Modellalapú szoftverfejlesztés

VII. előadás

Grafikus nyelvek modellezése

Dr. Mezei Gergely

1

Szöveges modellezés

Fordítóprogramok,
Nyelvfeldolgozás lépései.
Kódgenerálás,
Interpreterek

2

Grafikus modellezés

Szerkezet + megjelenítés,
Blockly, UML Profile,
Metamodellezés,
Szemantika

3

Modellfeldolgozás

Modellfeldolgozás,
Kódgenerálás,
Gráftranszformáció,
Modellalapú fejlesztés

MIRŐL LESZ SZÓ?

Grafikus nyelvek modellezése

I. Grafikus nyelvek/modellek

II. Absztrakt szintaxis UML alapon

III. Blockly

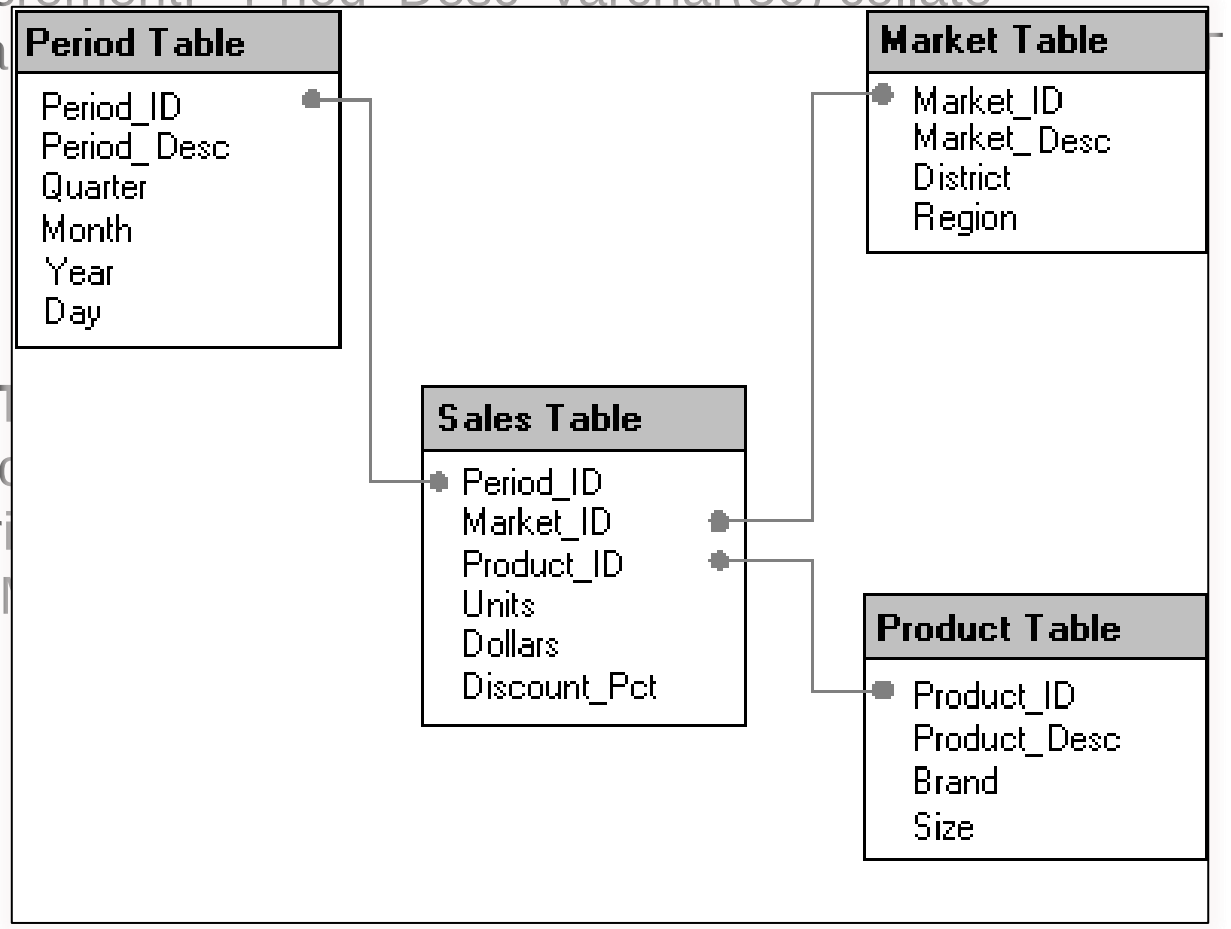
IV. Metamodellezés

V. Kényszerek

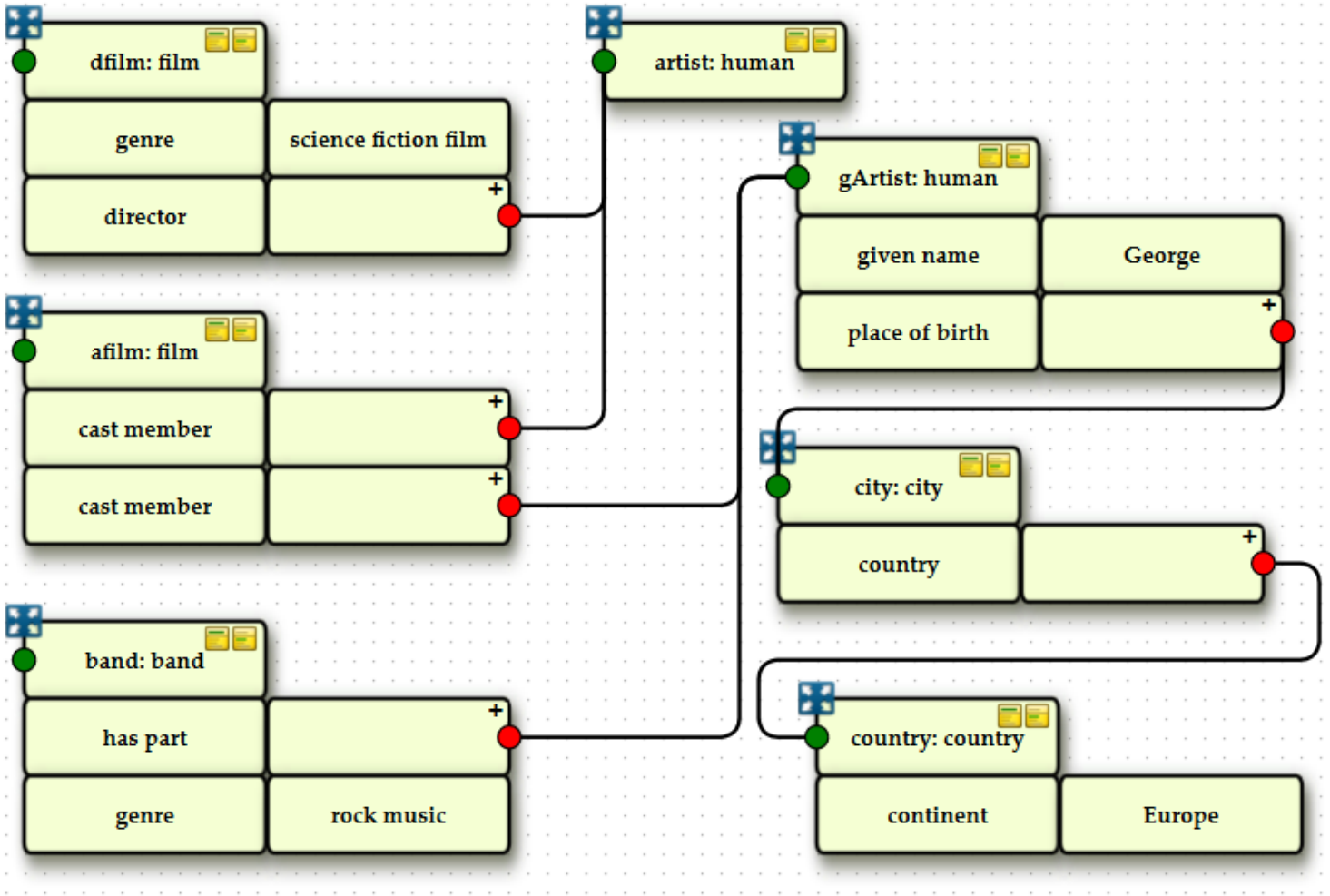
Adatbázis tervezés – SQL

CREATE TABLE IF NOT EXISTS `**Period Table**`
(`Period_ID` int(11) NOT NULL auto_increment, `Priod_Desc` varchar(50) collate latin2_hungarian_ci default NULL, `Qua
NULL, `Year` mediumint(9) NOT NULL,
(`Period_ID`));

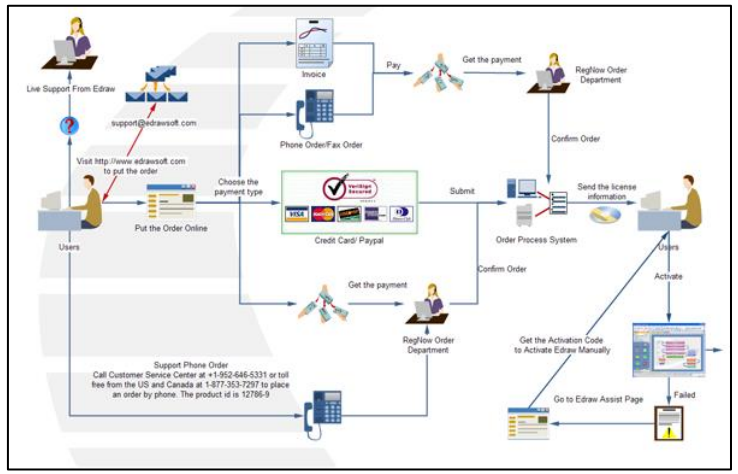
CREATE TABLE IF NOT EXISTS `**Market T**
(`Market_ID` int(11) NOT NULL auto_inc
latin2_hungarian_ci default NULL, `Distri
NULL, `Region` int(11) NOT NULL, PRI

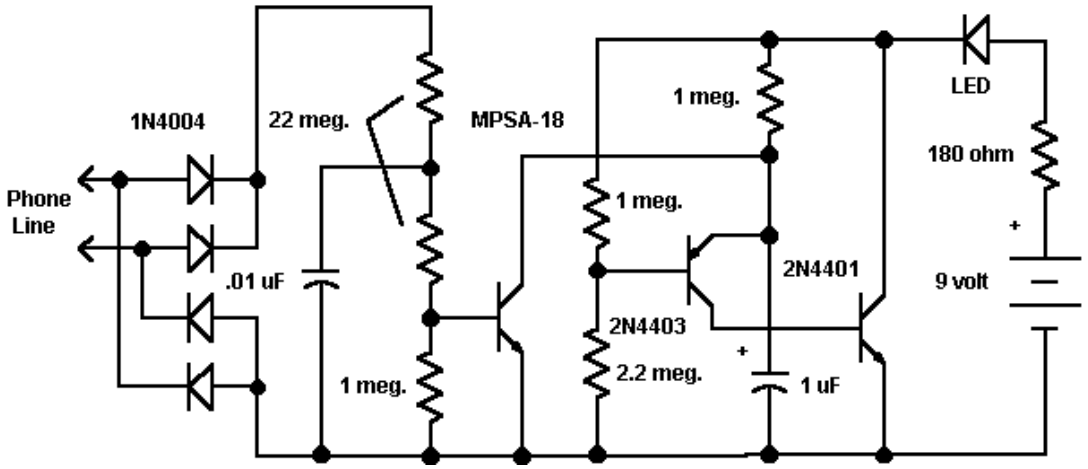


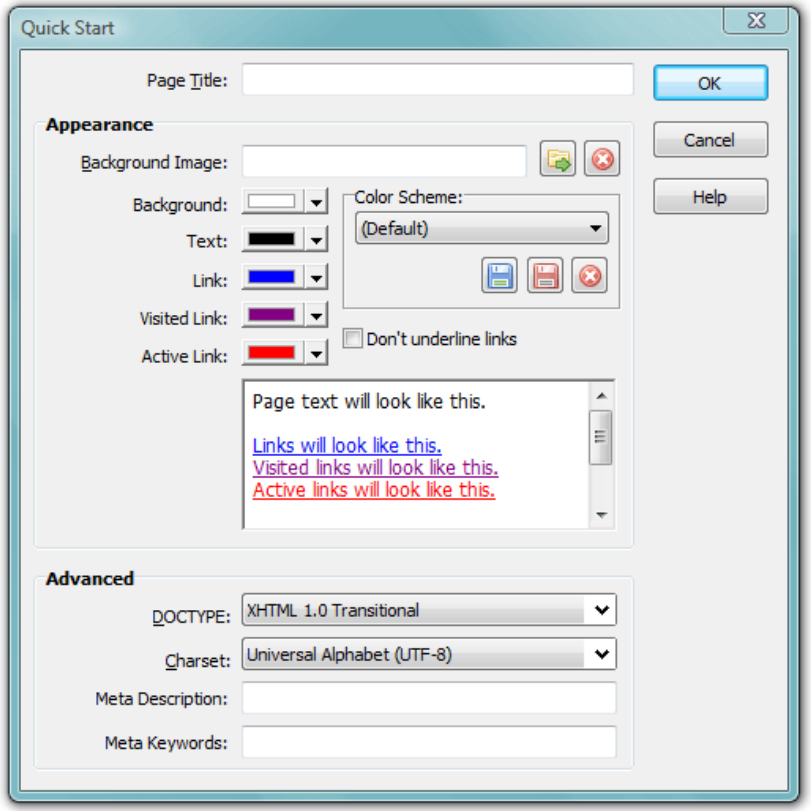
SPARQL vizuális lekérdezés



További példák







Vizuális modellezés előnyei

- A szoftverfejlesztés egyik legproblémásabb lépése a szoftverfejlesztő – szakértő kommunikáció
  - A szakértő általában nem programozó, nem beszél sem C#, sem Kotlin nyelvet!
  - Nem is akarja megtanulni!
  - Ismeri és megszokta viszont a szakterületi jelölésrendszert, *ami általában grafikus*
  - Ha a szakterületi jeleket használjuk, a kommunikáció sokkal könnyebb

Szöveges vs. grafikus (vizuális) nyelvek

Szöveges nyelvek

- Könnyű írni
  - Gyorsan megadható
  - Komplex összefüggések is jól leírhatóak
- Nehéz értelmezni
  - Nehéz átlátni egy bizonyos összetettség felett
  - A szintaxist meg kell tanulni
- Programozók számára jobban érthető
- Modellek tárolása, verziókezelése megoldott (pl. git)

Grafikus (vizuális) nyelvek

- Nehéz írni
  - Lassabb, nehézkesebb
- Könnyű értelmezni
  - A jelölés gyakran önleíró, intuitív
  - Gyors betanulási idő
- Átlagember számára jobban érthető
- Modellek tárolása, verziókezelés bonyolult (szerializálás)
- Van ahol az elrendezés a modell része

Miből áll a szakterületi nyelv?

