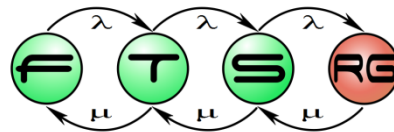


# Domain-specifikus modellezés az Eclipse Modeling Framework használatával



# Domain-specifikus nyelvek: miért?

- A mai szoftverfejlesztés kihívásai
  - Komplexitás
    - Növekvő fejlesztési idő és költség
  - Sokféleség
    - Domain, követelmény, implementációs technika, eszköz
  - Változás
    - Környezet, követelmények, hibajavítások...

# Domain-specifikus nyelvek

- Cél: a szakterületi fogalmakkal fejezzük ki magunkat
  - Absztrakciós szint növelése
  - Szakértő (nem fejlesztő) is értse a modellt!
- Domain-specifikus nyelv (Domain Specific Language)
  - Pl. HTML, SQL, reguláris kifejezések
- Lehetne általános célú nyelv (General Purpose Language)
  - Pl. Java, C stb.
  - De túl implementációspecifikus
  - Szakértő nem tudja hatékonyan használni

# Domain-specifikus nyelvek: mikor?

- Előnyök
  - Specifikus problémák gyorsabb megoldása
  - Követelmények kifejezése a problémater szintjén
  - Automatikus validáció
- Hátrányok
  - Jártasság szükséges a szakterületben (költség)
  - Nyelvtan megalkotása szubjektív, bonyolult
  - Eszköztámogatás fejlesztése költséges lehet (a behatárolt alkalmazhatósághoz képest)
  - Inkompatibilis DSL-ek elterjedhetnek ugyanarra a területre
    - Szabványosítás

# Nyelv-alapú fejlesztés

- **Nyelv:**
  - Modellek leírása
  - Emberek számára
  - Cél: problématerület áttekintése
- **Biztosítandó**
  - Futtatás
  - Analízis
  - Tesztelés
  - Implementáció
  - ...

# Nyelvek együttes kezelése

- Feladat: több nyelv együttes kezelése
  - Transzformáció
  - Aspektus integráció
  - Szinkronizáció
  - Finomítás és ekvivalencia vizsgálata
  - Evolúció
- Hasonlóan az aspektus-orientált programozáshoz
  - Több szempont
  - Azonos rendszer

# Az UML lehetőségei és korlátai

- Előnyök:
  - Standard közös nyelv
  - Vizuális
- Hátrányok:
  - Nem domain-specifikus
  - Korlátozott hatókör
    - UML Profilok
  - Nem formális szemantika
    - Nincs egy közös, általánosan használható szemantika
  - Unified (NOT Universal) Modeling Language

# Metamodellezés

Modellező nyelvek tervezése



# Modellező nyelvek tervezése

- Metamodellezés: Tervezési metodológia modellező nyelvekhez
- Metamodel: egy modellező nyelv modellje
- Részei:
  - Absztrakt szintaxis
  - Konkrét szintaxis
  - Jólformáltsági szabályok
- További lehetőségek:
  - Szemantika
  - Leképezések más nyelvekre

# Konkrét szintaxis

- Ahogy a felhasználó “látja”
- Nyelvenként lehet többféle is
- Fajtái
  - Szöveges szintaxis
  - Grafikus szintaxis

# Konkrét szintaxis

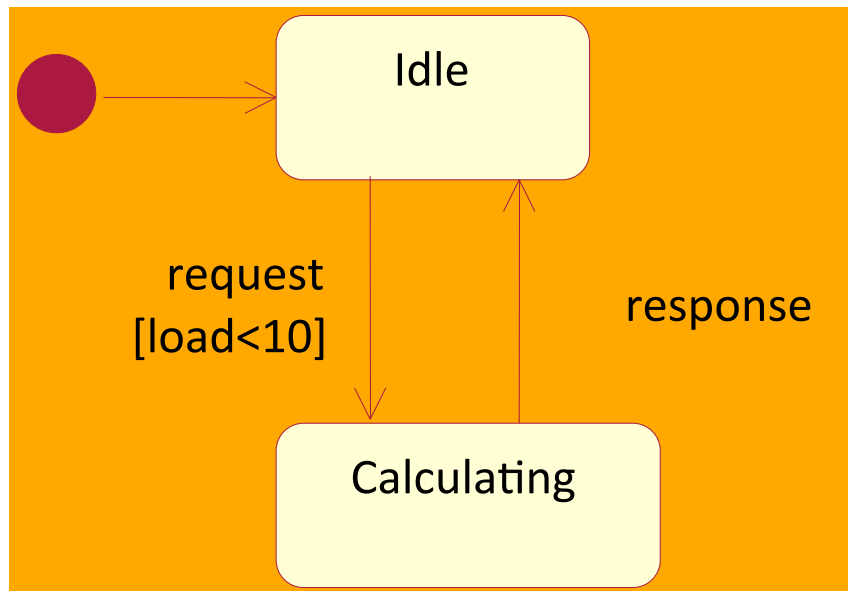
- Grafikus leírás
  - + Könnyen olvasható: Intuitív, könnyen érthető jelölésrendszer
  - + Biztonságosan írható: Csak szintaktikailag helyes modellek készíthetők
  - Nehéz írni: A grafikus szerkesztés lassabb...
  - Nem jól skálázódik
    - Nagy modellek kezelése problémás
- Eclipse technológiák: jövő héttől

# Konkrét szintaxis

- Szöveges leírás
  - + Könnyen írható
    - Komplex kifejezések megadása gyorsan
  - + Jól skálázódik
    - 10k soros fájl jól kezelhető
  - Nehéz olvasni
    - Áttekinthetőség, kapcsolatok megjelenítésének hiánya
  - Nehéz karbantartani
    - Névvél történő hivatkozások
- Eclipse technológiák: októberben

# Példa: konkrét szintaxis

- Grafikus jelölés



- Szöveges jelölés

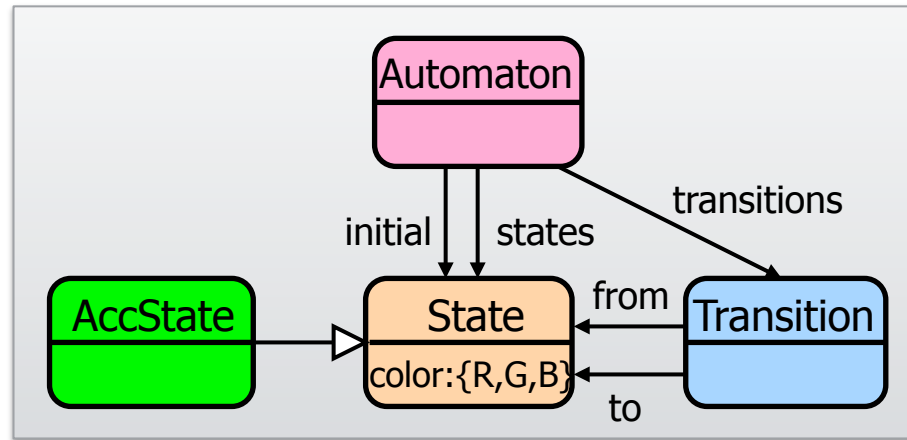
```
void request() {
    if(state == IDLE &&
        this.load < 10)
        state =
        CALCULATING;
}
```

```
void response() {
    if (state ==
        CALCULATING)
        state = IDLE;
}
```

# Absztrakt szintaxis (metamodell)

- Metamodell: a modellező nyelv modellje
- Cél: definiálni...
  - a nyelv által tartalmazott koncepciókat
  - és az ezek közötti lehetséges kapcsolatokat
- Tartalma:
  - Alapelemek definíciója
  - Kapcsolatok az elemek között
  - Absztrakció/finomítás (Taxonómia, Ontológia) az elemek között
  - Kényszerek (pl. számosság)
  - (Egyéb jólformáltsággal kapcsolatos szabályok)

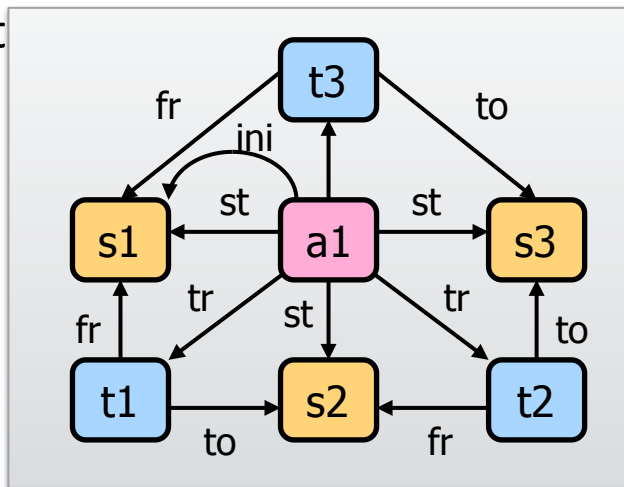
# Metamodellek és példányok



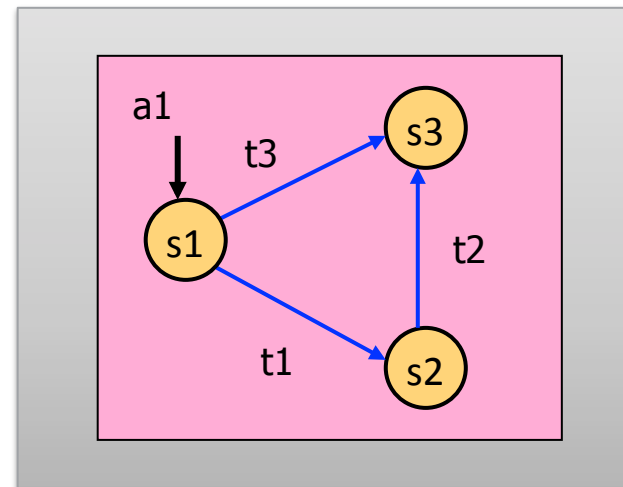
Metamodel

Metamodel szint

Modell szint

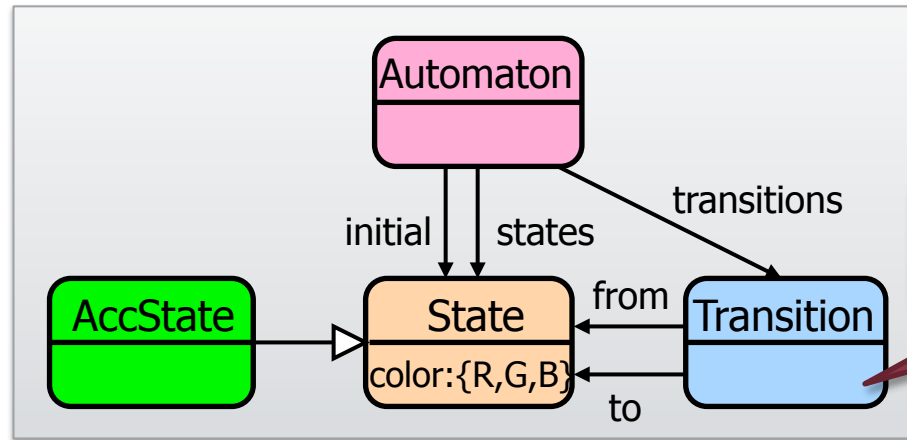


Absztrakt szintaxis



Konkrét szintaxis

# Metamodellek és példányok

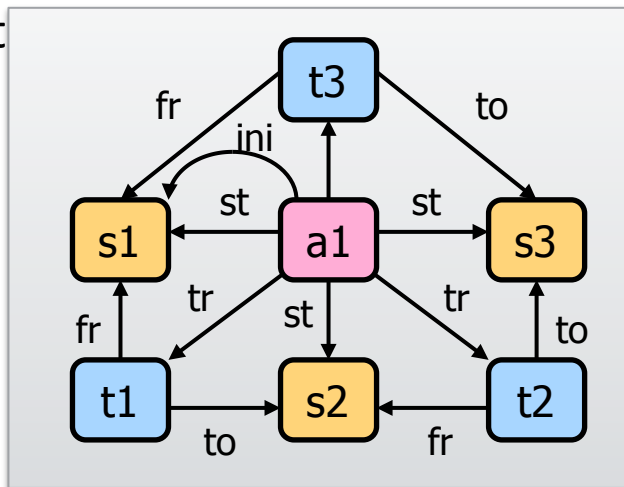


Osztály

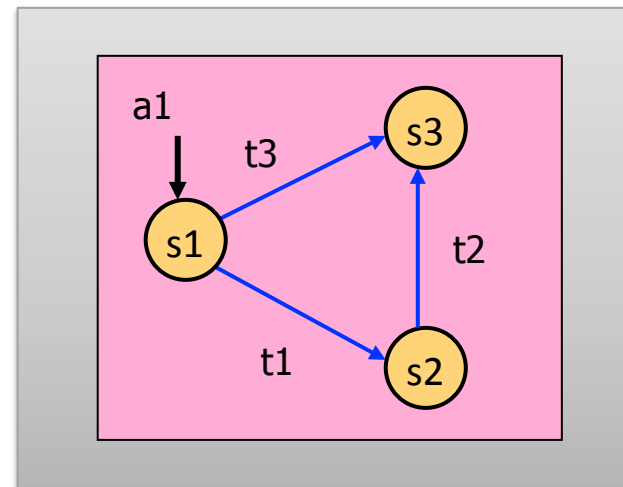
Metamodell szint

Metamodell

Modell szint



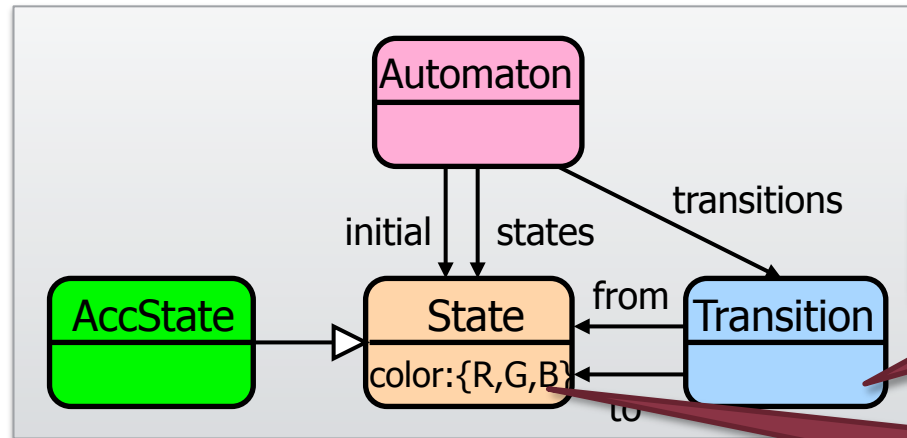
Absztrakt szintaxis



Konkrét szintaxis



# Metamodellek és példányok



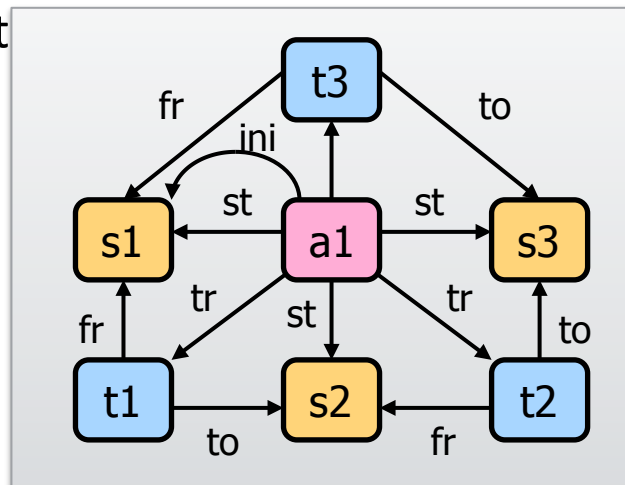
Osztály

Attribútum

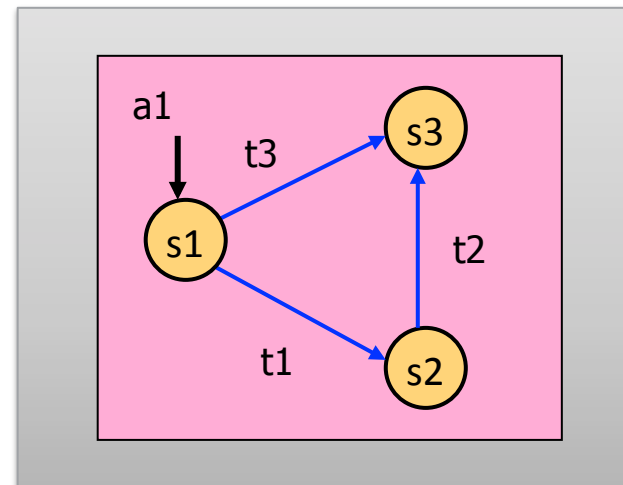
Metamodel

Metamodel szint

Modell szint

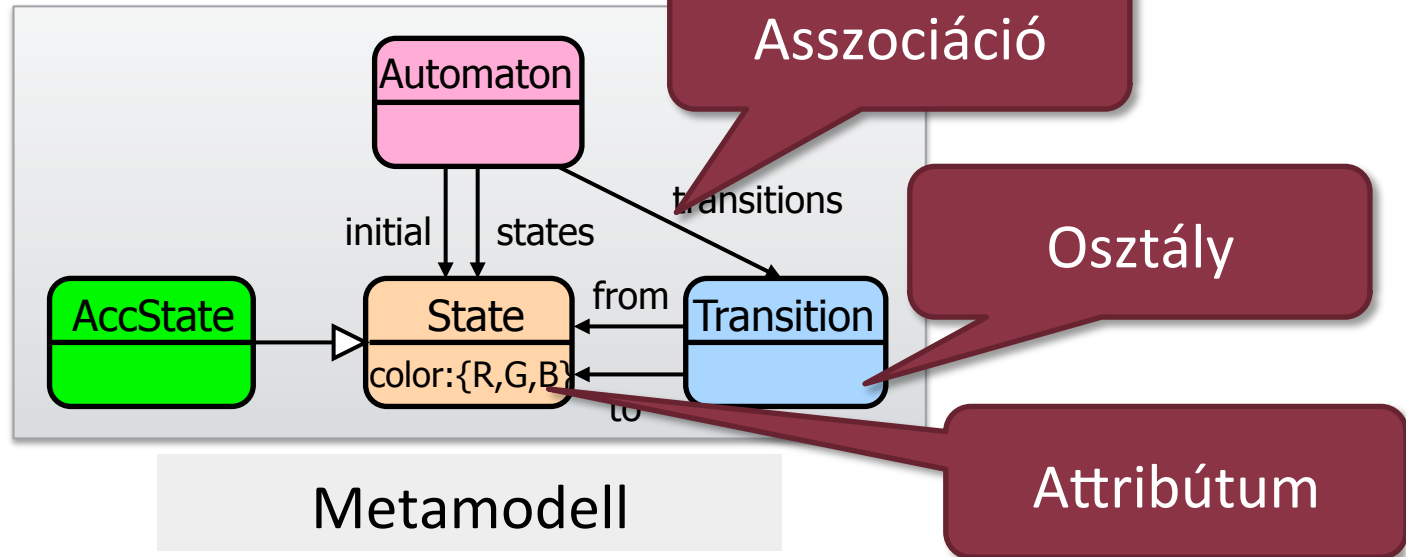


Absztrakt szintaxis



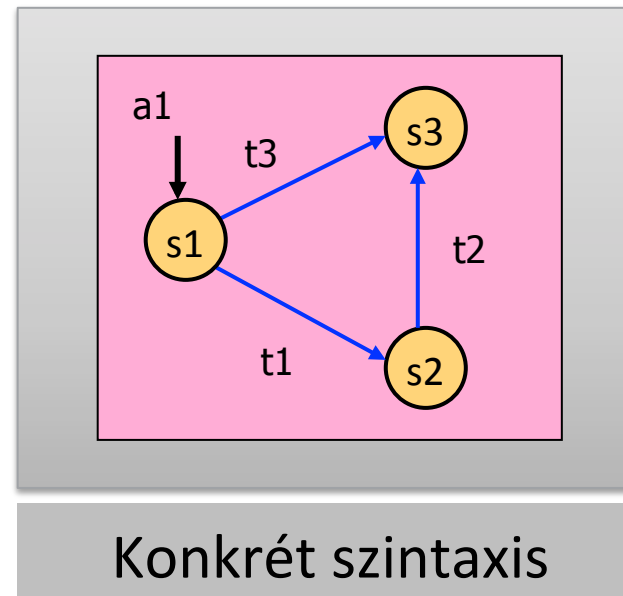
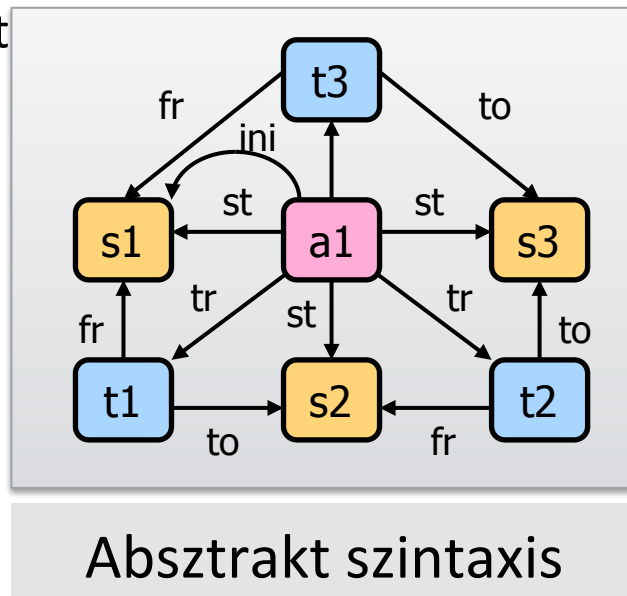
Konkrét szintaxis

