



Modellalapú szoftverfejlesztés

VIII. 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



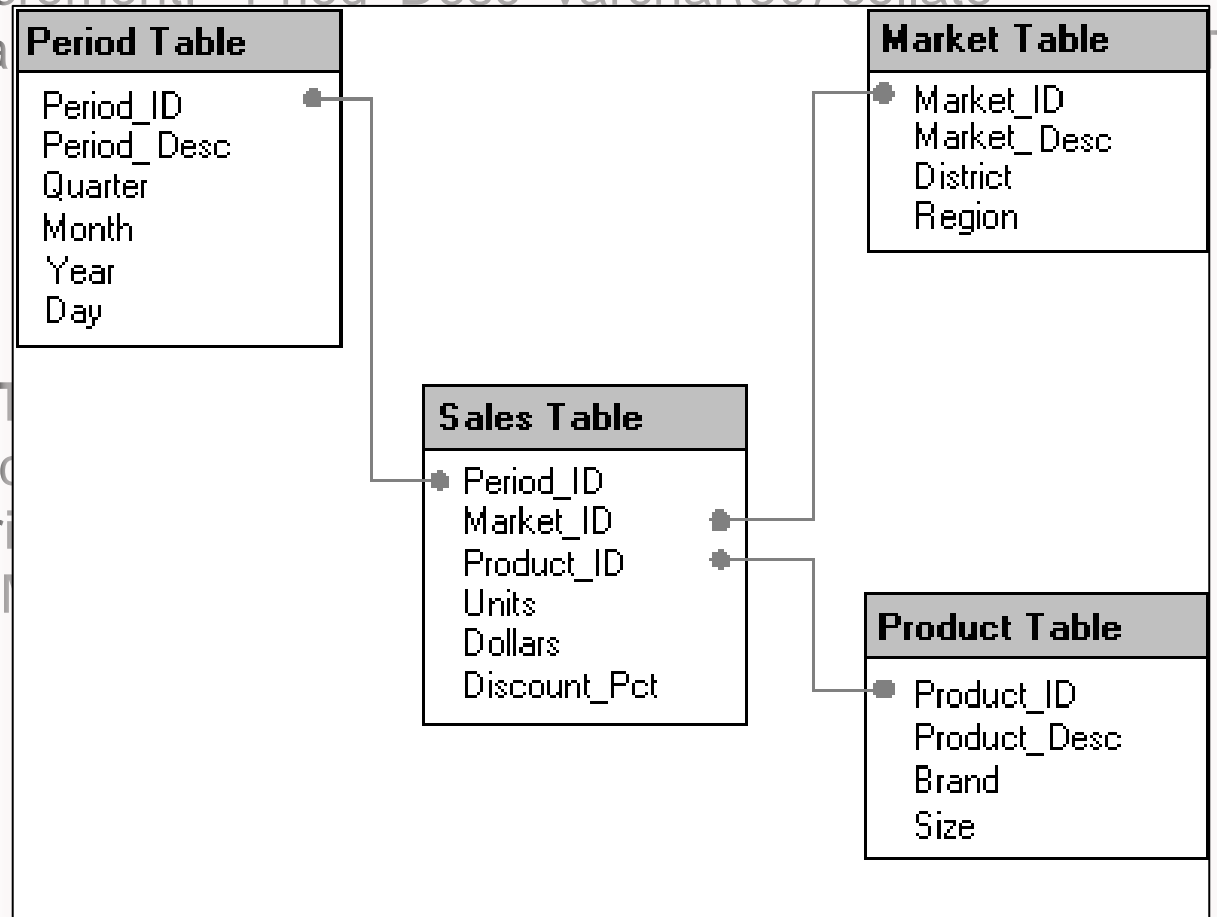
Adatbázis tervezés – SQL

CREATE TABLE IF NOT EXISTS `Period Table`

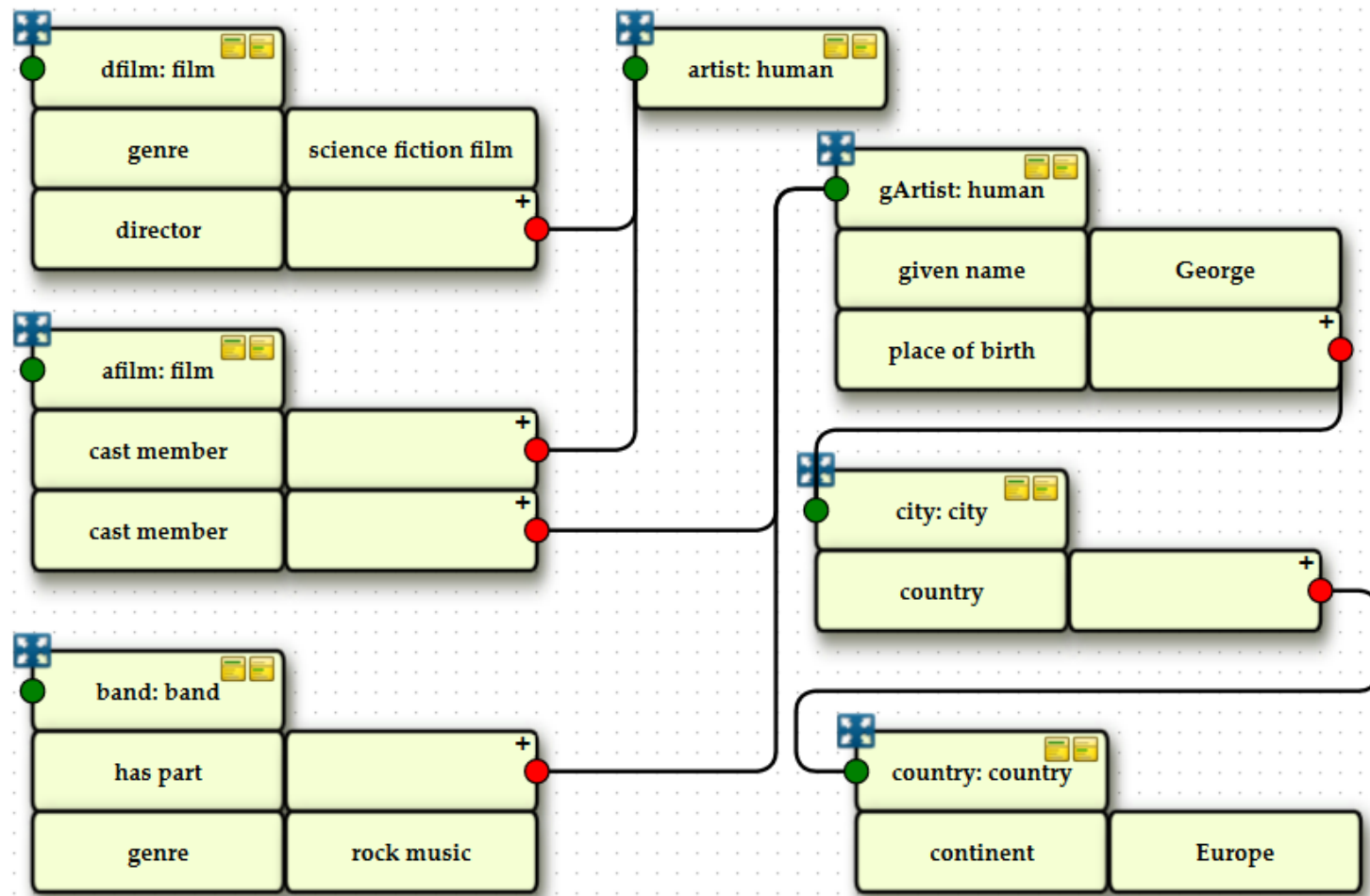
```
( `Period_ID` int(11) NOT NULL auto_increment, `Period_Desc` varchar(50) collate latin2_hungarian_ci default NULL, `Quarter` mediumint(9) NOT NULL, `Year` mediumint(9) NOT NULL, `Month` mediumint(9) NOT NULL, `Day` mediumint(9) NOT NULL, PRIMARY KEY (`Period_ID`));
```

CREATE TABLE IF NOT EXISTS `Market Table`

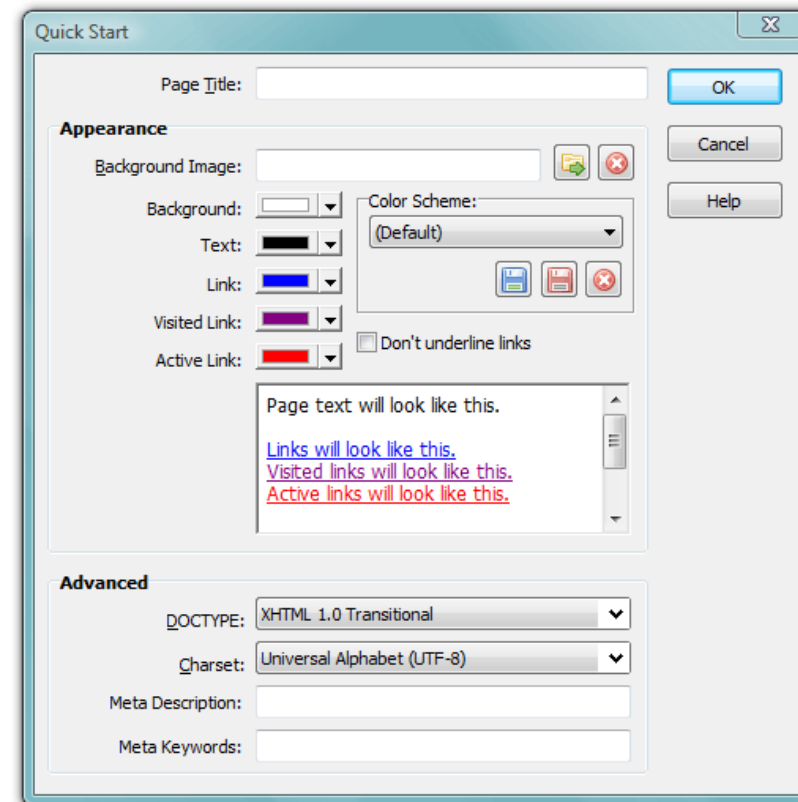
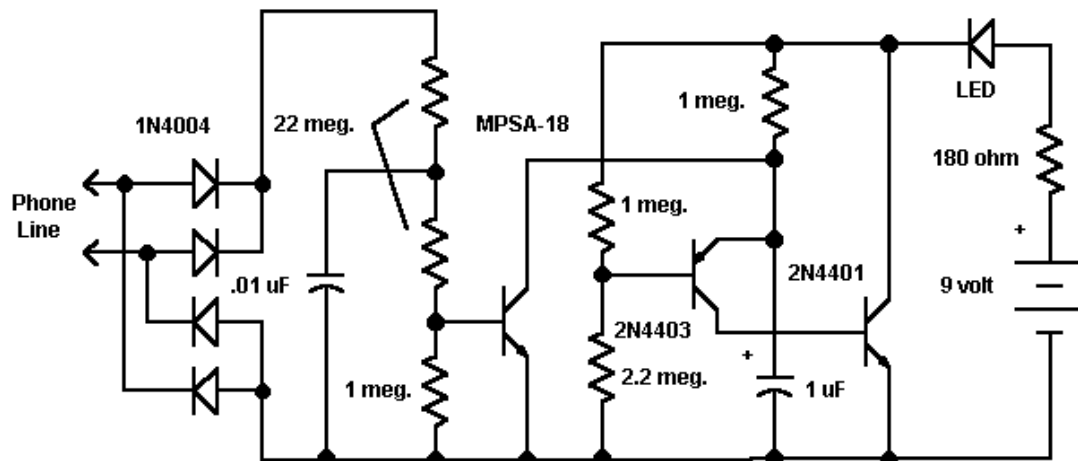
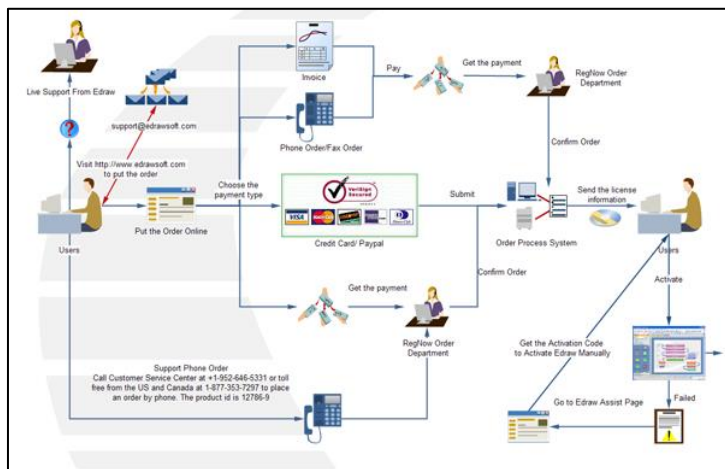
```
( `Market_ID` int(11) NOT NULL auto_increment, `Market_Desc` varchar(50) collate latin2_hungarian_ci default NULL, `District` mediumint(9) NOT NULL, `Region` int(11) NOT NULL, PRIMARY KEY (`Market_ID`));
```



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 összerakni
 - Gyorsan megadható
 - Komplex összefüggések is jól leírhatóak
- Nehéz értelmezni
 - Nehéz átlátni egy bizonyos összetettség felett
 - A szintaxist meg kell tanulni
- Programozók számára jobban érthető

