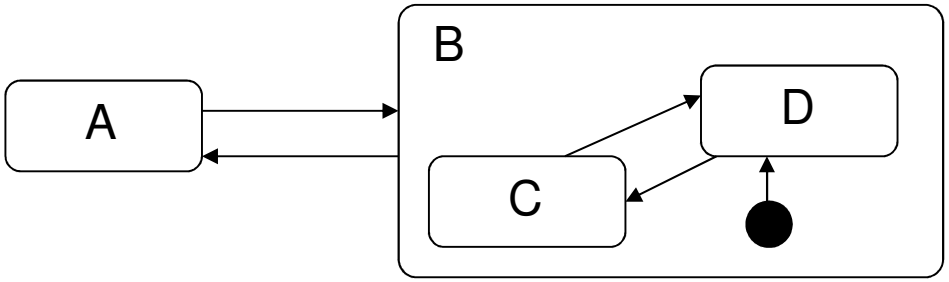
```
package Pack1::Pack2::Pack3
context X
....
```

- egyenértékűek

```
context Pack1::Pack2::Pack3::X
....
```

Állapotok modellje

- Az X osztályban legyen




```
oclInState(state_ref) : Boolean
```

```
context X
   this.oclInState(A)
   this.oclInState(B)
   this.oclInState(B::C)
   this.oclInState(B::D)
```

Tuple

- A struct vagy record OCL változata

```
Tuple {name: String = 'John', age: Integer = 10}
Tuple {name = 'John', age = 10}
Tuple {age = 10, name = 'John'}
```

- egyenértékűek
- mezők elérése:

```
Tuple {name = 'John', age = 10}.age = 10
```

Tupletype

- nincs neve

```
context Airline
def: full: Set(TupleType(
                flnr: String, d: Time, target: String)) =
flights->select
    ((1 - passengers->size()/maxNrPassengers) > 0.9)->
collect(Tuple {
   flnr: String = flightNumber,
   d: Time = departTime,
   target: String = destination.name})->asSet()
```

Undefined érték

- hiányzik vagy érvénytelen (void, invalid)
- nem definiált kifejezések, pl.

```
sequence->at(i)
```

ha i > a sequence hosszánál

- propagálódik

```
if bool_exp then expr_1 else expr_2
```

a kifejezés sem def, ha a bool nem

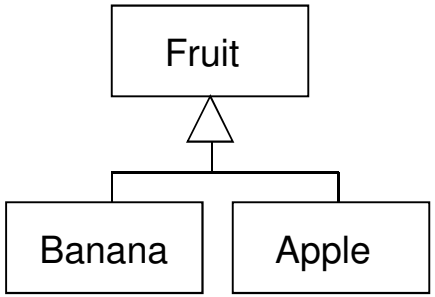
Undefined érték

- speciális esetek
  - true or undefined = true
  - false and undefined = false
  - false implies undefined = true
- *if*-ben csak a végrehajtásra kerülő ág érvényessége számít

elem típusa

- Szigorúan típusos nyelv
- Típus ellenőrzése (minden elemen):

```
oclIsTypeOf(type:OclType) : Boolean
```



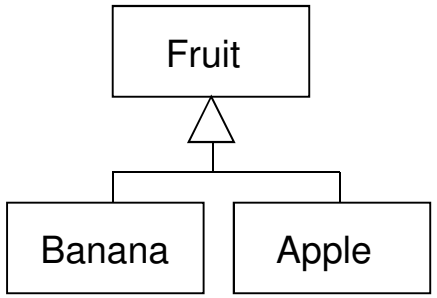

```
context Fruit
inv: self.oclIsTypeOf(Fruit)= true
inv: self.oclIsTypeOf(Apple)= false

context Apple
inv: self.oclIsTypeOf(Apple)= true
inv: self.oclIsTypeOf(Fruit)= false
```

típus konformancia

- A helyettesíthetőség elvén alapul
- Konformancia ellenőrzése:

```
oclIsKindOf(type:OclType) : Boolean
```




```
context Fruit
inv: self.oclIsKindOf(Fruit)= true
inv: self.oclIsKindOf(Apple)= false

context Apple
inv: self.oclIsKindOf(Apple)= true
inv: self.oclIsKindOf(Fruit)= true
```

típus konformancia

- Classifier-en
  - tranzitív, reflexív, anti-szimmetrikus
  - osztály konform
    - ősosztályokkal
    - implementált interfészekkel
  - interfész konform
    - ősinterfészekkel