# Metamodellek és példánvok

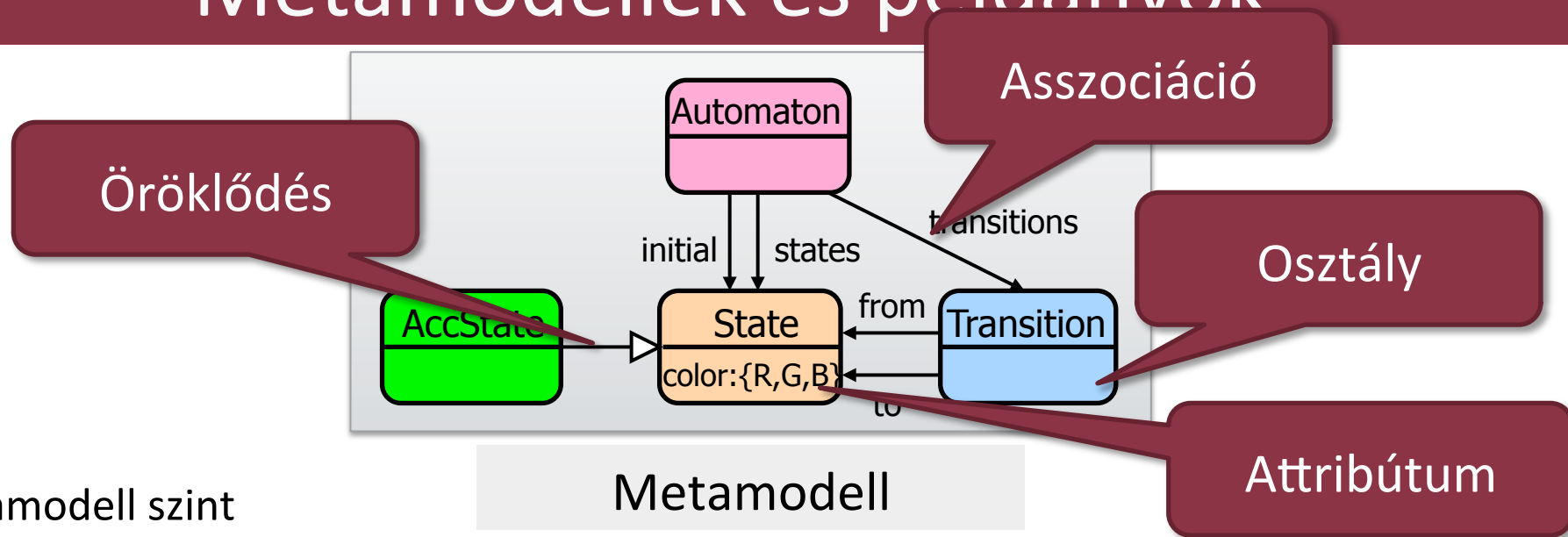


Metamodell szint

Modell szint



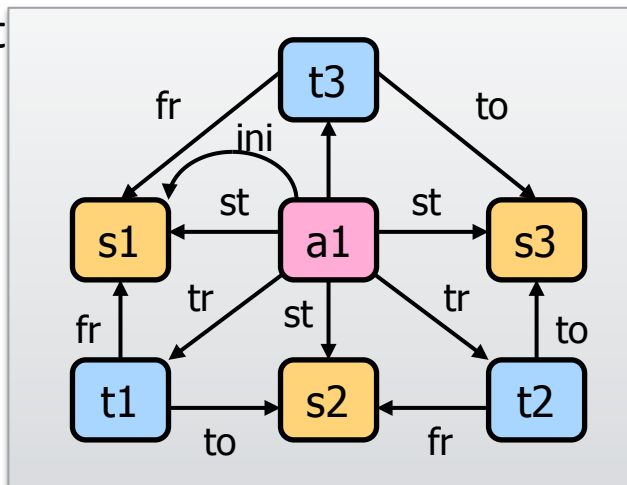
# Metamodellek és példánvok



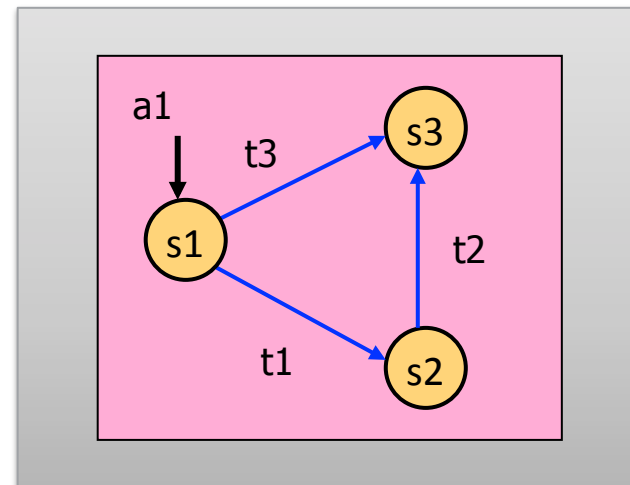
Metamodell szint

Metamodell

Modell szint

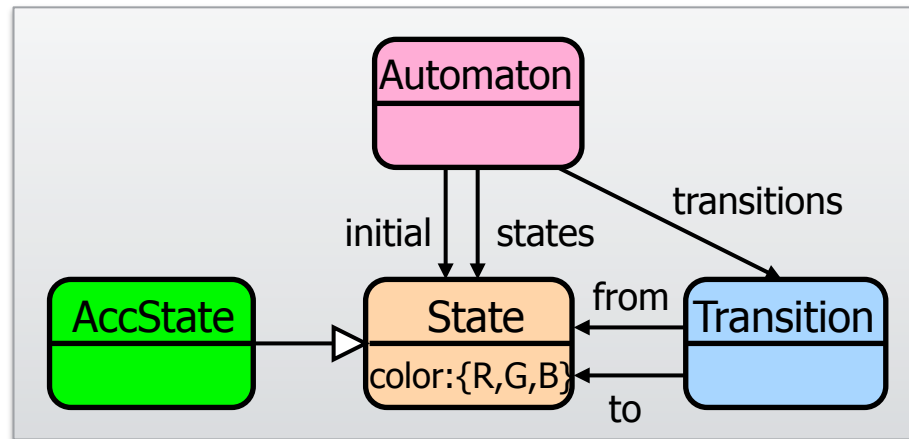


Absztrakt szintaxis



Konkrét szintaxis

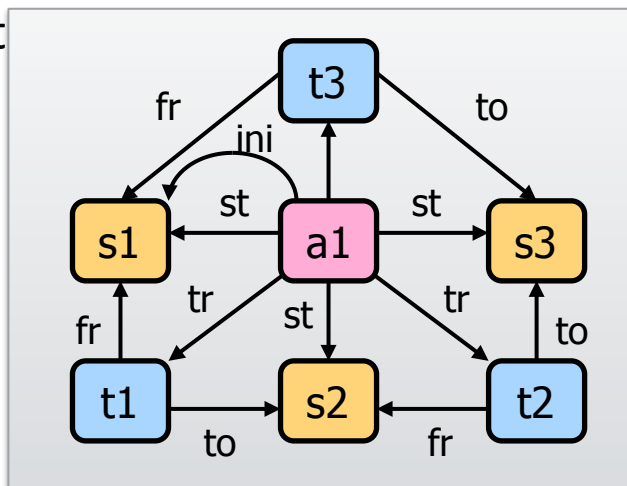
# Metamodellek és példányok



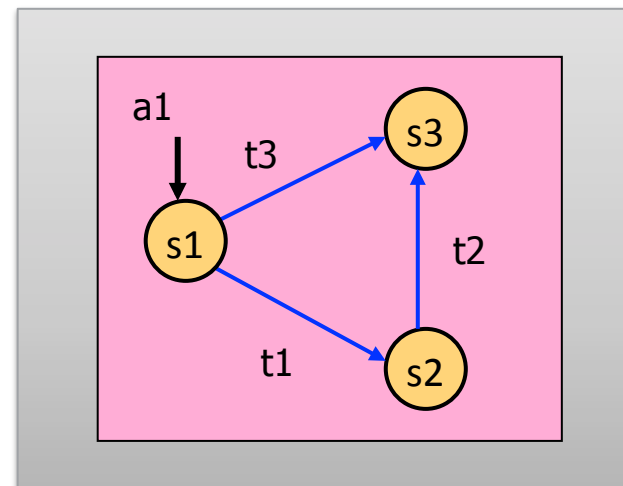
Metamodel

Metamodel szint

Modell szint

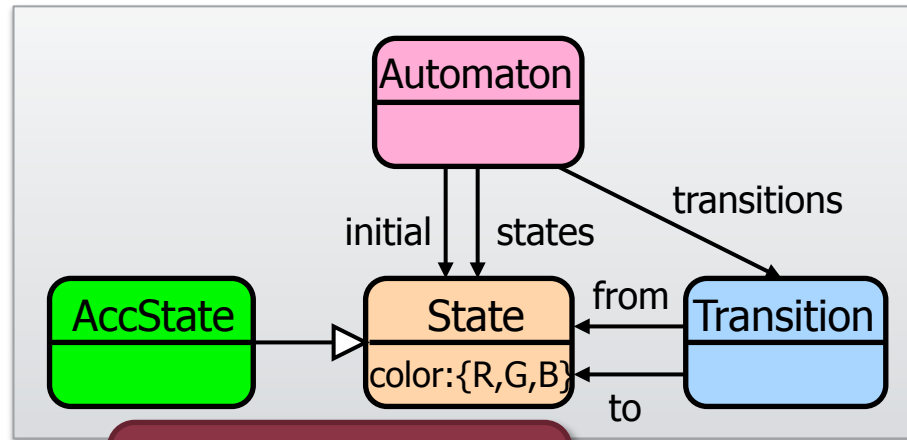


Absztrakt szintaxis



Konkrét szintaxis

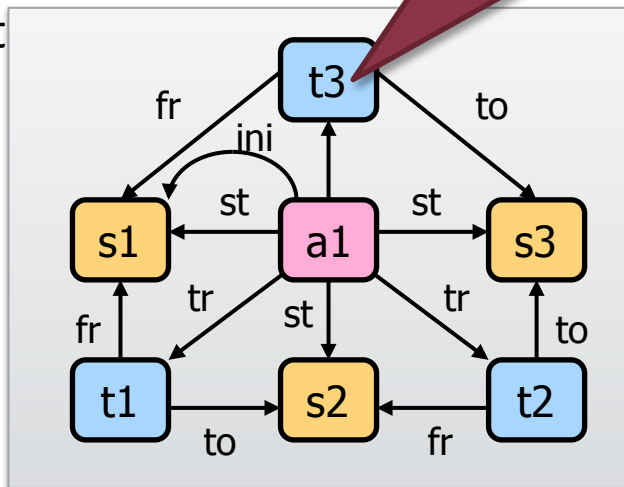
# Metamodellek és példányok



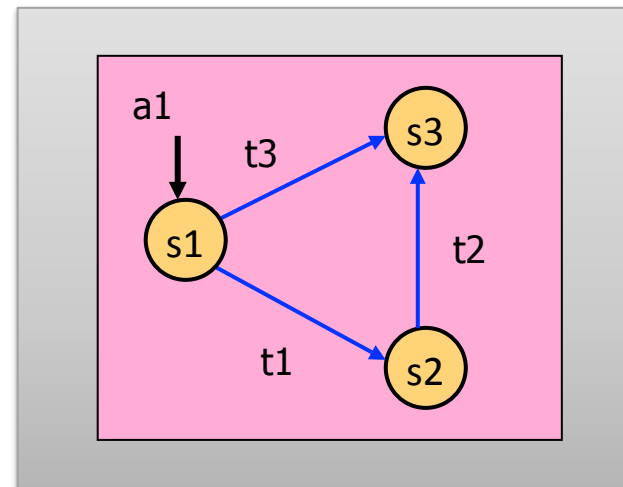
Objektum

Metamodell szint

Modell szint

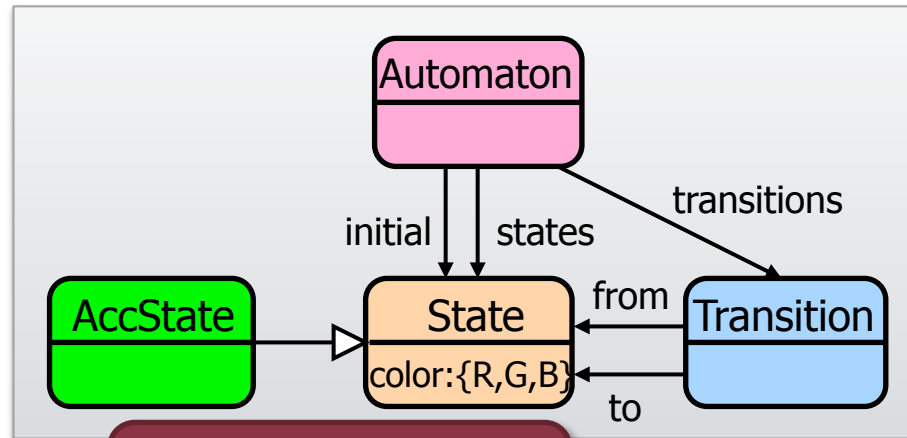


Absztrakt szintaxis



Konkrét szintaxis

# Metamodellek és példányok

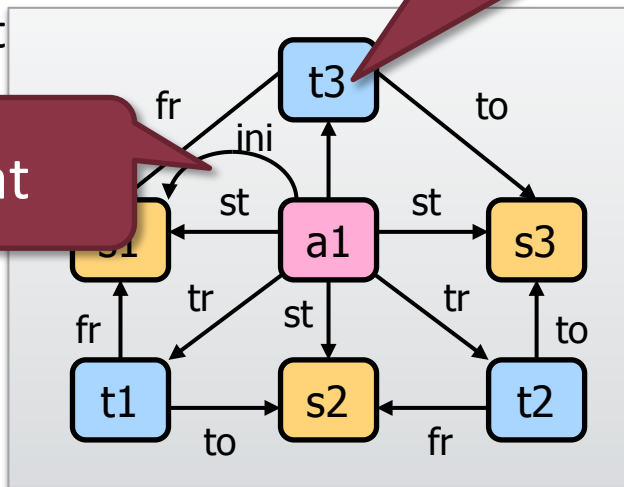


Objektum

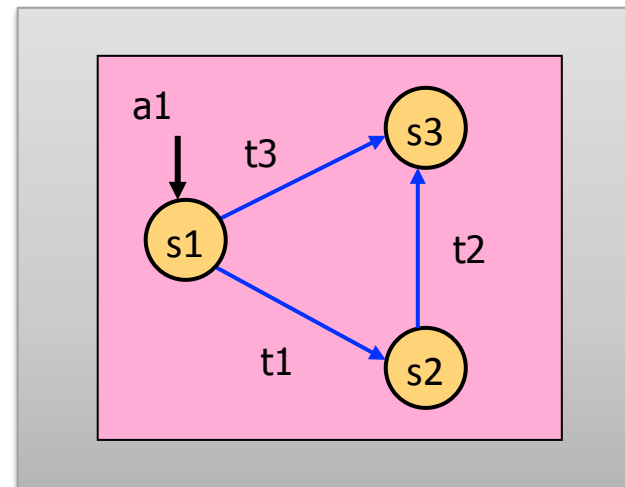
Metamodell szint

Modell szint

Kapcsolat

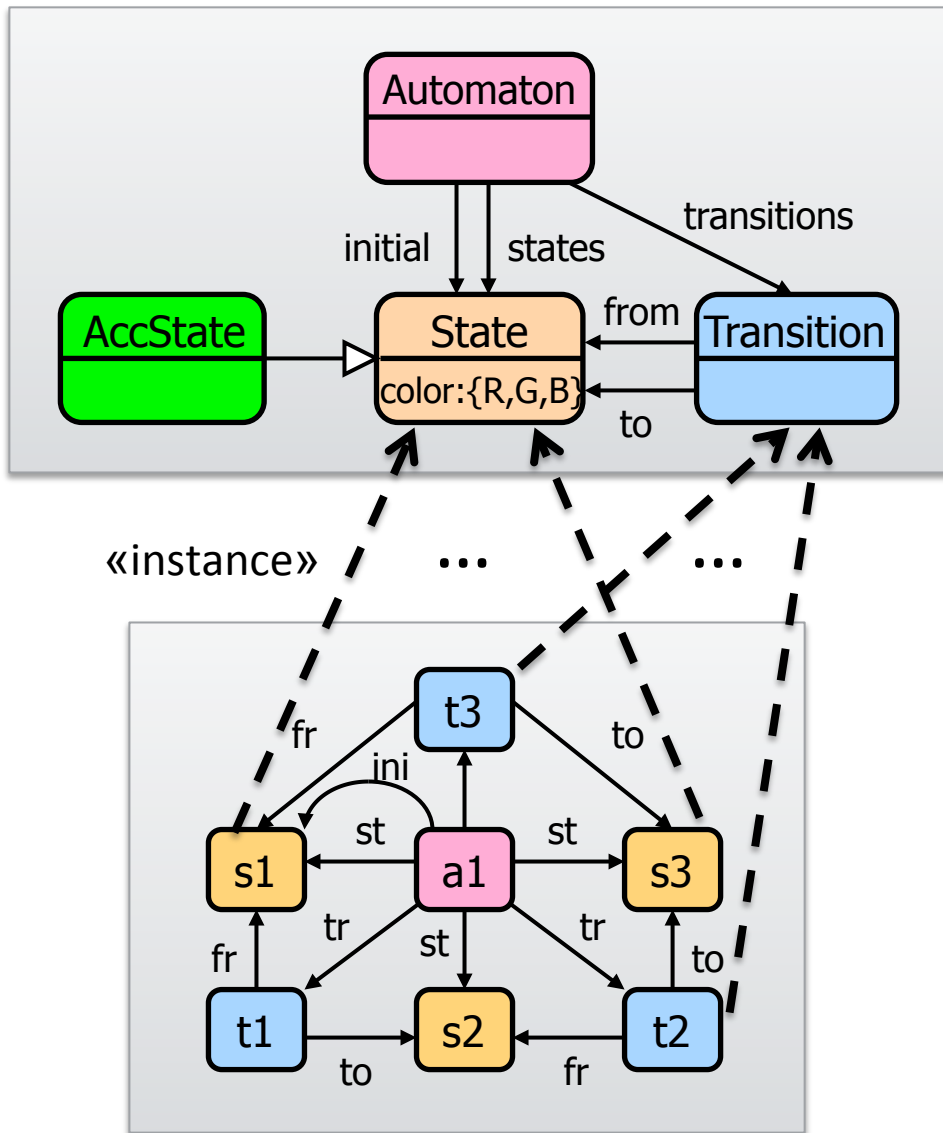


Absztrakt szintaxis

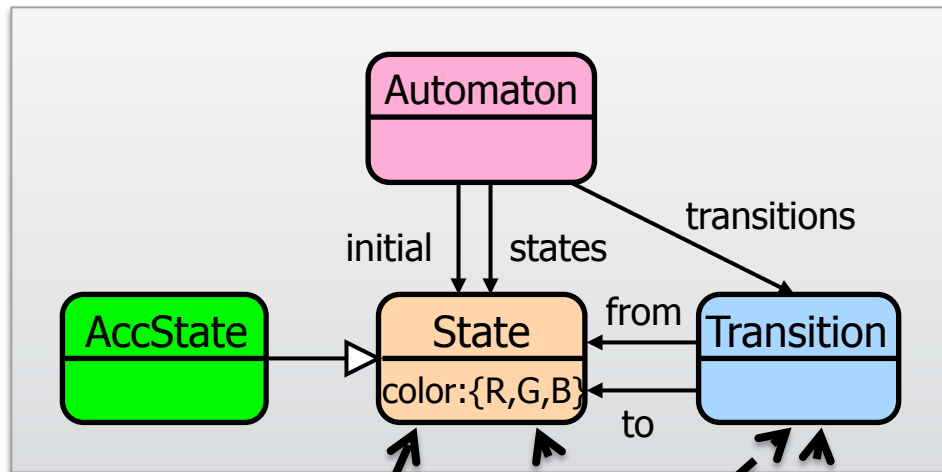


Konkrét szintaxis

# Példányosítás

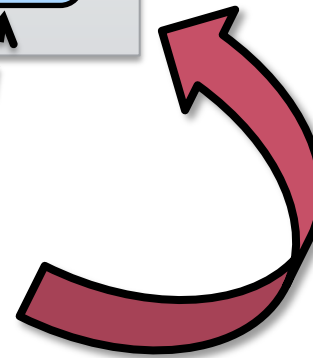
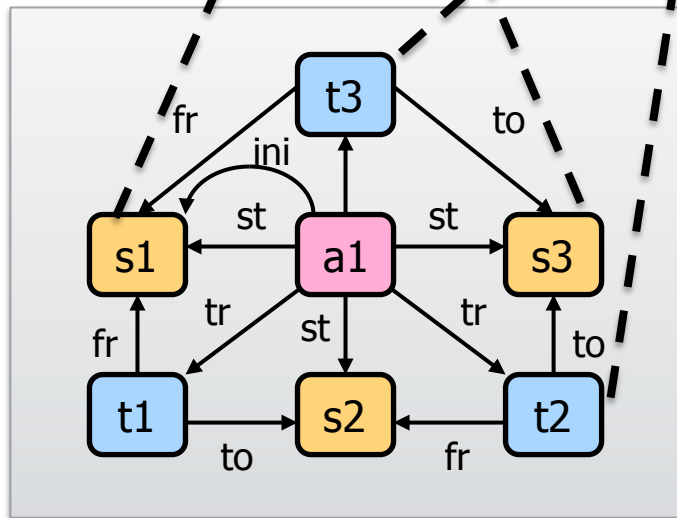


# Példányosítás



Kapcsolat  
modellszint és  
metamodell szint  
között

«instance»





# Létező metamodellek

- Különböző alkalmazási domainek az UML körül
  - SysML (rendszerfejlesztés)
  - SPEM (folyamat modellezés)
  - CWM (adattárházak)
  - GRM (általános erőforrás metamodell)
  - EDOC (nagyvállalati elosztott rendszerek)

# Jólformáltsági szabályok

- Számossági kényszerek
  - Legfeljebb egy: 0..1
  - Sok: \*
- Aggregáció/Tartalmazás
  - Minden modellelemnek legfeljebb egy szülő
- Nyelvspecifikus kényszerek
  - Pl. egyedi nevek
  - Pl. OCL-ben (Object Constraint Language) kifejezhető

# Szemantika

- Szemantika: a nyelv fogalmainak jelentése
- Hogyan értelmezzük a modellt
- Statikus szemantika
  - Eddig erről volt szó
- Dinamikus szemantika
  - Lehetséges viselkedés
  - Állapotváltozások

# Szemantika

- Fő megközelítések:
  - Denotációs
    - Egyik nyelv fogalmainak lefordítása a másik nyelvre
    - Fordított
  - Operációs
    - A nyelvi fogalmak viselkedésének modellezése
    - Interpretált
  - ~~Axiomatikus~~
    - ~~Logikai formulák~~
    - ~~Nehezen értelmezhető~~

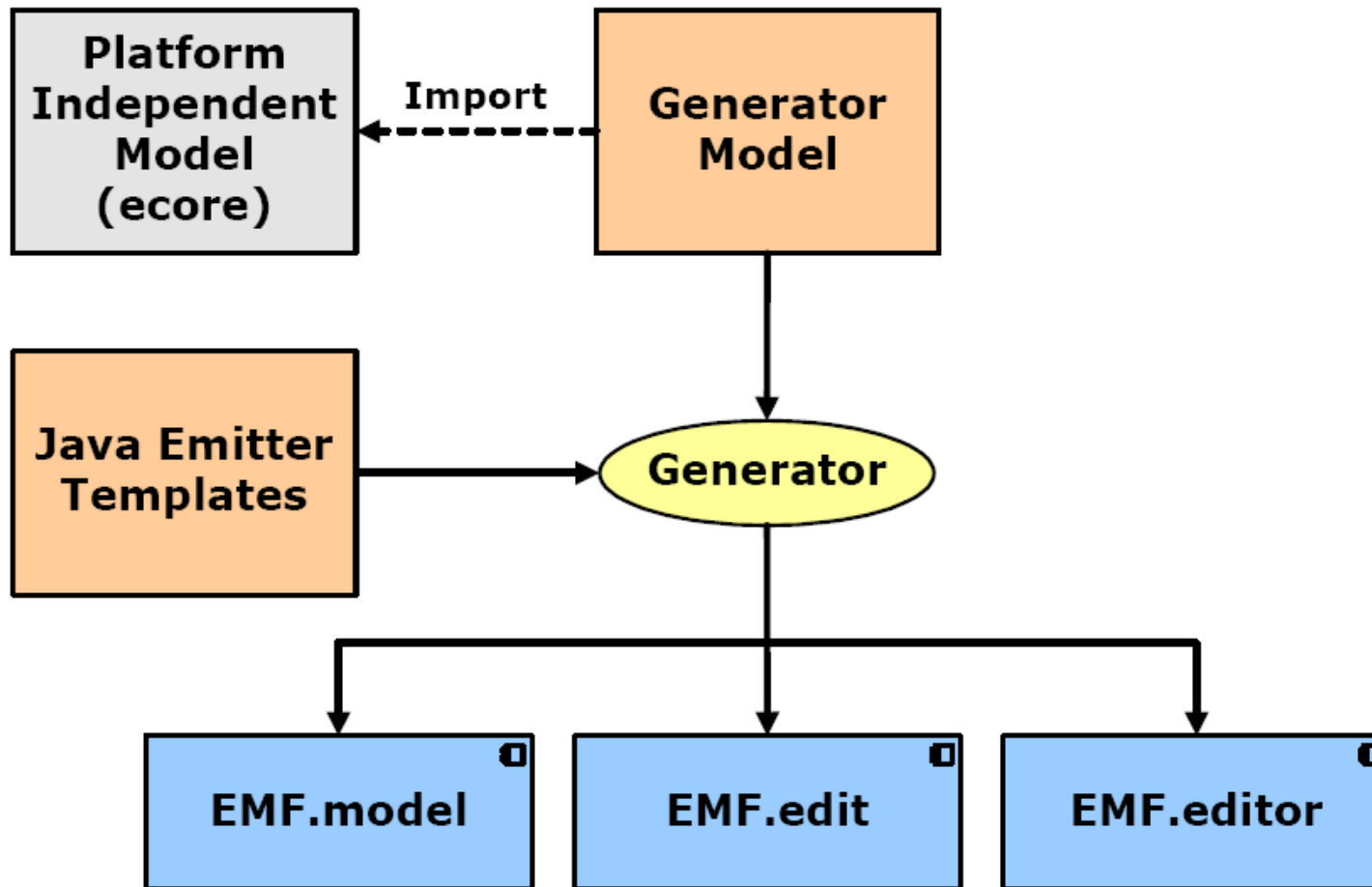
# Eclipse Modeling Framework (EMF)