Grafikus (vizuális) nyelvek

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

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:**

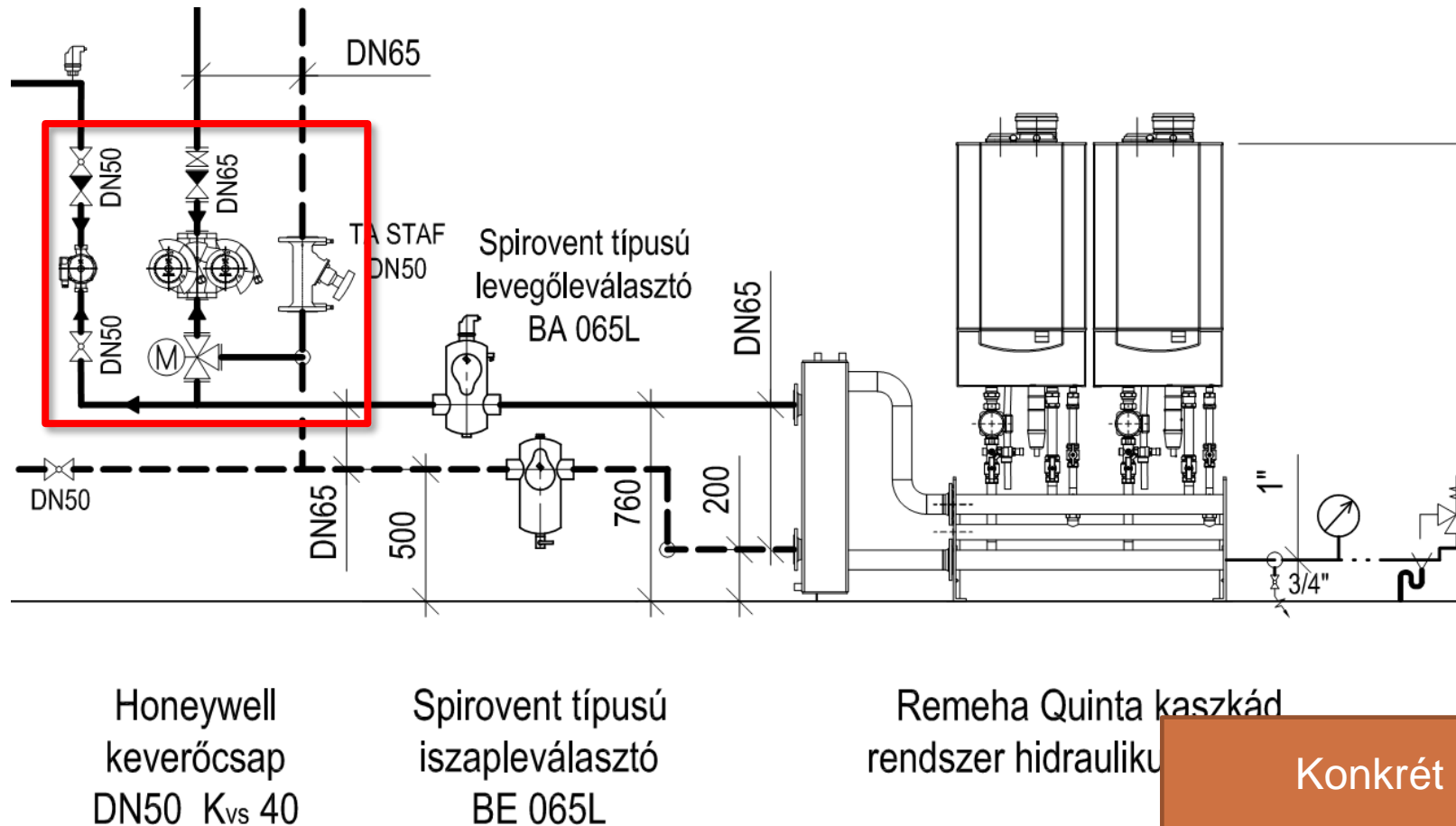
„Az „és” operátort jelöljük „&&”-el”

- **Szemantika:**

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

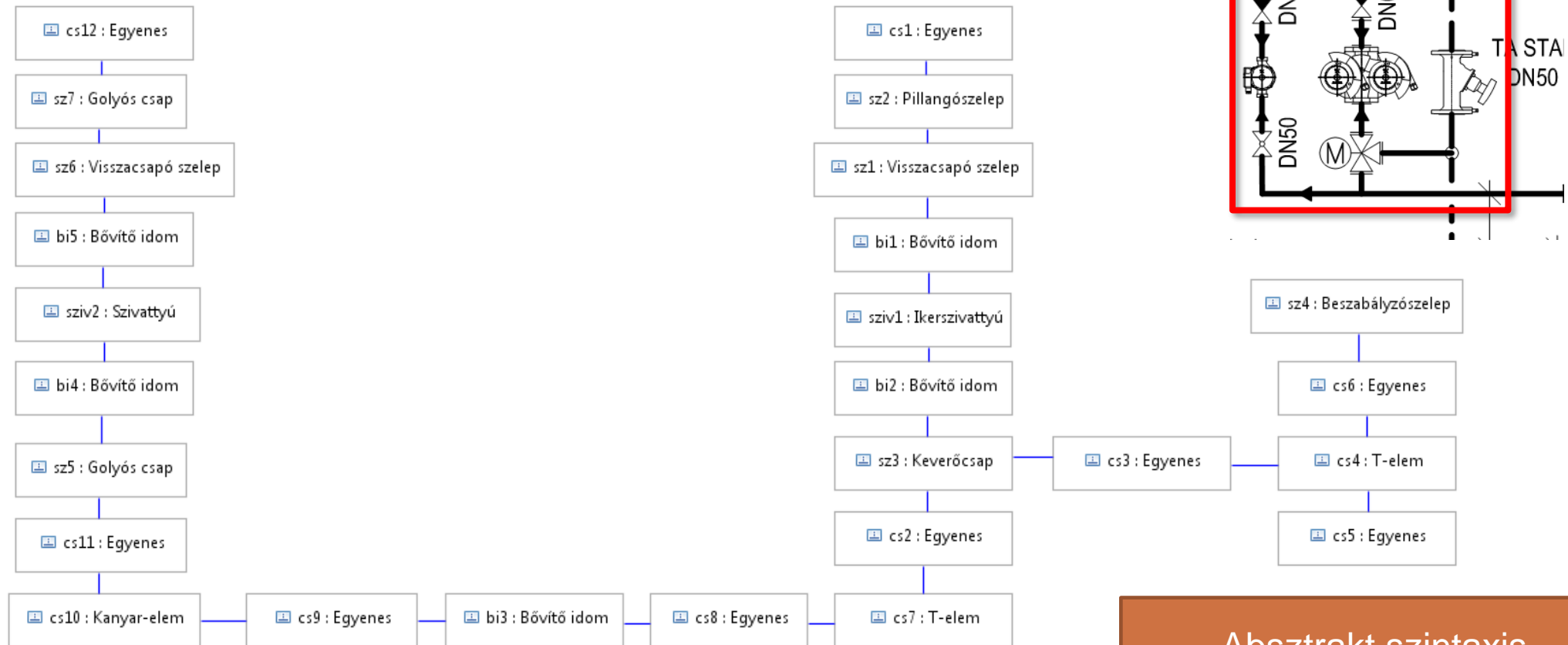
Konkrét szintaxis

■ Hogyan néz ki?



Absztrakt szintaxis: modell

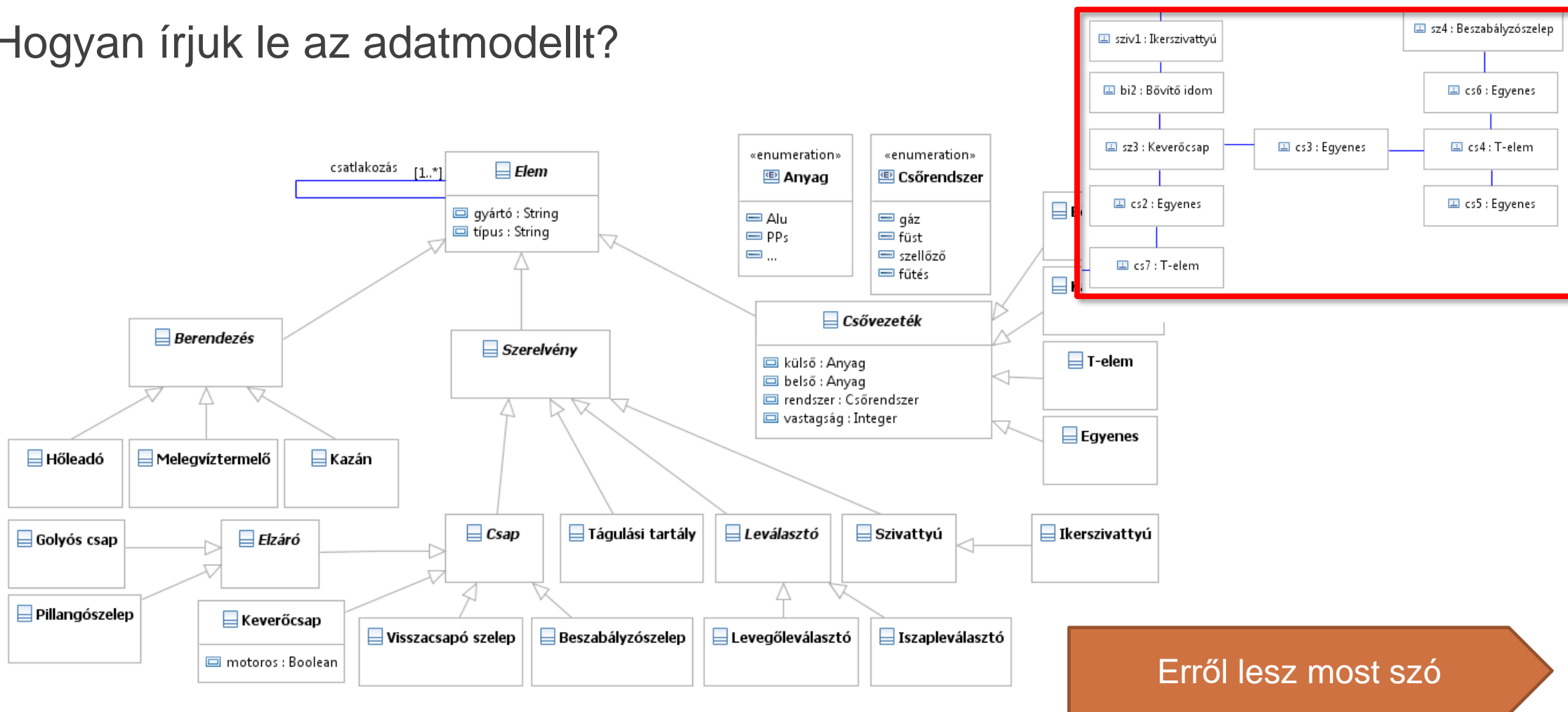
■ Hogyan reprezentáljuk a modellt?



Absztrakt szintaxis

Absztrakt szintaxis: a modell leírása

■ Hogyan írjuk le az adatmodellt?