- Mire van szükség egy szakterületi nyelv definiálásakor?

> Nyelv struktúrája
> Kiegészítő kényszerek

} Absztrakt szintaxis

> Megjelenítés

} Konkrét szintaxis

> Struktúra jelentése

} Szemantika

Absztrakt – konkrét szintaxis – szemantika

- **Absztrakt szintaxis**:

  „A nyelvünkben legyen egy „és” operátor, aminek két bemenő paramétere van és egy eredménye. Mindannyian bool típusúak.”

- **Konkrét szintaxis**:

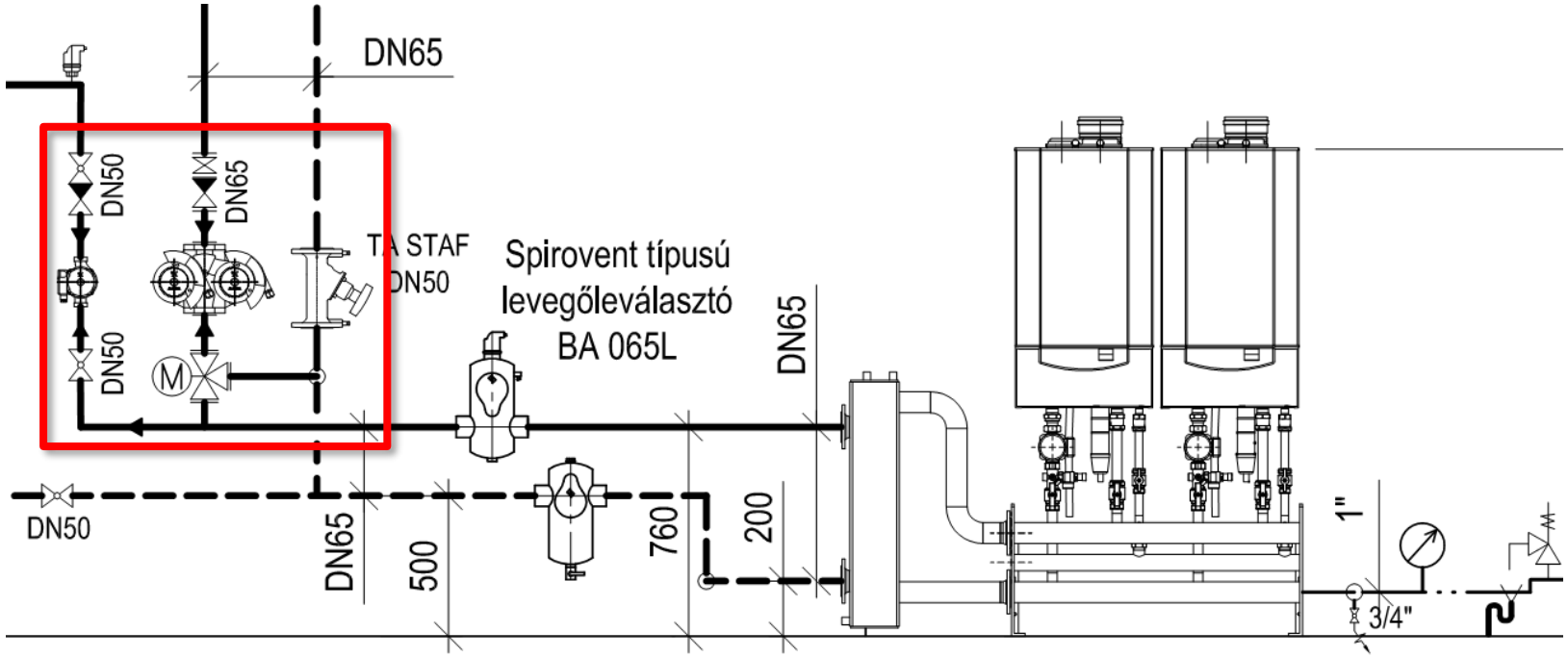
  „Az „és” operátort jelöljük „&&”-el”

- **Szemantika**:

  „Az „és” operátor pontosan akkor ad vissza igaz értéket, ha mindkét operandusa igaz értékű.

Konkrét szintaxis

- Hogyan néz ki?



Honeywell keverőcsap DN50 K_{VS} 40

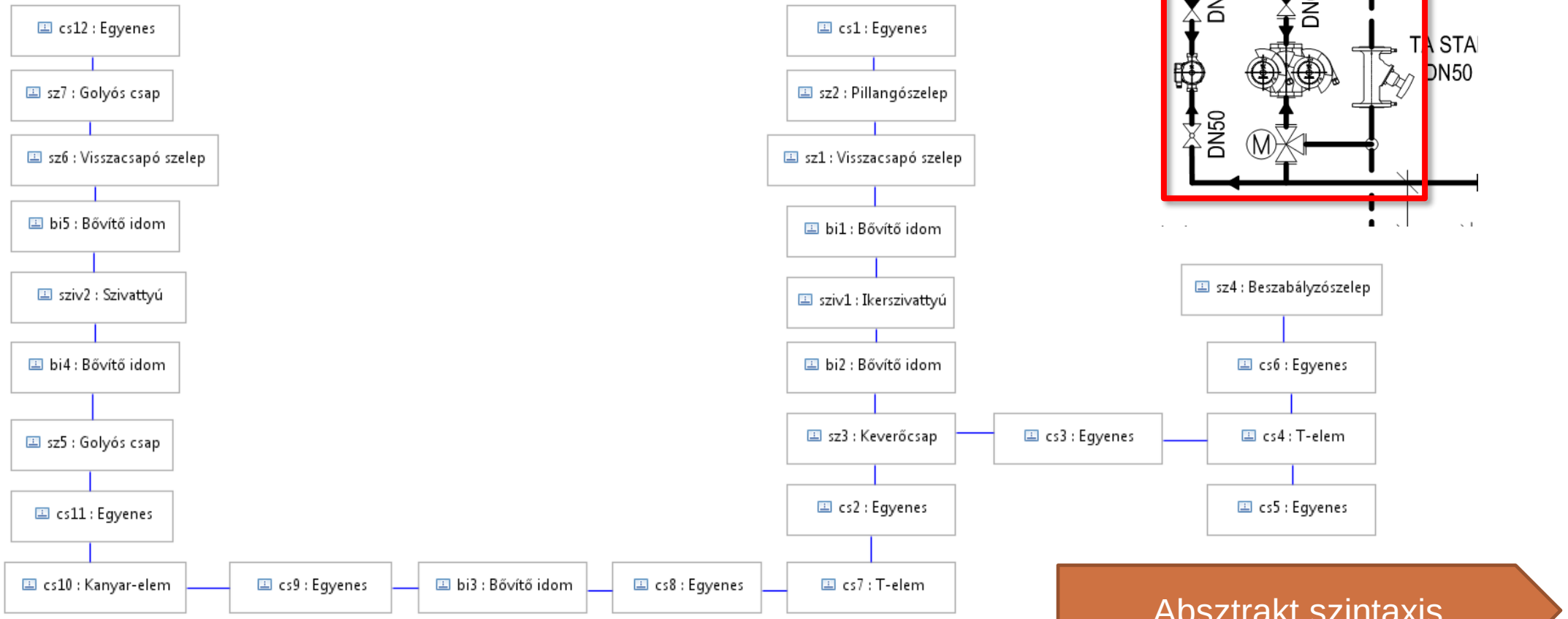
Spirovent típusú iszapleválasztó BE 065L

Remeha Quinta kaszkád rendszer hidrauliku

Konkrét szintaxis

Absztrakt szintaxis: modell

- Hogyan reprezentáljuk a modellt?



Absztrakt szintaxis

Absztrakt szintaxis: a modell leírása

- Hogyan írjuk le az absztrakt szintaxist?