típus konformancia

- Collection-ön
  - Collection(T1) konform Collection(T2)-vel
  - Bag(T1) konform Bag(T2)-vel
  - Set(T1) konform Set(T2)-vel
  - Sequence(T1) konform Sequence(T2)-vel
  - Ord.Set(T1) konform Ord.Set(T2)-vel
  - Subkoll.(T1) konform Collection(T2)-vel
    - ha T1 konform T2-vel

típus konformancia

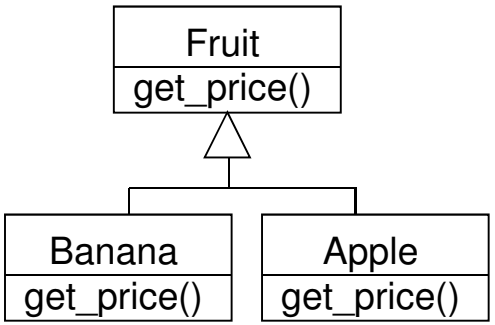
- Integer konform Real-lel
- Tuple1 konform Tuple2-vel, ha
  - minden név1 = név2
  - minden típus1 konform típus2-vel

castolás

```
object_Type2.oclAsType(Type1)
```

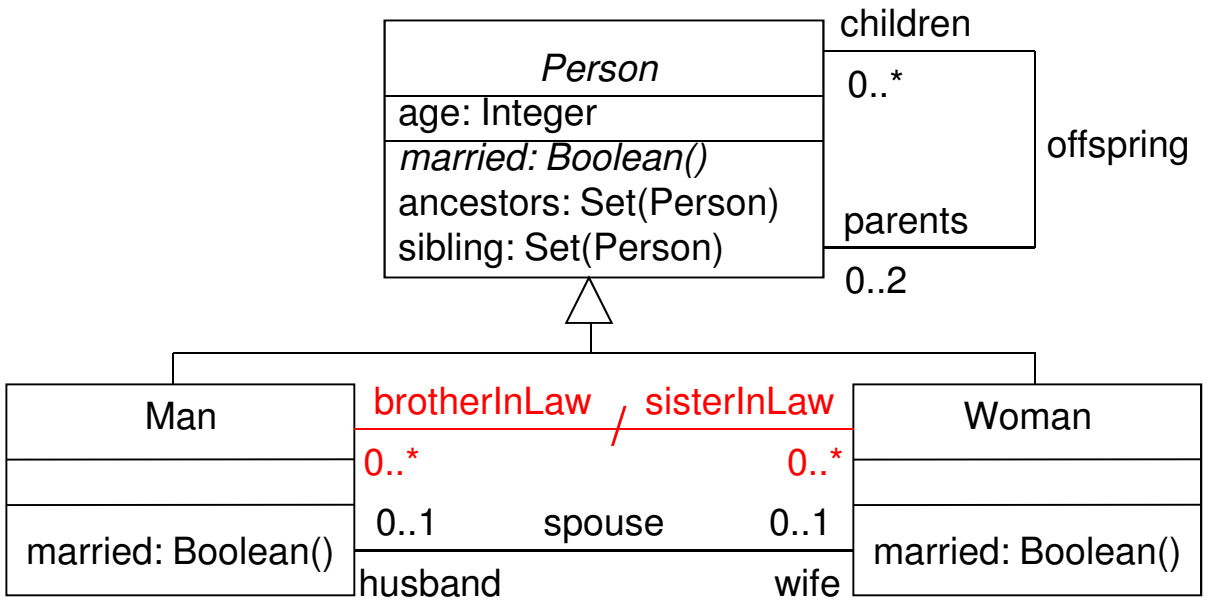
- object_Type2 típusa Type1 lesz
- Ha Type1 konform Type2-vel
- egyébként Undefined

override




```
context Banana
inv: self.get_price ..
                -- Banana::get_price
inv: self.oclAsType(Fruit).get_price ..
                -- Fruit::get_price
```