Metamodellezés Eclipse alatt

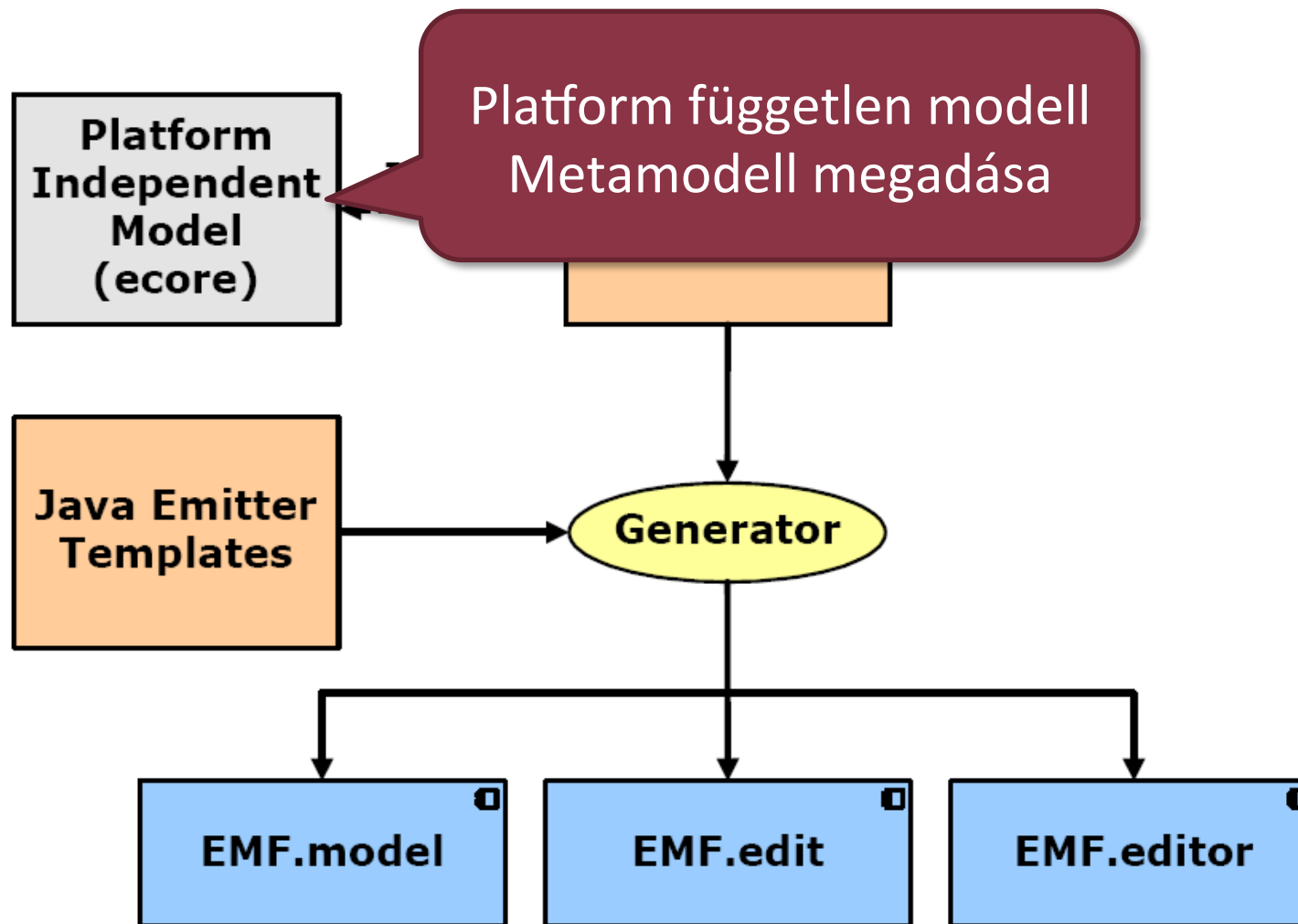
# Eclipse Modeling Framework

- Modellezési komponens Eclipse alá
- Lehetőség domain-specifikus nyelvek definiálására
- Szerkesztés
  - Alapvető parancsok
  - Értesítés változásokról
  - Visszavonás
  - Alapvető editor komponens
- XML/XMI export-import támogatás
- Saját metamodellező mag