Erről lesz most szó

Grafikus nyelvek modellezése

I. Grafikus nyelvek/modellek

II. Absztrakt szintaxis UML alapon

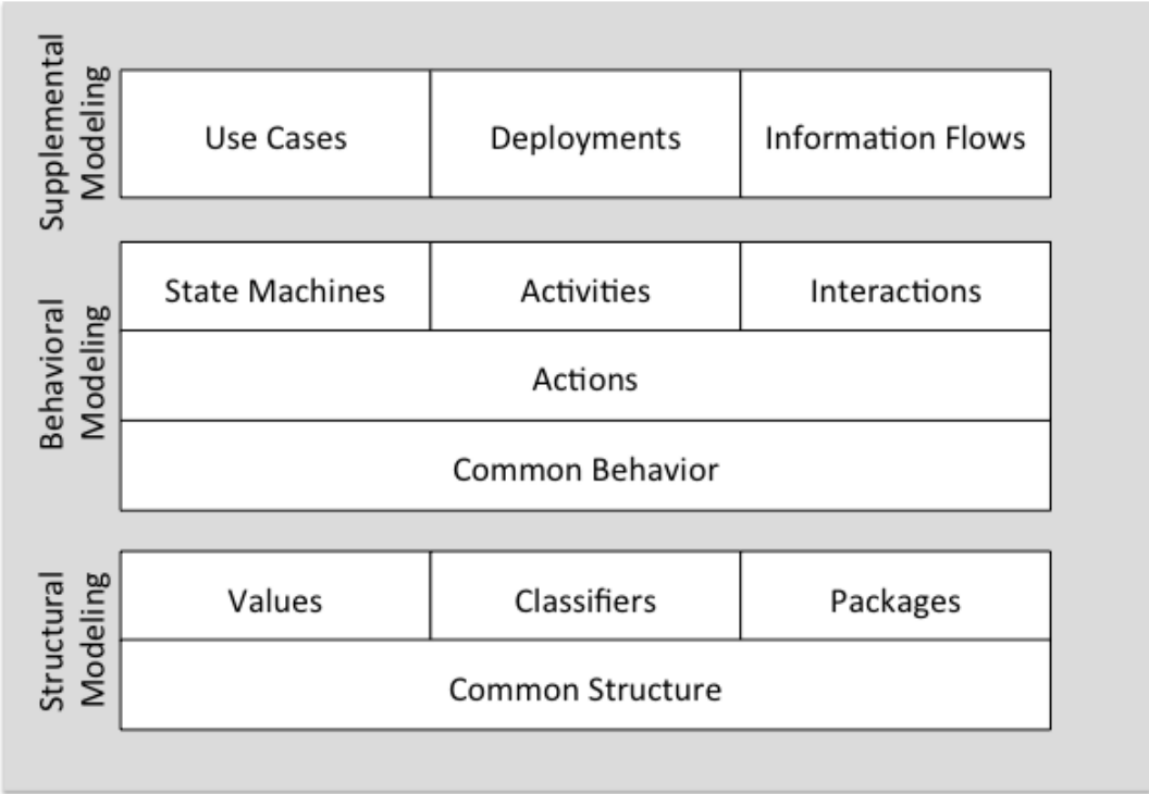
III. Blockly

IV. Metamodellezés

V. Kényszerek

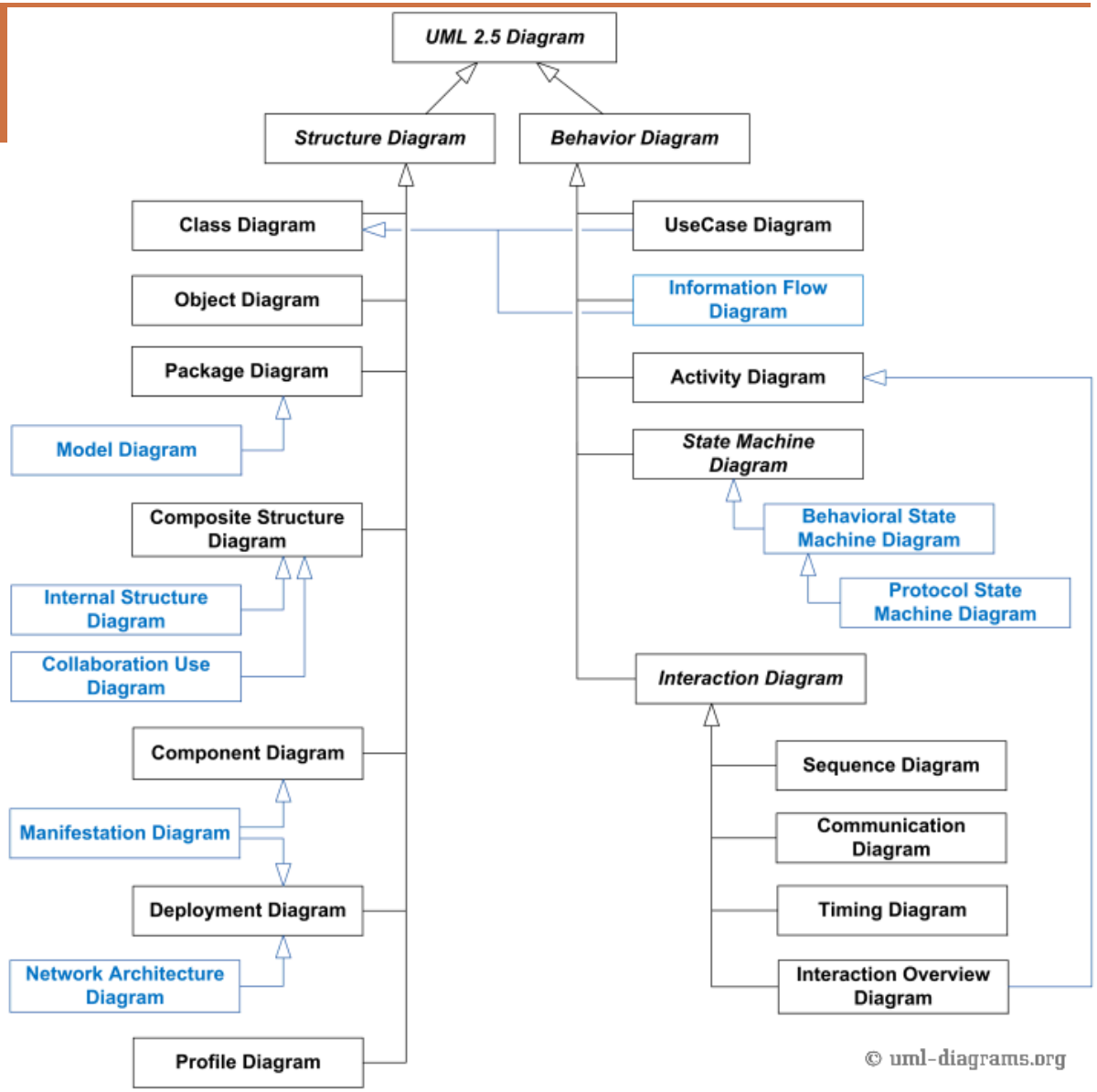
Az UML tulajdonságai

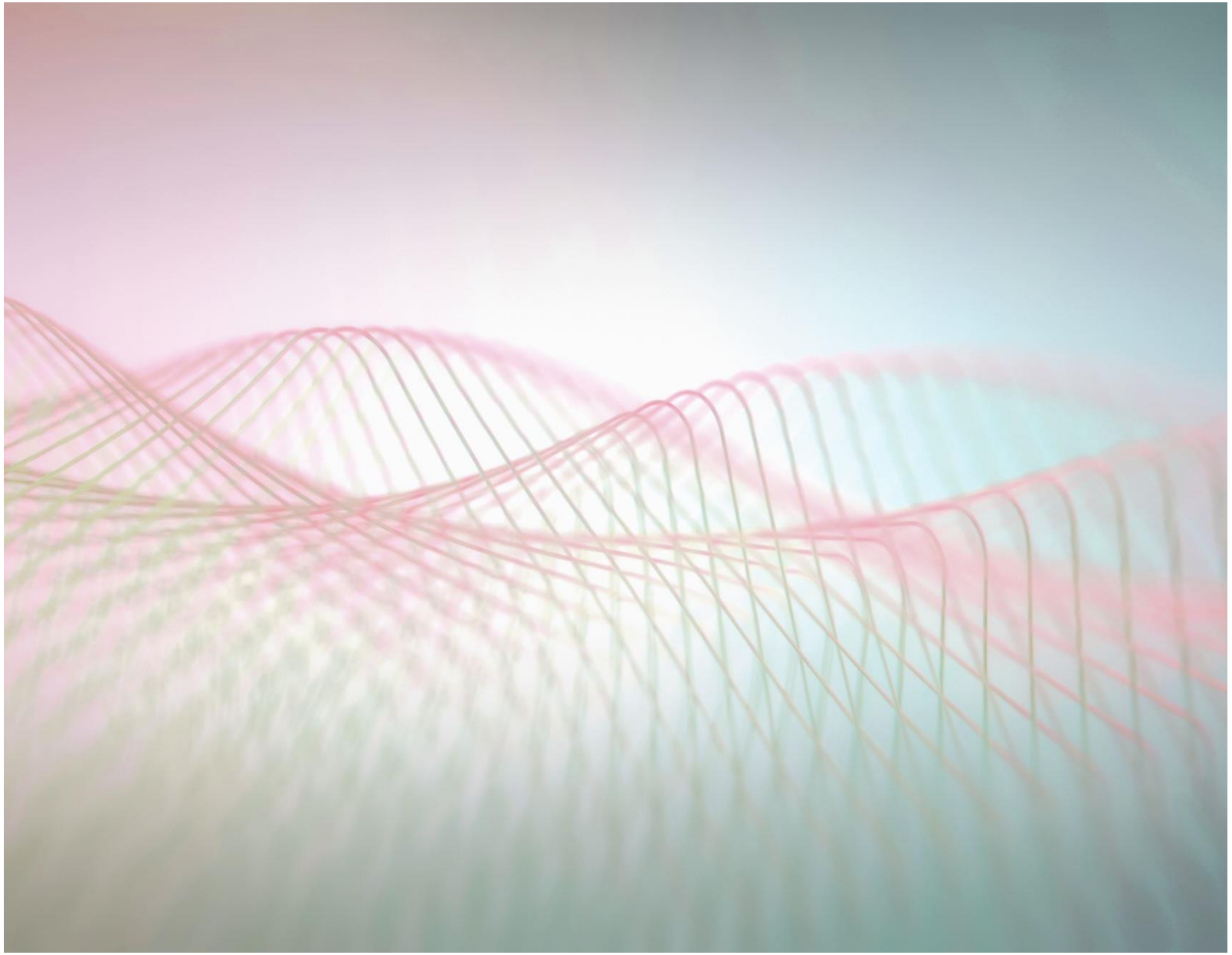
- Vizuális modellező nyelv
- Ötvözi az addig kialakult módszertanok előnyeit
- Programozási nyelvtől független
- Kiterjeszthető (főként: sztereotípiák)
- Szabványos (jelenleg 2.5.1 verzió)



UML nyelvek

- Az UML hatékony lehetőséget kínál a szoftvertervezés során a modellek használatára
  - 14 diagram típus (UML 2.5.1)
- Hogyan lehetne jobban testreszabni?





UML Profile

UML Profile

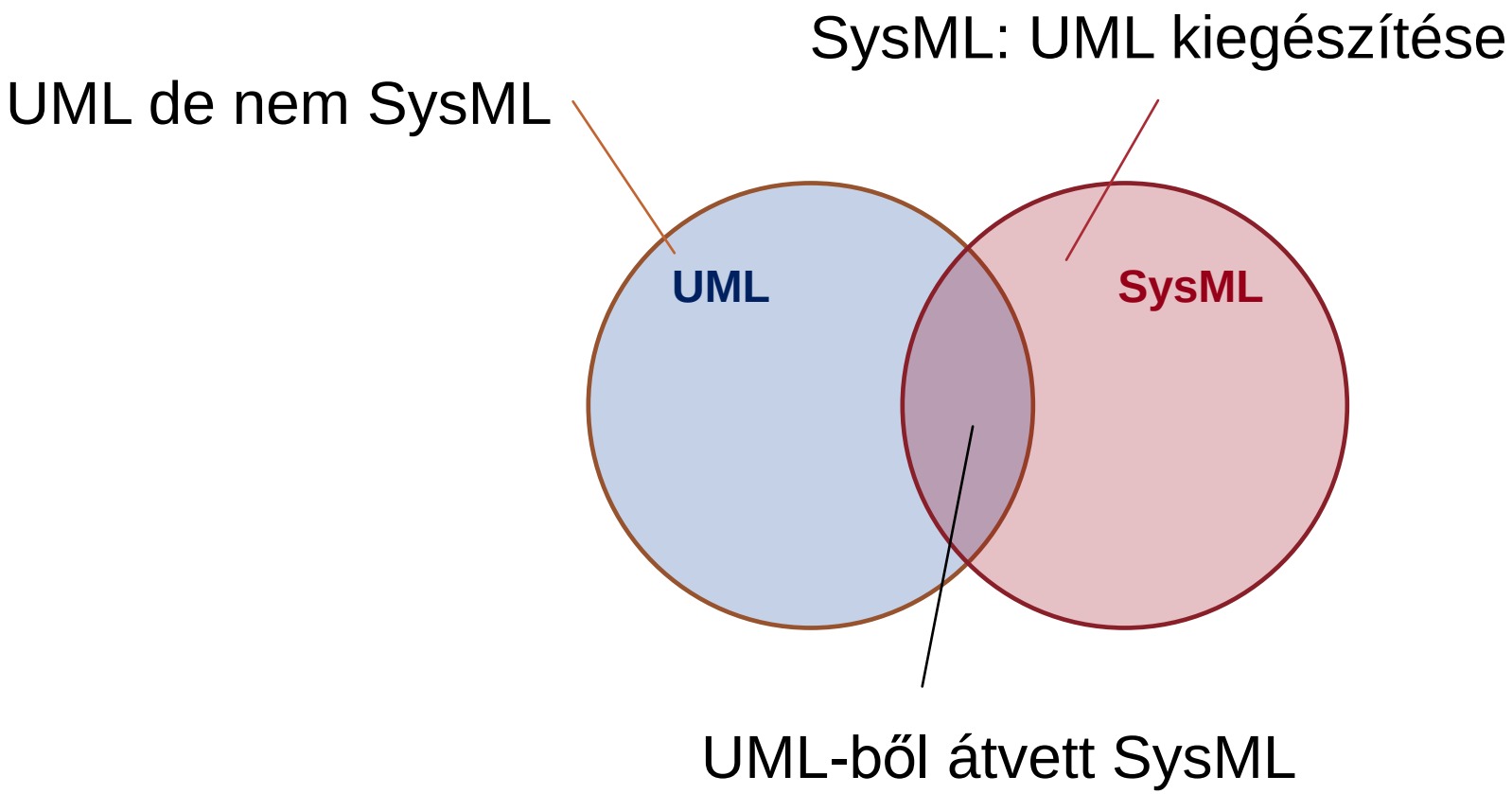
- Szükség lehet az UML által kínált általános nyelvek specializálására
  - Pl. Telekommunikációs modellek, beágyazott rendszerek
  - Szakterületi nyelveket hogyan lehet kifejezni UML-el?
- OMG megoldás: UML Profile
  - Az UML modellek egy részét jelöli ki, „jólformáltságot” biztosít a kijelölt tartományban
  - Általános elemek specializációját írja le, sztereotípiákkal (stereotype), címkékkel (tag) és kényszerekkel (constraint)
  - Nem mond ellent az eredeti specifikációnak

UML Profile példák

- Modeling and Analysis of Real Time and Embedded systems (MARTE)
- Service oriented architecture Modeling Language (SoaML)
- UML Profile for Advanced and Integrated Telecommunication Services (TelcoML)
- UML Testing Profile (UTP)
- UML Profile for Voice
- SysML

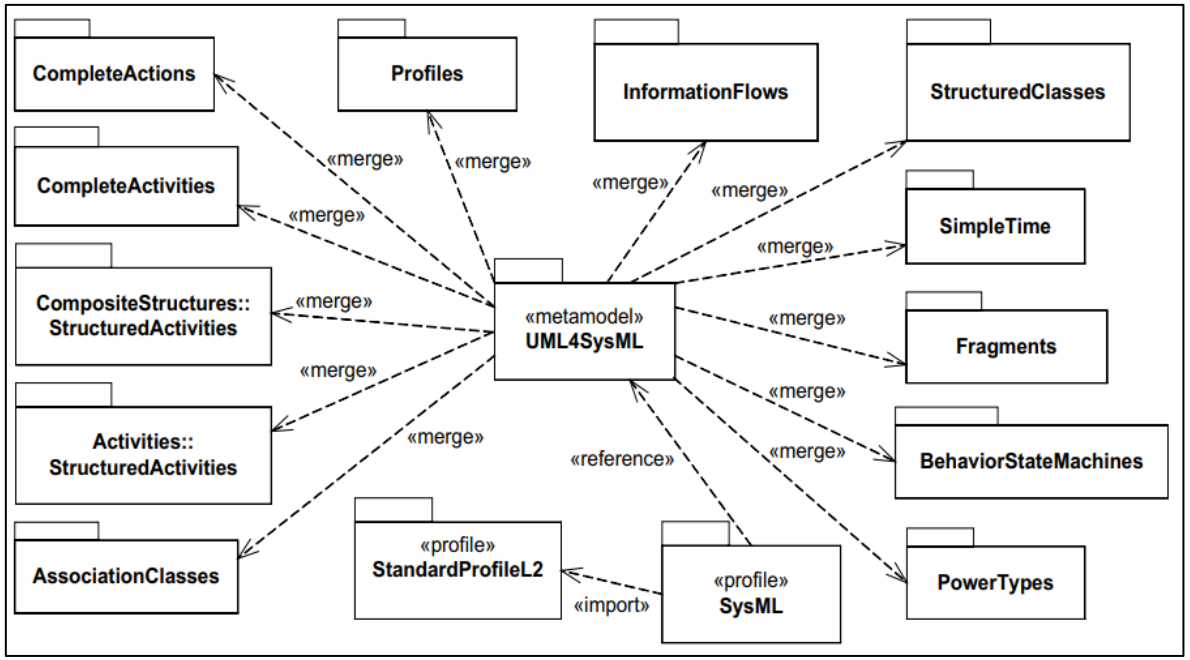
SysML

- Általános modellező, rendszertervezéshez
  - Specifikáció
  - Analízis
  - Tervezés
  - Verifikáció
  - Validáció