Példa


Person
age: Integer
married: Boolean()
ancestors: Set(Person)
sibling: Set(Person)
children
0..*
offspring
parents
0..2
Man
married: Boolean()
brotherInLaw
sisterInLaw
0..*
0..*
0..1
spouse
0..1
husband
wife
Woman
married: Boolean()

oclInvalid

- konform minden típussal
- az *InvalidType* metatípus egyetlen példánya
- egyetlen példánya az *invalid*
- az érték érvénytelenségét jelöli
- minden műveletre *oclInvalid*-ot ad
- kivéve az *oclIsUndefined()* -ot és *oclIsInvalid()* -ot

oclVoid

- konform minden típussal
- a *VoidType* metatípus egyetlen példánya
- egyetlen példánya a *null*
- az érték hiányát jelöli
- minden műveletre *oclInvalid*-ot ad
- kivéve az *oclIsUndefined()* -ot

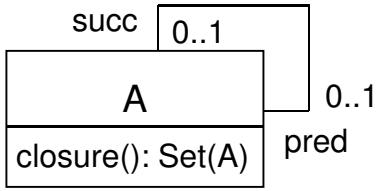
OclAny

- A kollekciók kivételével minden típus megvalósítja

object = (object2:OclAny)	Boolean	ugyanazok
object <> (object2:OclAny)	Boolean	nem ugyanazok
object.oclIsUndefined()	Boolean	oclVoid vagy oclInvalid
object.oclIsInvalid()	Boolean	oclInvalid
object.oclIsKindOf(type:OclType)	Boolean	típusa konform type-pal
object.oclIsTypeOf(type:OclType)	Boolean	típusa egyezik type-pal
object.oclIsNew()	Boolean	op. közben keletkezett
object.oclAsType(type:OclType)	type	type-ra castol
object.oclInState(st:StateName)	Boolean	st állapotban van
type::allInstances()	Set(type)	$ véges típusokra

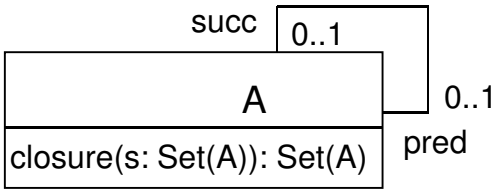
tranzitív lezárt

closure – a következők halmaza




```
context A::closure() : Set(A)
body: if succ.oclIsUndefined()
         then self->asSet()
         else succ.closure()->including(self)
      endif
```

tranzitív lezárt

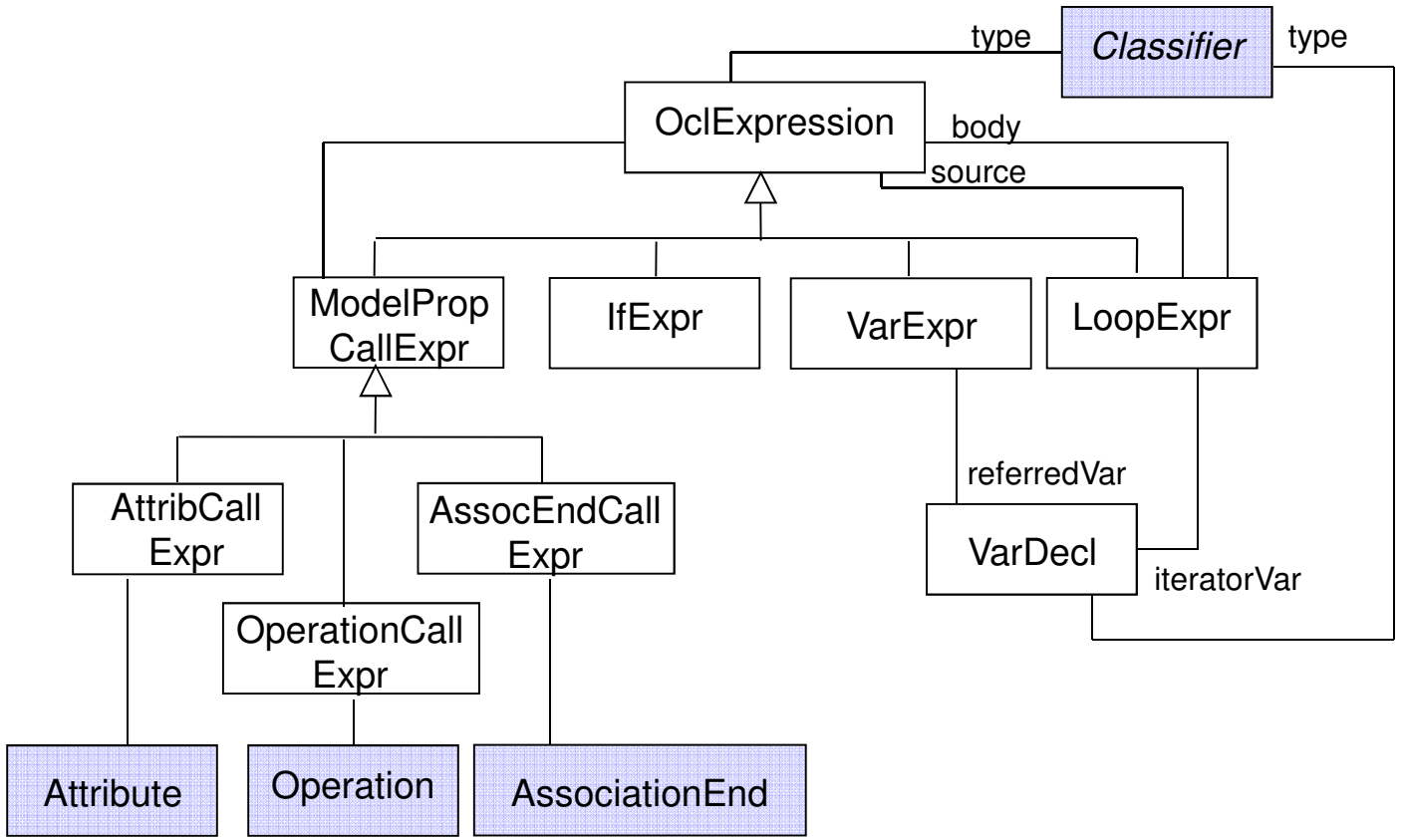



```
context A::closure(s : Set(A)) : Set(A)
body: if succ.oclIsUndefined() or s->includes(self)
         then s->including(self)
         else succ.closure(s->including(self))
      endif
```