# Az EMF eszközkészlet

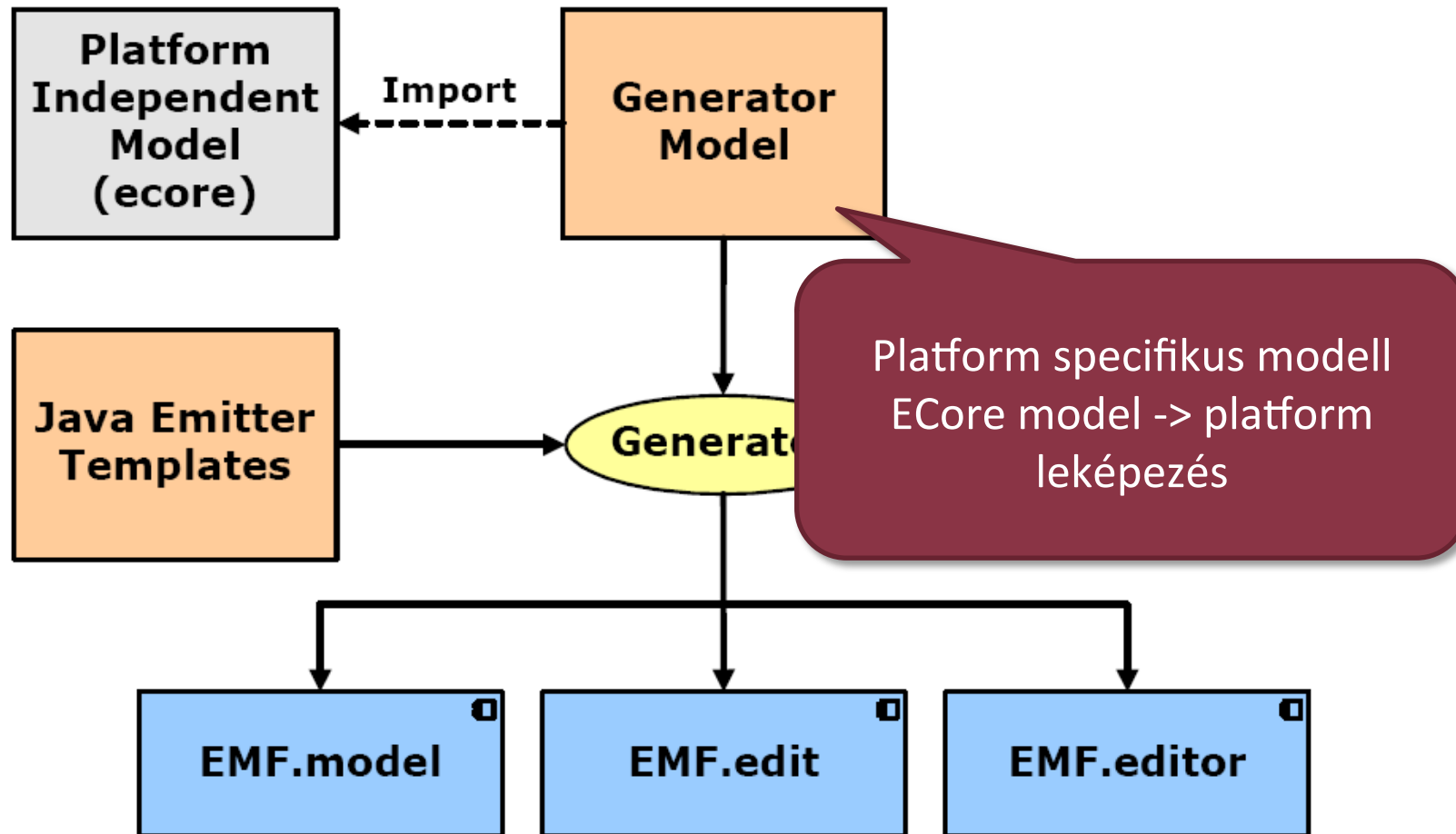


# Az EMF eszközkészlet

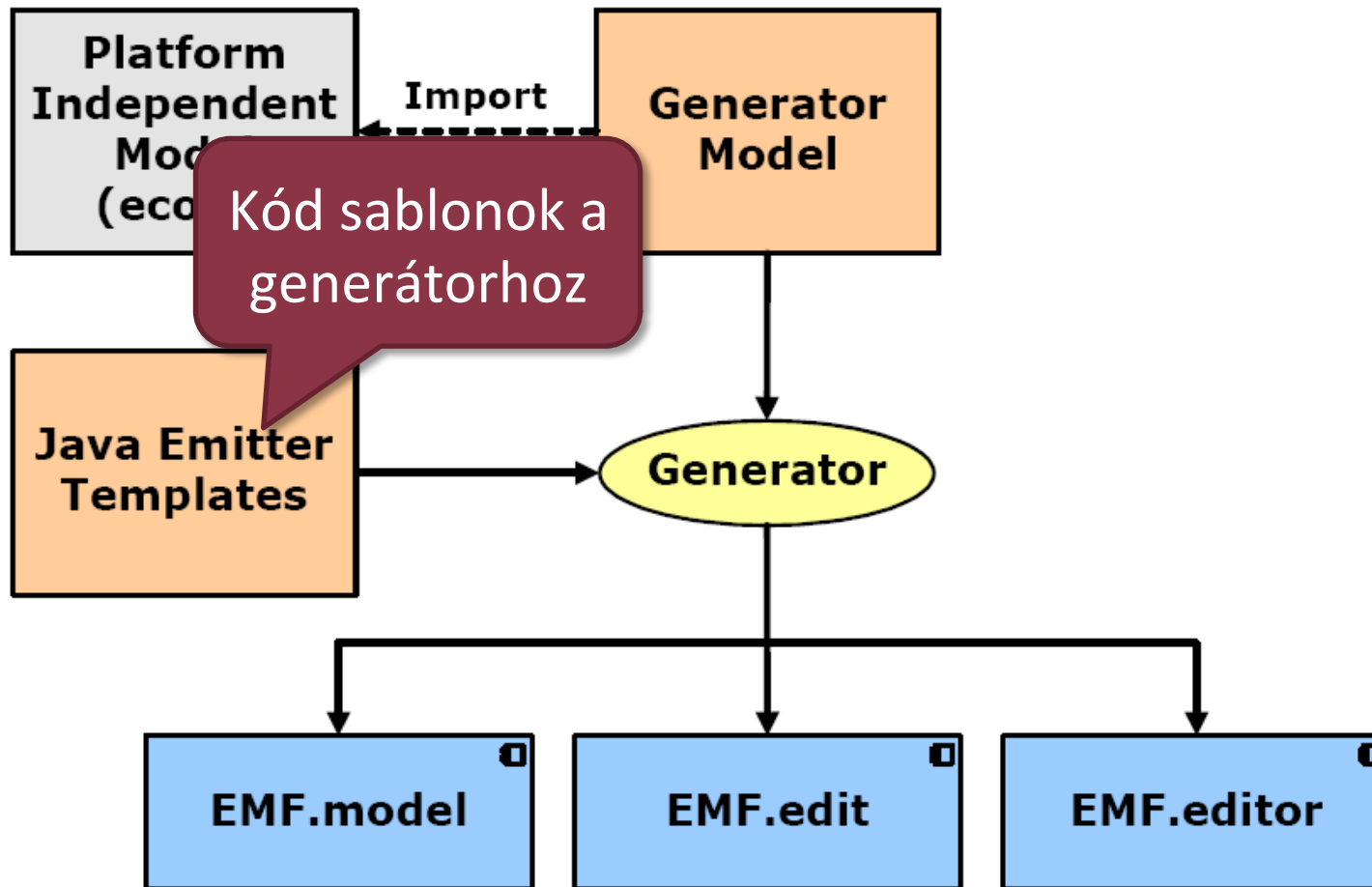




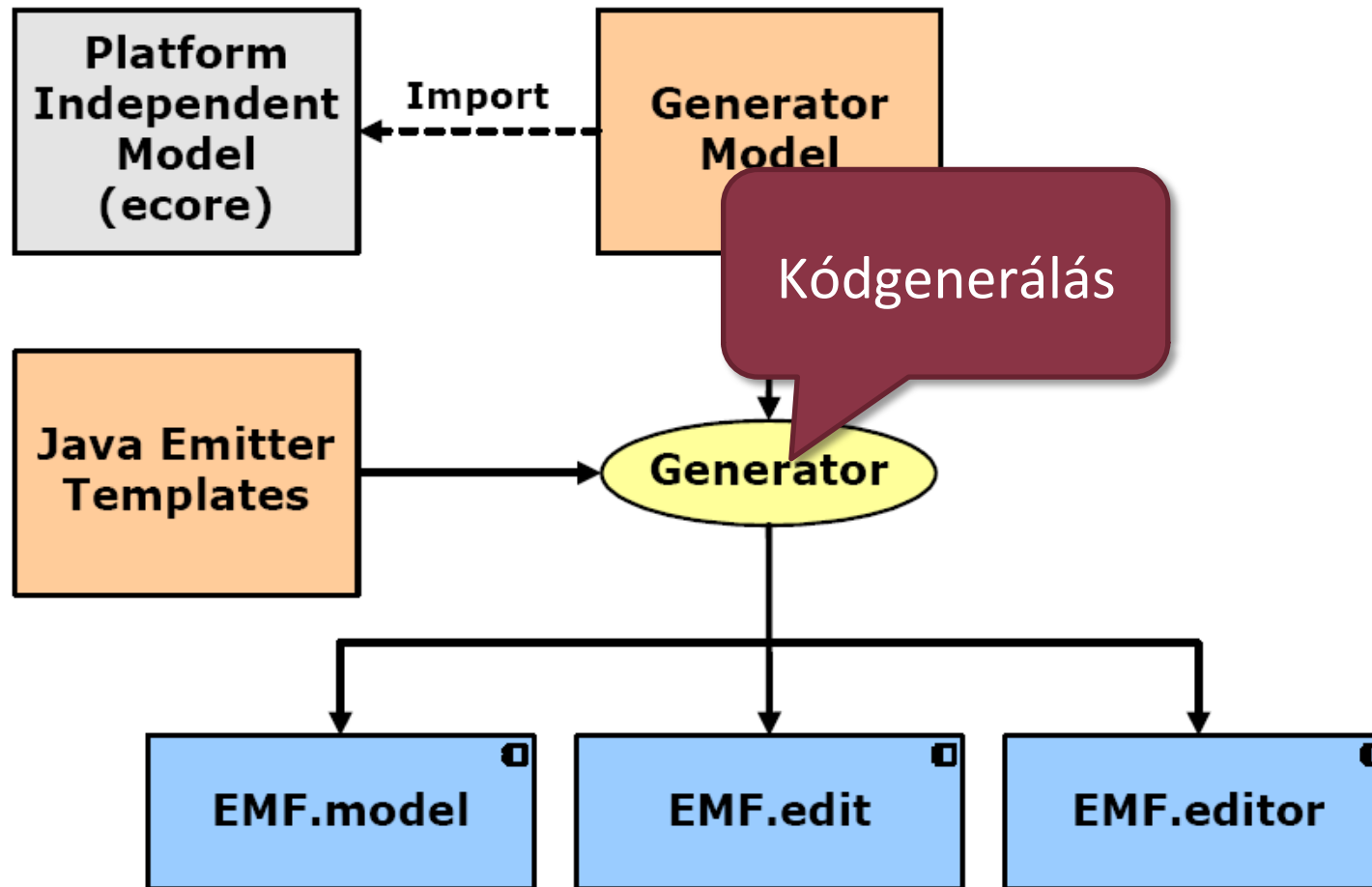
# Az EMF eszközkészlet



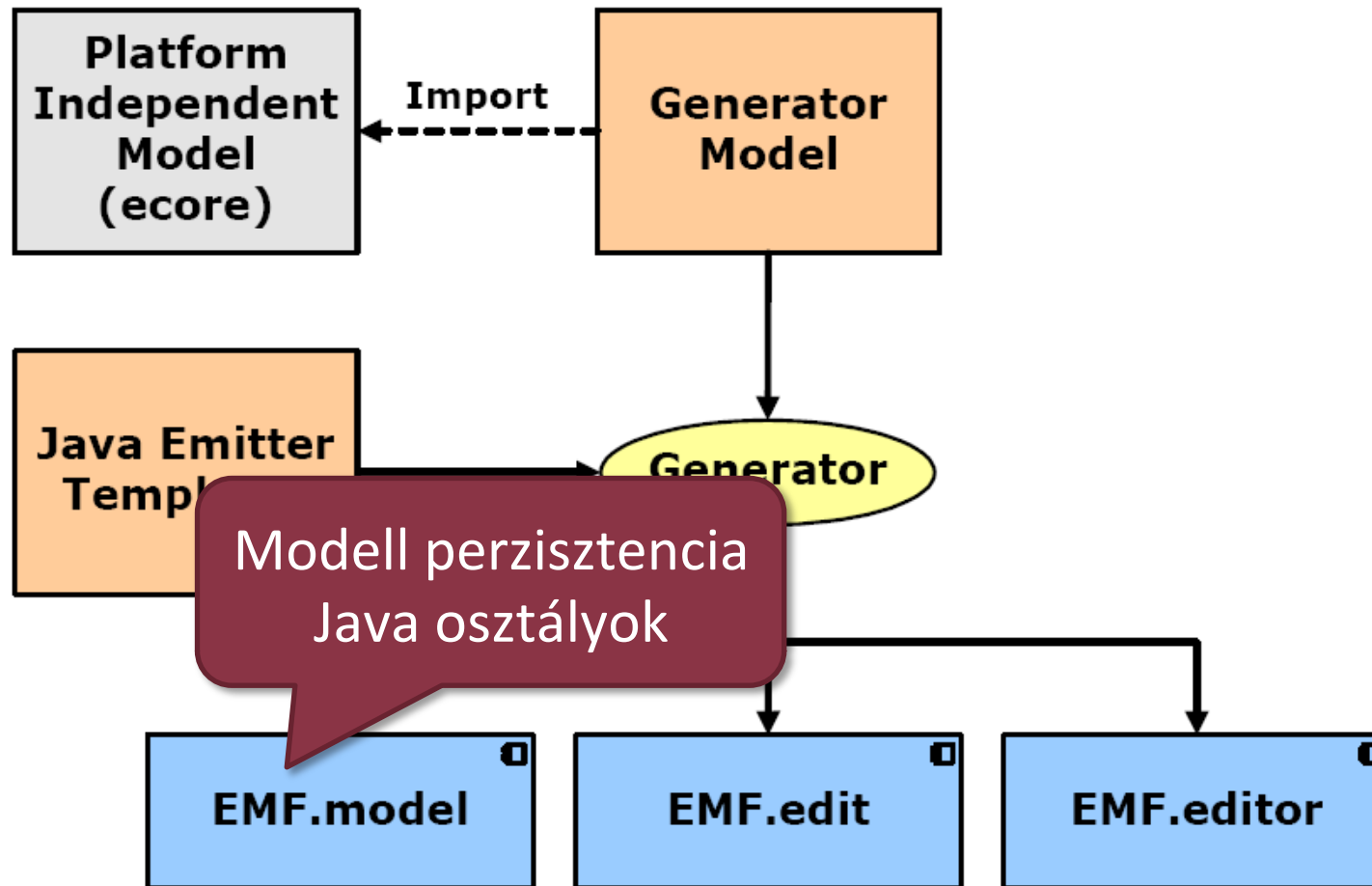
# Az EMF eszközkészlet



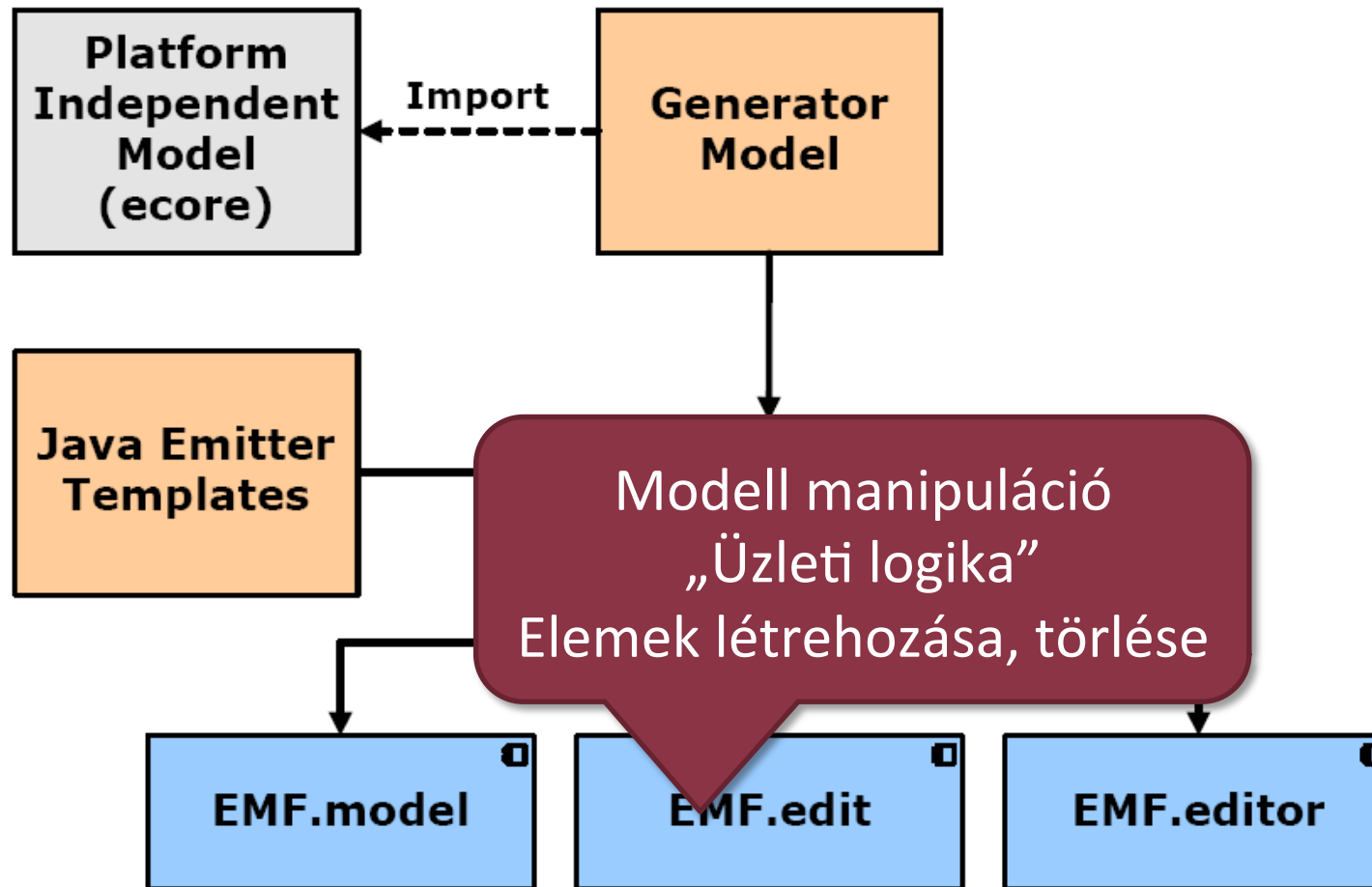
# Az EMF eszközkészlet



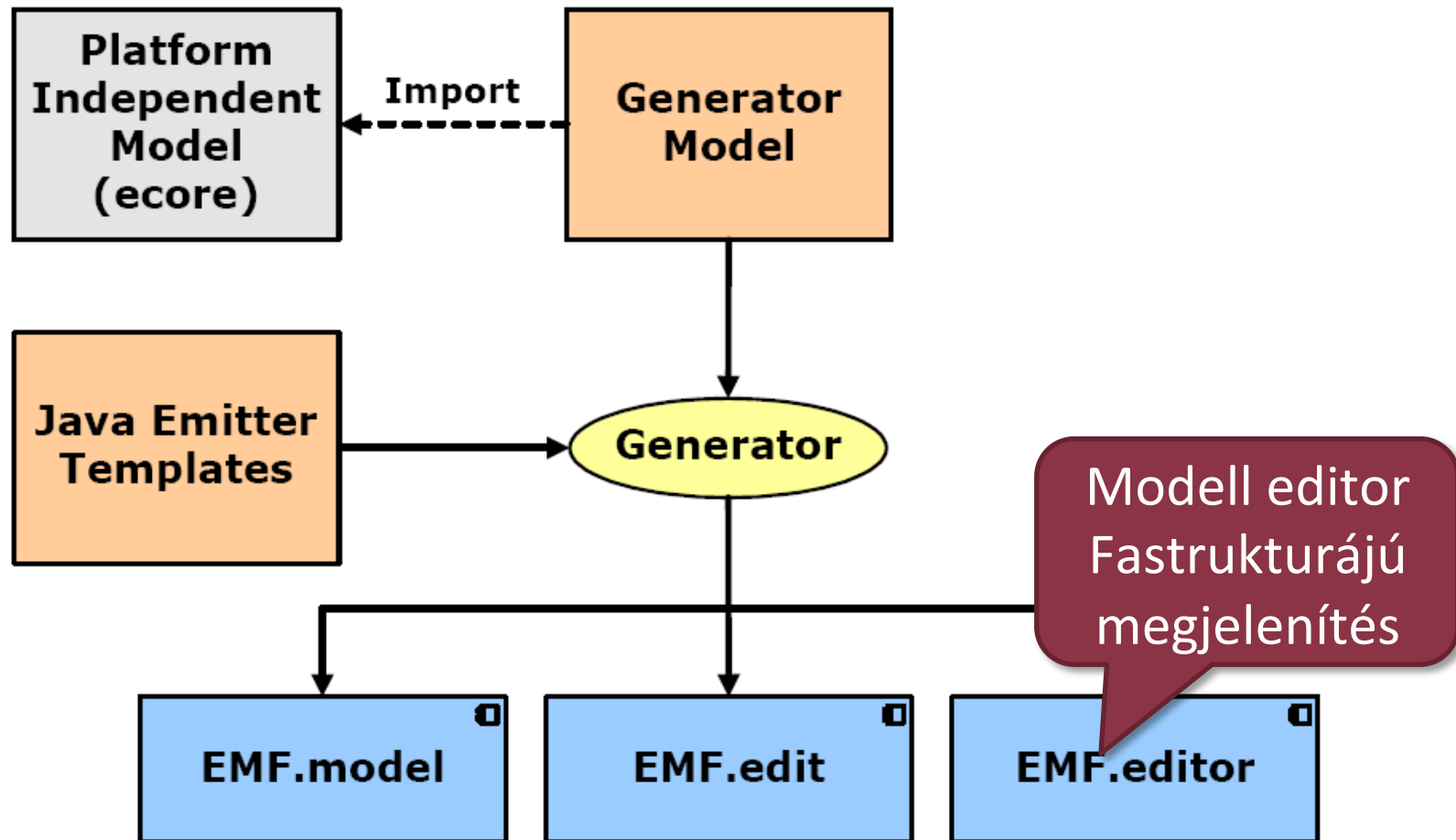
# Az EMF eszközkészlet



# Az EMF eszközkészlet



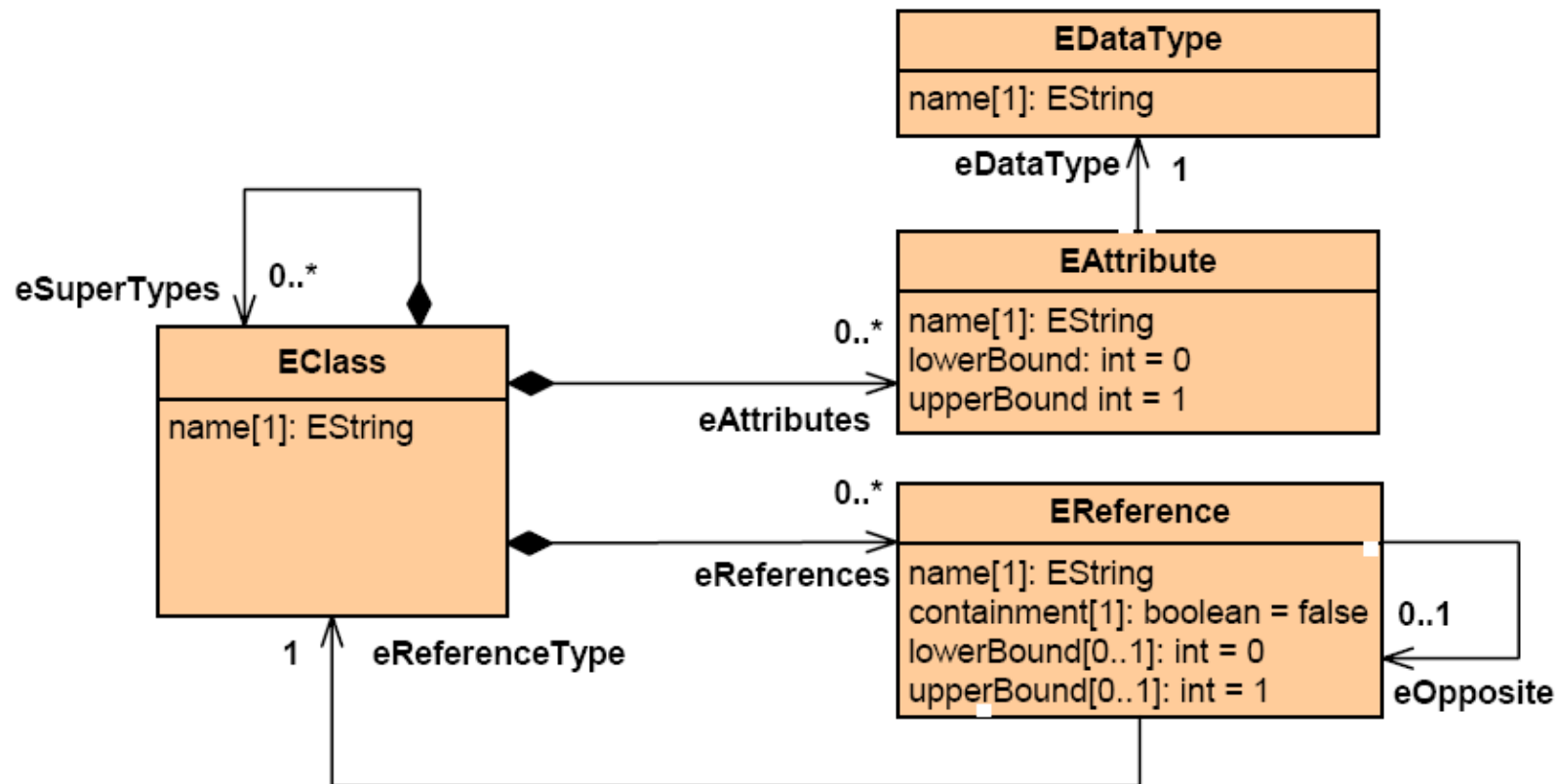
# Az EMF eszközkészlet



# ECore

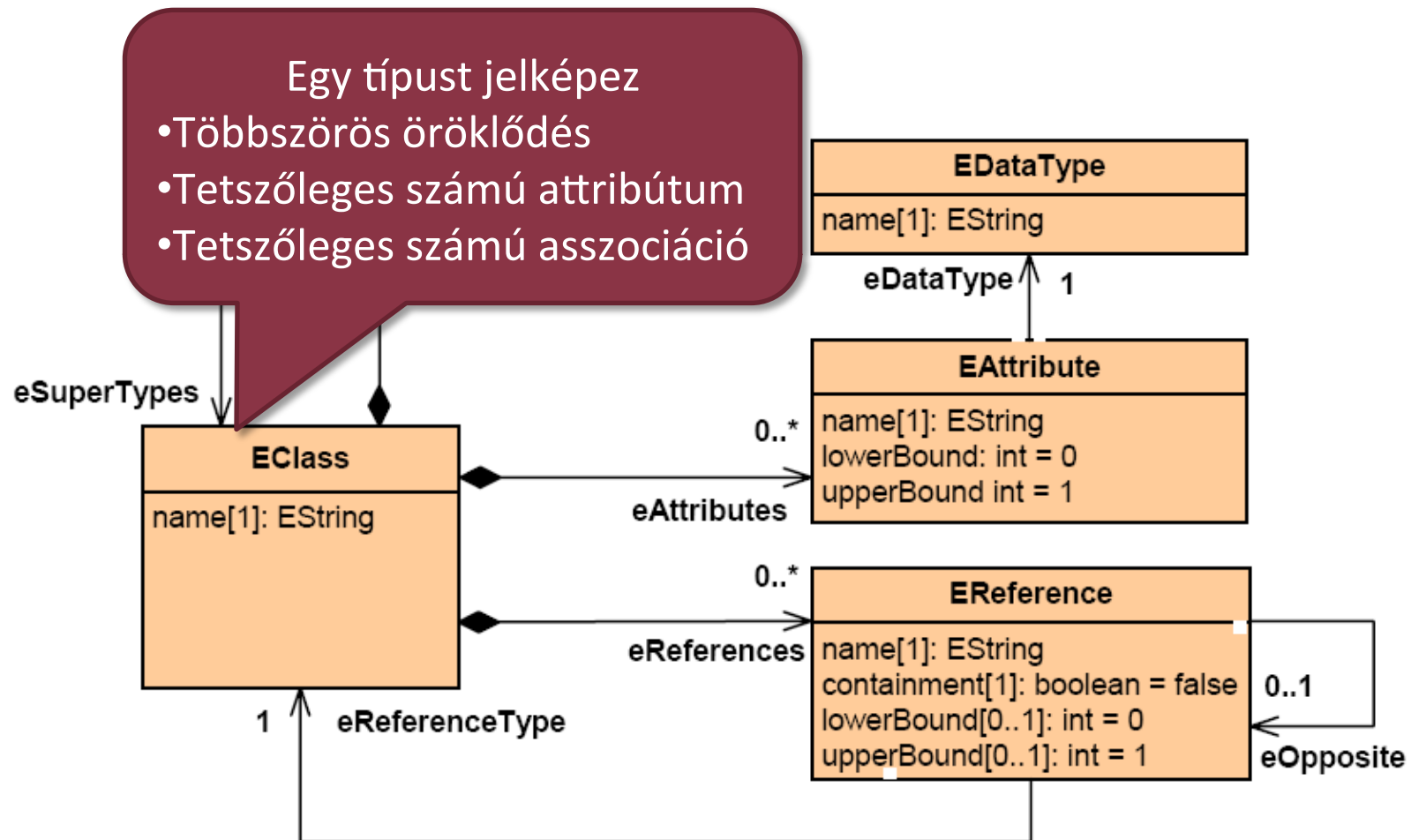
- Az EMF metamodellező nyelve