Grafikus nyelvek modellezése

I. Grafikus nyelvek/modellek

II. Absztrakt szintaxis UML alapon

III. Blockly

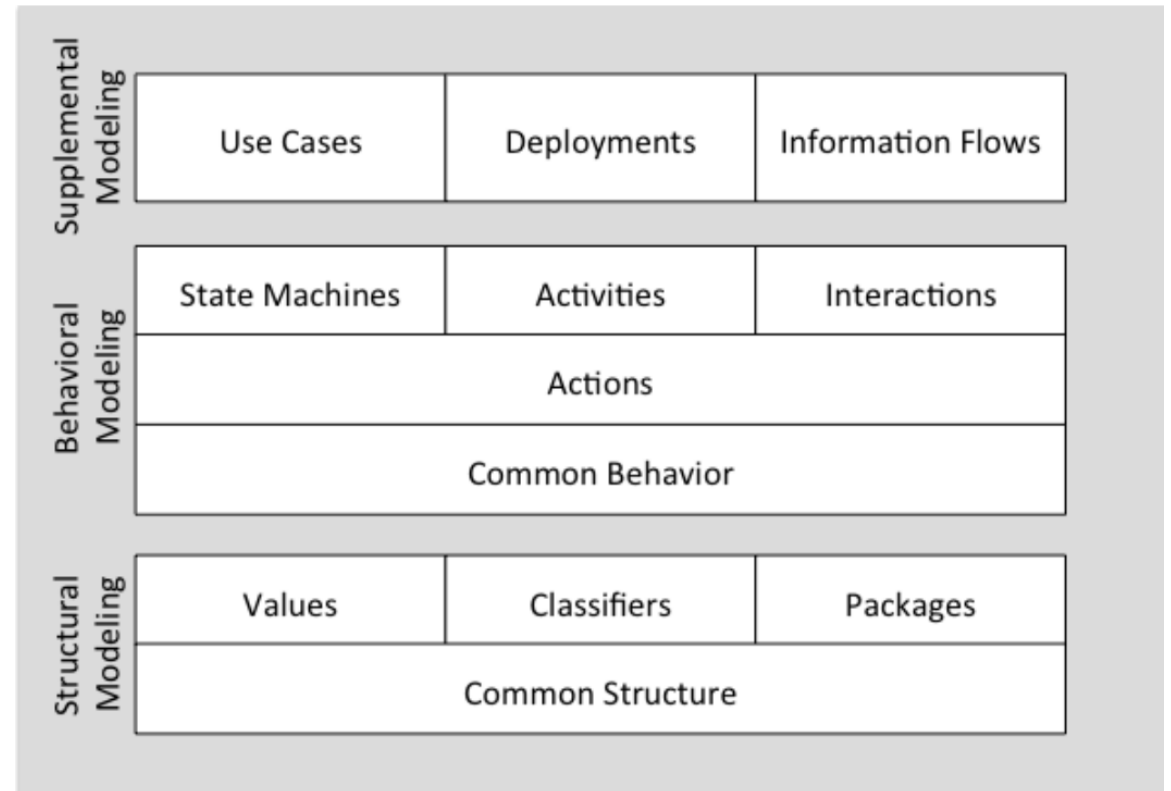
IV. Metamodellezés

V. Kényszerek



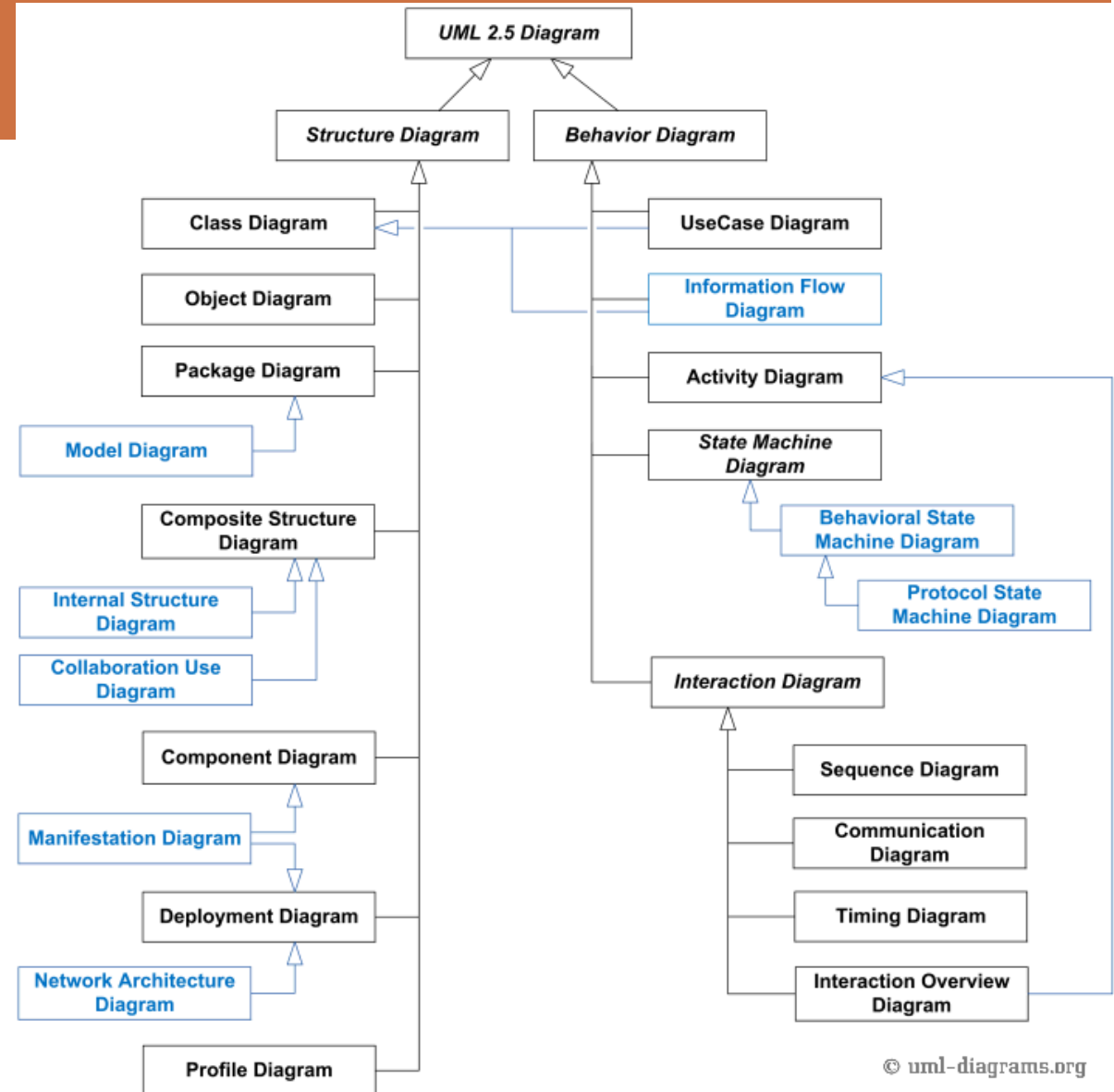
Az UML tulajdonságai

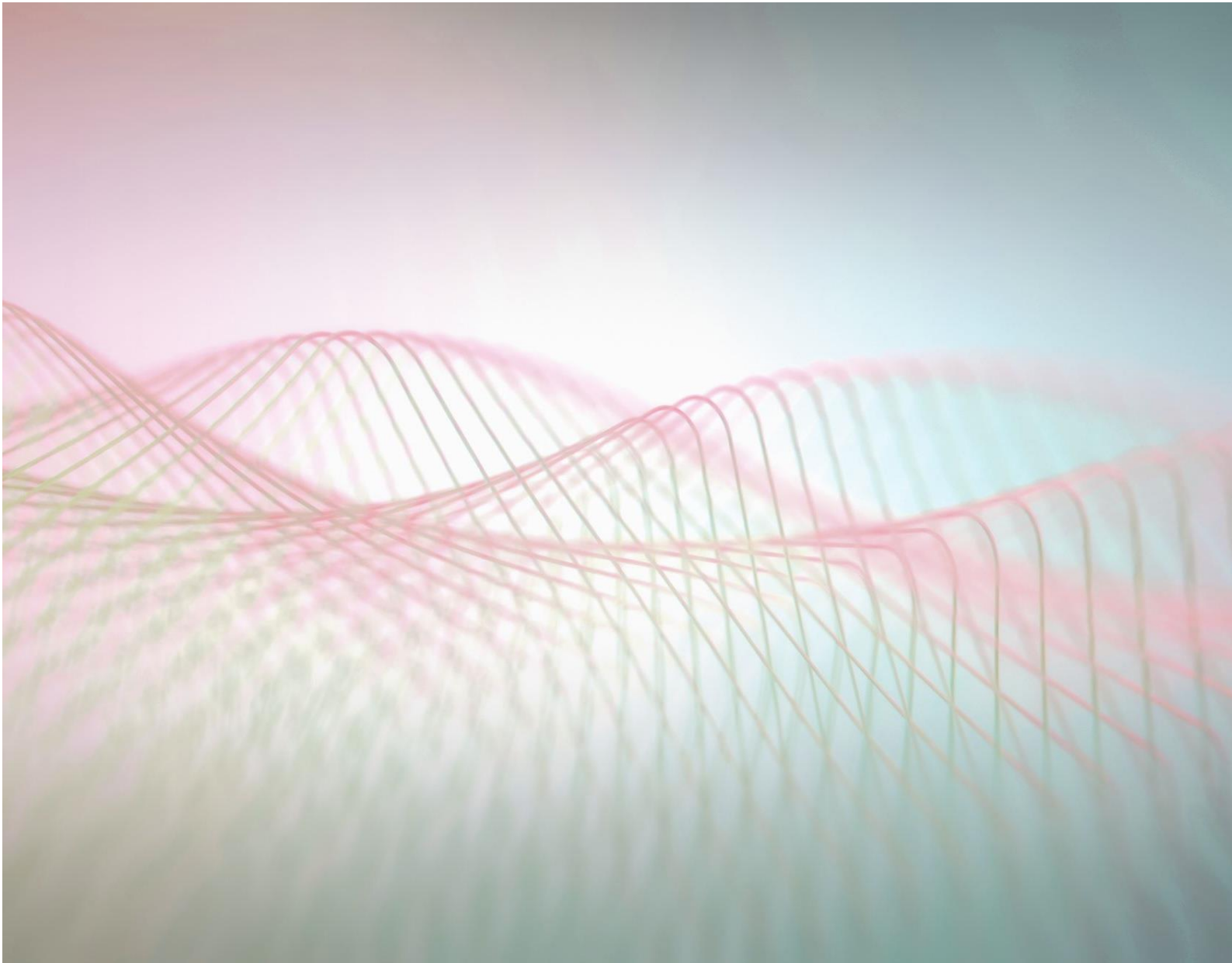
- Vizuális modellező nyelv
- Ötvözi az addig kialakult módszertanok előnyeit
- Programozási nyelvtől független
- Kiterjeszthető (főként: sztereotípiák)
- Szabványos (jelenleg 2.5.1 verzió)



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

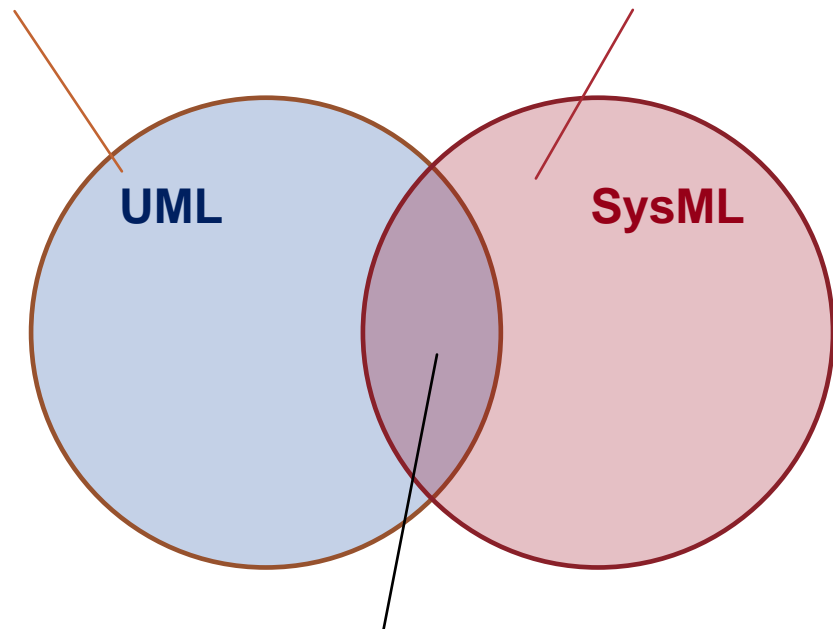
- Modeling and Analysis of Read Time and Enterprise systems (MARTE)
- Service oriented architecture Modeling Language (SoaML)
- UML Profile for Advanced and Integrated Telecommunication Services (TelcoML)
- UML Testing Profile (UTP)
- UML Profile for Voice
- SysML

SysML

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

UML de nem SysML

SysML UML Profile



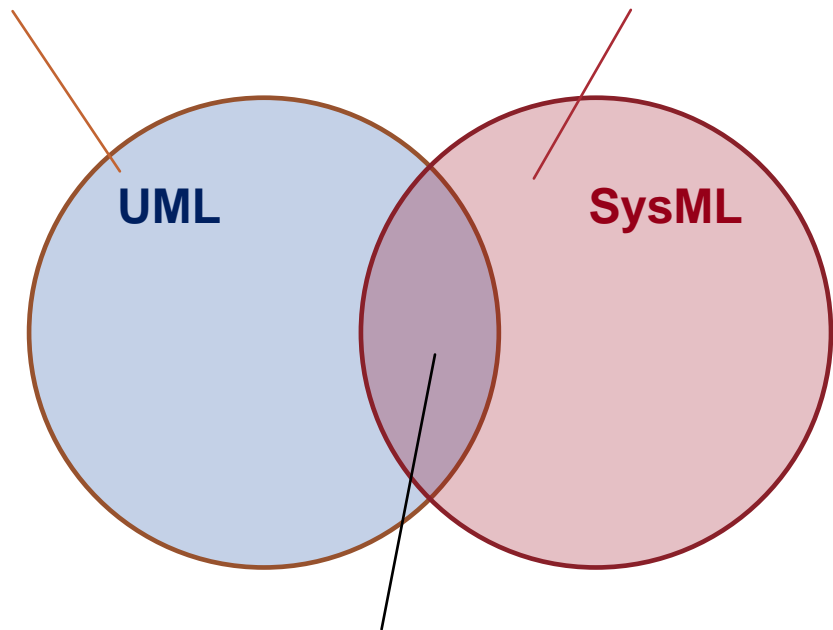
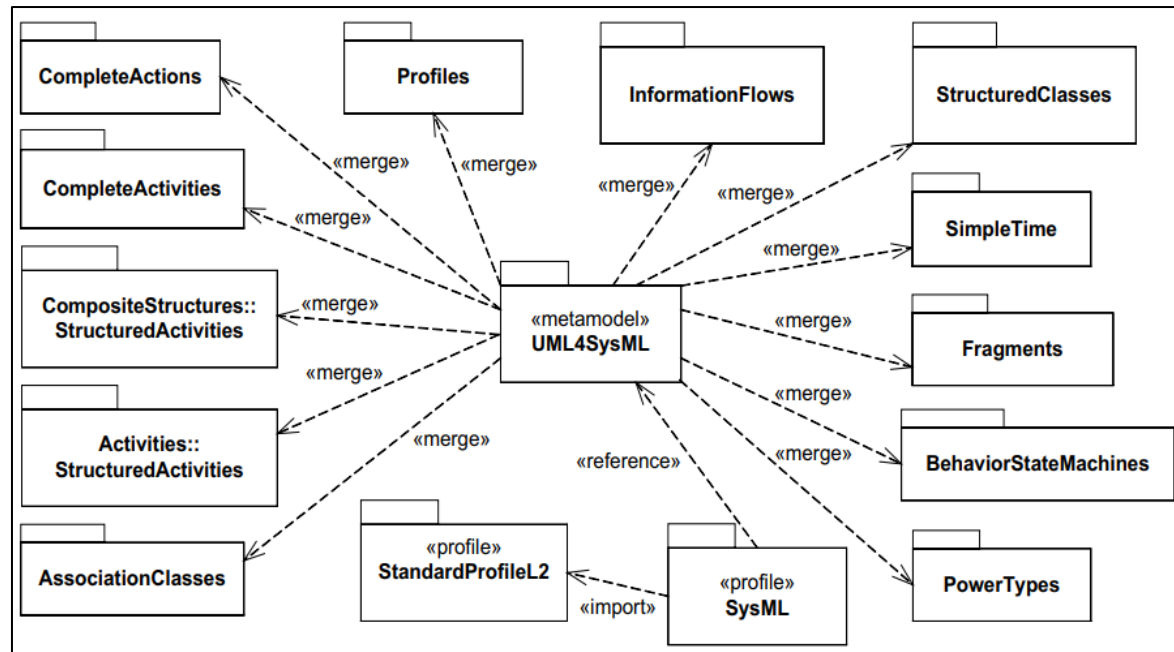
UML-ből átvett SysML