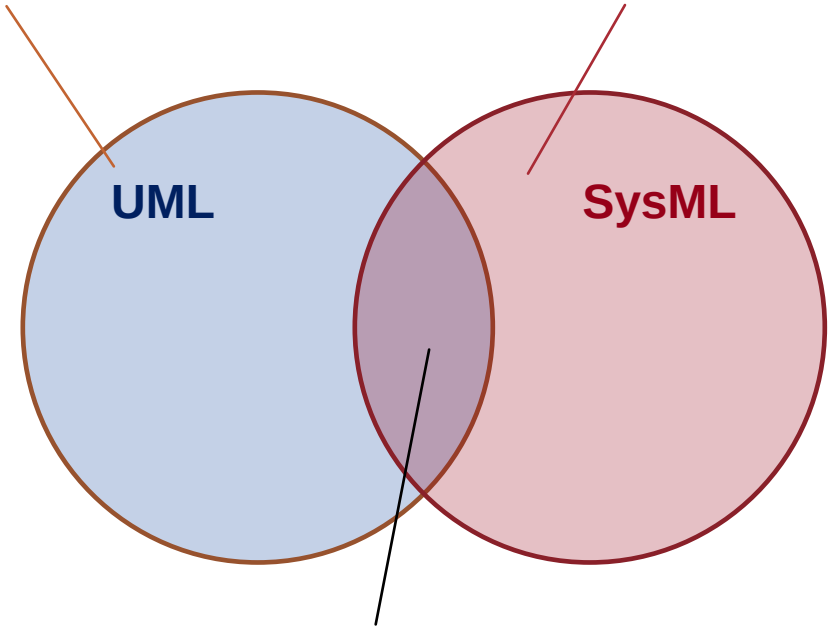
SysML

- Elhagy 7 UML diagramot



SysML: UML kiegészítése

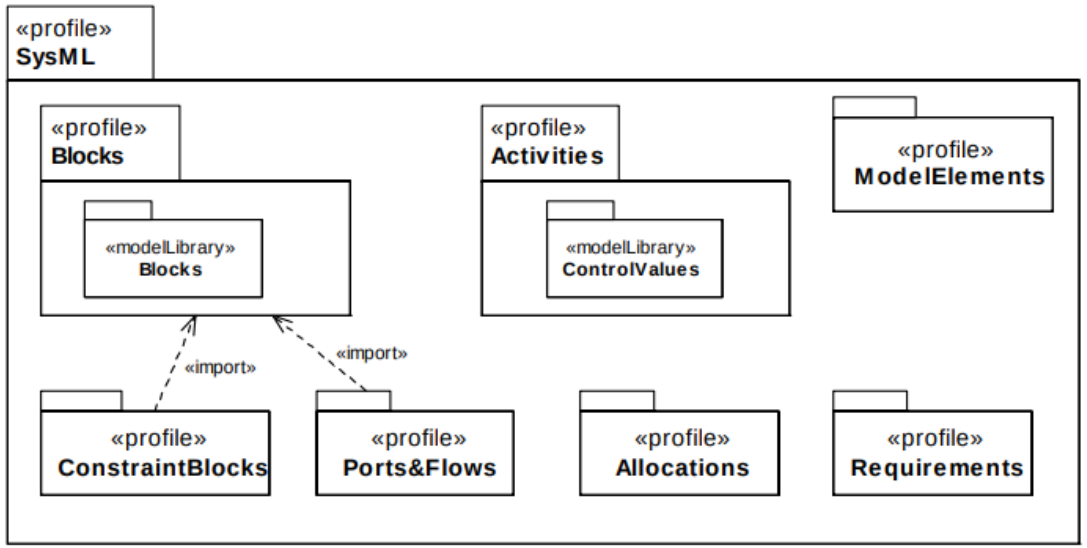
UML de nem SysML



UML-ből átvett SysML

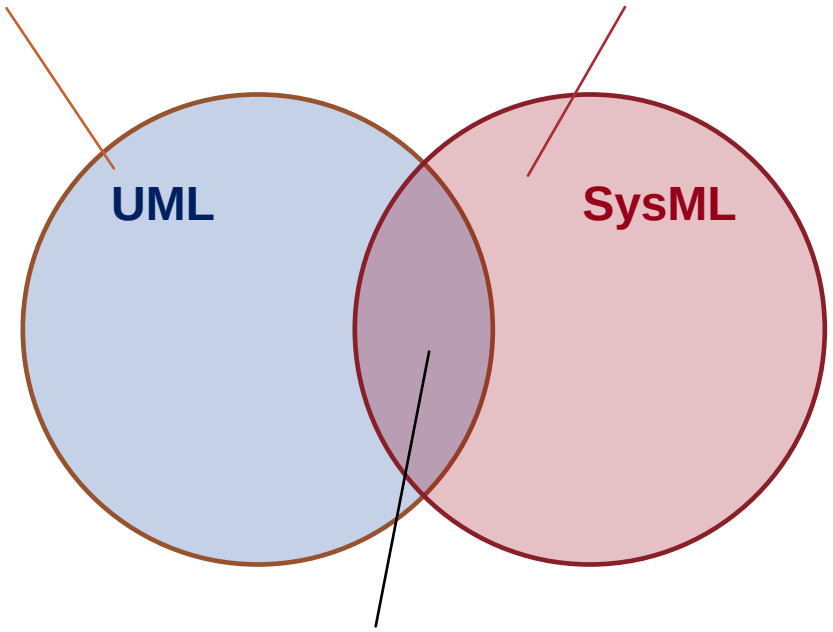
SysML

- Hozzáad új diagrammokat



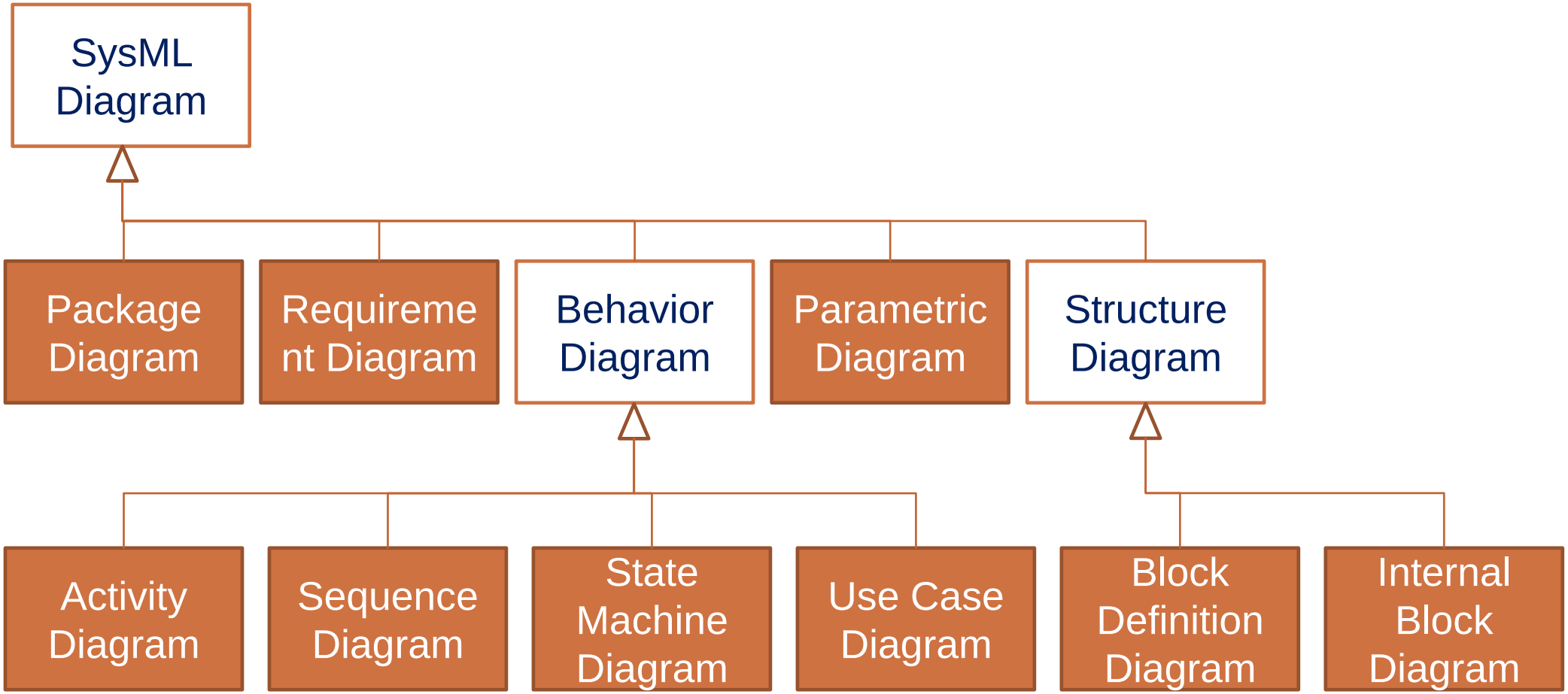
SysML: UML kiegészítése

UML de nem SysML



UML-ből átvett SysML

SysML diagram típusok



Grafikus nyelvek modellezése

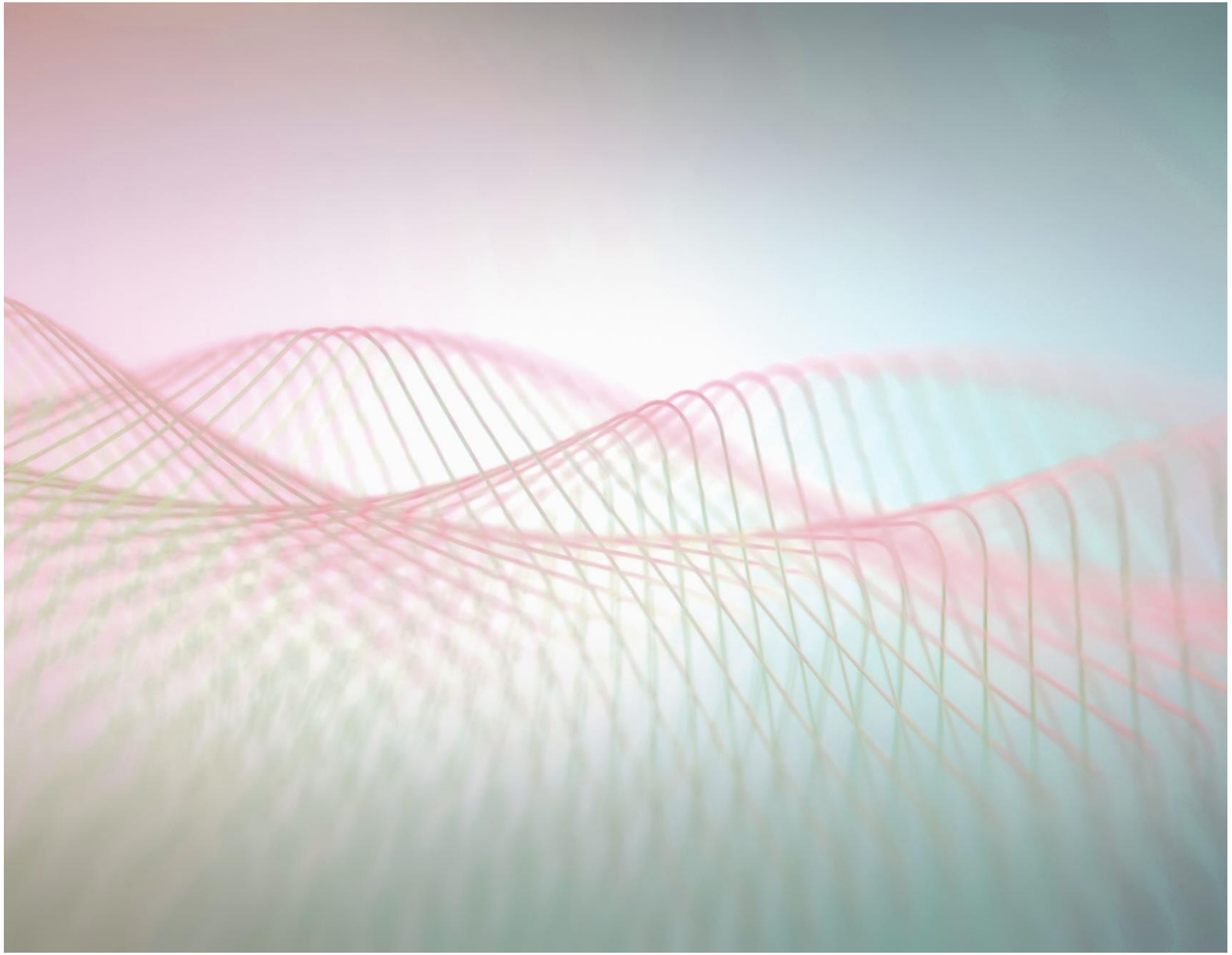
I. Grafikus nyelvek/modellek

II. Absztrakt szintaxis UML alapon

III. Blockly

IV. Metamodellezés

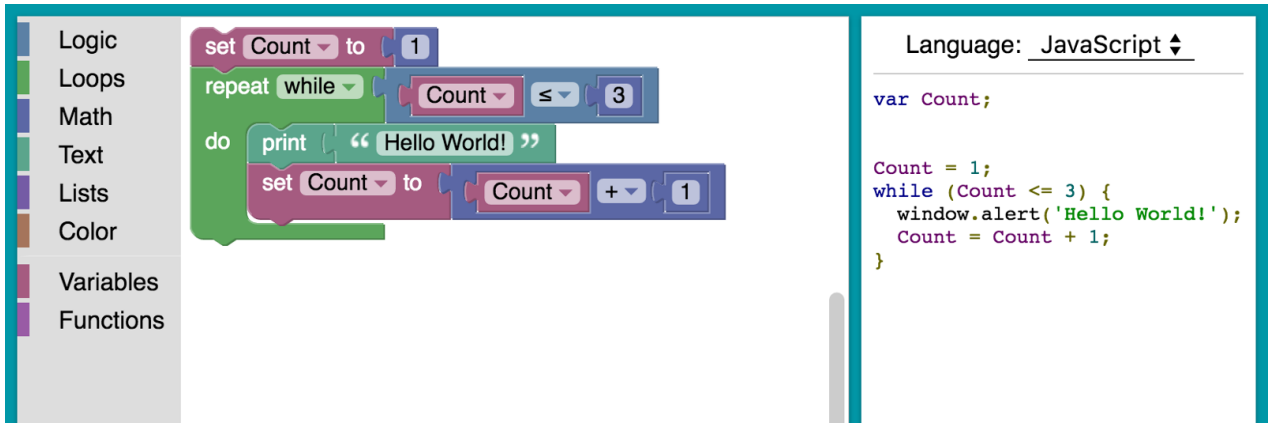
V. Kényszerek