  - Meta-nyelv: metamodellező nyelv
- A metamodellek platformfüggetlenek
  - További nyelv(ek) az implementáció-közeli modellezésre
- Strukturális modellek definíciójára szolgál

# Ecore – A legfontosabb elemek

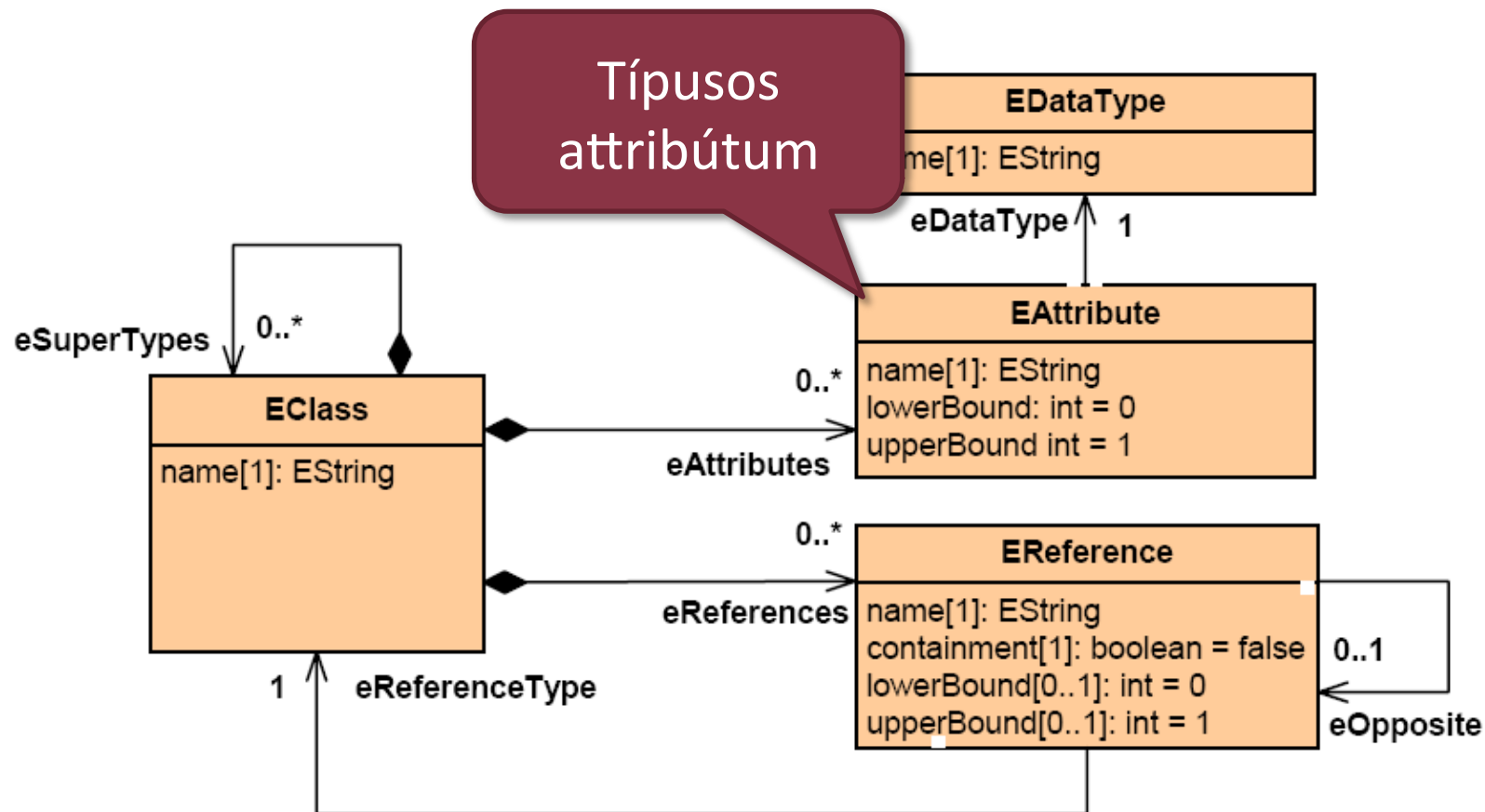




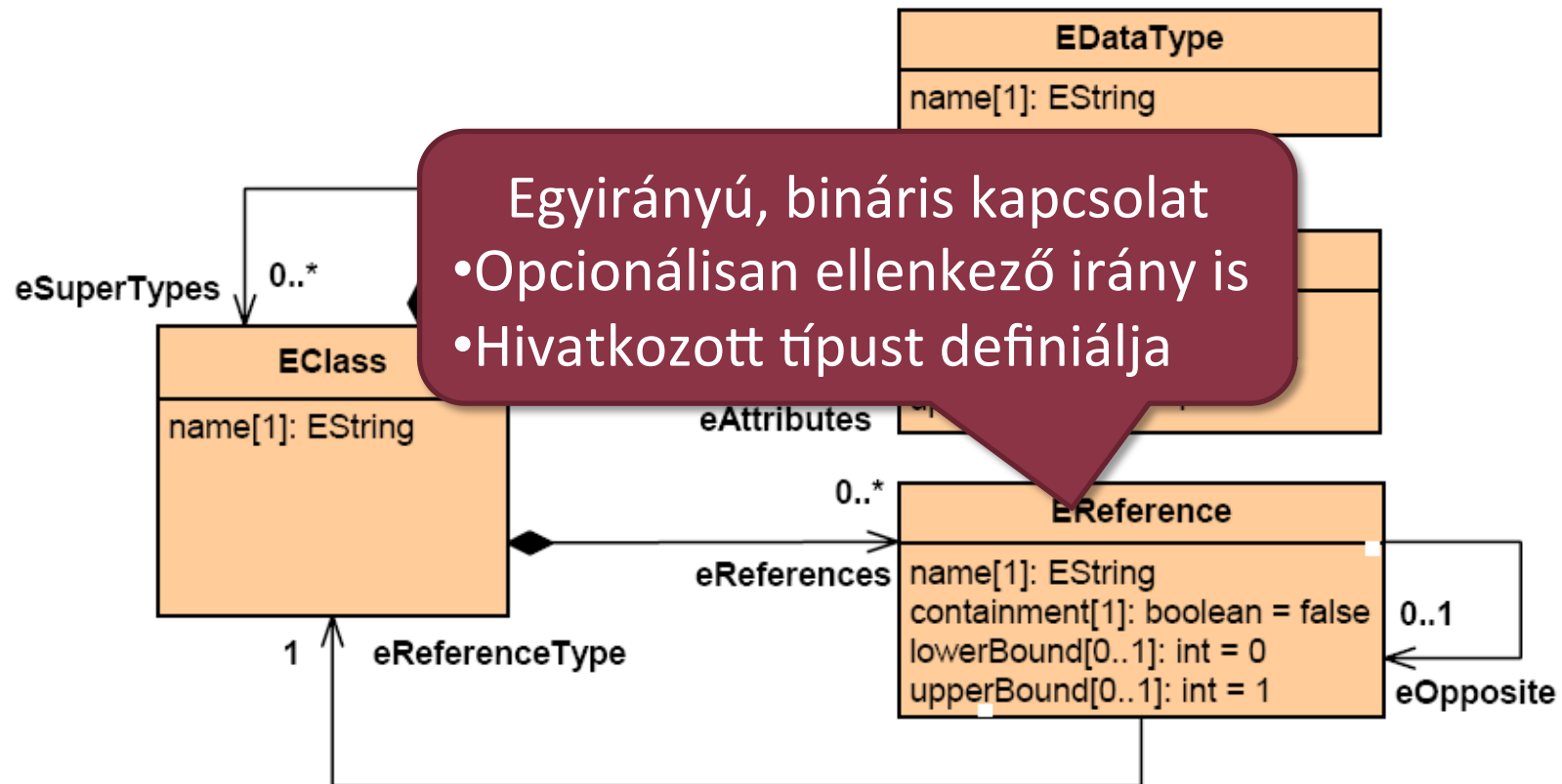
# Ecore – A legfontosabb elemek



# Ecore – A legfontosabb elemek

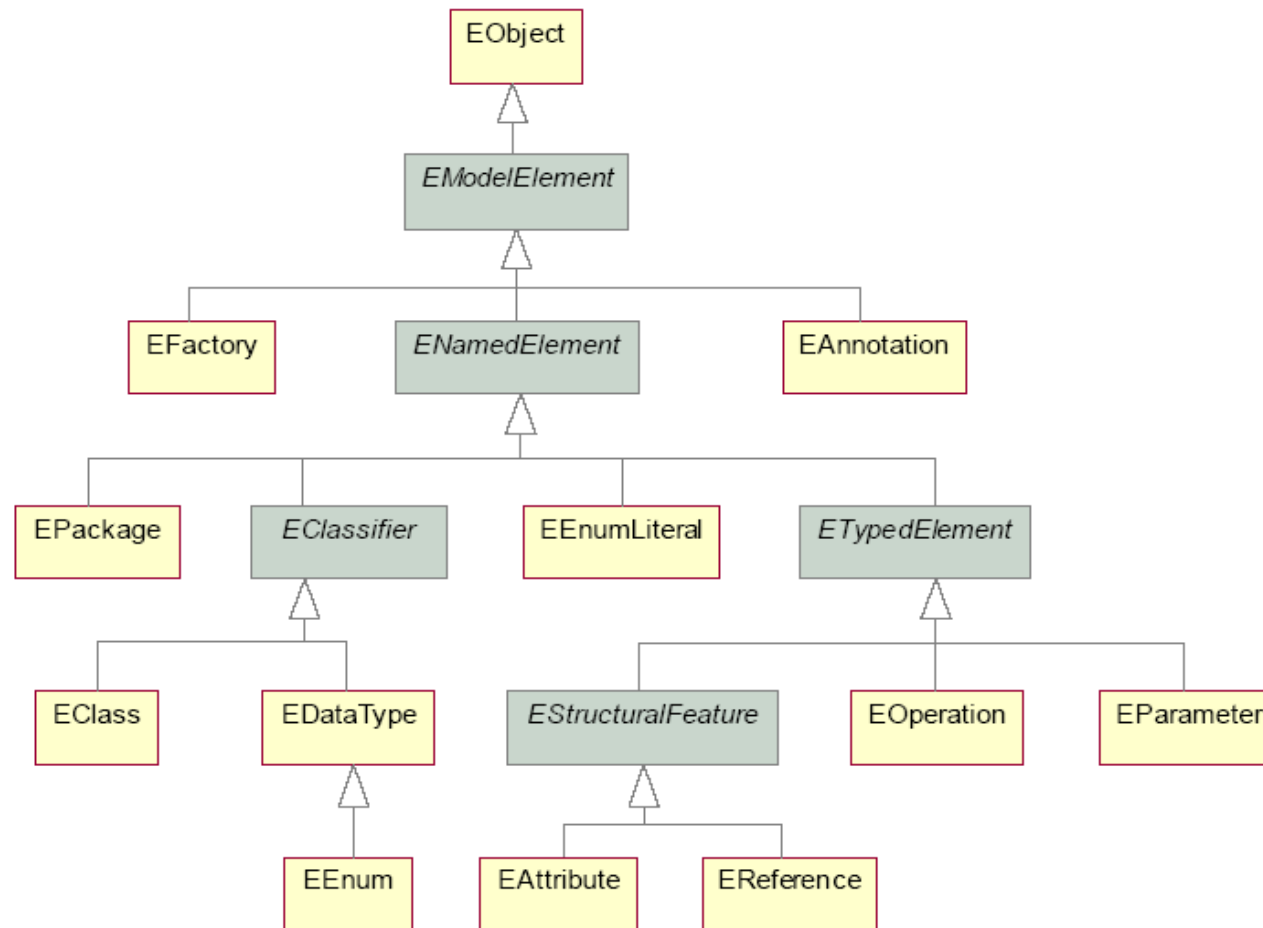


# Ecore – A legfontosabb elemek



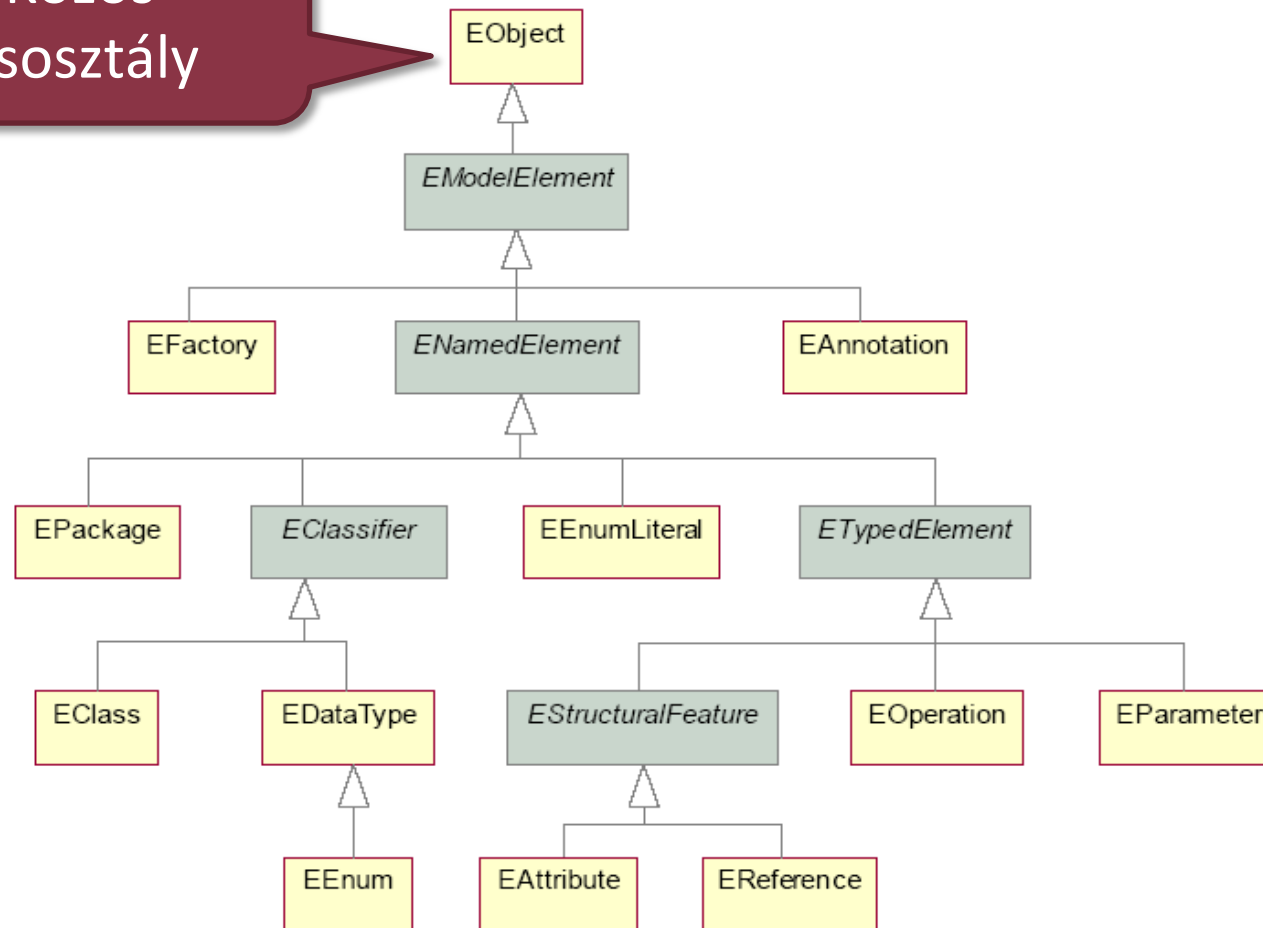
- Egyirányú, bináris kapcsolat
- Opcionálisan ellenkező irány is
  - Hivatkozott típust definiálja