SysML

- Elhagy 7 UML diagramot

UML de nem SysML

SysML UML Profile



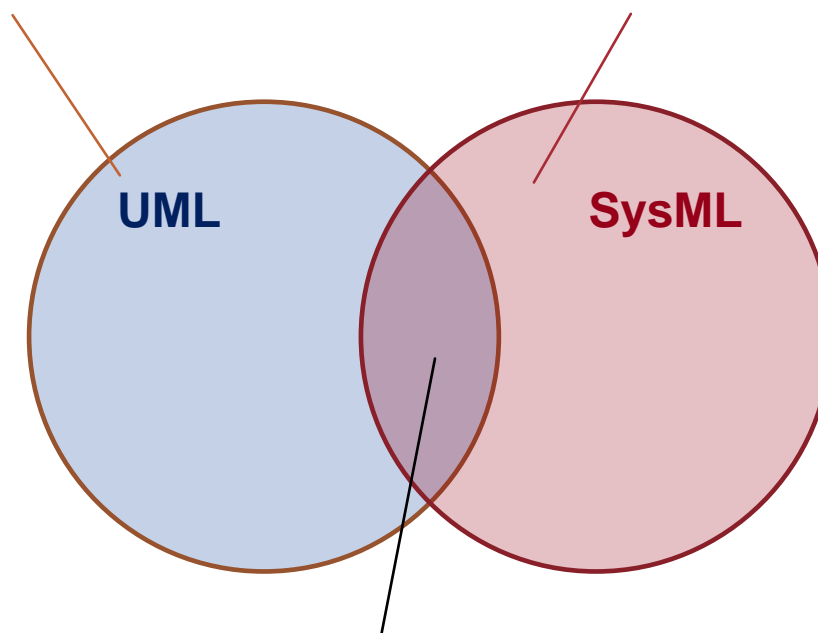
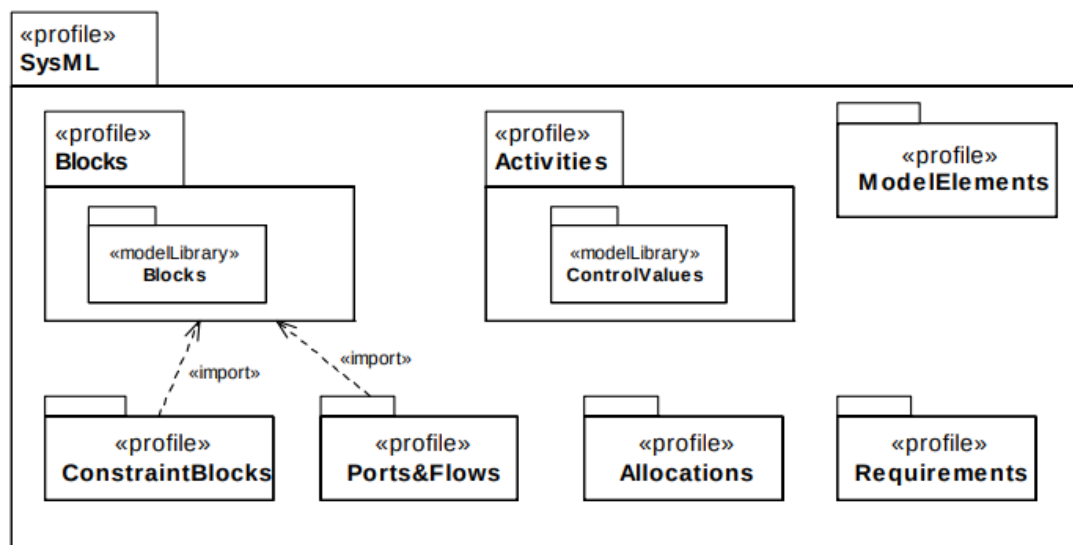
UML-ből átvett SysML