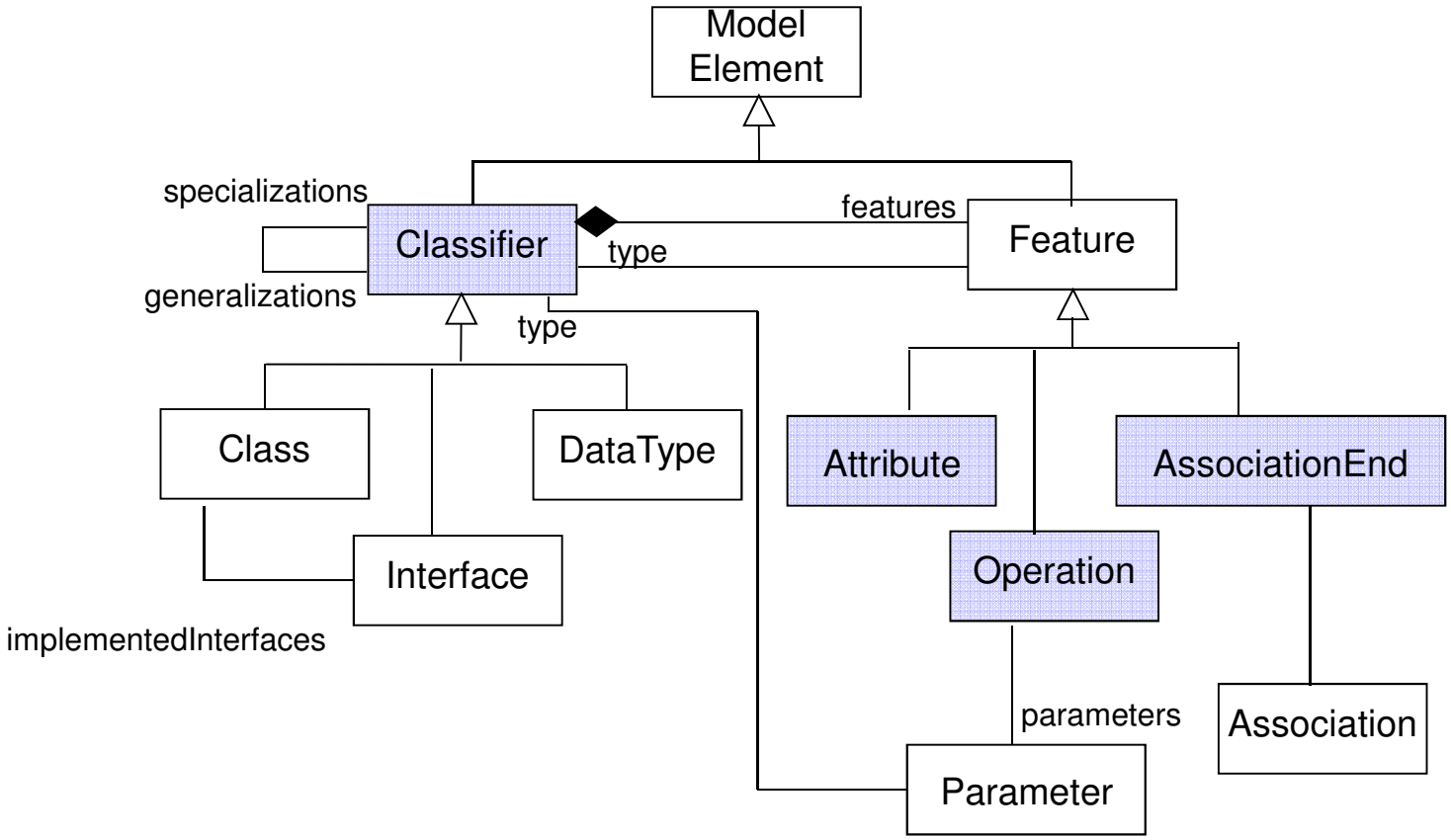
OCL és más UML modellek

- Feltételek, őrök, paraméterek
  - szekvencia diagramon és állapotgépen
- Állapot invariánsok
- Use-case-re elő- és utófeltételek
- Gond:
  - mi a kontextuális példány és típus
  - self ???

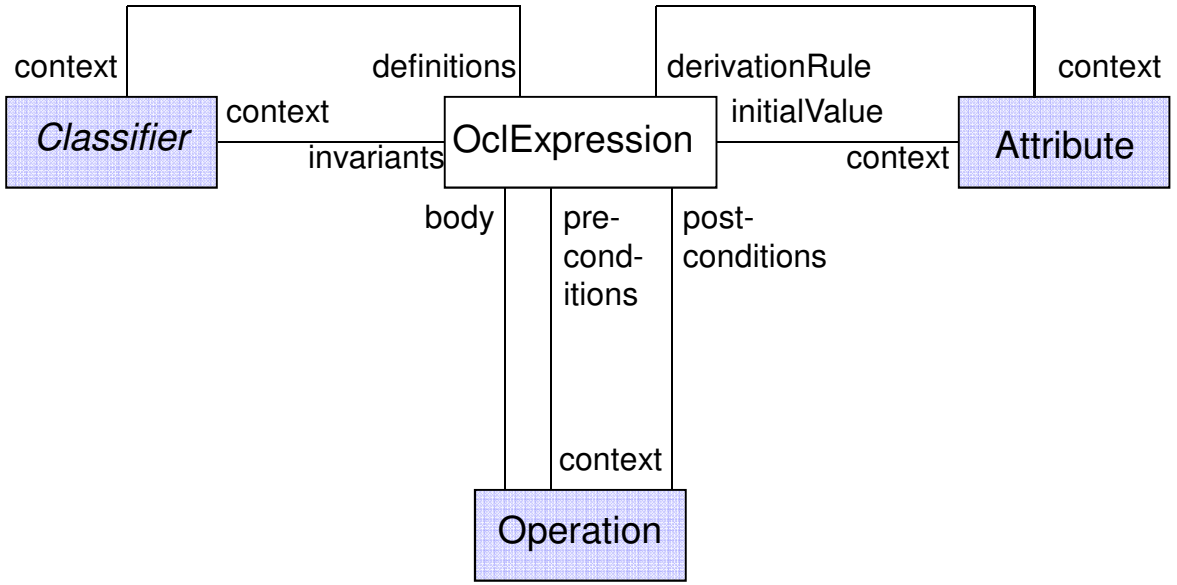
OCL Expr Metamodell



UML Metamodell



OCL – UML metaszinten



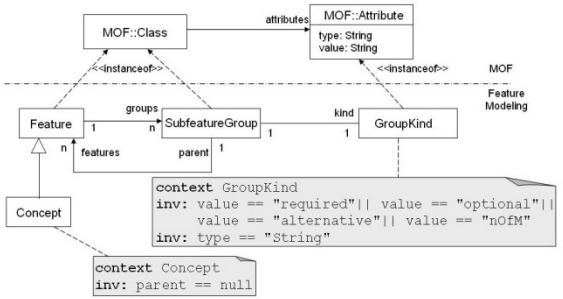
OCL – modellezés

- Elő- és utófeltételek
- Invariánsok
- Változók, asszociációk inicalizálása
- Származtatási szabályok
- Operációtörzsek
- Új attribútumok és operációk
- Állapot invariánsok
- Állapotgépen őrök
- interakciókon feltételek, őrök
- Activity diagramon feltételek