# Teljes ECore hierarchia



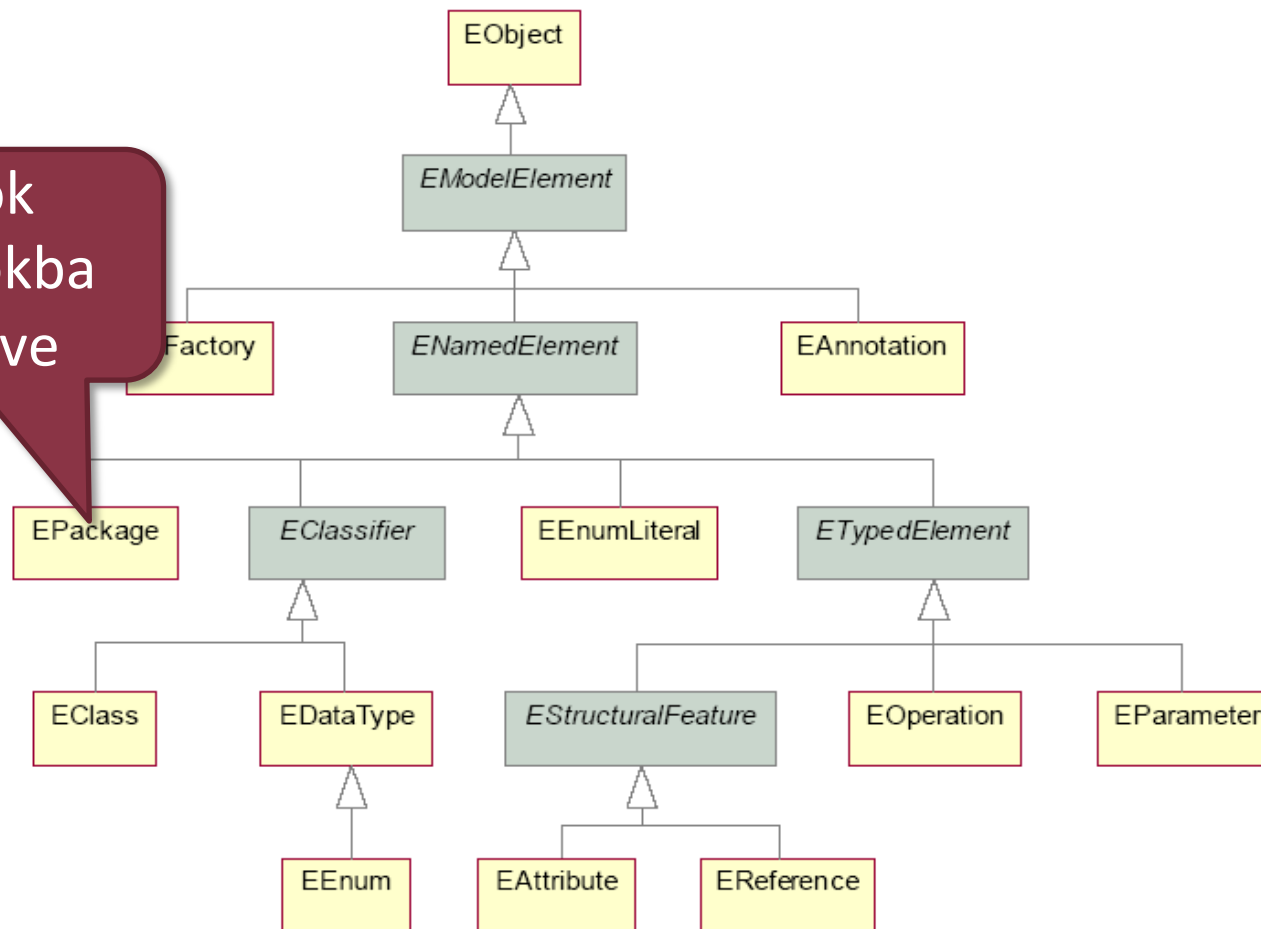
# Teljes ECore hierarchia

Közös  
ősosztály

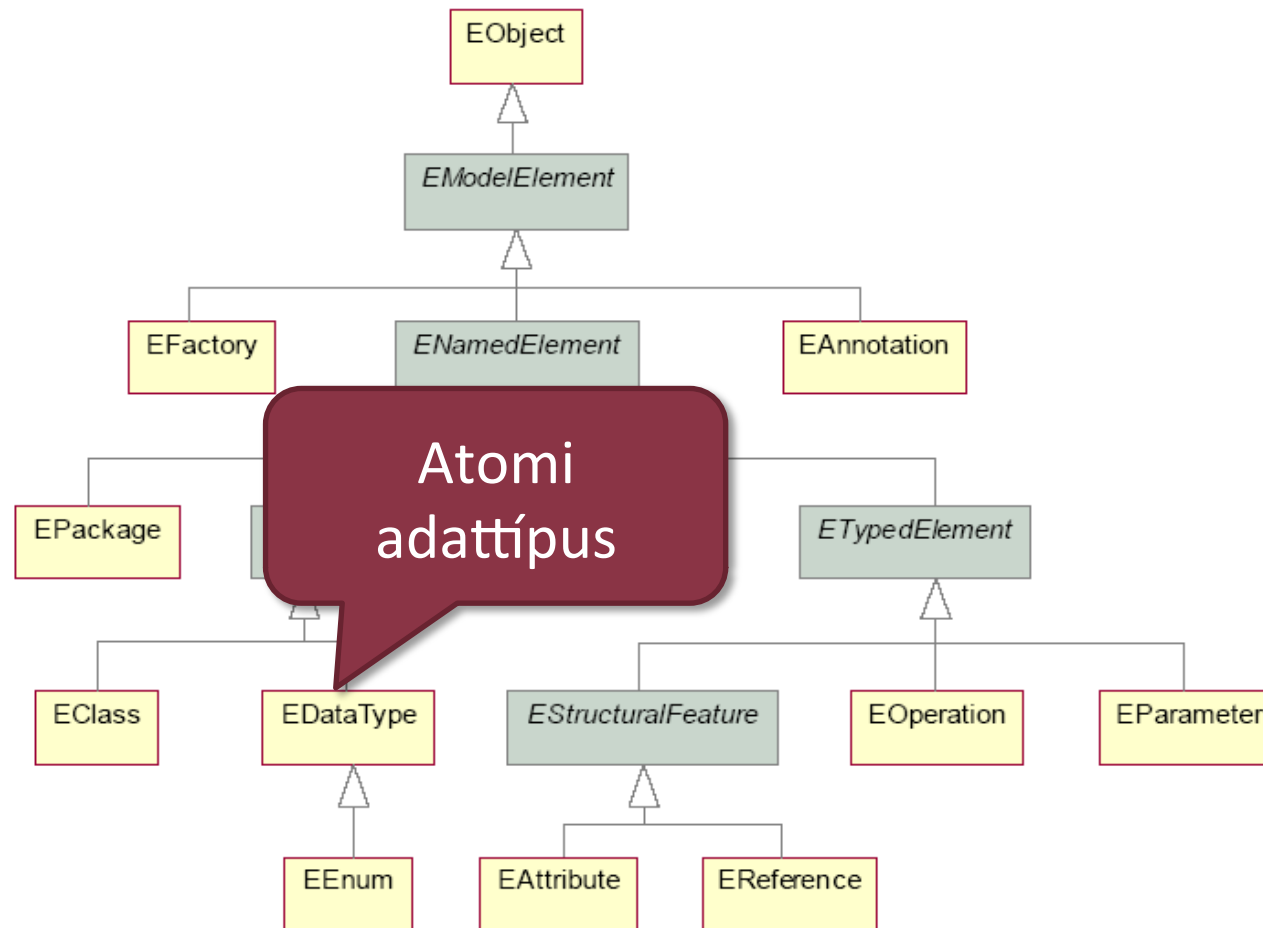


# Teljes ECore hierarchia

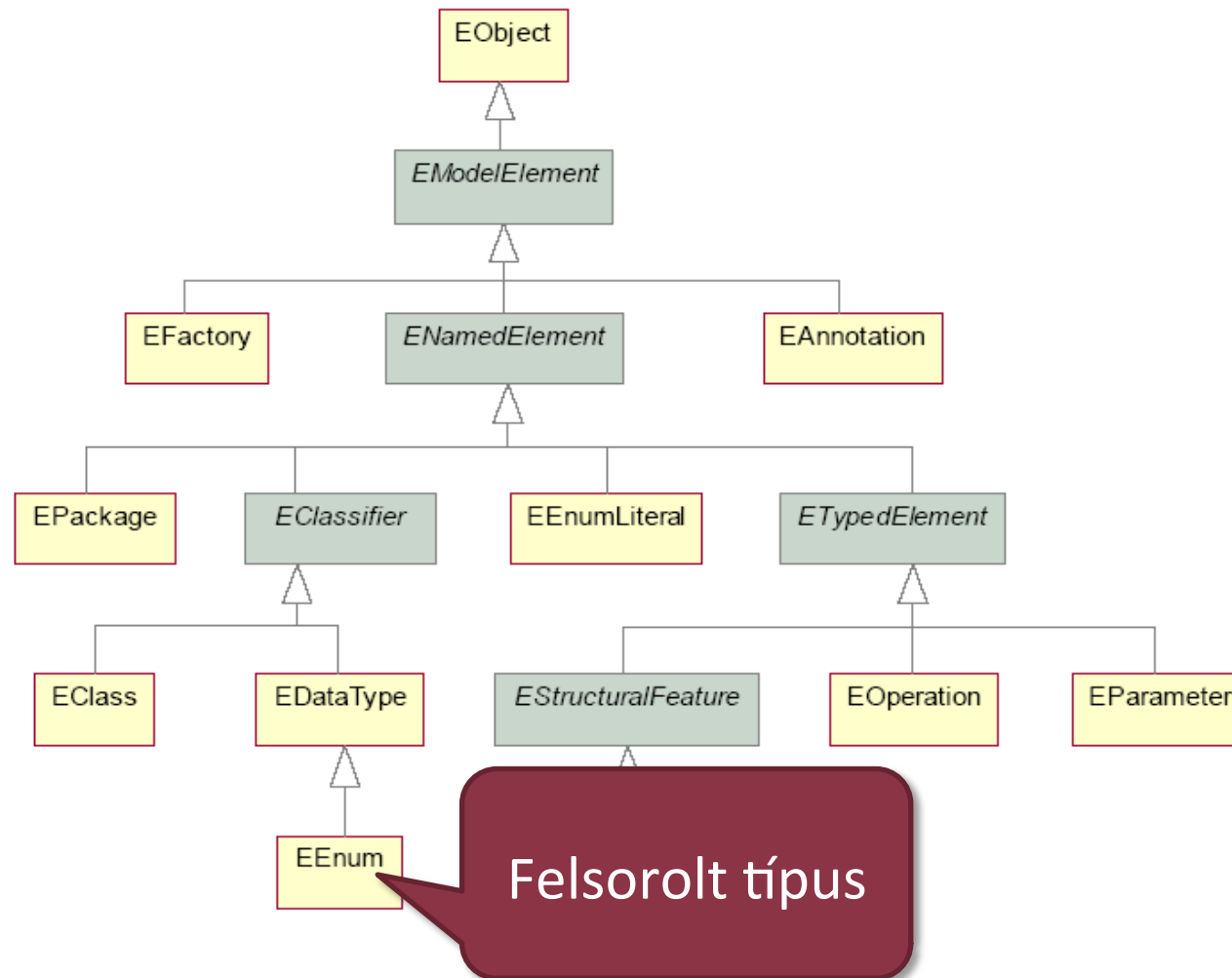
Típusok  
csomagokba  
szervezve



# Teljes ECore hierarchia

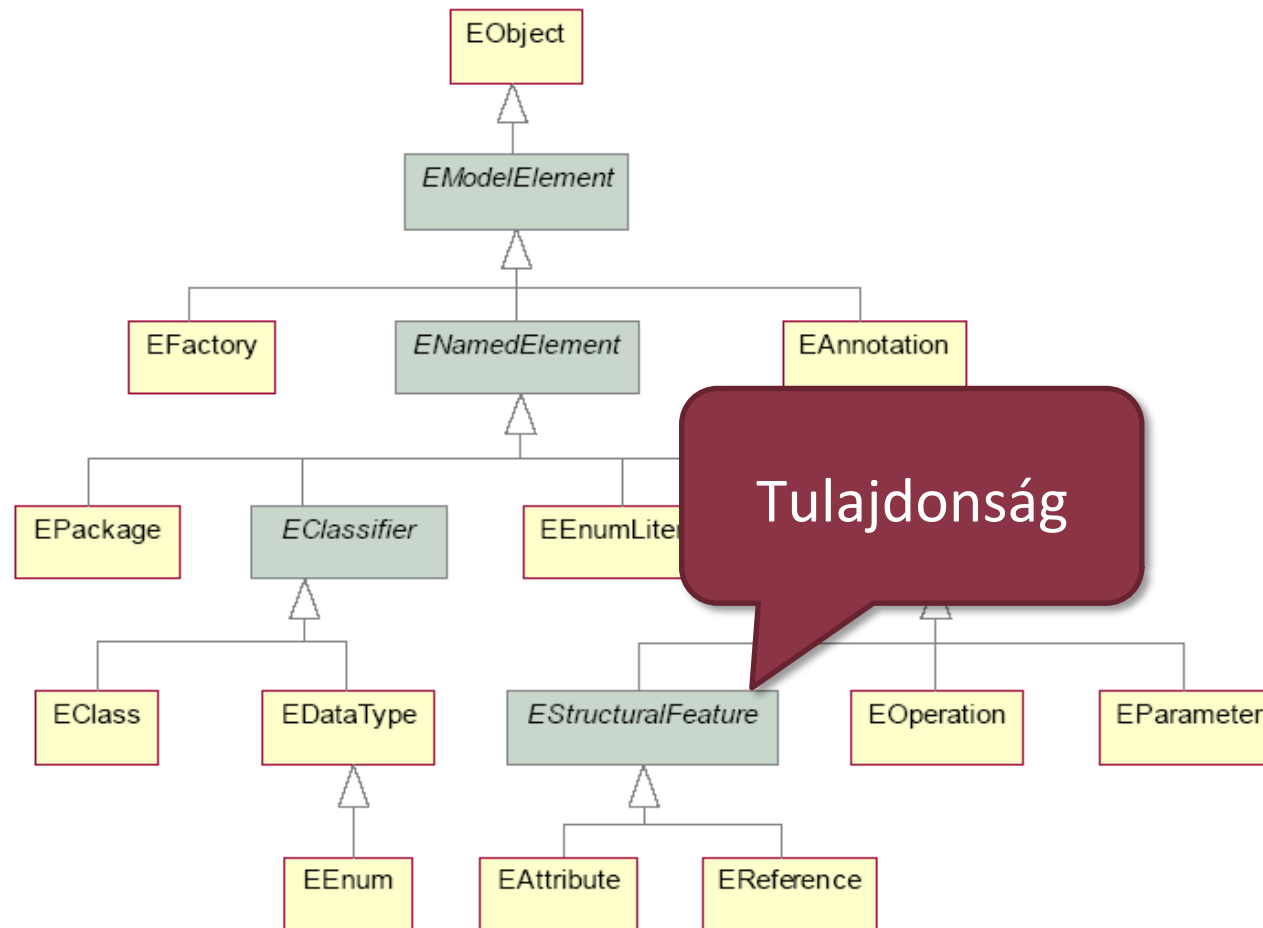


# Teljes ECore hierarchia

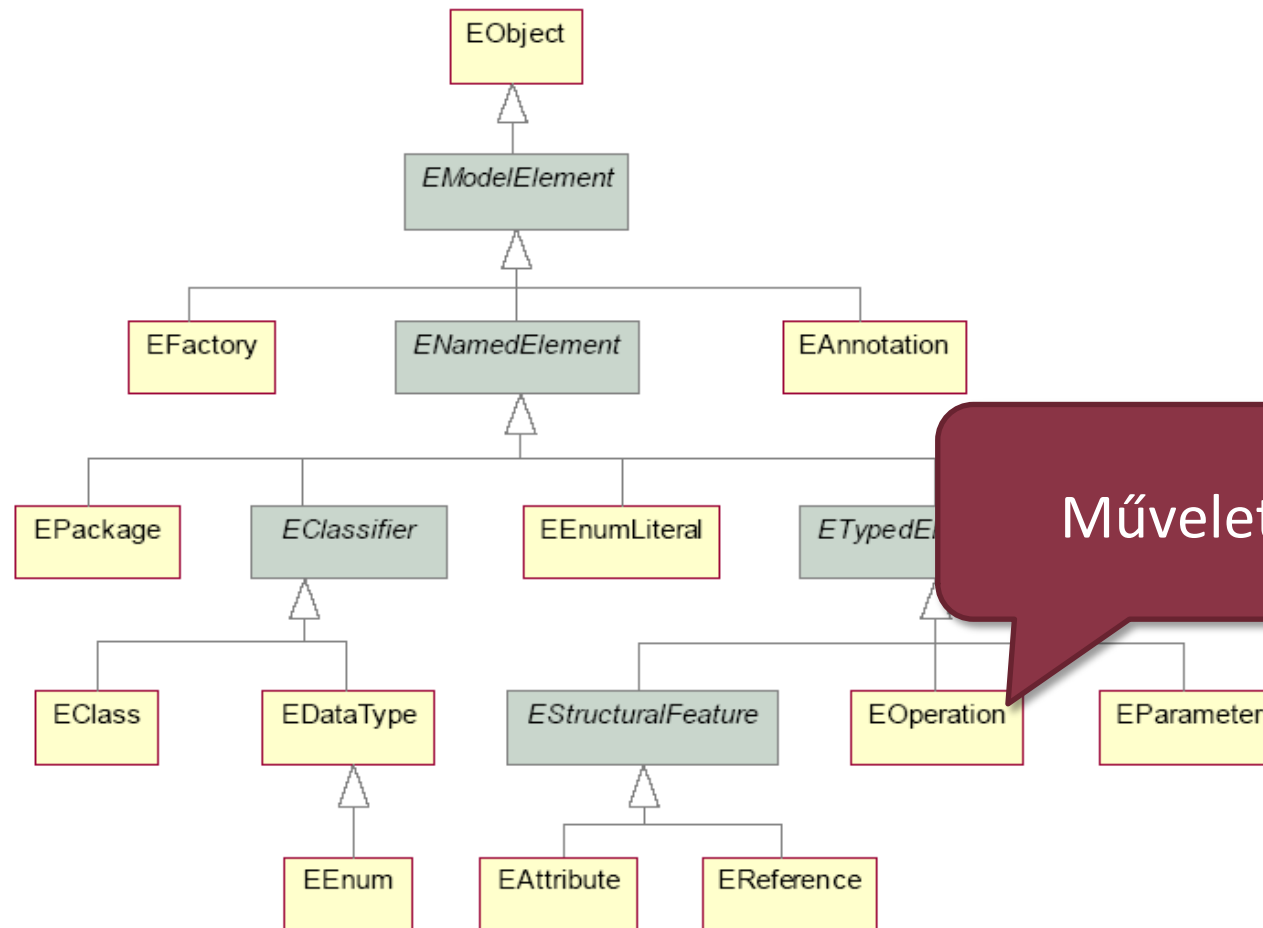




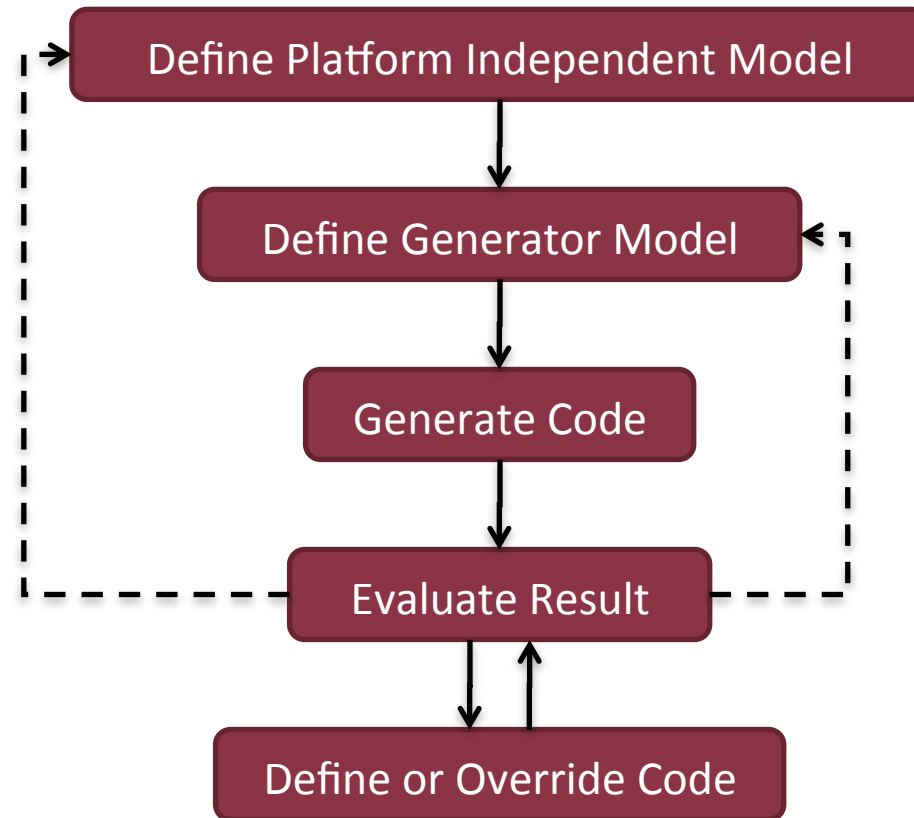
# Teljes ECore hierarchia



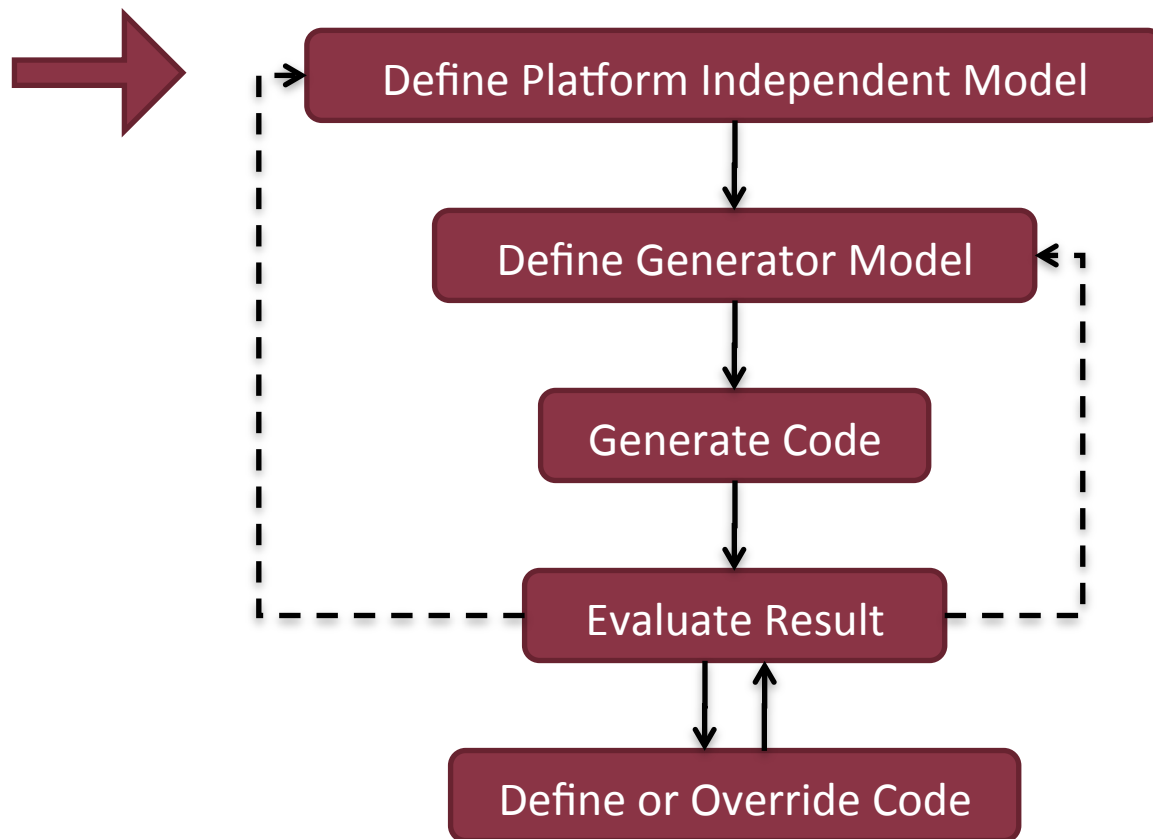
# Teljes ECore hierarchia



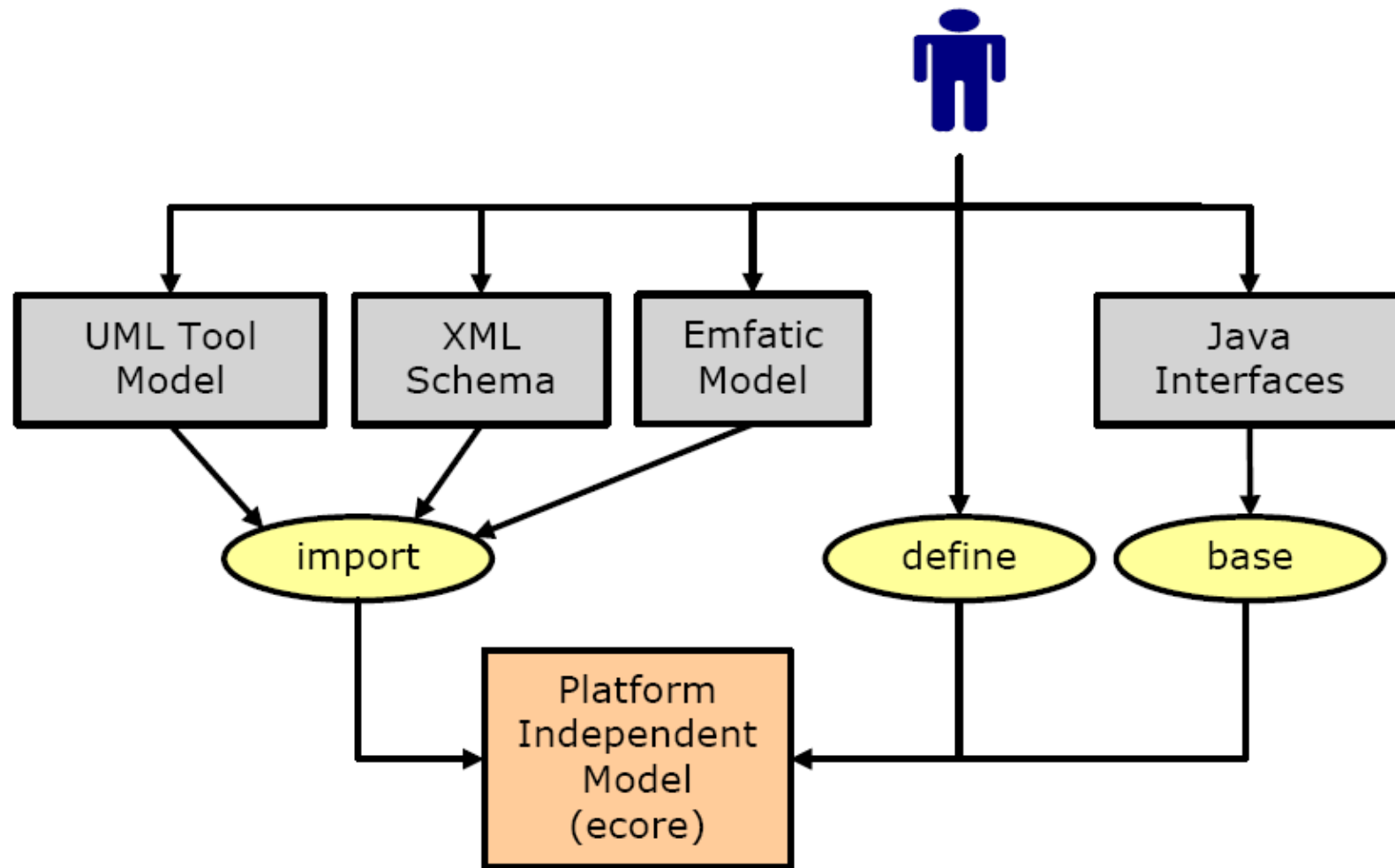
# Az EMF felhasználása



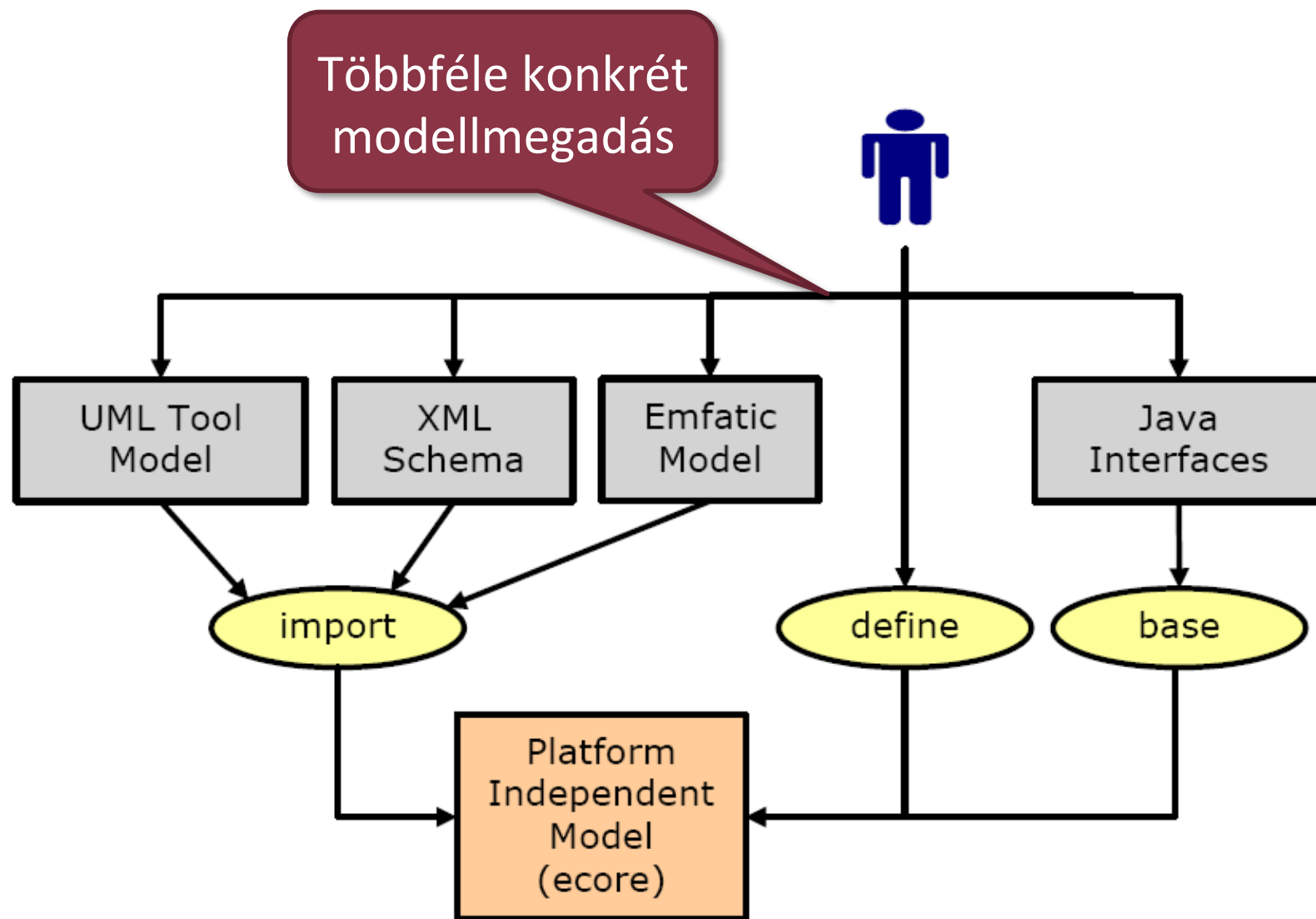
# Az EMF felhasználása



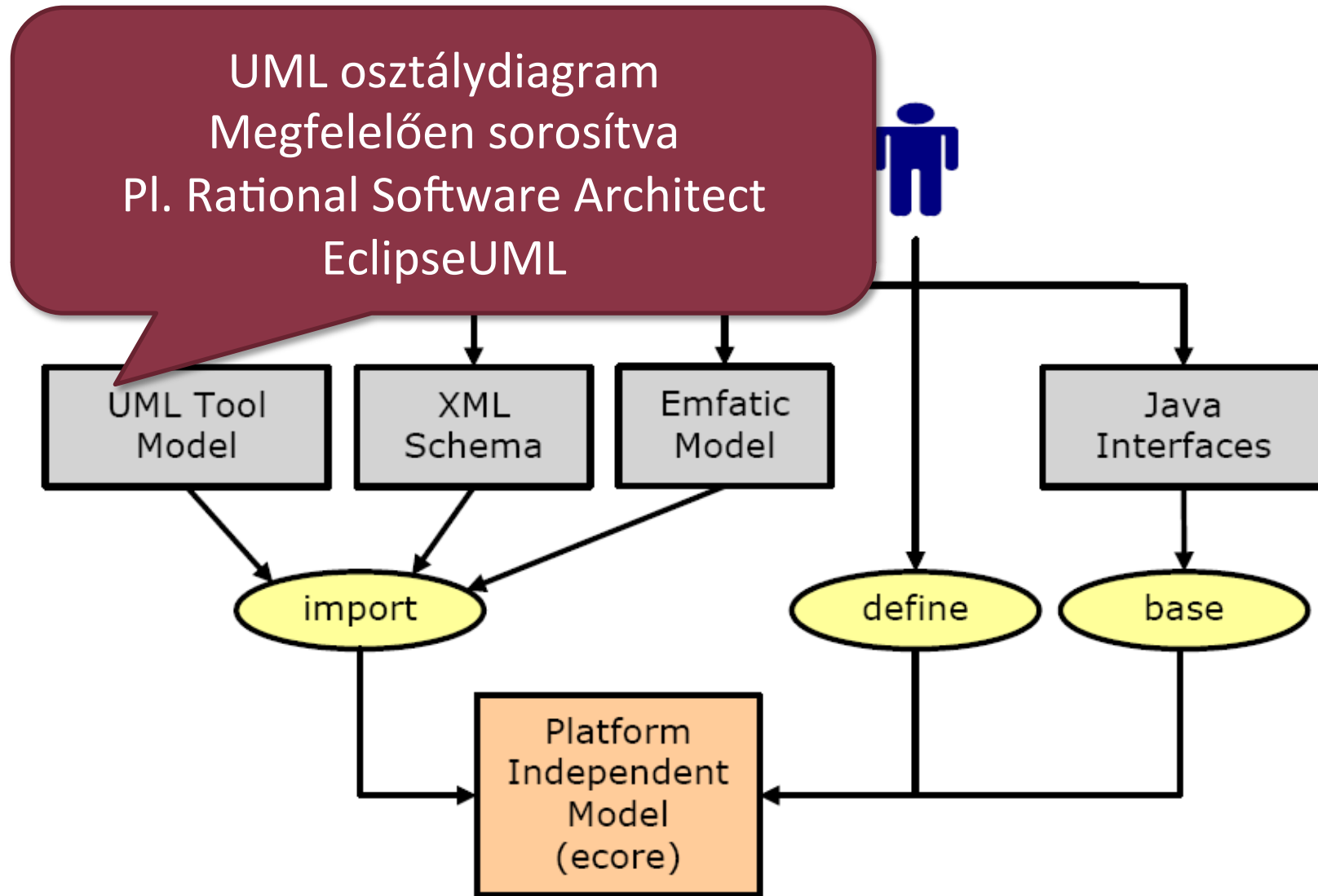
# ECore modell létrehozása



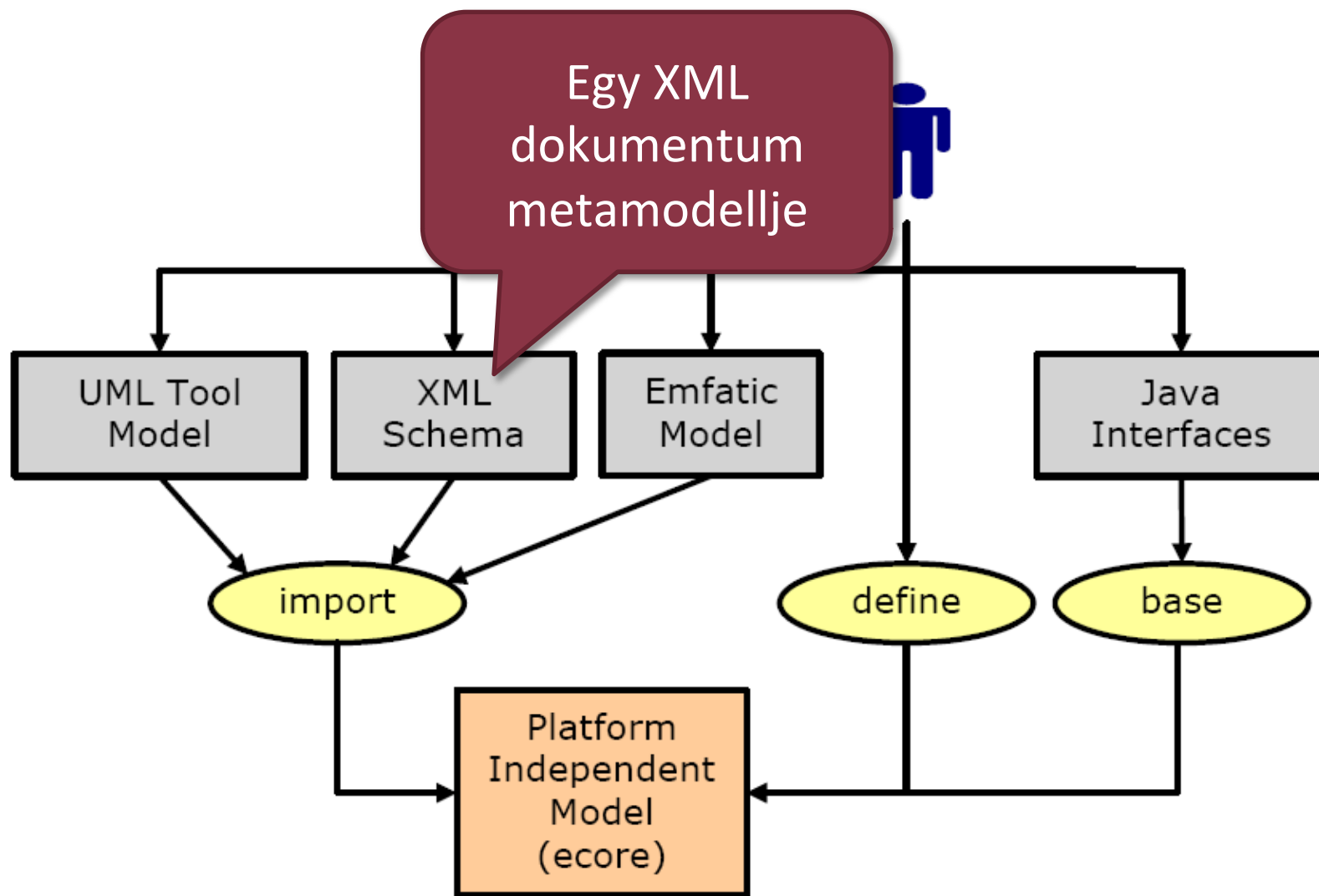
# ECore modell létrehozása



# ECore modell létrehozása

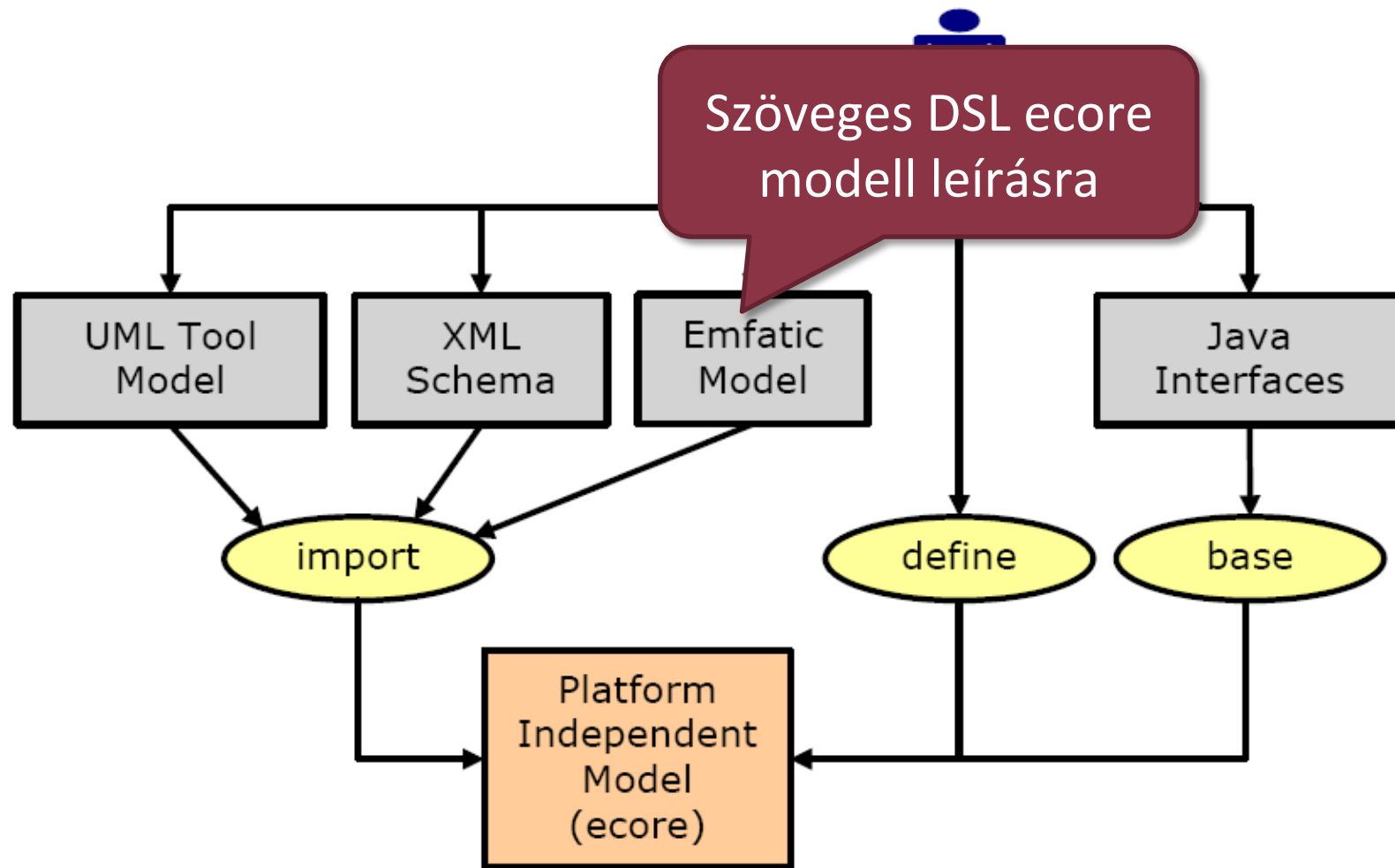


# ECore modell létrehozása

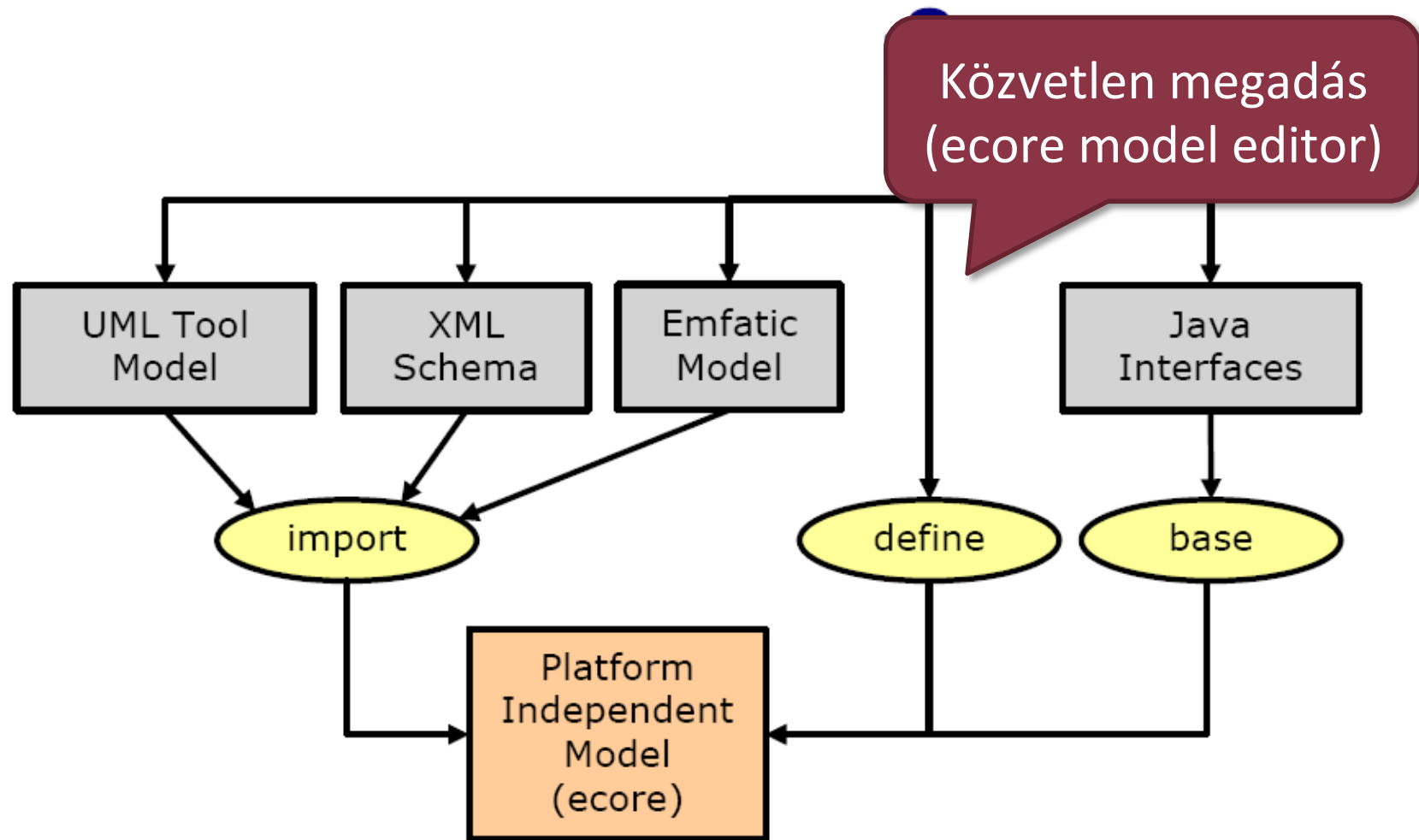




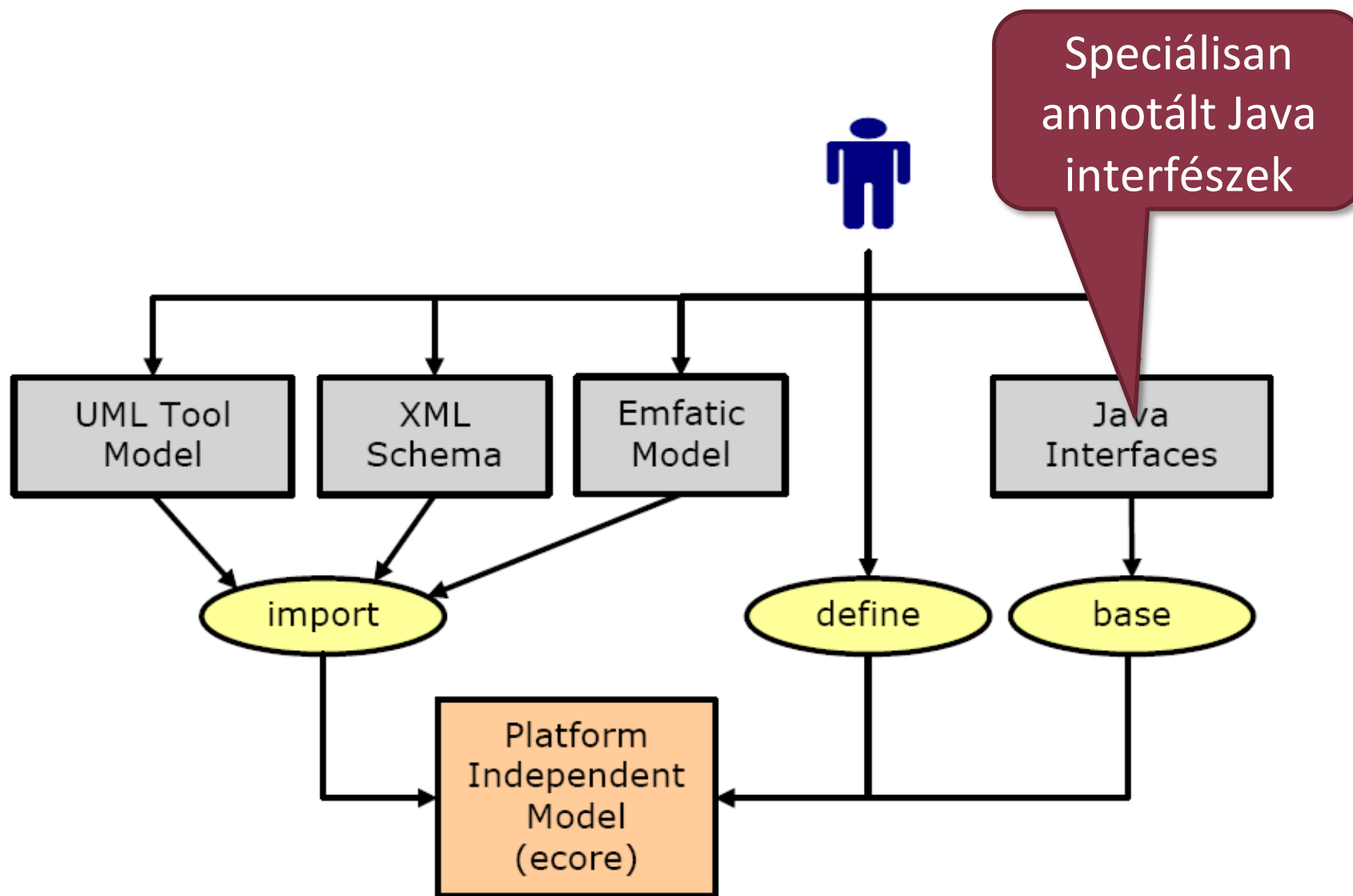
# ECore modell létrehozása



# ECore modell létrehozása



# ECore modell létrehozása



# Emfatic

- Szöveges DSL EMF metamodellek számára

The screenshot displays the Eclipse IDE interface with two main panels. The left panel shows the source code for a DSL metamodel named 'ppo.ecoredsl'. The code defines an import statement, package information, datatypes for Date and SKU, and two classes: Item and USAddress. The Item class has attributes for productName, quantity, USPrice, comment, shipDate, and partNum. The USAddress class has attributes for name, street, city, state, zip, and country, and a read-only attribute for