SysML

- Hozzáad új diagrammokat

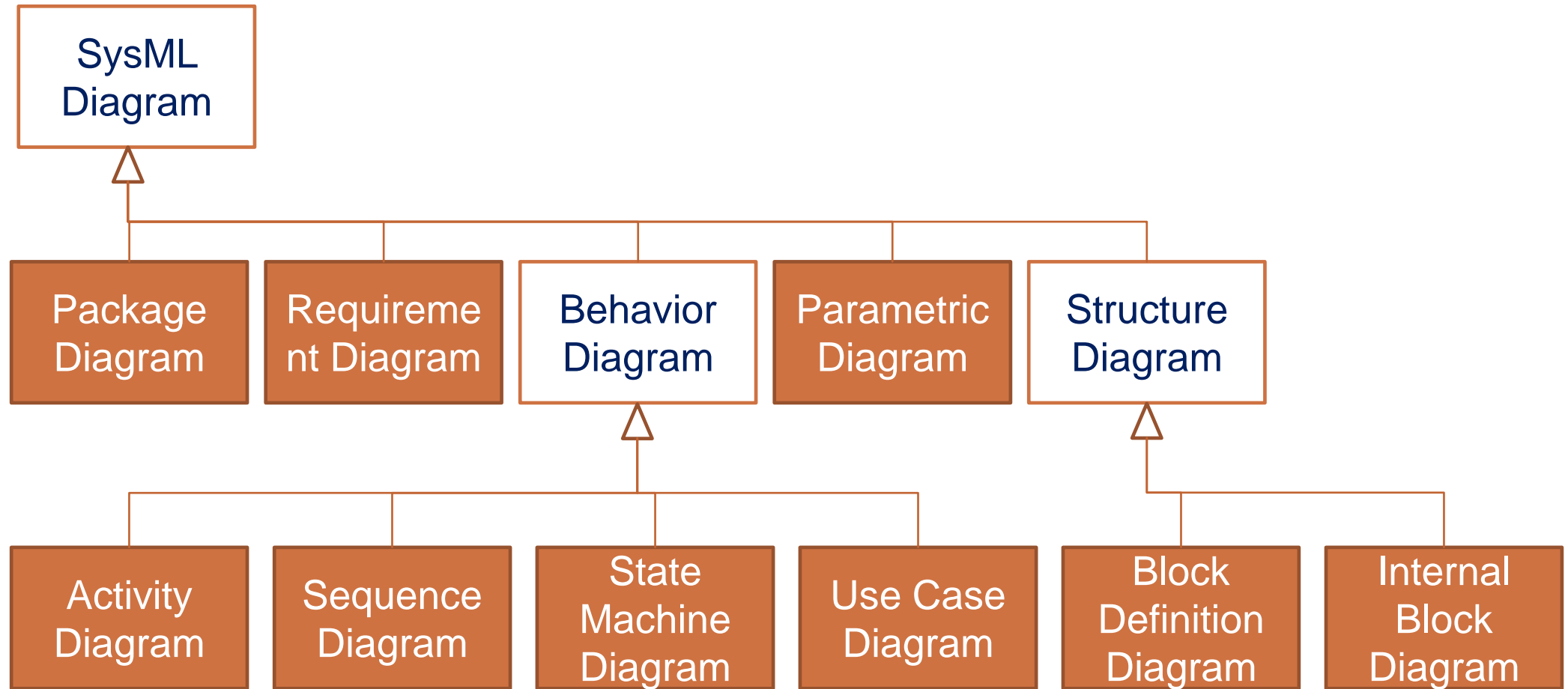
UML de nem SysML

SysML UML Profile

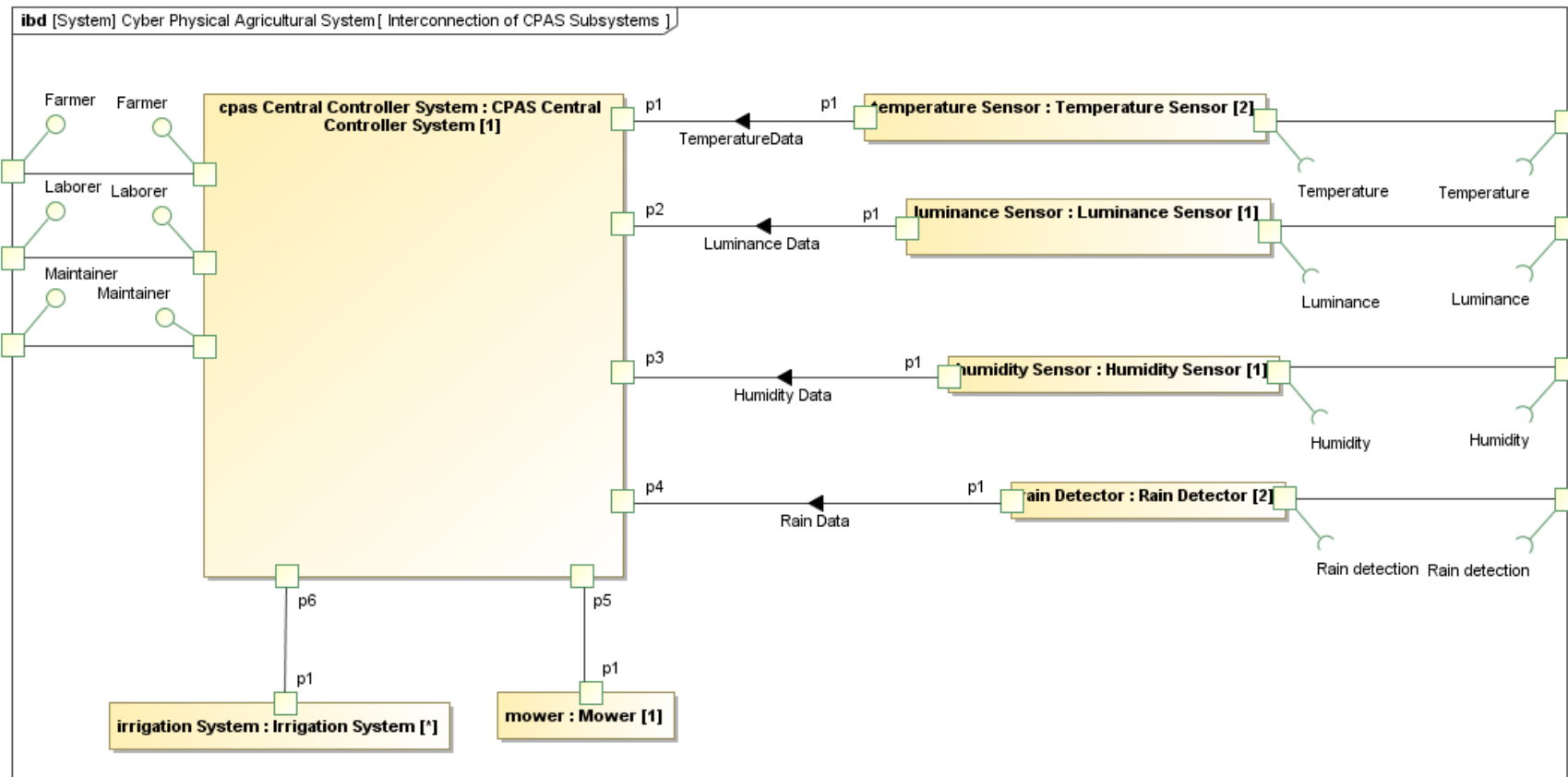


UML-ből átvett SysML

SysML diagram típusok



SysML diagram típusok



Internal
Block
Diagram

Grafikus nyelvek modellezése

I. Grafikus nyelvek/modellek

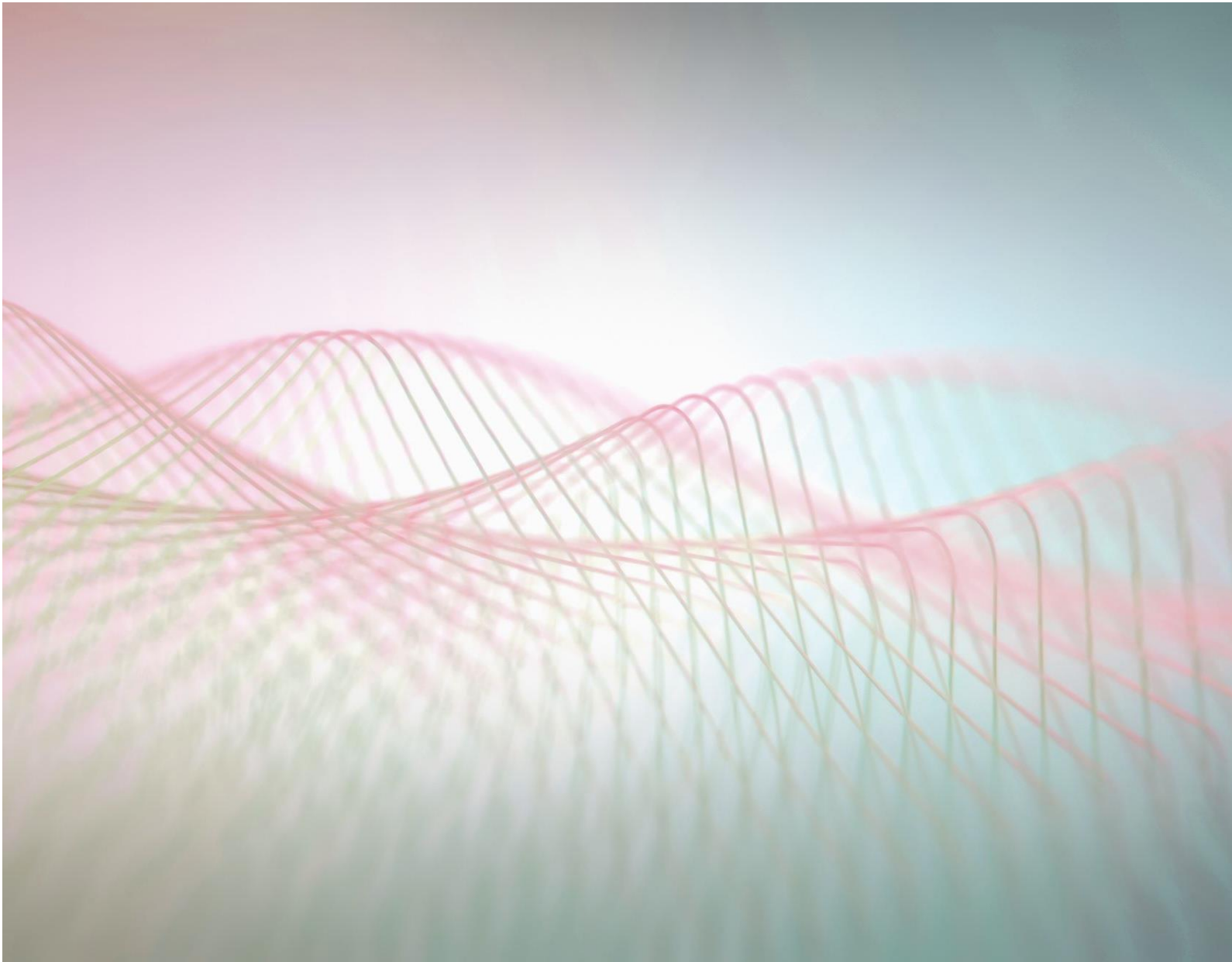
II. Absztrakt szintaxis UML alapon

III. Blockly

IV. Metamodellezés

V. Kényszerek

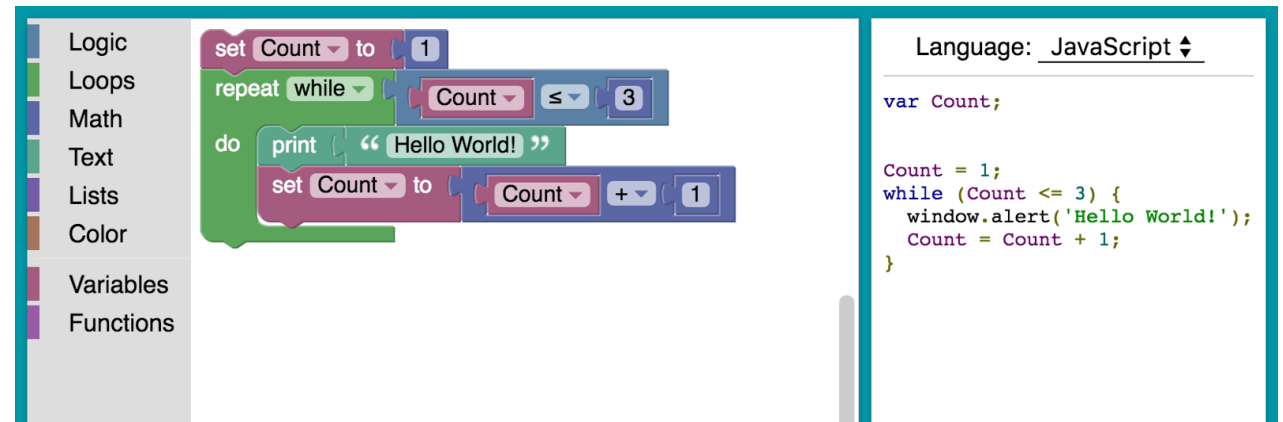




Blockly

Blockly

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



Blockly

■ Block factory – saját blokkok

← → ↻ <https://blockly-demo.appspot.com/static/demos/blockfactory/index.html>

Blockly > Demos > Block Factory

Input
Field
Type
Colour

name colour_rgb

inputs dummy input

fields left

- text infinite
- ⚠ text input 42 , NAME
- ⚠ angle input 90° , NAME
- ★ ⚠ dropdown NAME
- is red , R
- is green , G
- is blue , B
- There are 6 field blocks AME with this name.
- ⚠ variable item , NAME

value input GREEN

fields left

- text while

type

statement input NAME

fields left

- text do

type

external inputs

Preview: LTR

infinite 42 90° is re

while

do

Language code:

```
Blockly.Blocks['colour_rgb'] = function() {
  init: function() {
    this.setHelpUrl('http://www.blockly.com/');
    this.setColour(150);
    this.appendDummyInput('infinite')
      .appendField('infinite');
    this.appendField(new Blockly.FieldTextInput("42"), "NAME");
    this.appendField(new Blockly.FieldAngle("90°"), "NAME");
    this.appendField(new Blockly.FieldDropdown([["is red", "R"], ["is green", "G"], ["is blue", "B"]]), "NAME");
    this.appendField(new Blockly.FieldCheckbox("TRUE"), "NAME");
    this.appendField(new Blockly.FieldVariable("item"), "NAME");
  }
};
```

Generator stub: JavaScript

```
Blockly.JavaScript['colour_rgb'] = function(block) {
  var text_name = block.getFieldValue('NAME');
  var angle_name = block.getFieldValue('NAME');
  var dropdown_name = block.getFieldValue('NAME');
  var checkbox_name = block.getFieldValue('NAME') == 'TRUE';
  var colour_name = block.getFieldValue('NAME');
  var variable_name = Blockly.JavaScript.variableDB_.getName(block.getFieldValue('NAME'), 'variable');
  var value_green = Blockly.JavaScript.valueToCode(block, 'GREEN', Blockly.JavaScript.ORDER_ATOMIC);
  return 'while (' + text_name + ' is ' + colour_name + ')\n{\n  ' + value_green + ';\n}\n';
```

name repeat_block

inputs dummy input

fields left

- text repeat
- numeric input 0 , loop_var
- min 0 max Infinity precision 0
- text times

statement input loop_blocks

fields left

- text do

type any

automatic inputs

top+bottom connections

tooltip

help url

top type

bottom type

colour

repeat 5 times

do

“ This block can repeat the embedded blocks. ”

“ ”

any

any

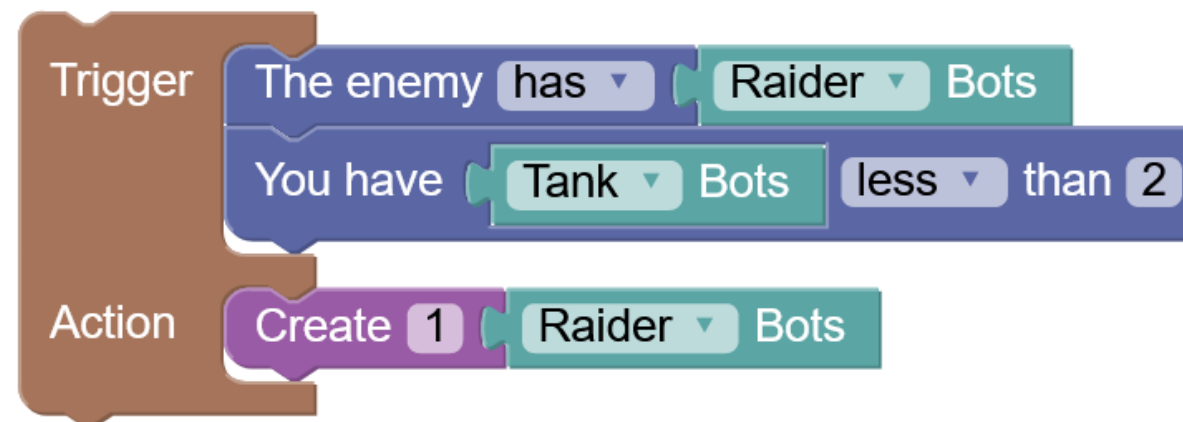
hue: 180°

Blockly példa – Stratégiai AI nyelv

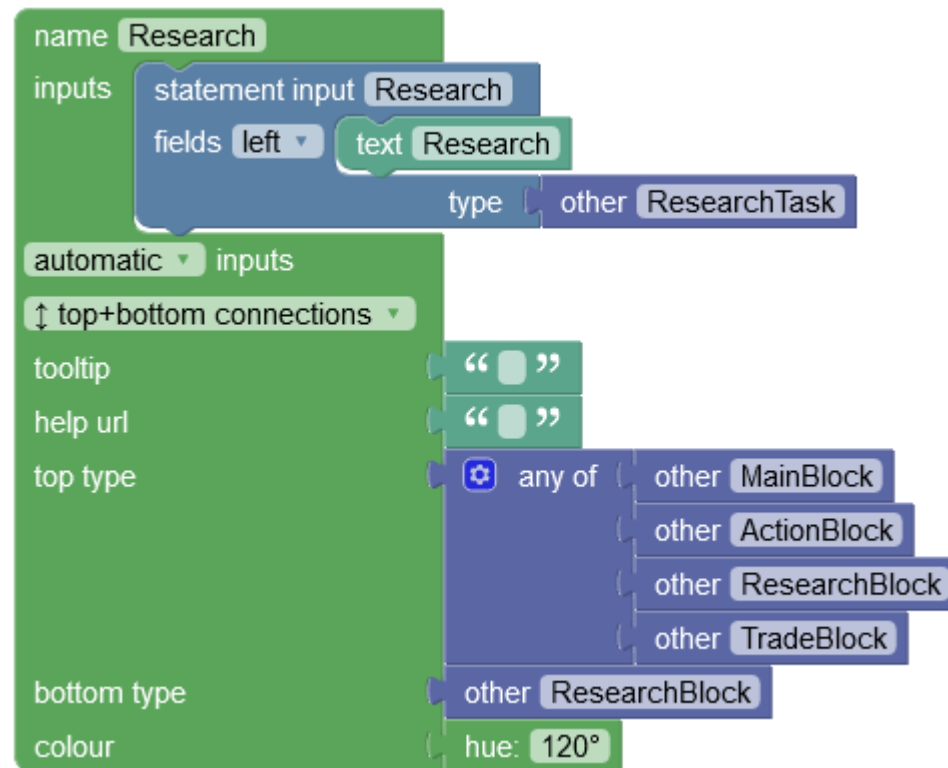
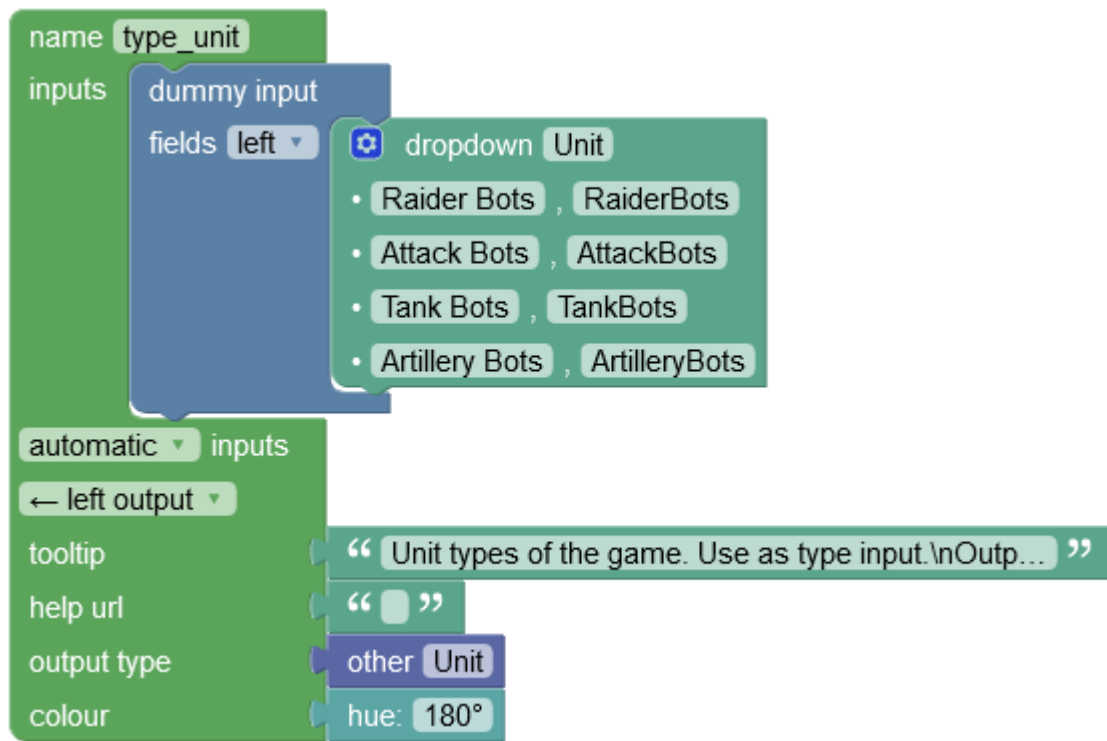
Turn	Player	Steel	RoboSteel	Energy	Crystals	Energy Cores	Credits
13	Player	20	5	60	16	7	85
13	Server AI	10	5	65	6	7	120