OCL - MDA

- Nyelvet definiálni
  - A MOF és UML a példa
  - M3 és M2 szinten új
  - Meta2 – sorozat 4-8 oldalak, használtuk

Metamodellezés példa 1



Metamodellek... © BME IIT, László Zoltán 4

OCL Java implementálása

- UML modell elemek

UML	Java
Osztály	Osztály
Operáció	Metódus
Attribútum	private + get-set a láthatóságra figyelemmel
Asszociáció	mint Attribútum + Kollekciók
Állapot	állapotonként egy bool változó
Esemény	Metódus
Enumeráció	static Java megoldás
Interfész	Interfész

OCL Java implementálása

- OCL elemi típusok

OCL	Java
Integer	int
Real	Float
String	String
Boolean	boolean
oclType	Class
oclAny	Object

OCL Java implementálása

- OCL kollekciók
  - set – HashSet
  - többi – ArrayList
- Tuples
  - java.utol.Map ???
  - különálló osztály

OCL Java implementálása

- Attribútum definíció és kezdőérték
  - értelem szerint
- Body – a metódus törzse
- derivációs szabály
  - metódus - lekérdezéskor újraszámol
  - attribútum - obszerver minta szerint
    - nincs publikus set-je

OCL Java implementálása

- Invariáns
  - bool értékű fv
  - mikor hívjuk ???
- Elő- és utófeltétel
  - magában az operációban
  - runtime változatban marad ?
  - mi legyen a hibával ?

OCL Java implementálása

- Őrök, feltételek állapotgépben
  - állapotgéptől függ
    - esemény = metódus --- előfeltétel
    - interpreter --- az akciók végrehajtásába
- Egyéb elemekre (pl. use-case) nincs szisztematikus módszer