country. The right panel shows the Outline view, which displays the hierarchical structure of the metamodel. The root is 'Ecore Dsl', which contains an 'Import Statement Decl' and a package 'ppo'. The 'ppo' package contains several classes: 'Ecore', 'Date [java.util.Date]', 'SKU [java.lang.String]', 'Item', and 'USAddress'. The 'USAddress' class is expanded to show its attributes: 'name : EString', 'street : EString', 'city : EString', 'state : EString', 'zip : EString', and 'country : EString'. Additionally, there is a class 'hasUSState(EDiagnosticChain, EBoolean)' which is expanded to show its attributes: 'EBoolean', 'diagnostics : EDiagnosticChain', and 'context : EMap<EObject, EObject>'. The 'context' attribute is further expanded to show its value: 'EObject'.

```
import "platform:/resource/org.xtext.dsl.test/src/Ecore.ecore";
@ "http://www.eclipse.org/emf/2002/Ecore" (
  constraints="NonNegativeQuantity ValidShipDate"
)
package ppo nsURI="http://com/example/ppo.ecore" nsPrefix=com.example.ppo {

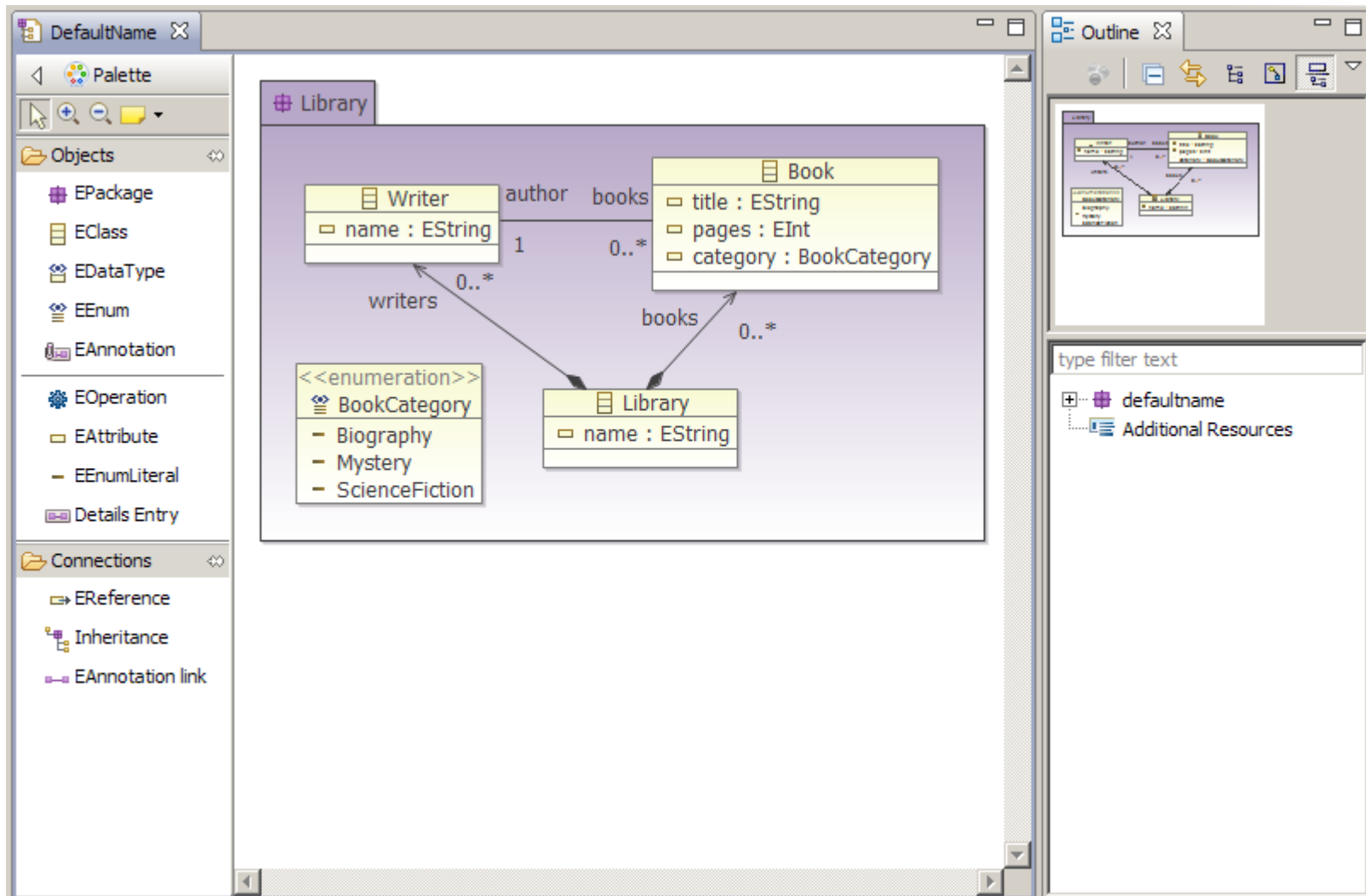
  datatype Date : java.util.Date;
  datatype SKU : java.lang.String;

  class Item {
    attr EString productName;
    attr EInt    quantity;
    attr EInt    USPrice;
    attr EString comment;
    attr Date    shipDate;
    attr SKU     partNum;
  }

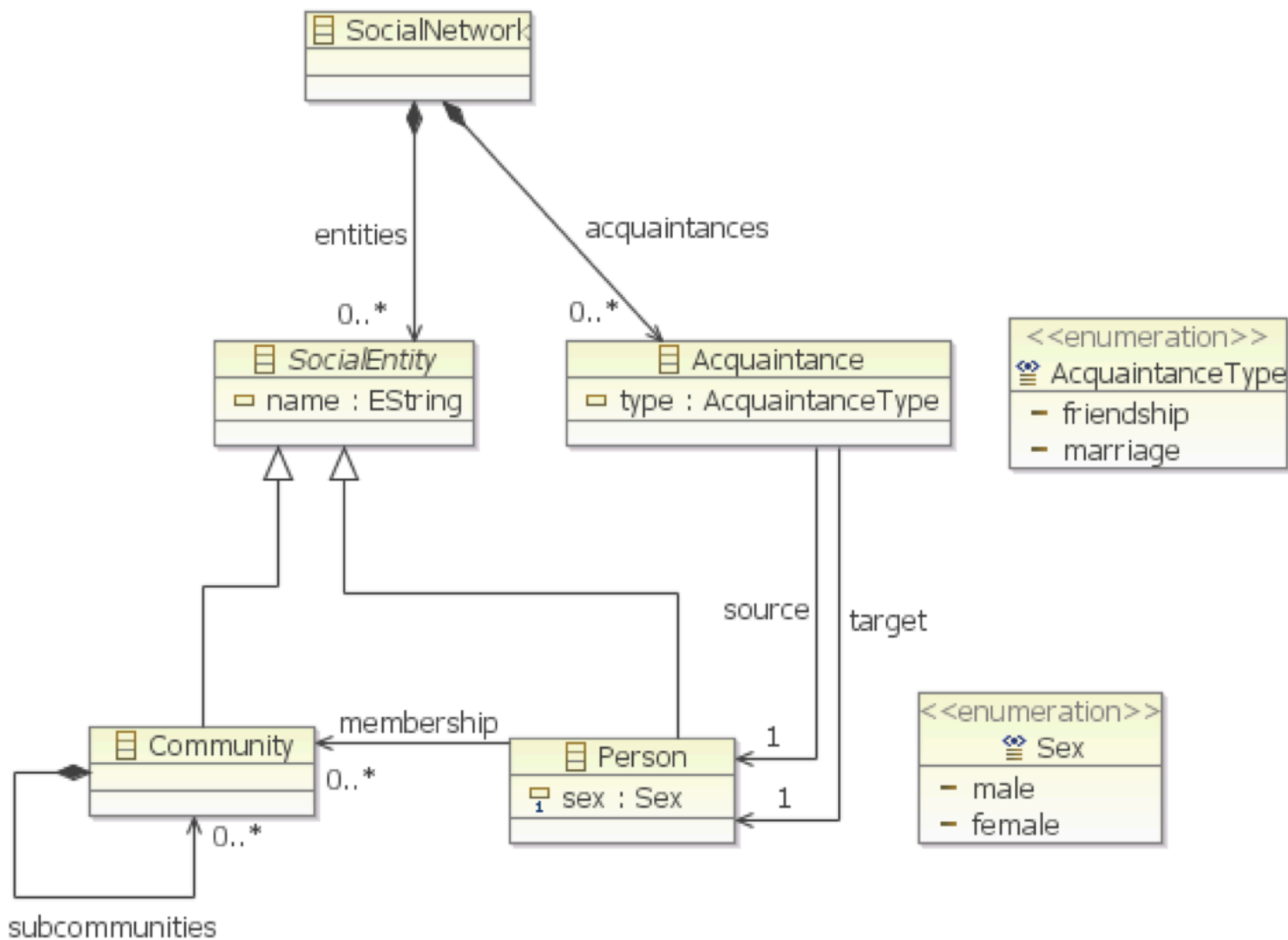
  class USAddress {
    attr EString name;
    attr EString street;
    attr EString city;
    attr EString state;
    attr EString zip;
    readonly attr EString country = "US";
    op EBoolean hasUSState (
      EDiagnosticChain diagnostics,
      EMap<EObject,EObject> context);
  }
}
```