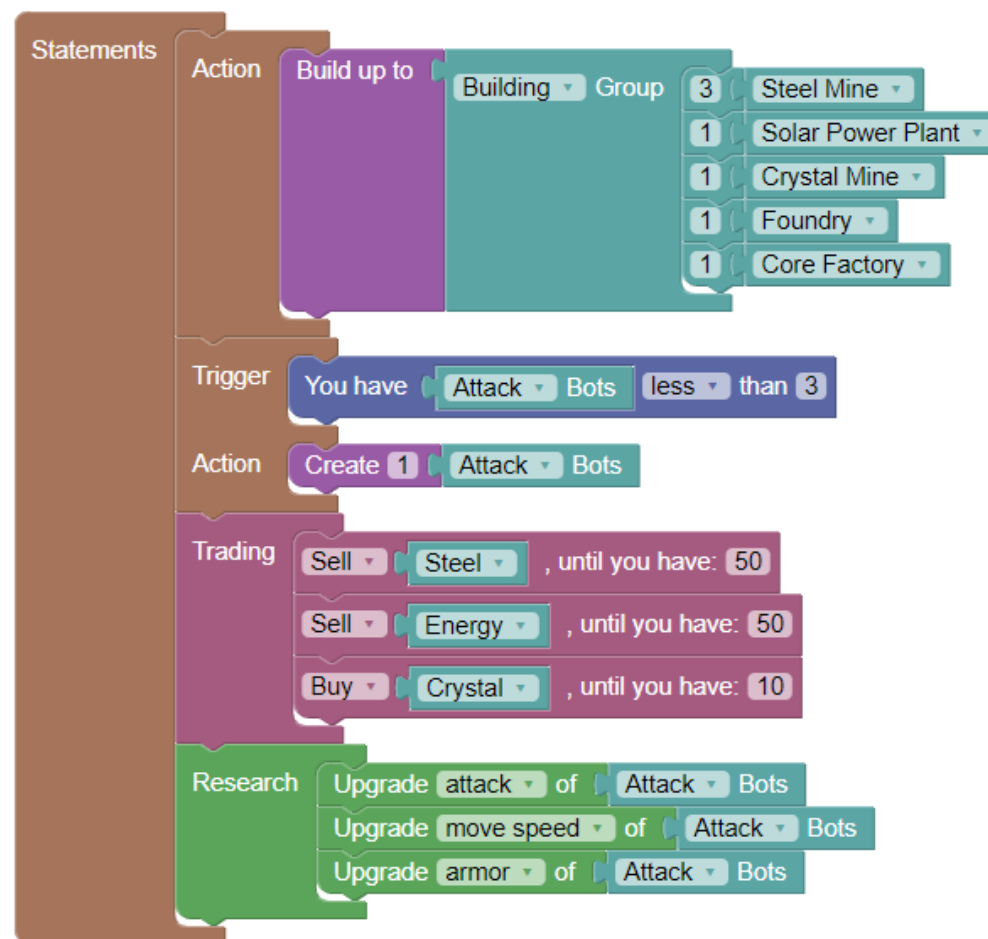
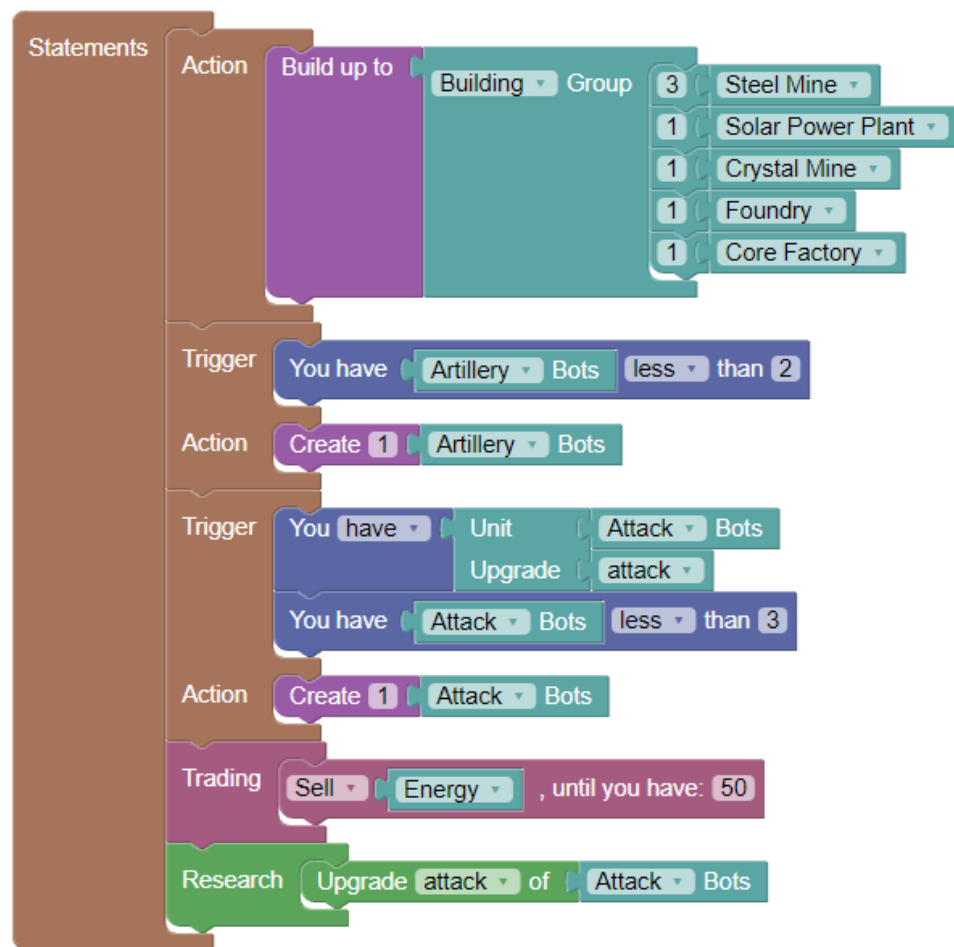
Player
artilleryBotUpgrades
attackBotUpgrades
raiderBotUpgrades
tankBotUpgrades
Server AI
artilleryBotUpgrades
attackBotUpgrades
raiderBotUpgrades
tankBotUpgrades



Blockly példa – Stratégiai AI nyelv



Blockly példa – Stratégiai AI nyelv



Grafikus nyelvek modellezése

I. Grafikus nyelvek/modellek

II. Absztrakt szintaxis UML alapon

III. Blockly

IV. Metamodellezés

V. Kényszerek



Metamodellezés

- Metamodel: 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 ...

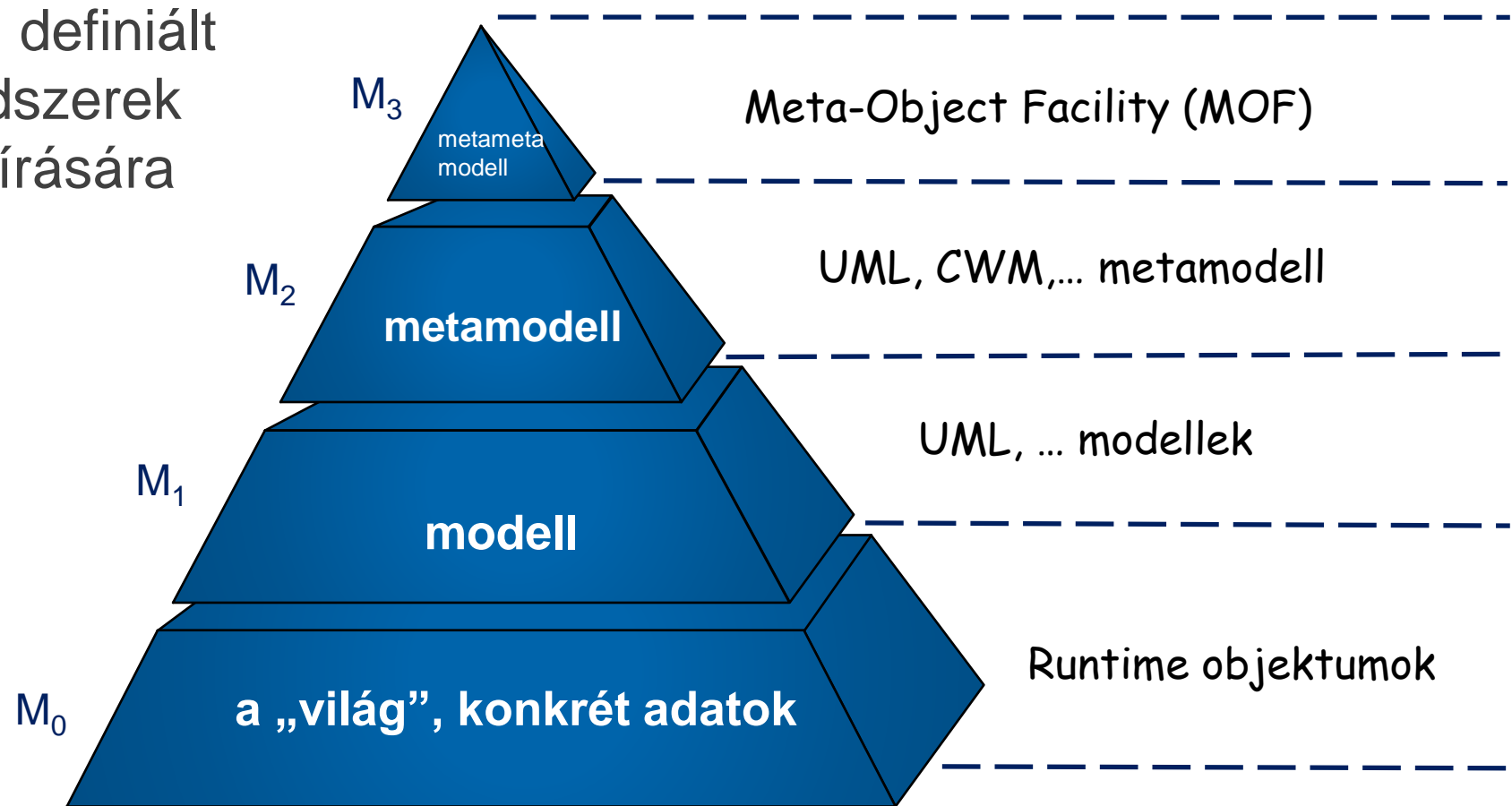
- UML profile
 - > Nem kötik az UML szabályai
 - > Rugalmasabb, testreszabhatóbb
- Blockly
 - > Példányosítás vs. sablon-alapú működés
 - > Korlátozott ellenőrzések a sablon mezőire
 - > Kapcsolatok, ismételt információ megadása
- Ad-hoc saját megoldás
 - > Formális, szabványos → automatikus feldolgozás
 - > Közös nyelv, átjárhatóság

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	Metametamodelld 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

MOF variánsok

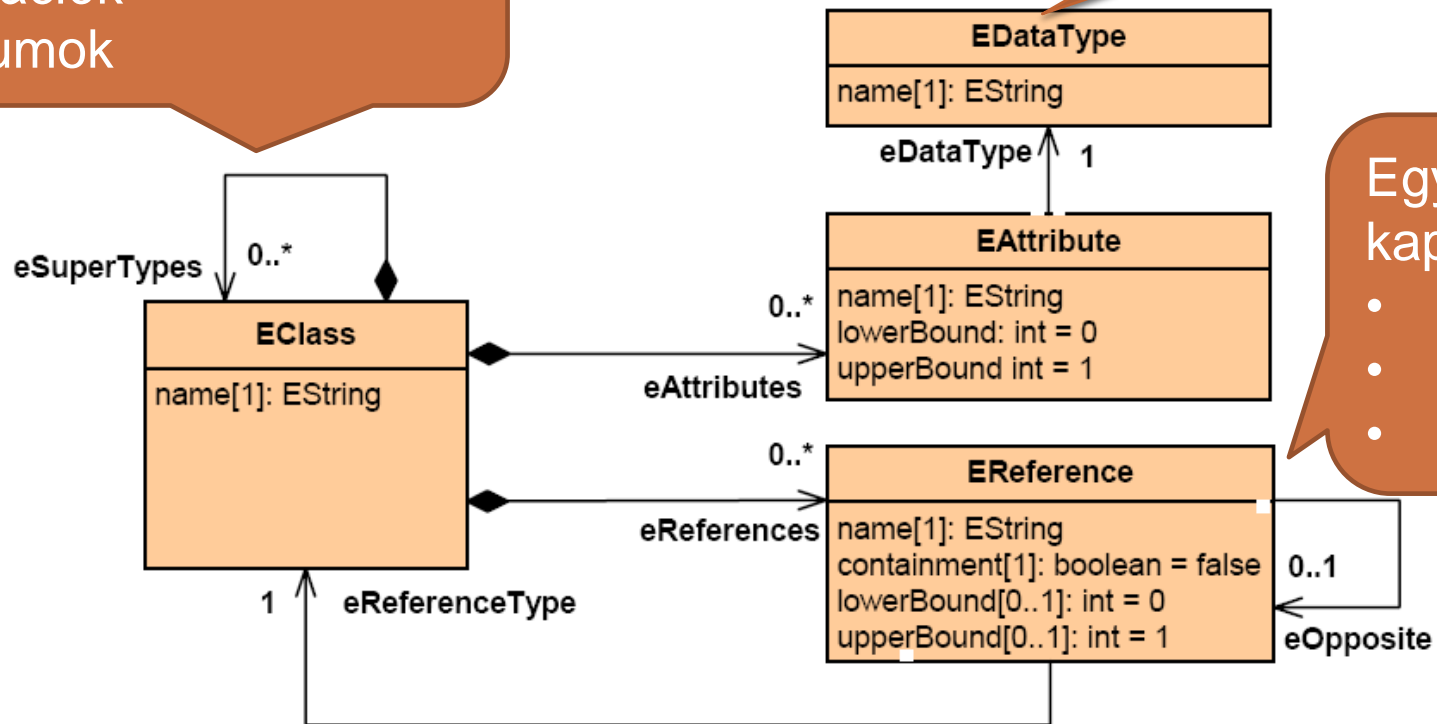
- EMOF (Essential MOF)
 - > Alap (OO-hoz és XML-hez kapcsolódó) funkciók
 - > Cél: MOF modellek leképezése JMI és XMI formára
 - > Egyszerű metamodellekhez
 - > Támogatja a kibővítéseket
 - > ECore
- CMOF (Complete MOF)
 - > „Teljes verzió” (UML 2.0 kiegészítések)
 - > UML 2.0 jellegű nyelvek definiálásához