Blockly

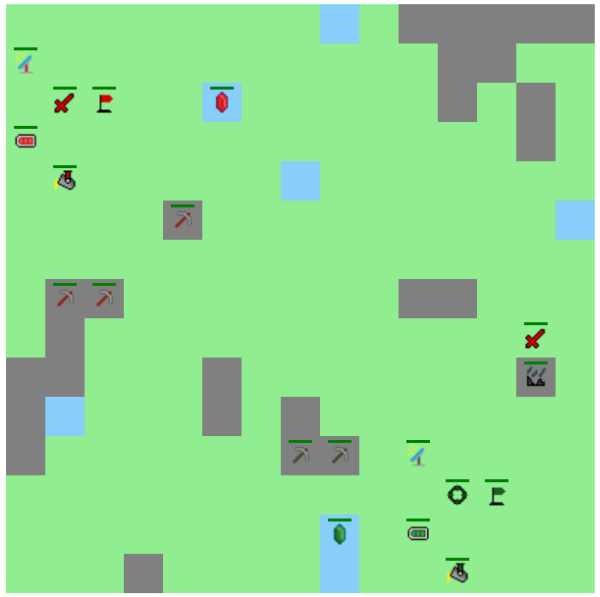
Blockly

- Egymásba ágyazott blokkok
- Egyszerű grafikus programmozási nyelv
- Általános, testreszabható
- Projekcióalapú szerkesztő
- Kódgenerálás sablon alapon

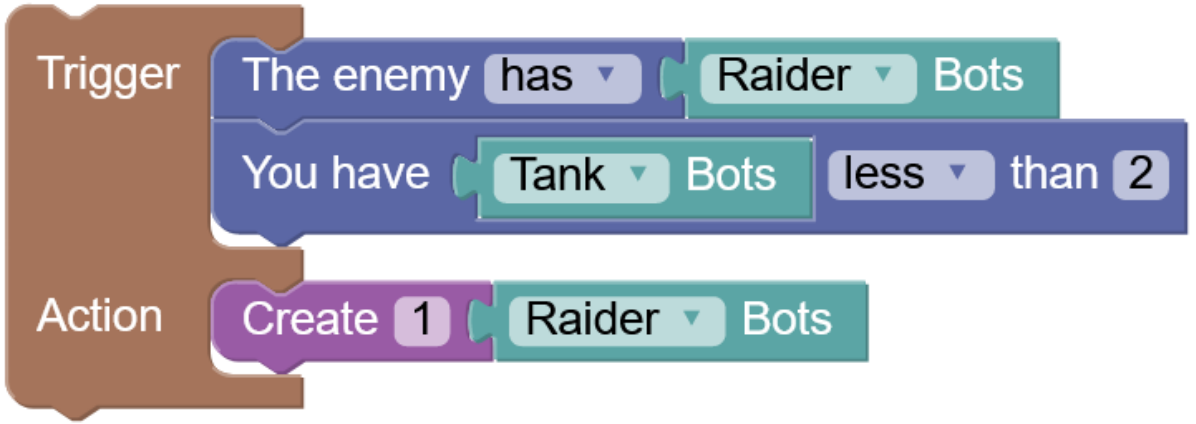


Blockly példa – Stratégai AI nyelv

Turn	Player	Steel	RoboSteel	Energy	Crystals	Energy Cores	Credits
13	Player	20	5	60	16	7	85
13	Server AI	10	5	65	6	7	120

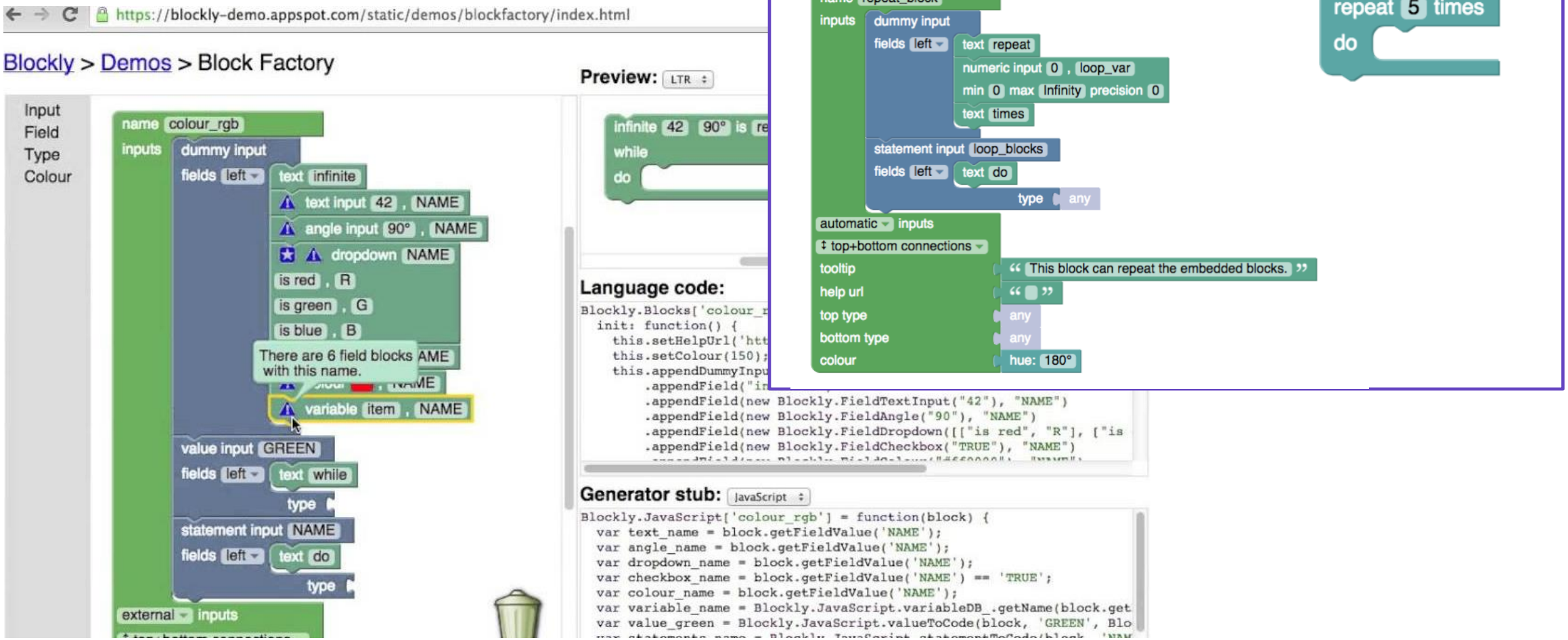


Player
artilleryBotUpgrades
attackBotUpgrades
raiderBotUpgrades
tankBotUpgrades
Server AI
artilleryBotUpgrades
attackBotUpgrades
raiderBotUpgrades
tankBotUpgrades