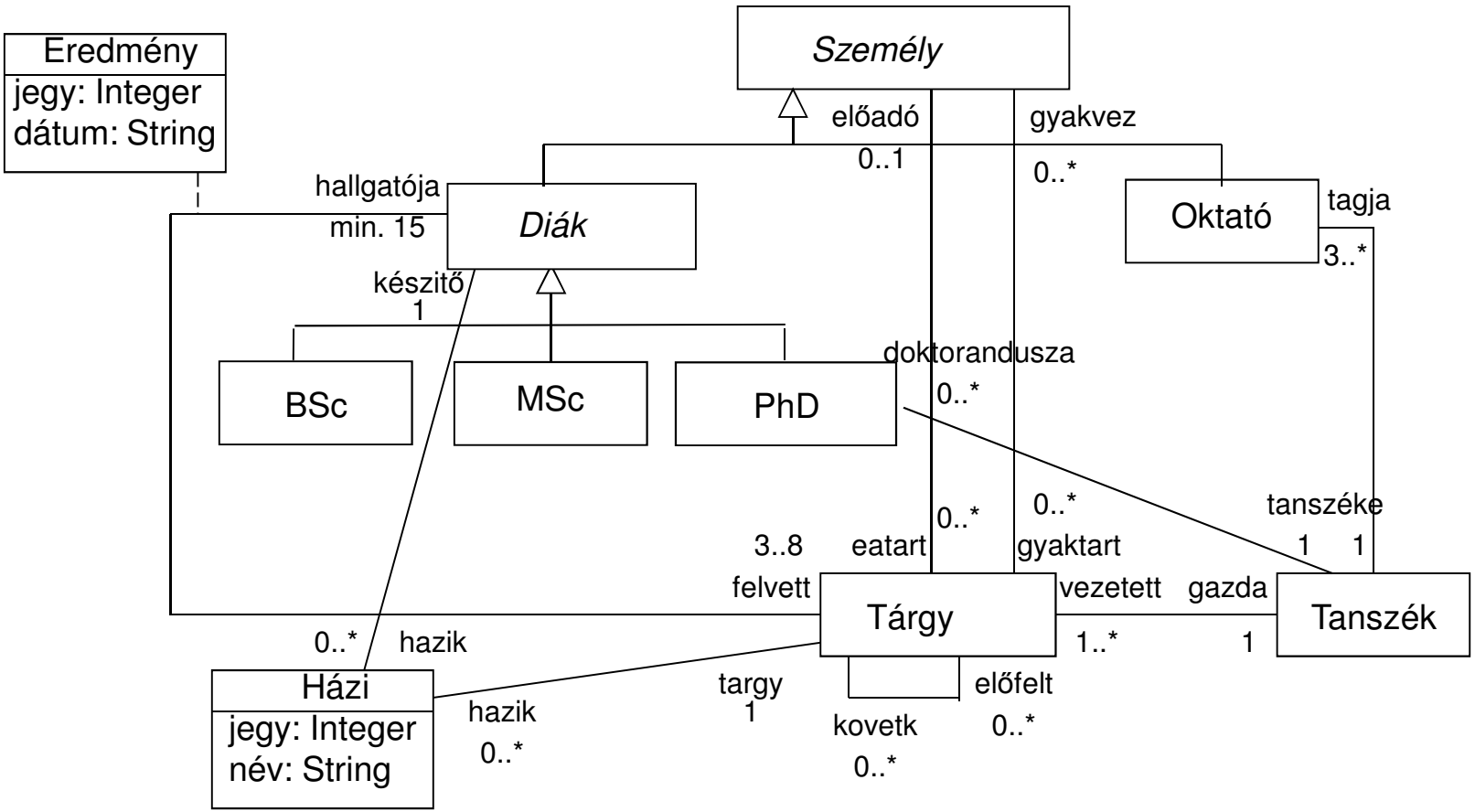
Octopus

- **OC**L **To**ol for **P**recise **U**ML **S**pecifications
  - Eclipse alapú IDE
  - UML modelt importál XMI-ból (gyenge)
  - UML text modelt használ
  - OCL standard szintaxis
  - OCL validátor
    - AST-t is készít
  - Java kódgenerátor

Házi feladat

- Octopus használatával UML modelhez szöveges korlátozásokból OCL leírás készítése
- Model módosítása - UML text model
- Java kódgenerálás
- OCL validálása - tesztesetek
- Beadás: nov 28. 16 óráig http://devil.iit.bme.hu:9180/hercules/ feltölteni
- Bemutatás: dec 4.-én, az órán

Class Diag


Eredmény
jegy: Integer
dátum: String
Személy
előadó
0..1
gyakvez
0..*
hallgatója
min. 15
Diák
Oktató
tagja
3..*
készitő
1
BSc
MSc
PhD
doktorandusza
0..*
0..*
0..*
tanszéke
1 1
3..8
eatart
gyaktart
felvett
Tárgy
vezetett
gazda
Tanszék
0..*
hazik
1..*
1
Házi
jegy: Integer
név: String
hazik
0..*
targy
1
kovetk
0..*
előfelt
0..*

Korlátozások 1

- ha egy tárgynak nincs előadója, akkor biztosan van gyakvezér(ek).
- előadó oktató vagy PhD hallgató lehet.
- gyakvezér lehet még az MSc-s hallgató is.
- hallgató csak 3.0 átlag felett kaphat ösztöndíjat.
- PhD hallgatónak legalább egy tárgyat oktatnia és legalább egyet hallgatnia is kell, valamint a két tárgy nem lehet ugyanaz.

Korlátozások 2

- egy diáknak legfeljebb két tárgyból lehet házi feladata.
- egy diák nem taníthat olyan hallgatót, akivel egy tárgyat hallgat.
- ha a diák által felvett tárgynak van előfeltétele, akkor valamennyi előfeltétel tárgyat a diáknak teljesítenie kell (legalább 2-st kell elérnie).

Feladat

- Adjon a modellhez állapotot
- Adhat a modellhez attribútumot, metódust, asszociációt
- Adjon a modellhez egy új osztályt
- Készítsen még 3 OCL állítást (egyenként minimum 3 osztályt érintsen)

Beadandó

- Bővített modell class diagram
- Bővített modell Octopus modellje
- Korlátozások OCL nyelven
- Korlátozások szövegesen
- Java source – tesztek IS !
- Zipelve – herculesre feltöltve