Ecore alapok

Fogalom

- ősfogalmak
- asszociációk
- attribútumok

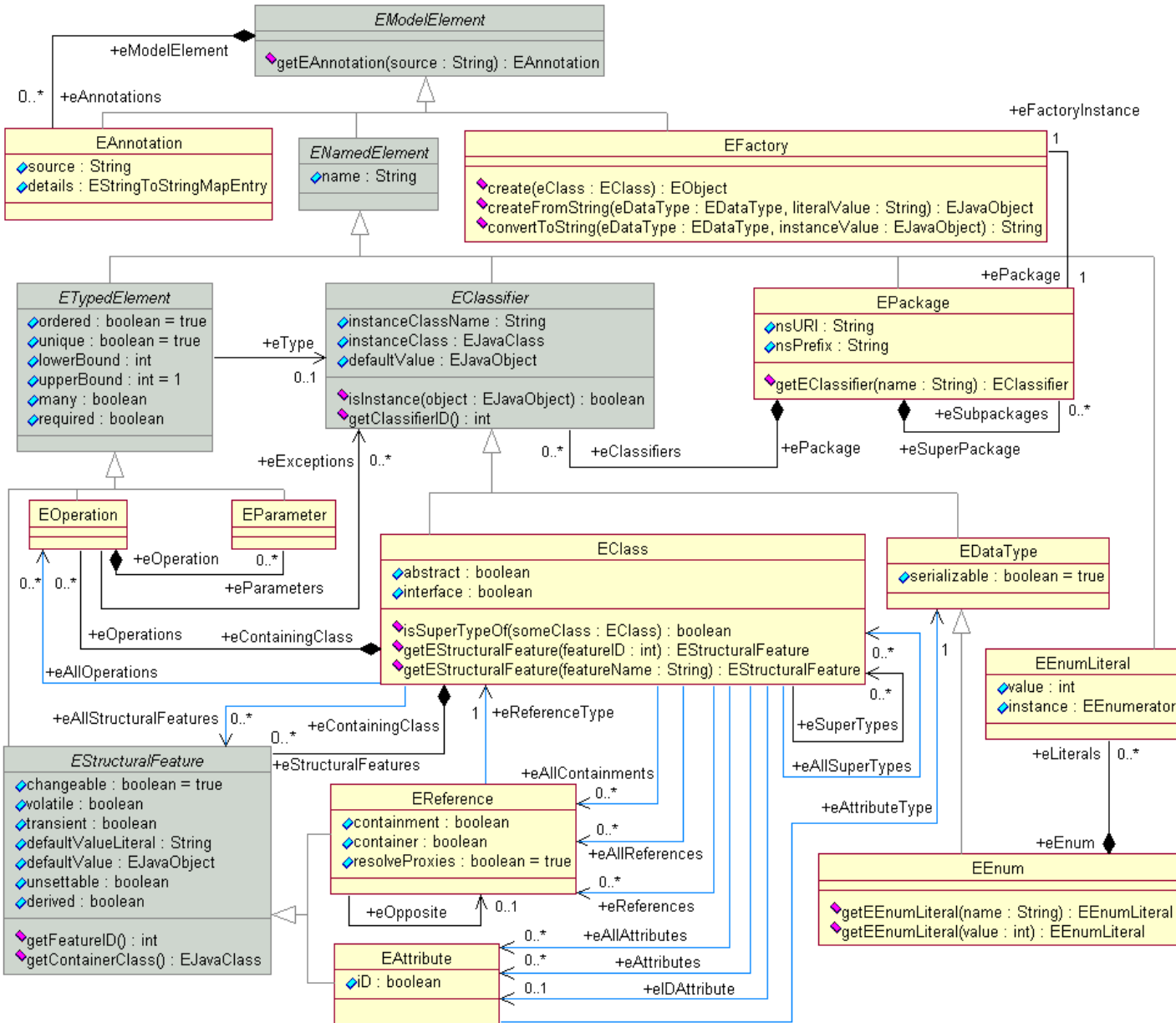
Típusos attribútum



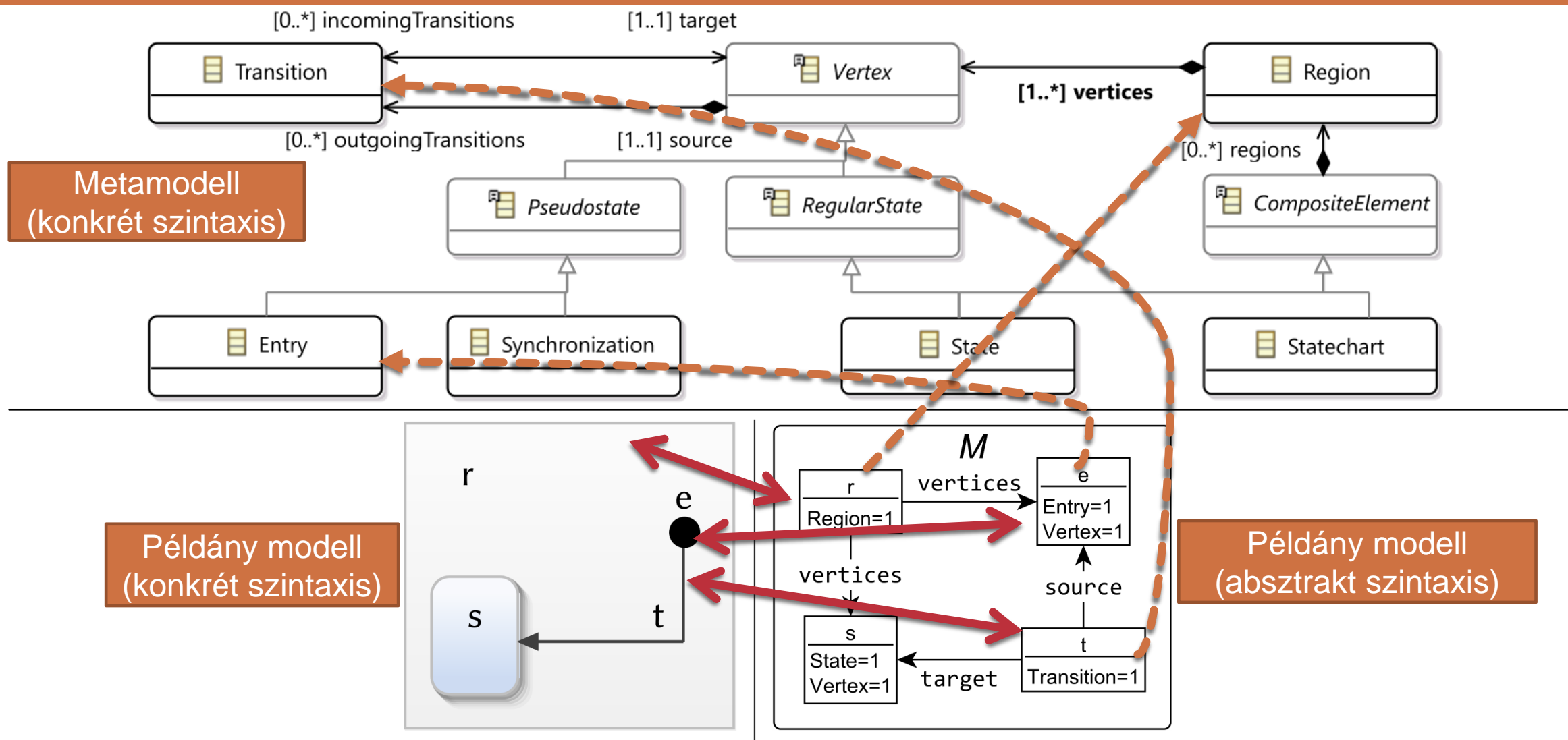
Egyirányú bináris kapcsolat

- Típus
- Opcionális inverz
- Multiplicitás

Ecore – EMOF*

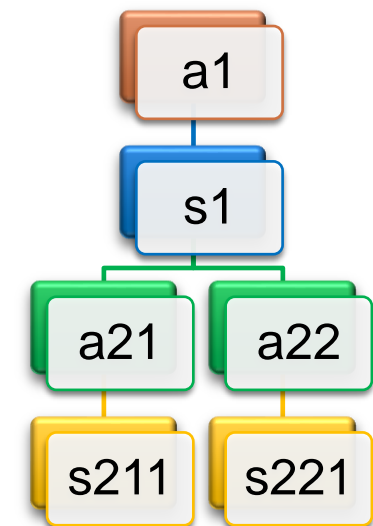
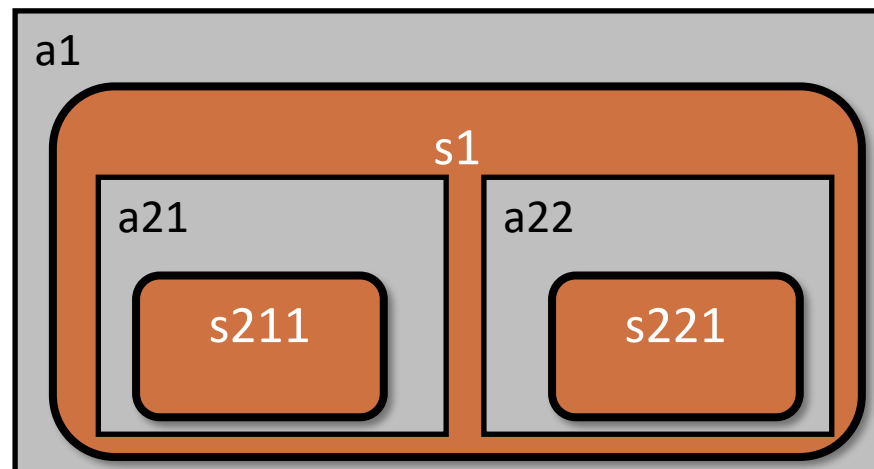
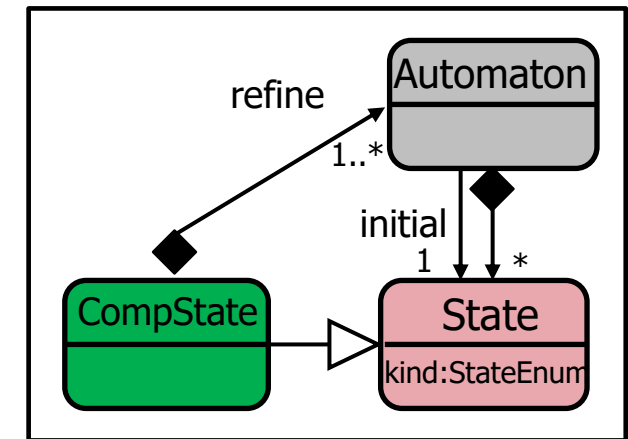


Yakindu példa



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 relációknál lehetséges



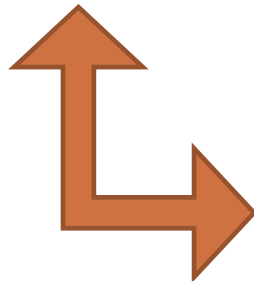
Öröklés és példányosítás

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

- ✓ $1+2 = \text{Morzsi egy kutya}$
- ✓ $1+2+3 = \text{Morzsi egy állat}$
- ! $1+4 = \text{Morzsi nem egy (kutya)fajta}$
- ! $2+5 = \text{A puli nem egy faj}$
- Az öröklés (SupertypeOf) tranzitív
- A példányosítása (InstanceOf) nem tranzitív

Metaadatok sorosítása - XMI

- Modellek átadása heterogén környezetben
- **XML Metadata Interchange (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

- > Megjelenítés

- > Struktúra jelentése

}

Absztrakt szintaxis

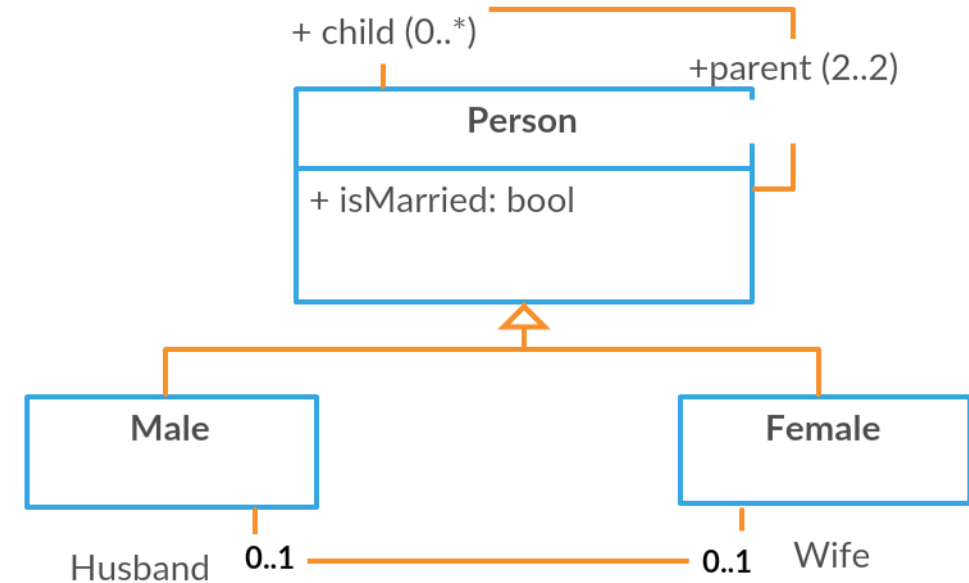
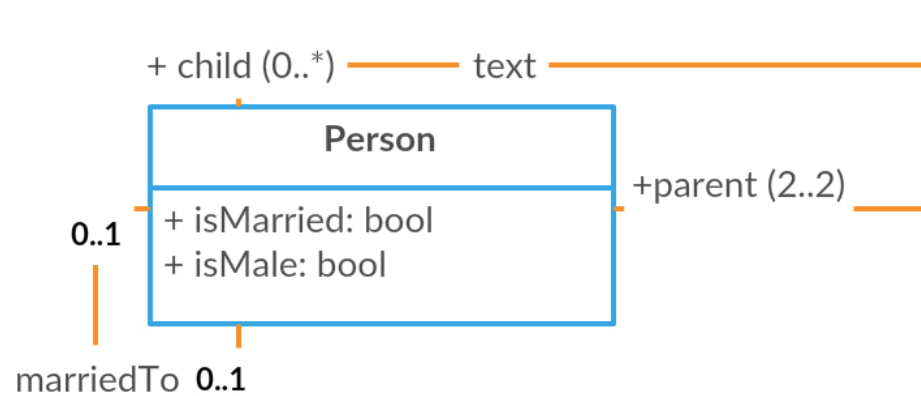
}