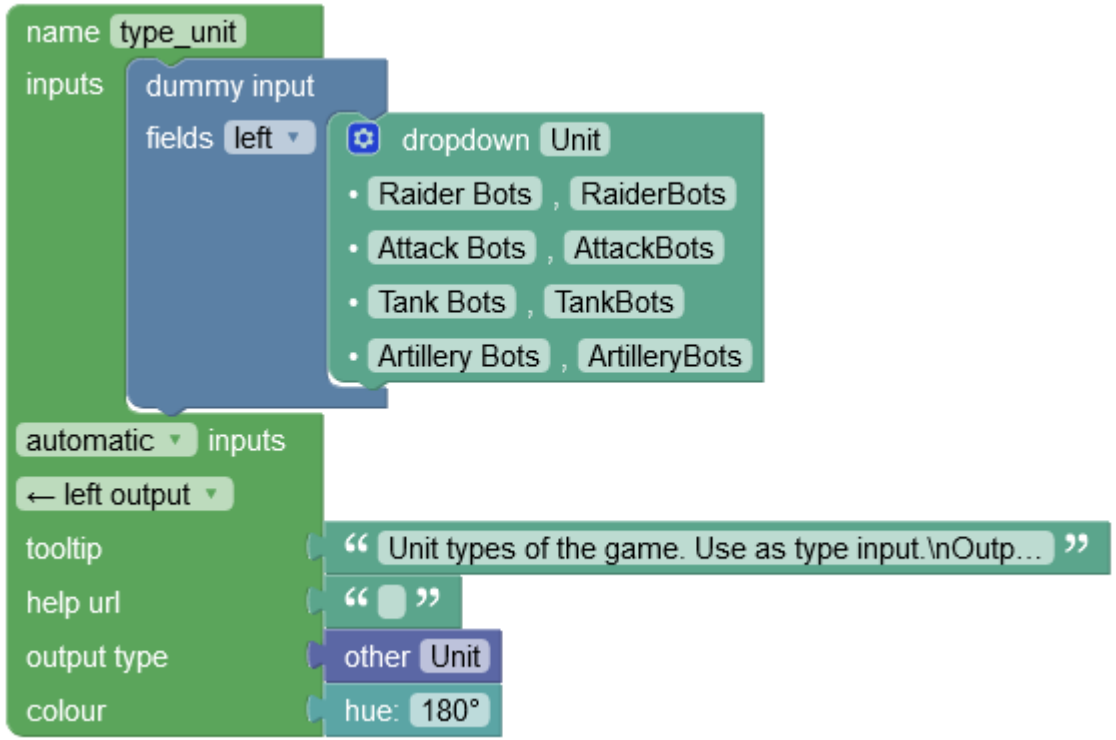


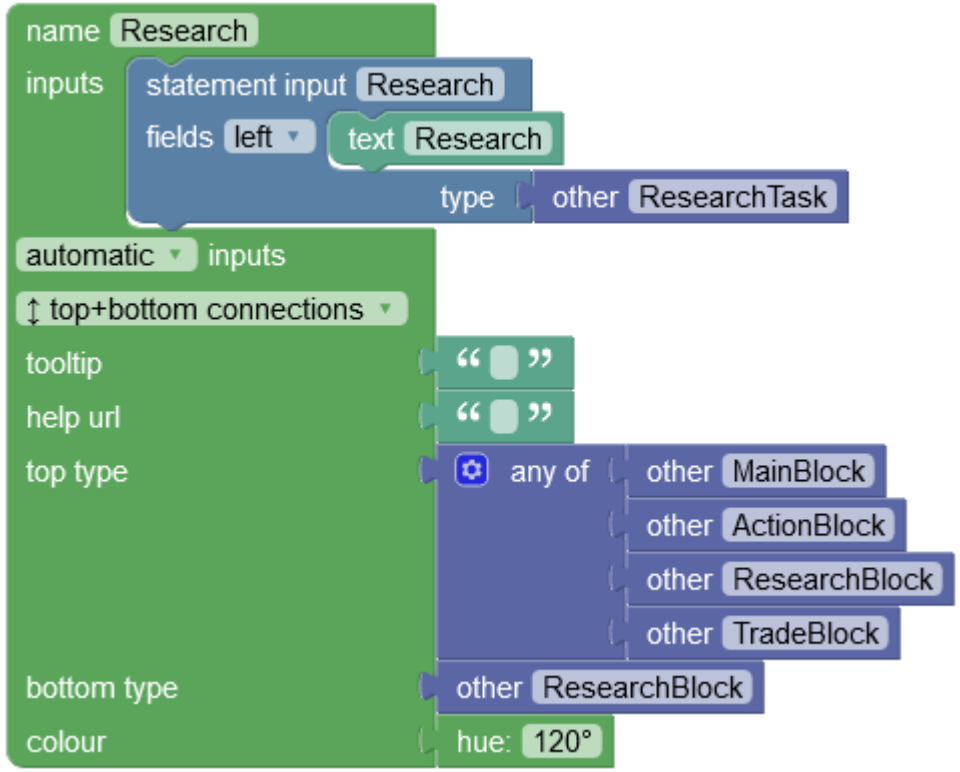
Blockly

- Block factory – saját blokkok

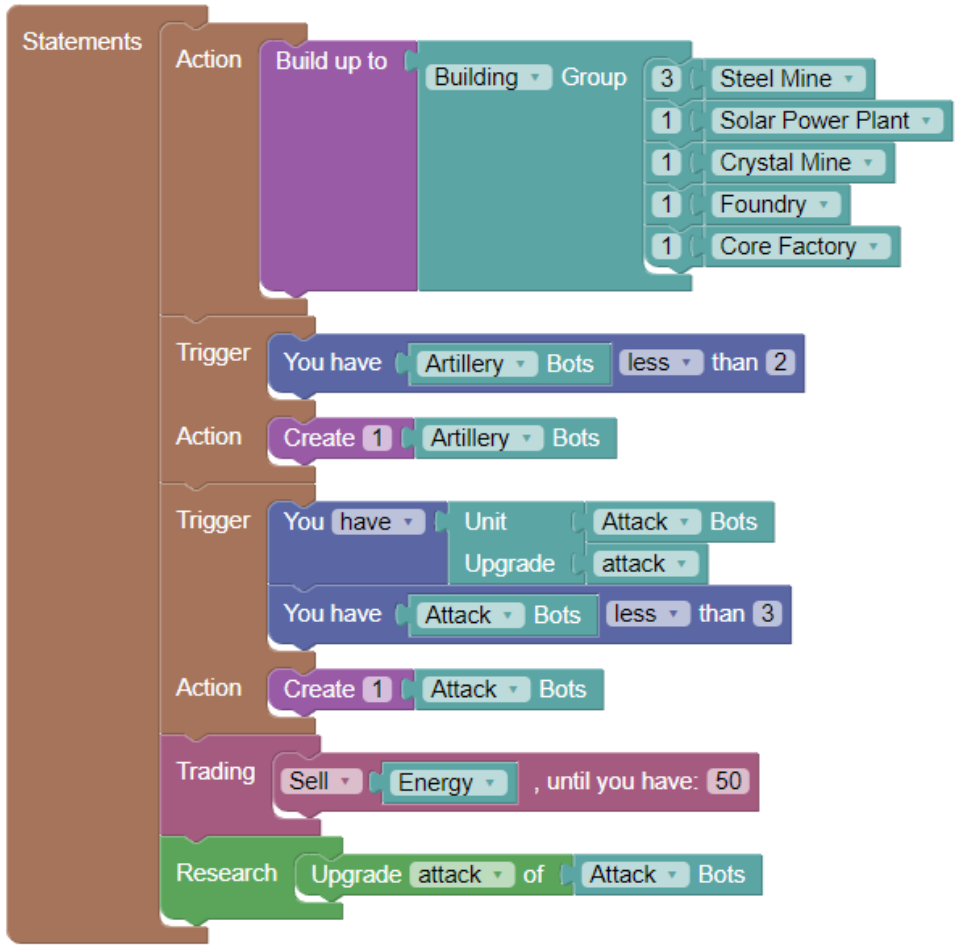


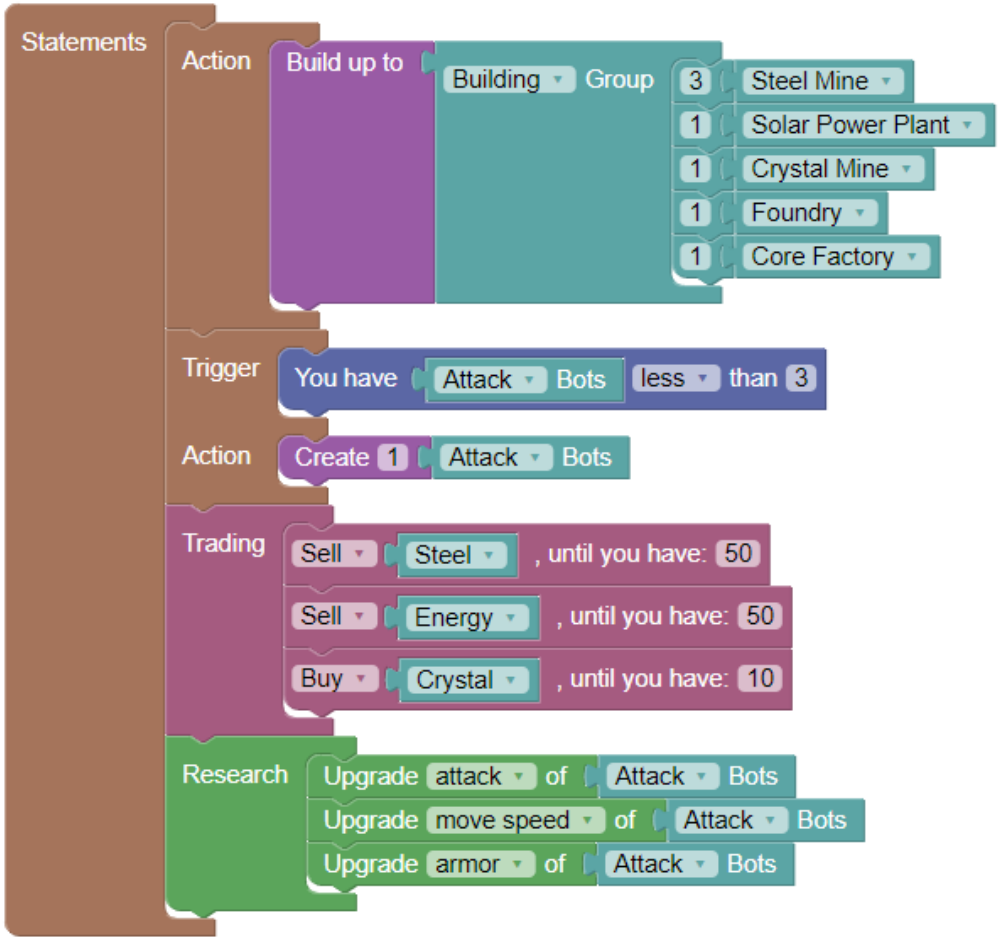
Blockly példa – Stratégai AI nyelv





Blockly példa – Stratégai AI nyelv





Grafikus nyelvek modellezése

I. Grafikus nyelvek/modellek

II. Absztrakt szintaxis UML alapon

III. Blockly

IV. Metamodellezés

V. Kényszerek

Metamodellezés

- Metamodell: definiálja a szakterületen jellemző alapvető modellezési elemeket, a köztük lévő kapcsolatokat és struktúra megkötéseket
  - > Modell elemek
  - > Kapcsolatok az elemek között
  - > Attribútumok (elemek és kapcsolatok attribútumai)
- Kiegészíthető: kényszerek és szabályszerűségek

Metamodellezés vs …

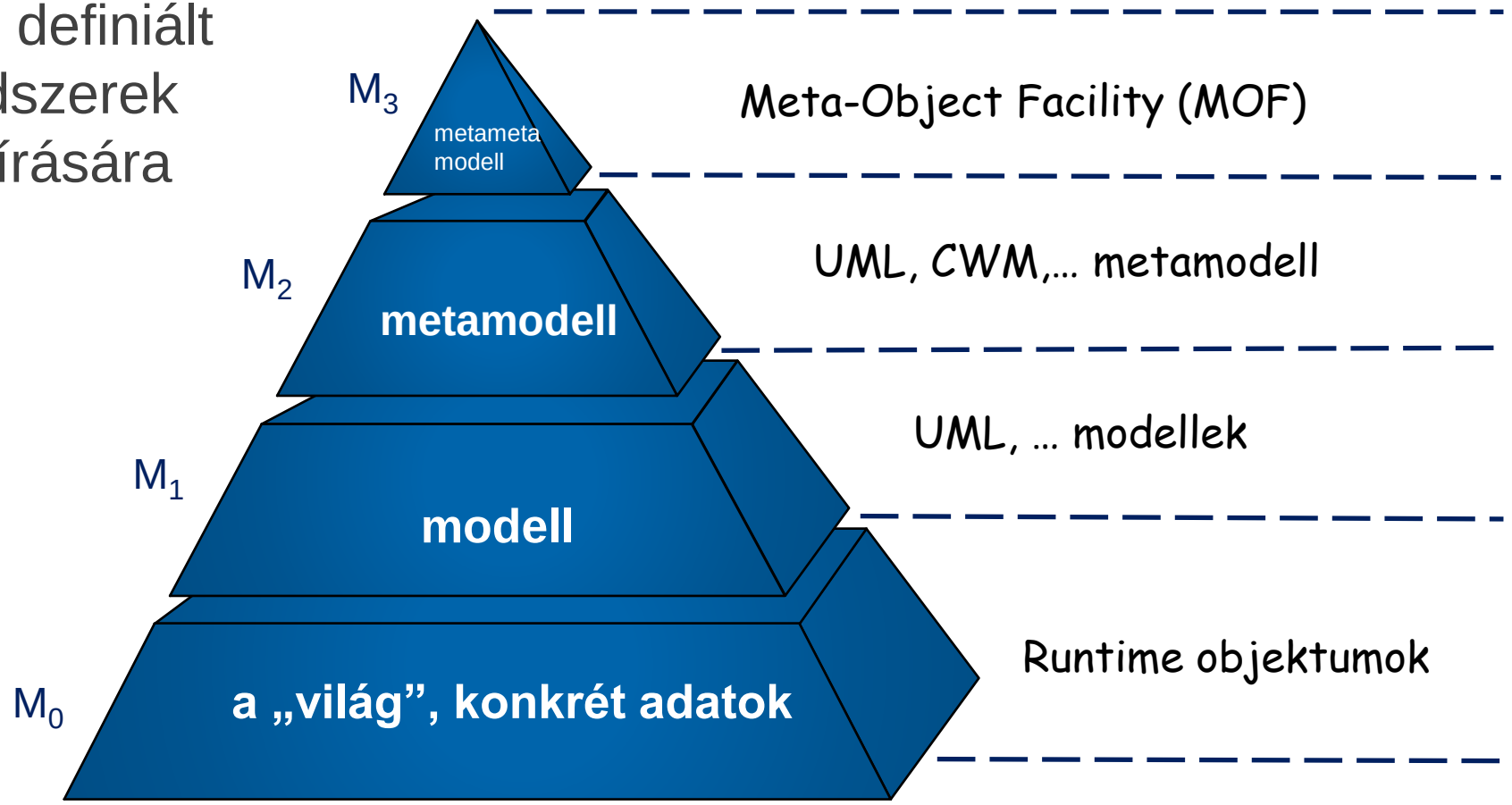
- UML profile
  - Kötik az UML szabályai
  - Kevésbé rugalmas/testreszabható
- Blockly
  - Korlátozott ellenőrzések a sablon mezőire
  - Kapcsolatok, ismételt információ megadása (függvényhívás)
- Ad-hoc saját megoldás
  - Nem formális/szabványos → automatikus feldolgozás nehézkes
  - Közös nyelv, átjárhatóság hiánya

A metamodellező nyelv

- A modellek absztrakt szintaxisát a metamodellek definiálják
- Mi adja meg a *metamodellek absztrakt szintaxisát*?
- Metamodellező nyelv
  - > Maga is egy (speciális) szakterületi nyelv
  - > Metamodellek definiálására használható
  - > Definiálja a modellezőnyelvek *lehetséges* absztrakt szintaxisát

Meta-Object Facility (MOF)

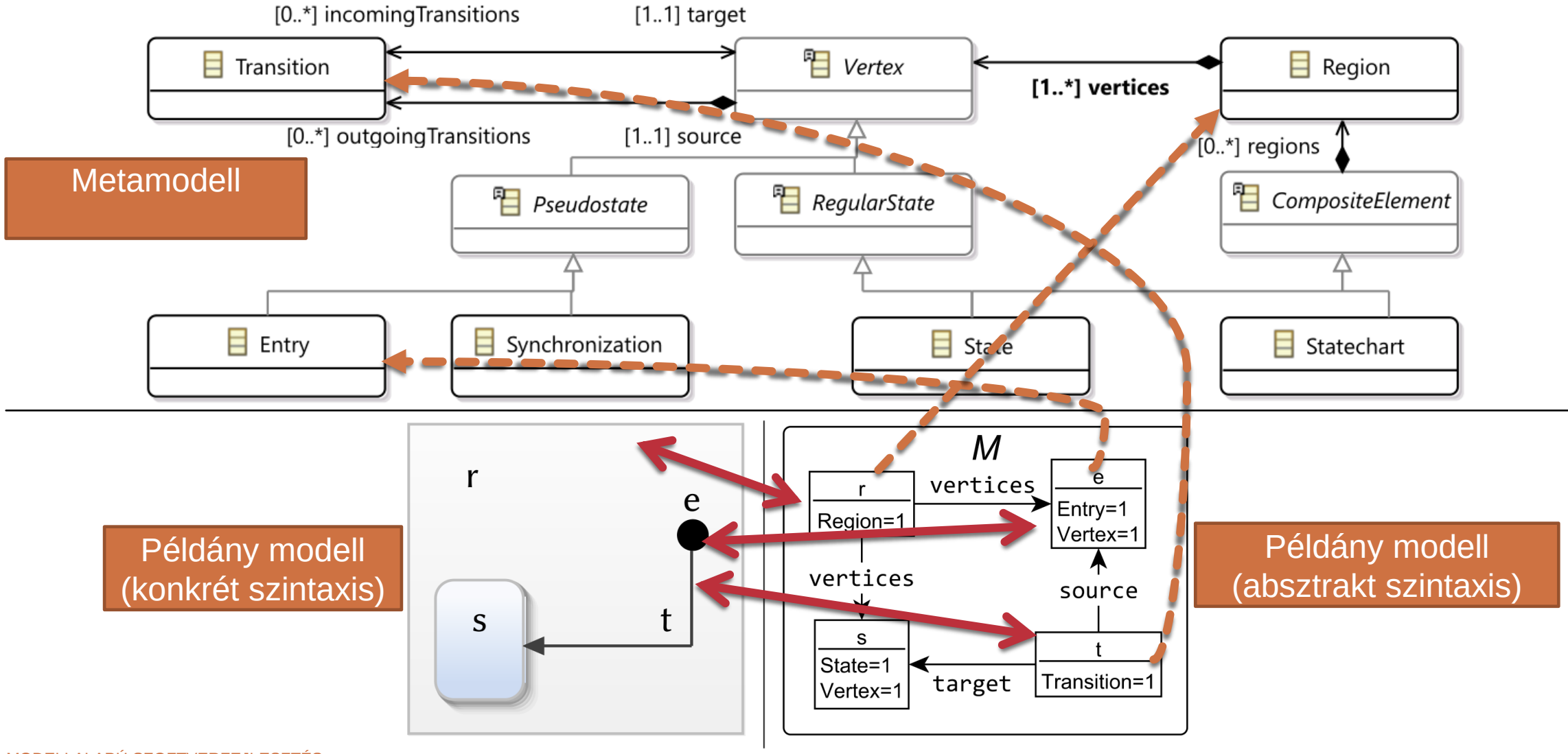
- Object Management Group (OMG) szabvány
- Szemantikailag jól definiált modell a típusrendszerek (metamodellek) leírására



MOF – 4 szint

Szint	Leírás	Példák	Ki írja?
Metameta-modell	Metamodellezési architektúra Metamodellező nyelvek létrehozására.	MetaClass, MetaAttribute, MetaOperation	Kutatók, MOF készítők
Metamodell	Metametamodell példánya. Szakterületi és modellezési nyelvek létrehozására.	Class, Attribute, Operation	Szabvány készítők, Szakterületi nyelv készítők
Modell	A metamodell példánya, konkrét szakterületi modellek.	Ember osztály, Csoport osztály	Szakemberek, felhasználók
Objektumok, adatok	A modell példánya, konkrét, adatokkal kitöltött modell	„Kovács József”, ”Csiga futóklub”	Runtime létrejövő objektumok

Yakindu példa

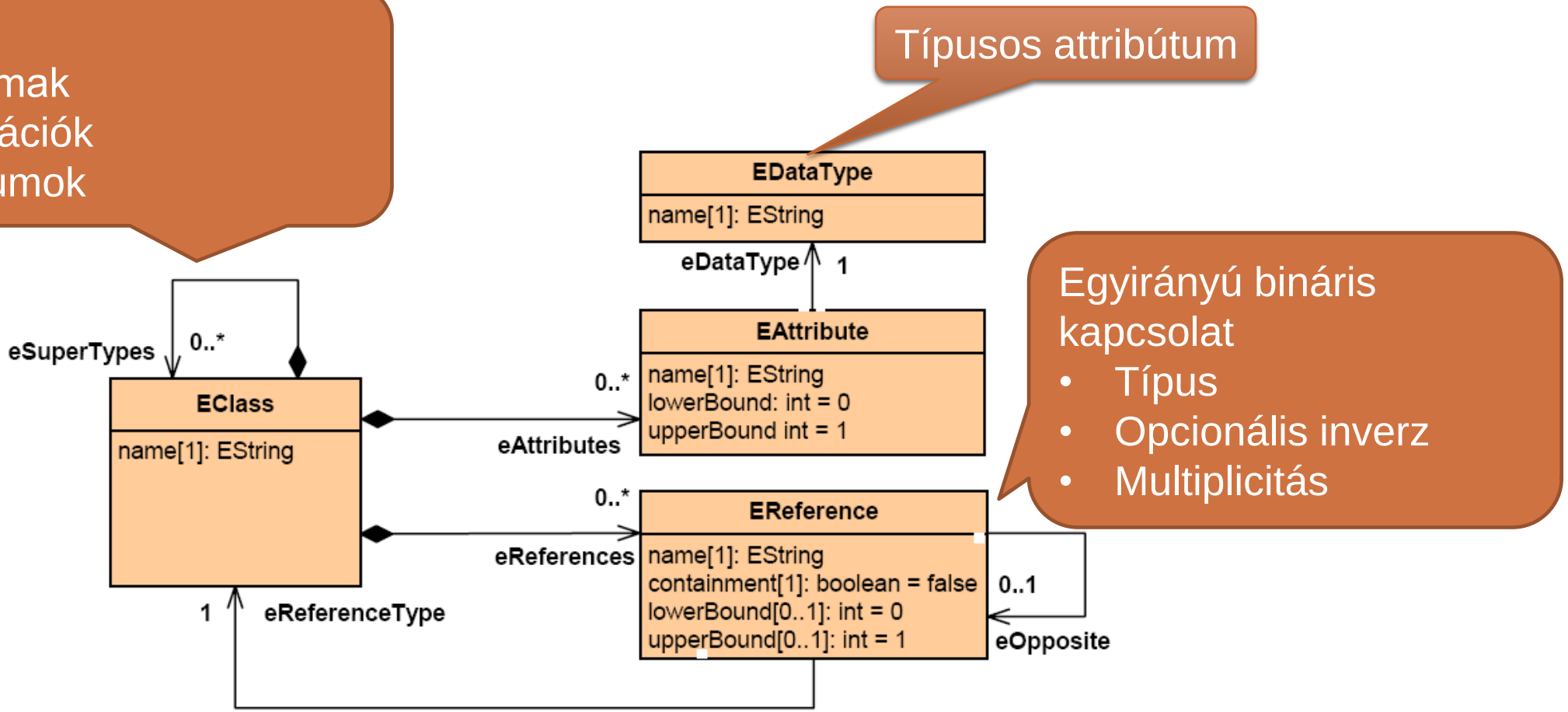


MOF variánsok

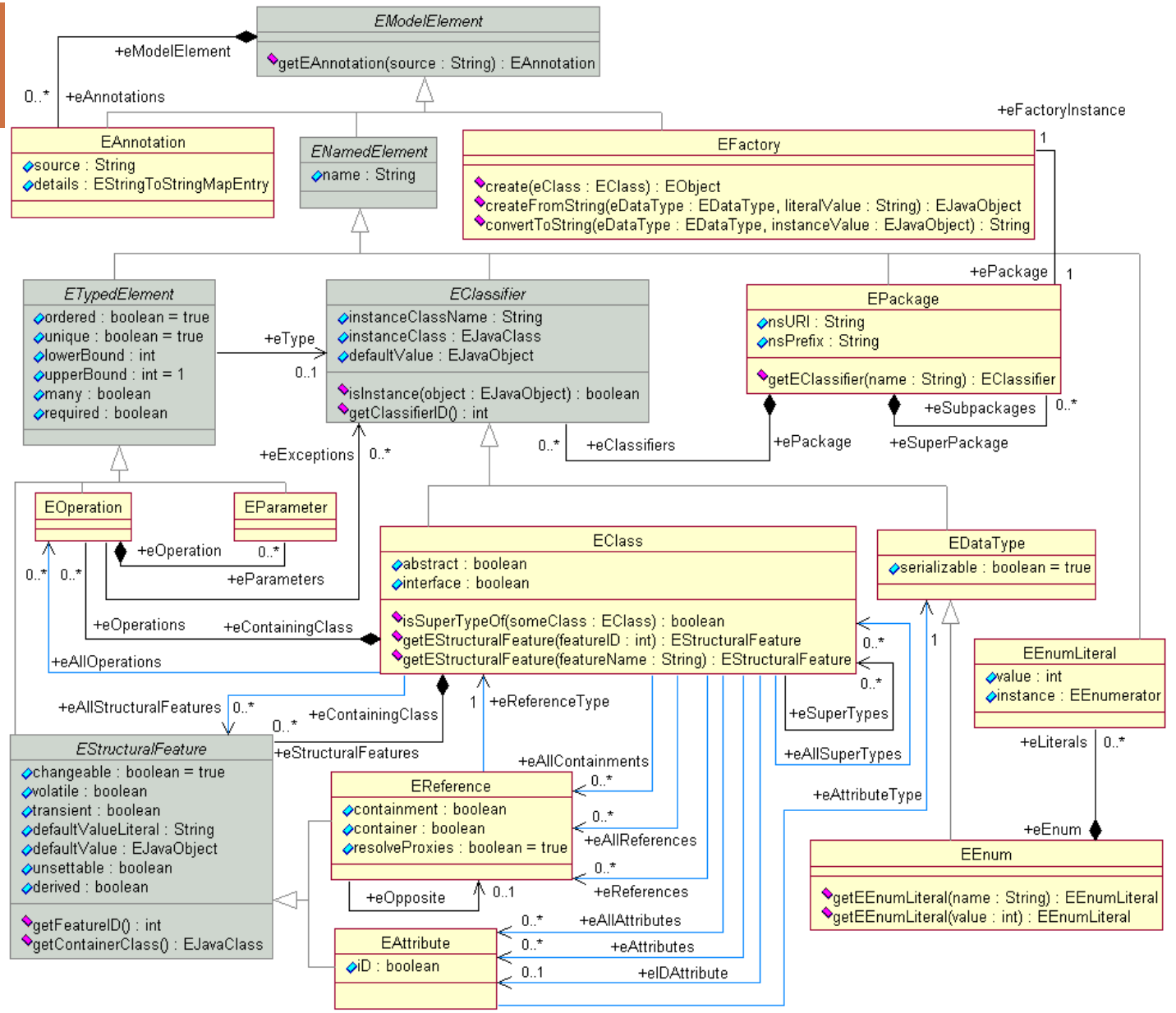
- EMOF (Essential MOF)
  - Alap (OO-hoz és XML-hez kapcsolódó) funkciók
  - Cél: MOF modellek leképzése JMI és XMI formára
  - Egyszerű metamodellekhez
  - Támogatja a kibővítéseket
  - ECore
- CMOF (Complete MOF)
  - „Teljes verzió” (UML 2.0 kiegészítések)
  - UML 2.0 jellegű nyelvek definiálásához