Konkrét szintaxis

}

Szemantika

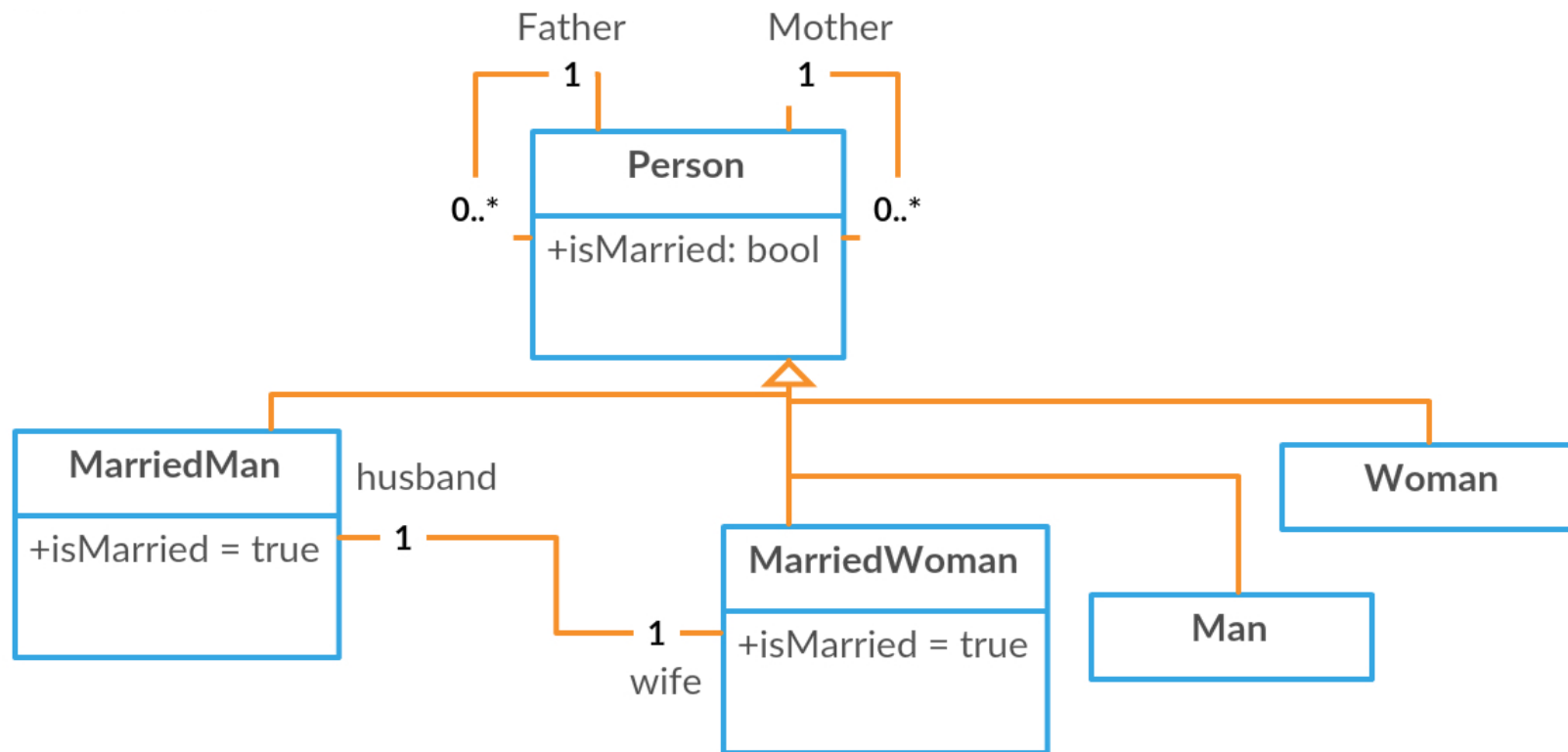
Hogyan fejezzem ki?



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

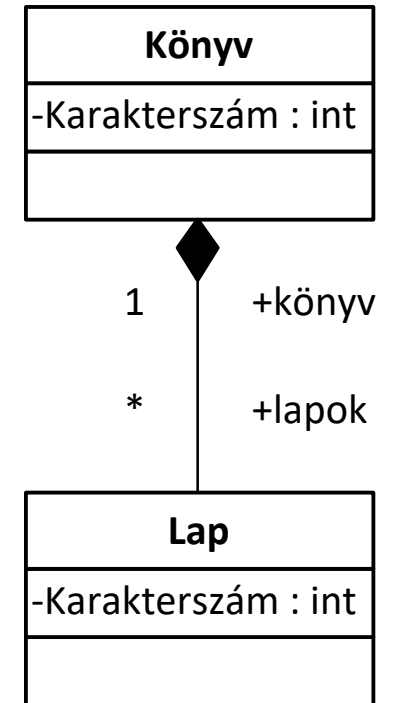
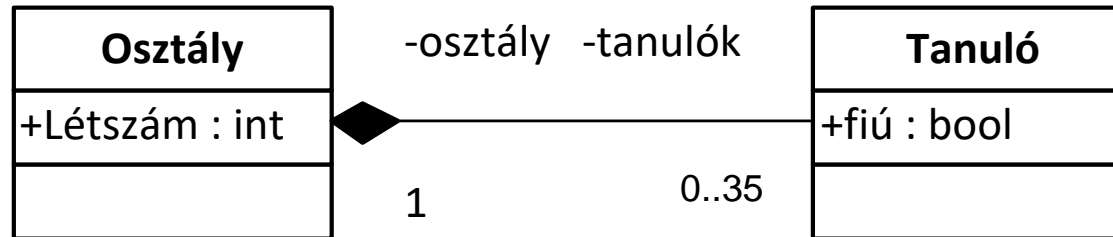
Hogyan fejezzem ki?

- Tartalmazza az előző két megkötést, de nem használható



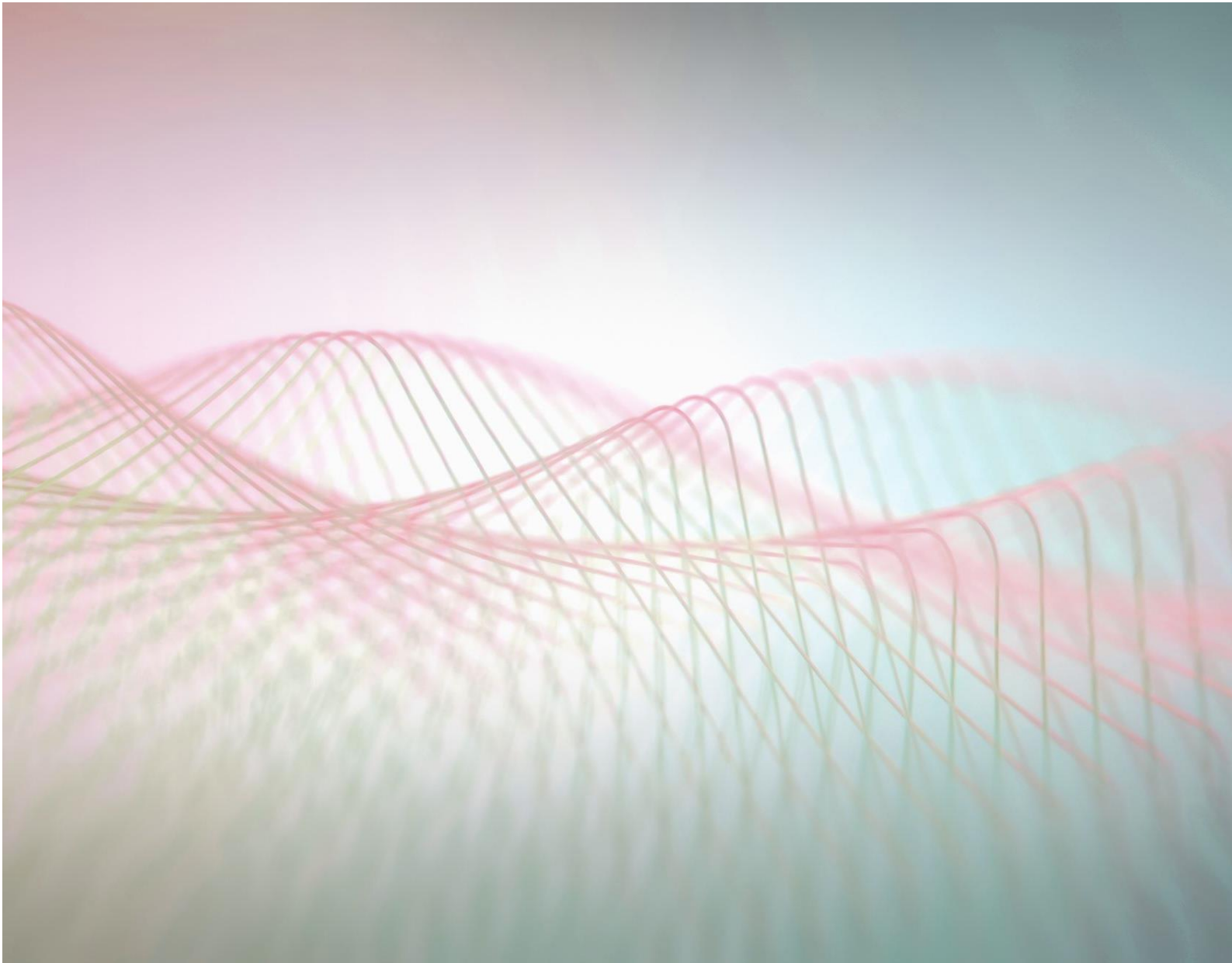
Kényszerek - Motiváció

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

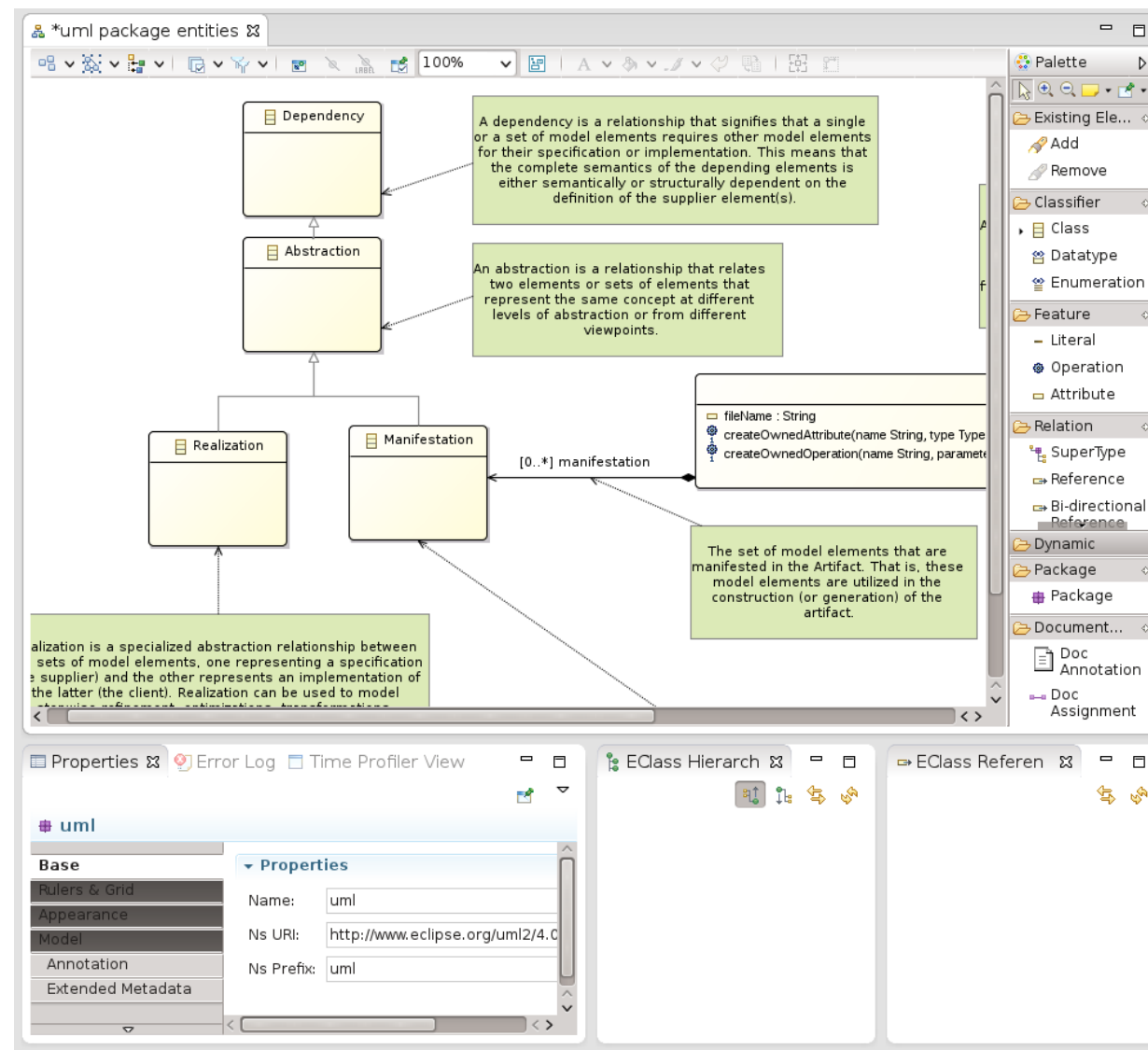


ÖTÖDIK GYAKORLAT

Eclipse Modeling Framework (EMF)

Metamodellezés EMF technológiával

- De Facto modellezési szabvány
- Hogyan metamodellezzünk?
- Hogy épülnek fel a leggyakoribb modellezőeszközök?
- Modellek perzisztenciája
- Modellek szerkesztése
- Adatszerkezetek modellek reprezentációjához





Köszönöm a figyelmet!