Ecore alapok

Fogalom
- ősfogalmak
- asszociációk
- attribútumok

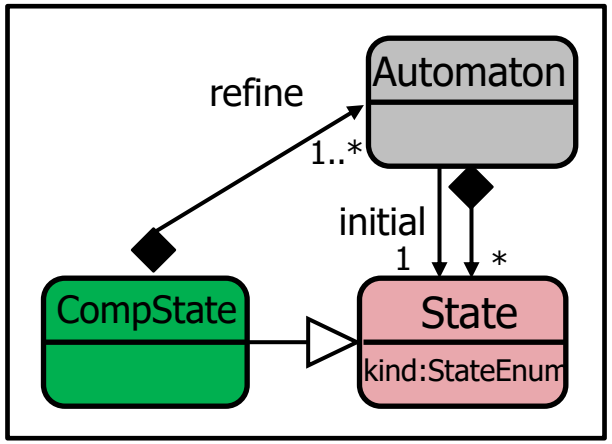


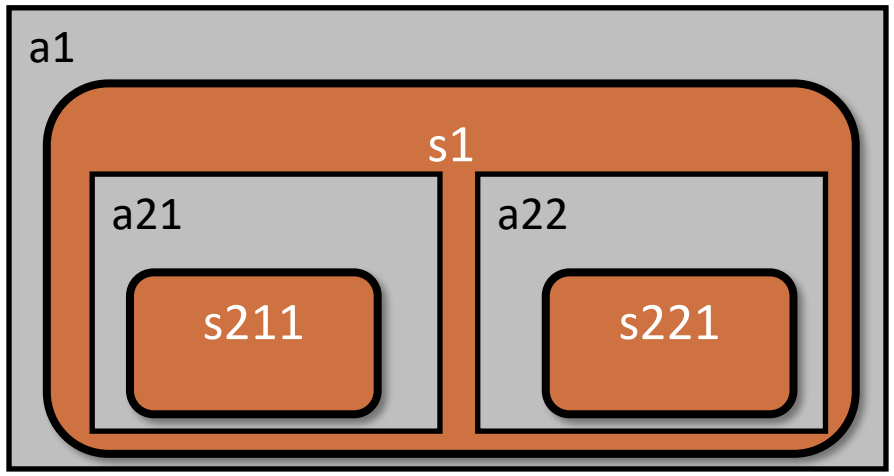
Ecore – EMOF*

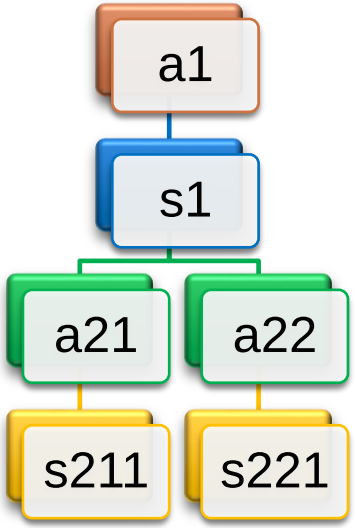

EModelElement
getEAnnotation(source : String) : EAnnotation
+eModelElement
0..*
+eAnnotations
EAnnotation
source : String
details : EStringToStringMapEntry
ENamedElement
name : String
EFactory
create(eClass : EClass) : EObject
createFromString(eDataType : EDataType, literalValue : String) : EJavaObject
convertToString(eDataType : EDataType, instanceValue : EJavaObject) : String
+eFactoryInstance
1
ETypedElement
ordered : boolean = true
unique : boolean = true
lowerBound : int
upperBound : int = 1
many : boolean
required : boolean
+eType
0..1
EClassifier
instanceClassName : String
instanceClass : EJavaClass
defaultValue : EJavaObject
isInstance(object : EJavaObject) : boolean
getClassifierID() : int
+ePackage
1
EPackage
nsURI : String
nsPrefix : String
getEClassifier(name : String) : EClassifier
+eSubpackages
0..*
+eSuperPackage
+eClassifiers
0..*
+ePackage
+eExceptions
0..*
EOperation
EParameter
+eOperation
0..*
+eParameters
0..*
0..*
+eOperations
+eContainingClass
+eAllOperations
EClass
abstract : boolean
interface : boolean
isSuperTypeOf(someClass : EClass) : boolean
getEStructuralFeature(featureID : int) : EStructuralFeature
getEStructuralFeature(featureName : String) : EStructuralFeature
EDataType
serializable : boolean = true
EEnumLiteral
value : int
instance : EEnumerator
1
+eLiterals
0..*
+eEnum
EEnum
getEEnumLiteral(name : String) : EEnumLiteral
getEEnumLiteral(value : int) : EEnumLiteral
+eAllStructuralFeatures
0..*
0..*
+eContainingClass
+eStructuralFeatures
1
+eReferenceType
0..*
+eSuperTypes
+eAllSuperTypes
+eAttributeType
EStructuralFeature
changeable : boolean = true
volatile : boolean
transient : boolean
defaultValueLiteral : String
defaultValue : EJavaObject
unsettable : boolean
derived : boolean
getFeatureID() : int
getContainerClass() : EJavaClass
+eAllContainments
0..*
EReference
containment : boolean
container : boolean
resolveProxies : boolean = true
0..*
+eAllReferences
0..*
+eReferences
+eOpposite
0..1
0..*
+eAllAttributes
EAttribute
iD : boolean
0..*
+eAttributes
0..1
+eIDAttribute

Tartalmazás (containment)

- Minden modellelemnek van pontosan egy tartalmazója
- A tartalmazás reláció is modellezett
  - > Speciális él a metaelemek közt
  - > Multiplicitás szabályokkal
- Körkörös tartalmazás nem megengedett
  - > De a metamodellben a típusok szintjén lehetséges







Öröklés és példányosítás

1. Morzsi egy puli
2. A puli egy kutya
3. A kutya egy állat
4. A puli egy (kutya)fajta
5. A kutya egy faj

✓ 1+2 = Morzsi egy kutya

✓ 1+2+3 = Morzsi egy állat

! 1+4 = Morzsi nem egy (kutya)fajta

! 2+5 = A puli nem egy faj

- Az öröklés (SupertypeOf): részhalmaz, tranzitív
- A példányosítás (InstanceOf): ‘sablon’ kitöltése, nem tranzitív

Metaadatok sorosítása - XMI

- Modellek átadása heterogén környezetben
- **X**ML **M**etadata **I**nterchange (XMI)
  - > OMG szabvány
  - > Része a MOF → XML leképezés


```
<Classifiers xsi:type="Class" name="Department">
   <StructuralFeatures xsi:type="Attribute" name="id" Type="String"/>
   <StructuralFeatures xsi:type="Reference" name="member"
upperBound="-1"
       Type="#//Employee" containment="true"/>
 </Classifiers>
 <Classifiers xsi:type="Class" name="Employee">
   <StructuralFeatures xsi:type="Attribute" name="name" Type="String"/>
 </Classifiers>
```

Grafikus nyelvek modellezése

I. Grafikus nyelvek/modellek

II. Absztrakt szintaxis UML alapon

III. Blockly

IV. Metamodellezés

V. Kényszerek

Miből áll a vizuális szakterületi nyelv?

- Mire van szükség egy vizuális szakterületi nyelv definiálásakor?

> Nyelv struktúrája
> Kiegészítő kényszerek

} Absztrakt szintaxis

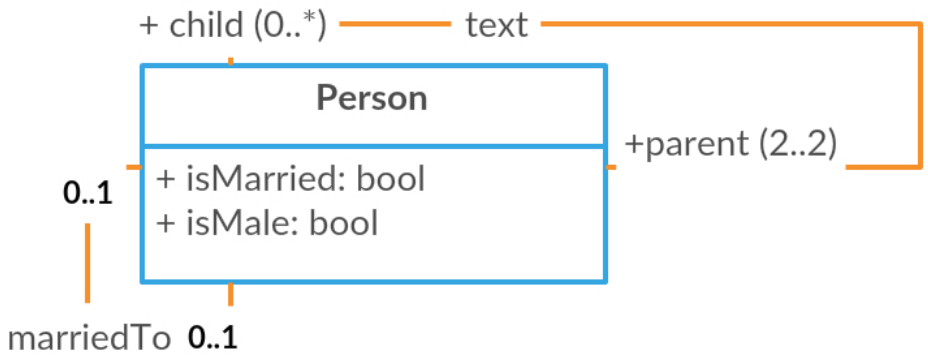
> Megjelenítés

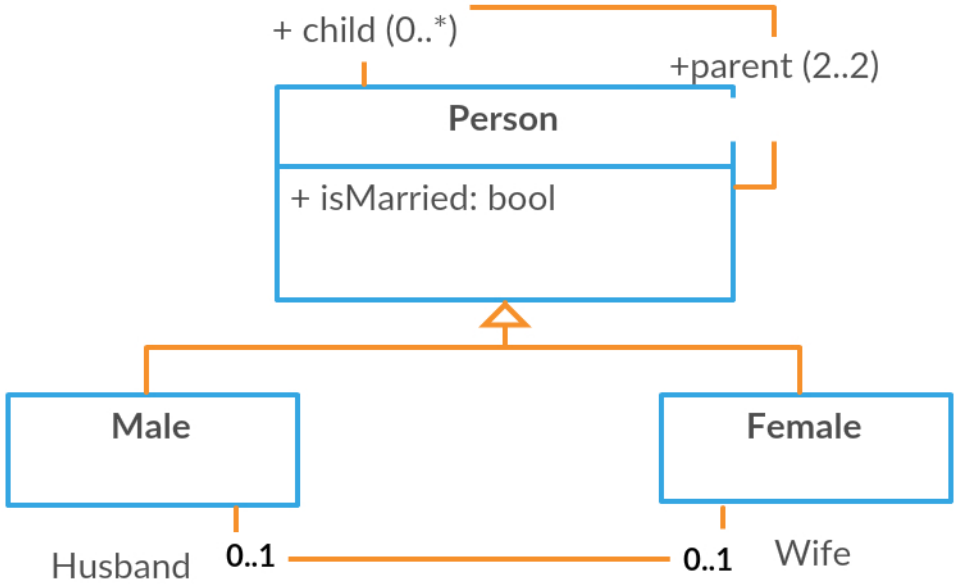
} Konkrét szintaxis

> Struktúra jelentése

} Szemantika

Hogyan fejezzem ki?

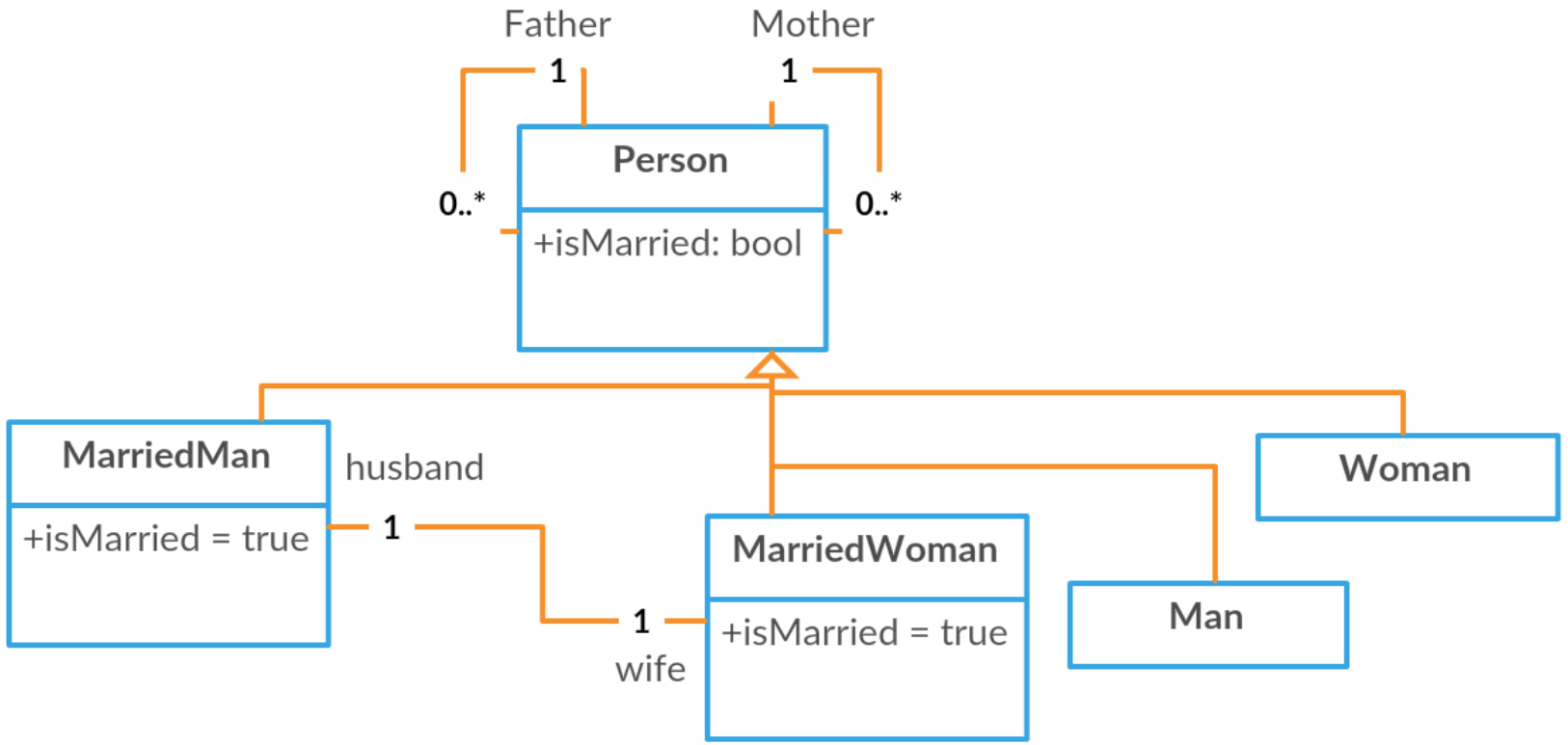




- Egy embernek egy apja és egy anyja van
- Ha valakinek van felesége, akkor az házas

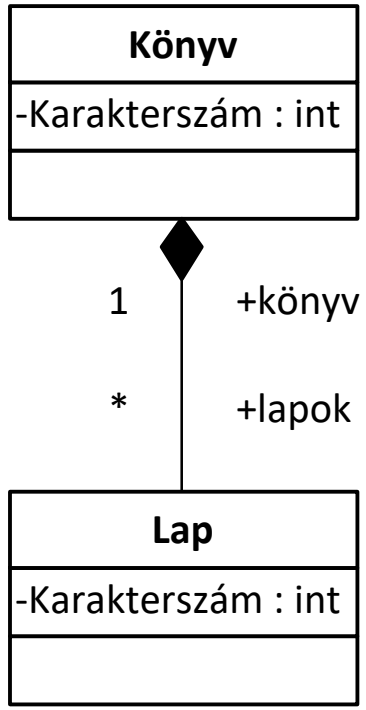
Hogyan fejezzem ki?

- Tartalmazza az előző két megkötést, de nem használható



Kényszerek - Motiváció

- Probléma: bonyolult összefüggések leírása
  - > Két érték egymástól függése
    „Egy könyv pontosan annyi betűt tartalmaz, mint amennyit a lapjai összesen”
  - > Összetett korlátozások
    „Egy osztályba járhatnak fiúk és lányok, tetszőleges felosztásban, de összesen max. 35-en lehetnek”



Kényszerek

- Mi a kényszer?
  - > *A kényszer egy megszorítás a metamodell egy vagy több elemén, értékén.*
- A struktúrális megadás kényelmes, bonyolult kényszereket azonban körülményes struktúraként megfogalmazni
  - > A hiányosság **nem** azért áll elő, mert a metamodellező nyelvet rosszul konstruáltuk meg!
- Hogyan legyen megadható egy kényszer?

Köszönöm a figyelmet!