# ECore Tools: ECore Diagram Editor

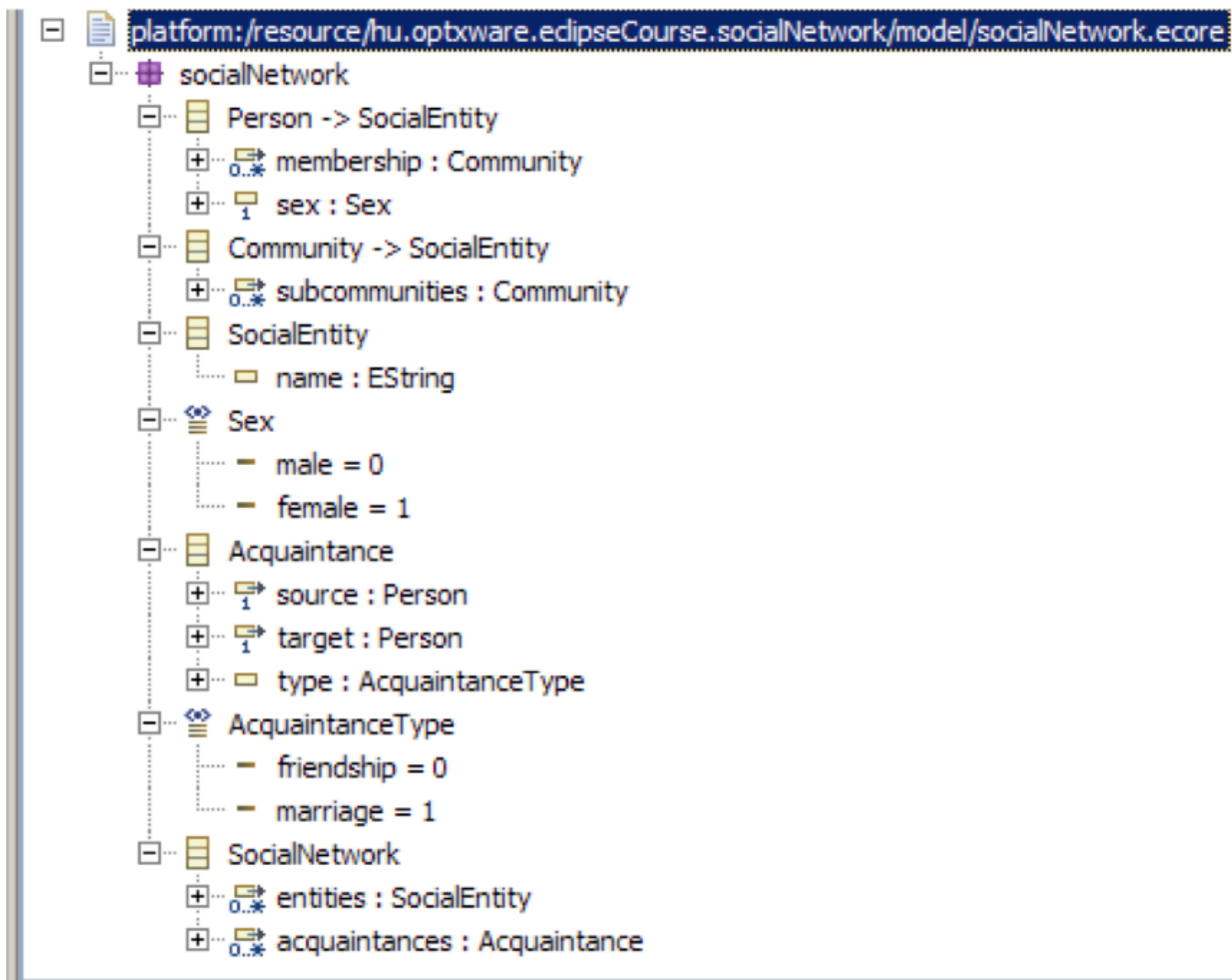
- Grafikus DSL EMF metamodellek számára



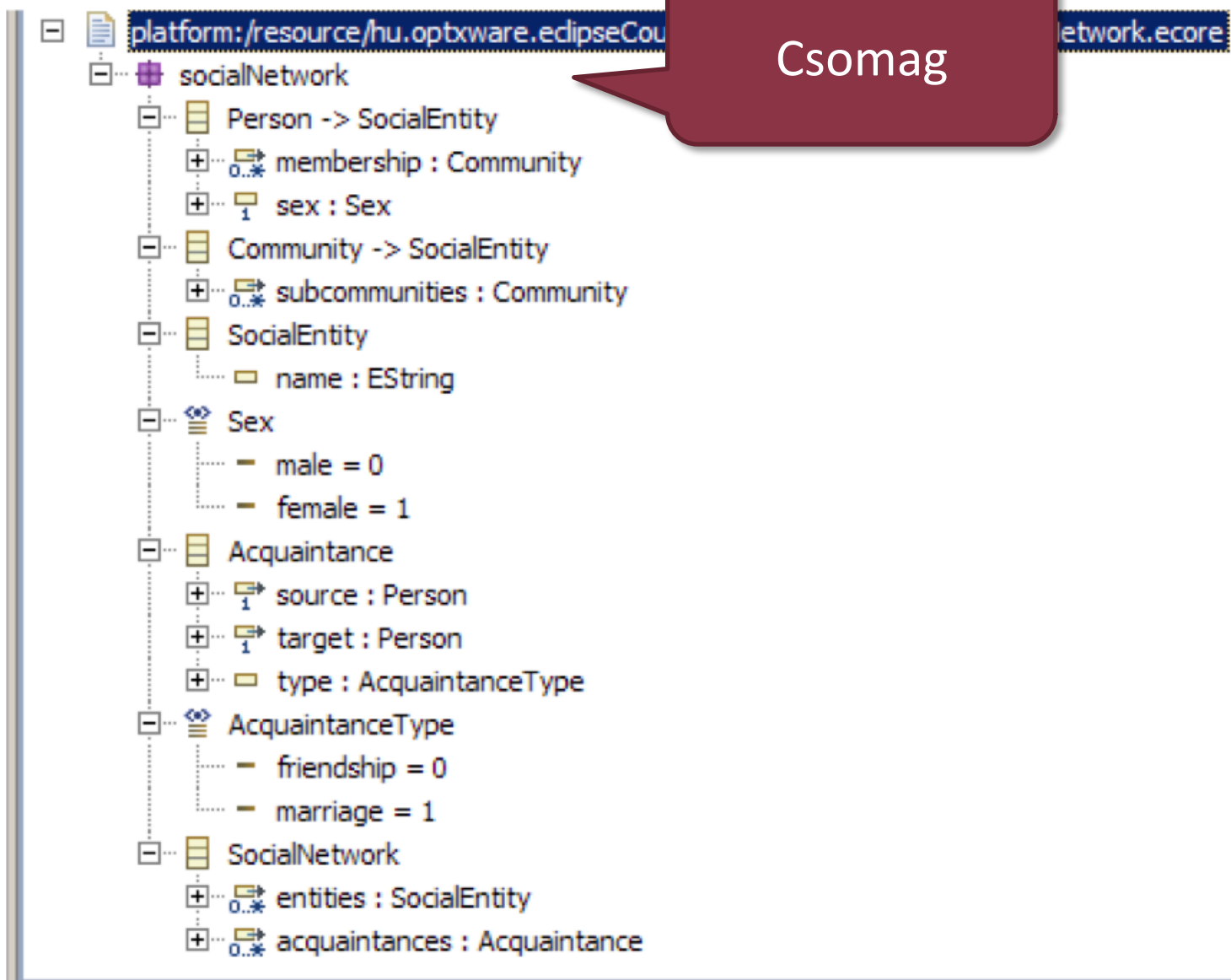
# Példa: Kapcsolati háló



# Példa: Az ECore metamodel

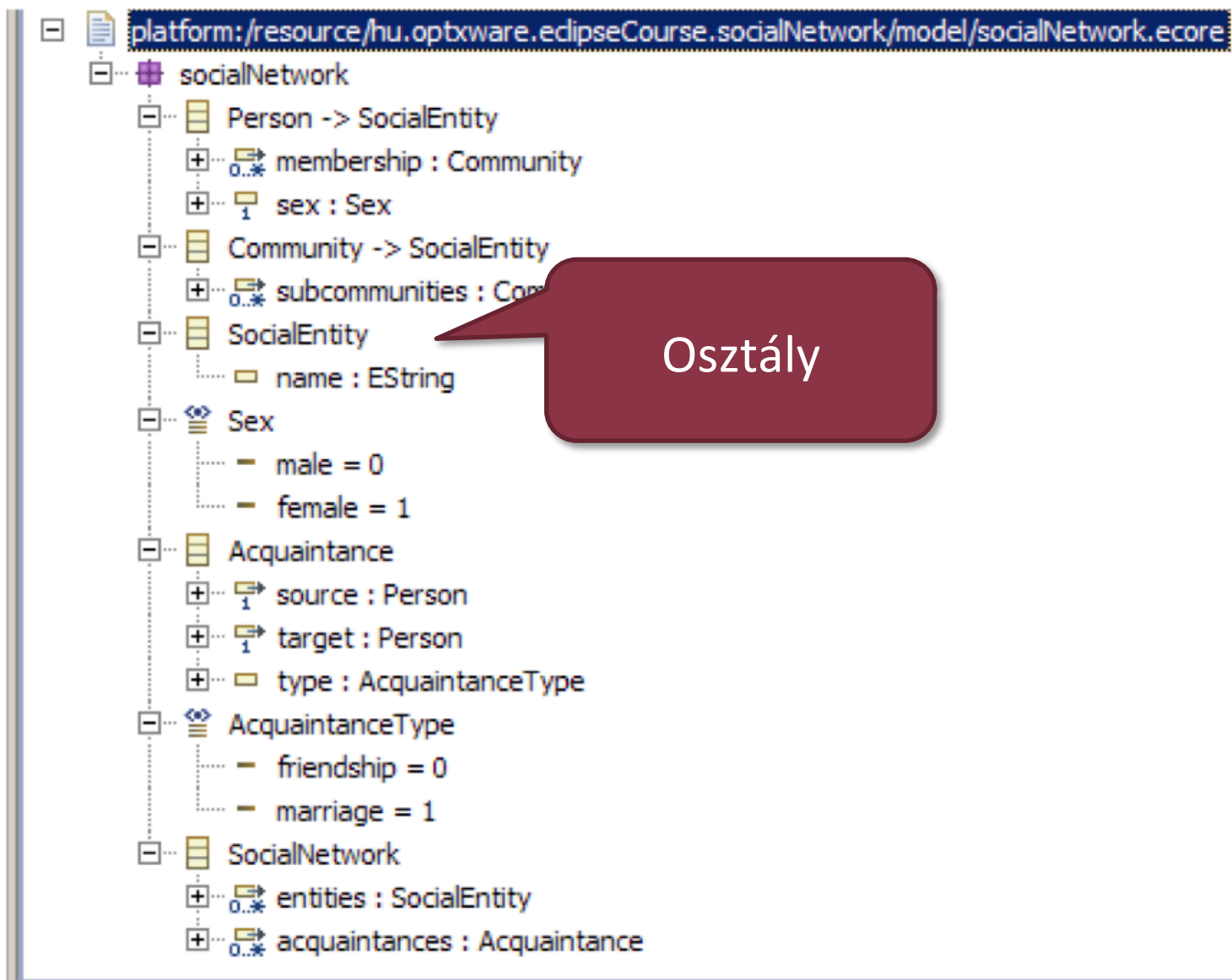


# Példa: Az ECore metamodel

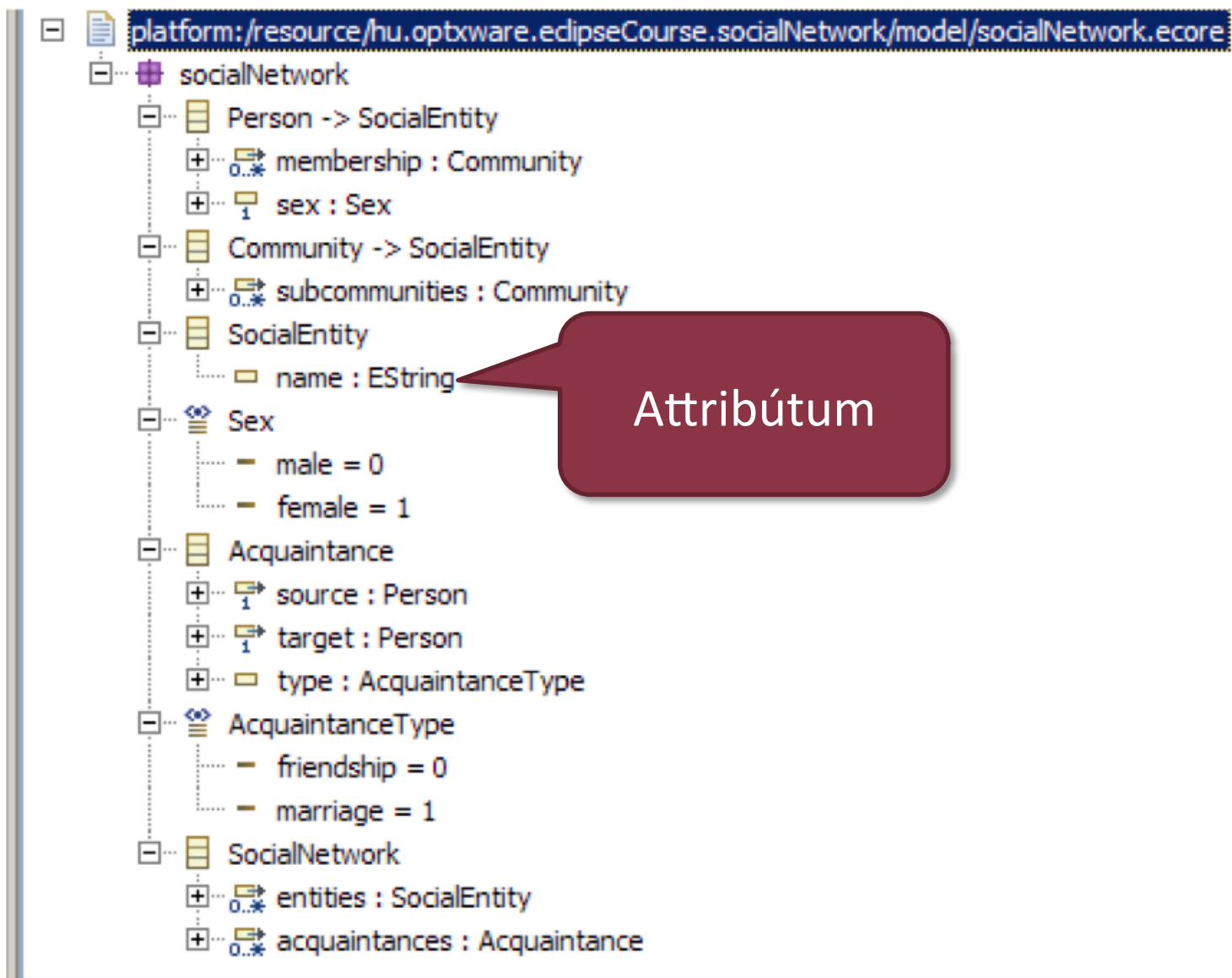




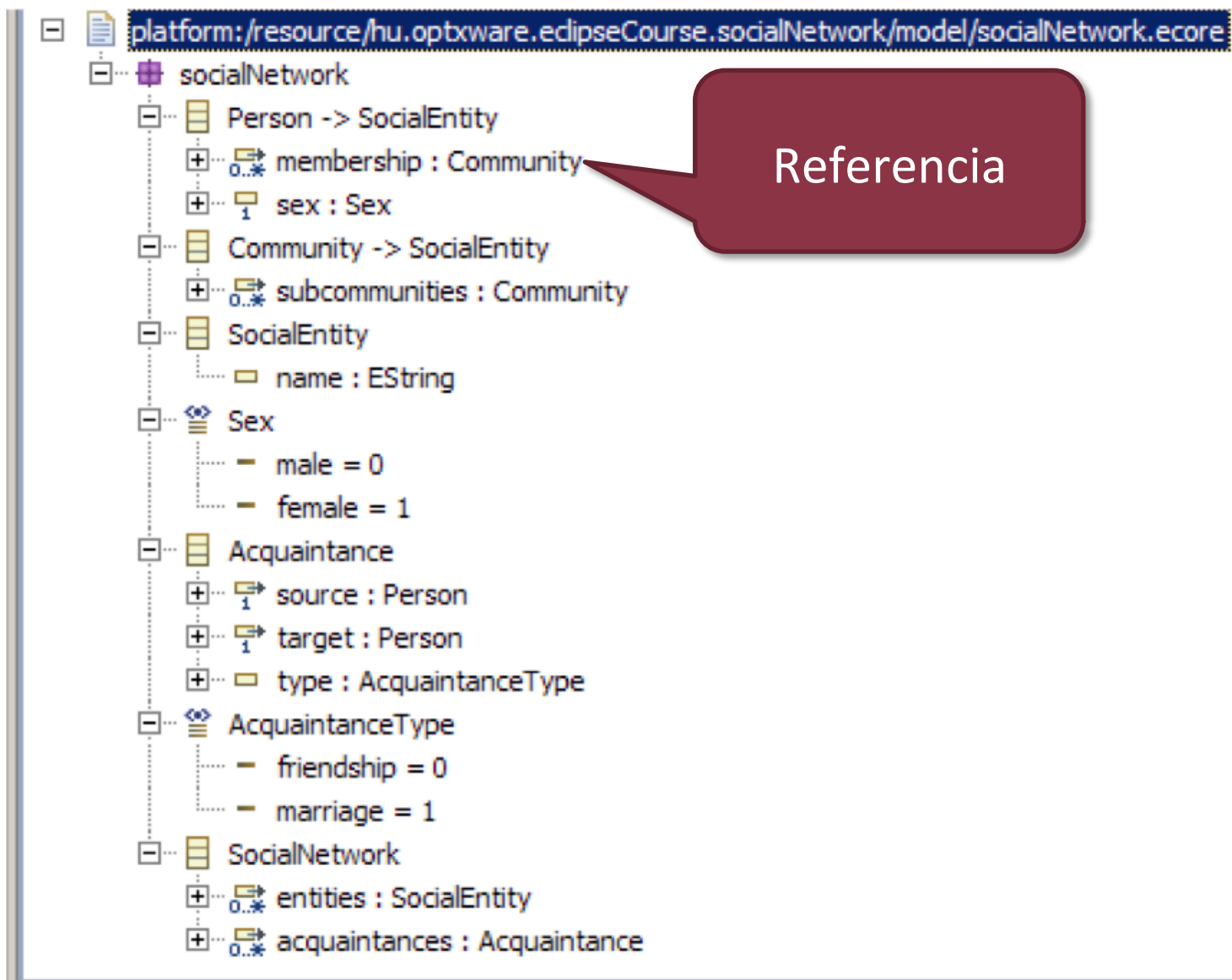
# Példa: Az ECore metamodel



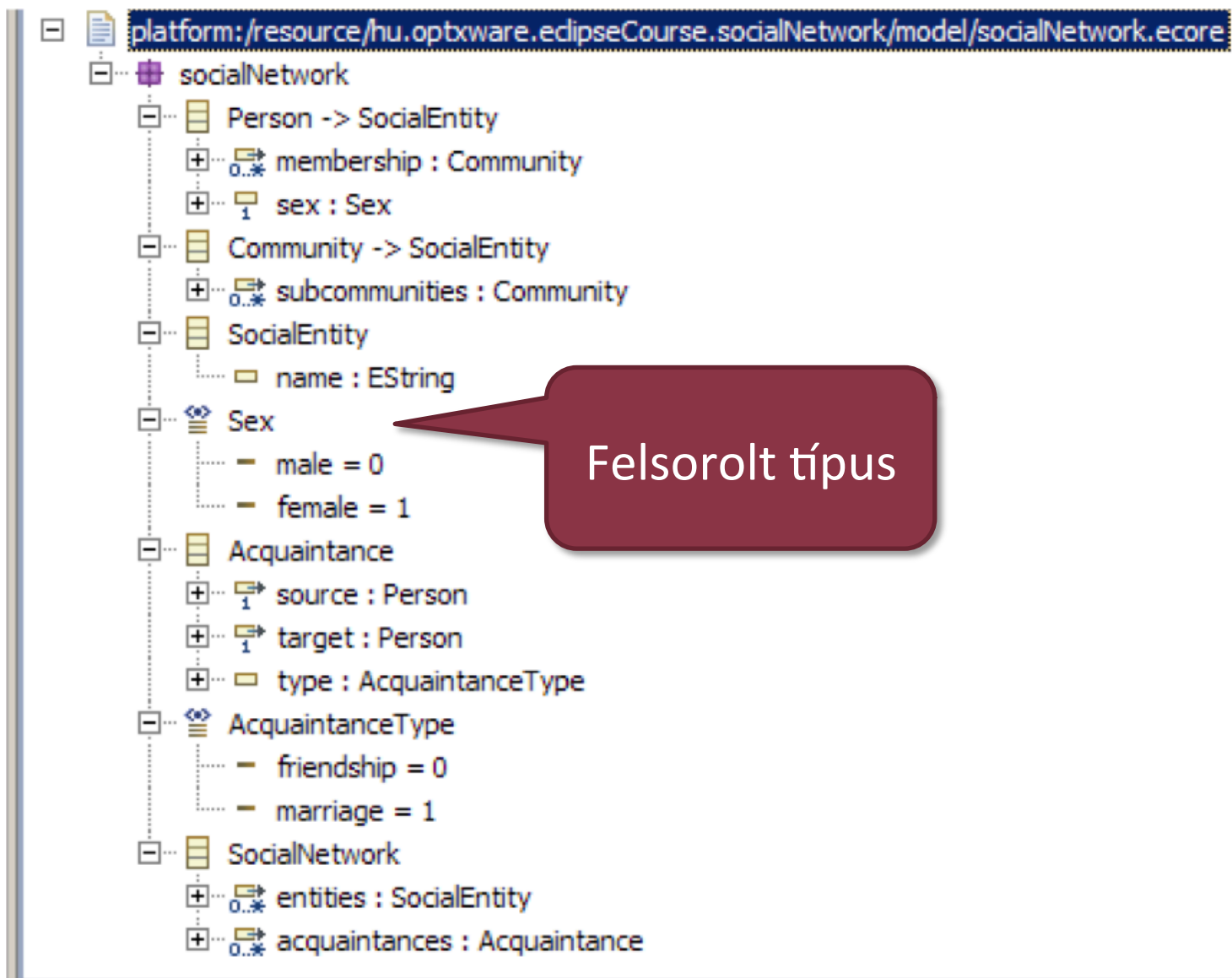
# Példa: Az ECore metamodel



# Példa: Az ECore metamodel

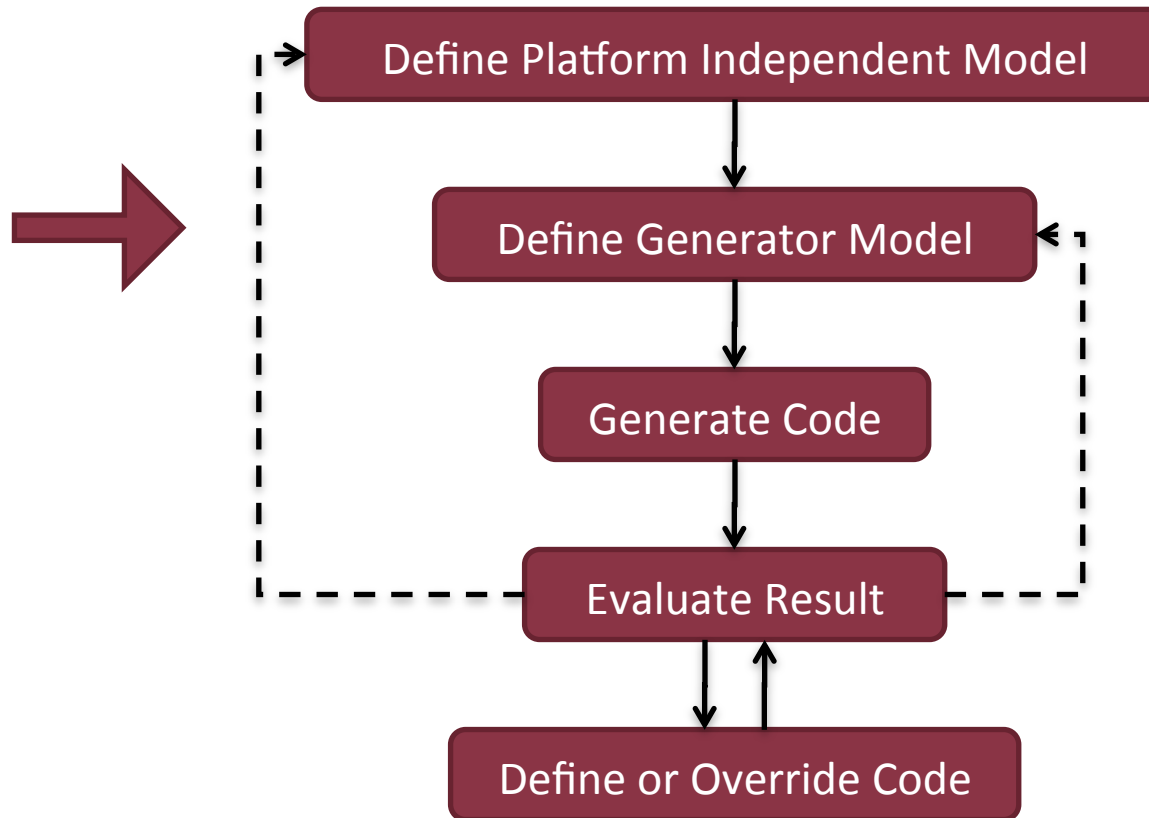


# Példa: Az ECore metamodel



Felsorolt típus

# Az EMF felhasználása



# Generátor modell

- Cél:
  - Kódgenerálási beállítások megadása
- EMF modell
  - Tree Editor
  - Hivatkozik a saját ecore modellünkre
- Kódgenerálási beállítások
  - Java verzió (pl. Java 5 esetén enumok használata)
  - Package/projektnevek
  - ...

# Generátor modell

The screenshot shows the Eclipse IDE interface. The top-left pane displays the project structure for 'Social Network', which includes sub-projects like 'Person -> SocialEntity', 'Community -> SocialEntity', 'SocialEntity', 'Acquaintance', 'SocialNetwork', 'Sex', and 'AcquaintanceType'. The bottom pane shows a table of properties for the selected project.

Property	Value
All	
Bundle Manifest	true
Compliance Level	5.0
Copyright Fields	false
Copyright Text	
Language	
Model Name	Social Network
Non-NLS Markers	false
Runtime Compatibility	false
Runtime Jar	false
Runtime Version	2.5
Edit	
Color Providers	false
Creation Commands	true
Creation Icons	true
Edit Directory	/hu.optxware.eclipseCourse.socialNetwork.edit/src
Edit Plug-in Class	socialNetwork.provider.SocialNetworkEditPlugin
Edit Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.edit
Edit Plug-in Variables	
Font Providers	false
Optimized Has Children	false
Provider Root Extends Class	
Table Providers	false
Editor	
Creation Sub-menus	false
Editor Directory	/hu.optxware.eclipseCourse.socialNetwork.editor/src
Editor Plug-in Class	socialNetwork.presentation.SocialNetworkEditorPlugin
Editor Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.editor
Editor Plug-in Variables	
Rich Client Platform	false
Model	
Array Accessors	false
Binary Compatible Reflective Methods	false
Class Name Pattern	

# Generátor modell

Hivatkozott ECore  
modellelemek  
(akár több ECore  
modellből is)

The screenshot shows the Eclipse IDE interface. The top part displays the 'Social Network' project structure in the Package Explorer, including packages like SocialNetwork, Person, Community, SocialEntity, Acquaintance, and classes like Sex and AcquaintanceType. The bottom part shows the Properties view for the selected package, listing various properties and their values.

Property	Value
Bundle Manifest	true
Compliance Level	5.0
Copyright Fields	false
Copyright Text	
Language	
Model Name	Social Network
Non-NLS Markers	false
Runtime Compatibility	false
Runtime Jar	false
Runtime Version	2.5
Color Providers	false
Creation Commands	true
Creation Icons	true
Edit Directory	/hu.optxware.eclipseCourse.socialNetwork.edit/src
Edit Plug-in Class	socialNetwork.provider.SocialNetworkEditPlugin
Edit Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.edit
Edit Plug-in Variables	
Font Providers	false
Optimized Has Children	false
Provider Root Extends Class	
Table Providers	false
Creation Sub-menus	false
Editor Directory	/hu.optxware.eclipseCourse.socialNetwork.editor/src
Editor Plug-in Class	socialNetwork.presentation.SocialNetworkEditorPlugin
Editor Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.editor
Editor Plug-in Variables	
Rich Client Platform	false
Array Accessors	false
Binary Compatible Reflective Methods	false
Class Name Pattern	



# Generátor modell

The screenshot shows the Eclipse IDE interface. The top part displays the project structure for 'Social Network', including sub-projects like 'Person -> SocialEntity', 'Community -> SocialEntity', 'SocialEntity', 'Acquaintance', 'SocialNetwork', 'Sex', and 'AcquaintanceType'. Below this is the 'Properties' dialog box, which is currently showing the 'All' tab. The dialog lists various properties and their values, such as 'Bundle Manifest' (true), 'Compliance Level' (5.0), 'Model Name' (Social Network), and 'Runtime Version' (2.5). A red speech bubble with the text 'Általános paraméterek' (General parameters) points to the 'Properties' dialog.

Property	Value
Bundle Manifest	true
Compliance Level	5.0
Copyright Fields	false
Copyright Text	
Language	
Model Name	Social Network
Non-NLS Markers	false
Runtime Compatibility	false
Runtime Jar	false
Runtime Version	2.5
Color Providers	false
Creation Commands	true
Creation Icons	true
Edit Directory	/hu.optxware.eclipseCourse.socialNetwork.edit/src
Edit Plug-in Class	socialNetwork.provider.SocialNetworkEditPlugin
Edit Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.edit
Edit Plug-in Variables	
Font Providers	false
Optimized Has Children	false
Provider Root Extends Class	
Table Providers	false
Creation Sub-menus	false
Editor Directory	/hu.optxware.eclipseCourse.socialNetwork.editor/src
Editor Plug-in Class	socialNetwork.presentation.SocialNetworkEditorPlugin
Editor Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.editor
Editor Plug-in Variables	
Rich Client Platform	false
Array Accessors	false
Binary Compatible Reflective Methods	false
Class Name Pattern	

# Generátor modell

The screenshot shows the Eclipse IDE interface. The top-left pane displays the project structure for 'Social Network', including sub-projects like 'Person -> SocialEntity', 'Community -> SocialEntity', 'SocialEntity', 'Acquaintance', 'SocialNetwork', 'Sex', and 'AcquaintanceType'. The bottom pane shows the 'Properties' dialog box for the selected project. The dialog is divided into several sections: 'All', 'Edit', 'Editor', and 'Model'. The 'Edit' section is highlighted, showing properties such as 'Color Providers', 'Creation Commands', 'Creation Icons', 'Edit Directory', 'Edit Plug-in Class', 'Edit Plug-in ID', 'Edit Plug-in Variables', 'Font Providers', 'Optimized Has Children', 'Provider Root Extends Class', and 'Table Providers'. A red callout box points to the 'Edit' section with the text 'Modellmanipuláció paramétere'i'.

Property	Value
Bundle Manifest	true
Compliance Level	5.0
Copyright Fields	false
Copyright Text	
Language	
Model Name	Social Network
Non-NLS Markers	false
Runtime Compatibility	false
Runtime Jar	false
Runtime Version	2.5
<b>Edit</b>	
Color Providers	false
Creation Commands	true
Creation Icons	true
Edit Directory	/hu.optxware.eclipseCourse.socialNetwork.edit/src
Edit Plug-in Class	socialNetwork.provider.SocialNetworkEditPlugin
Edit Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.edit
Edit Plug-in Variables	
Font Providers	false
Optimized Has Children	false
Provider Root Extends Class	
Table Providers	false
<b>Editor</b>	
Creation Sub-menus	false
Editor Directory	/hu.optxware.eclipseCourse.socialNetwork.editor/src
Editor Plug-in Class	socialNetwork.presentation.SocialNetworkEditorPlugin
Editor Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.editor
Editor Plug-in Variables	
Rich Client Platform	false
<b>Model</b>	
Array Accessors	false
Binary Compatible Reflective Methods	false
Class Name Pattern	

Modellmanipuláció  
paramétere'i

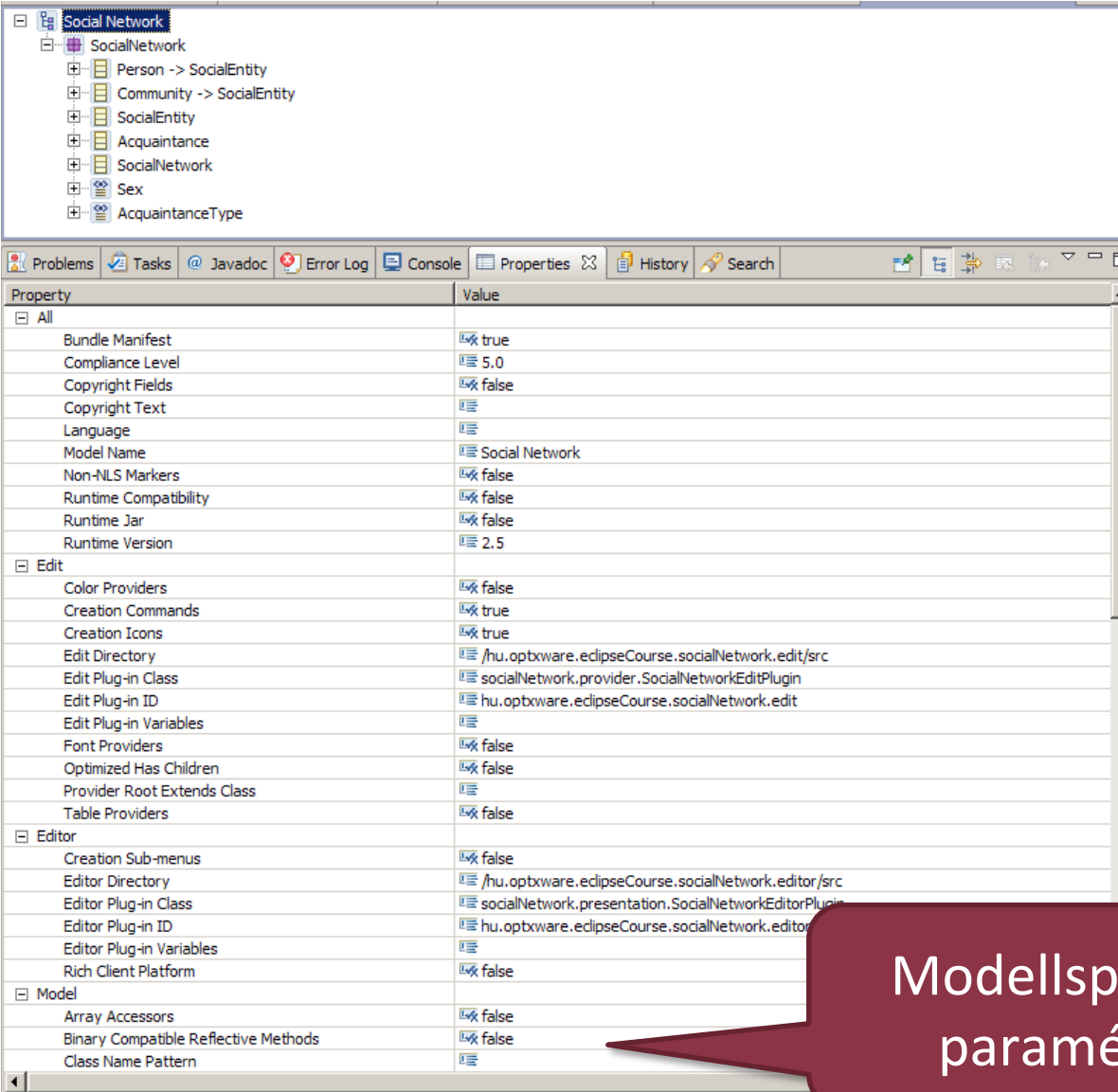
# Generátor modell

The screenshot shows the Eclipse IDE interface. The top part displays the project structure for 'Social Network', which includes sub-projects like 'Person -> SocialEntity', 'Community -> SocialEntity', 'SocialEntity', 'Acquaintance', 'SocialNetwork', 'Sex', and 'AcquaintanceType'. Below this is a table of properties for the selected project.

Property	Value
Bundle Manifest	true
Compliance Level	5.0
Copyright Fields	false
Copyright Text	
Language	
Model Name	Social Network
Non-NLS Markers	false
Runtime Compatibility	false
Runtime Jar	false
Runtime Version	2.5
Color Providers	false
Creation Commands	true
Creation Icons	true
Edit Directory	/hu.optxware.eclipseCourse.socialNetwork.edit/src
Edit Plug-in Class	socialNetwork.provider.SocialNetworkEditPlugin
Edit Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.edit
Edit Plug-in Variables	
Font Providers	false
Optimized Has Children	false
Provider Root Extends Class	
Table Providers	false
Creation Sub-menus	false
Editor Directory	/hu.optxware.eclipseCourse.socialNetwork.editor
Editor Plug-in Class	socialNetwork.presentation.SocialNetworkEditorPlugin
Editor Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.editor
Editor Plug-in Variables	
Rich Client Platform	false
Array Accessors	false
Binary Compatible Reflective Methods	false
Class Name Pattern	

Szerkesztőspecifikus paraméterek

# Generátor modell



The screenshot displays the Eclipse IDE interface for a project named 'Social Network'. The left-hand side shows a package explorer with the following structure:

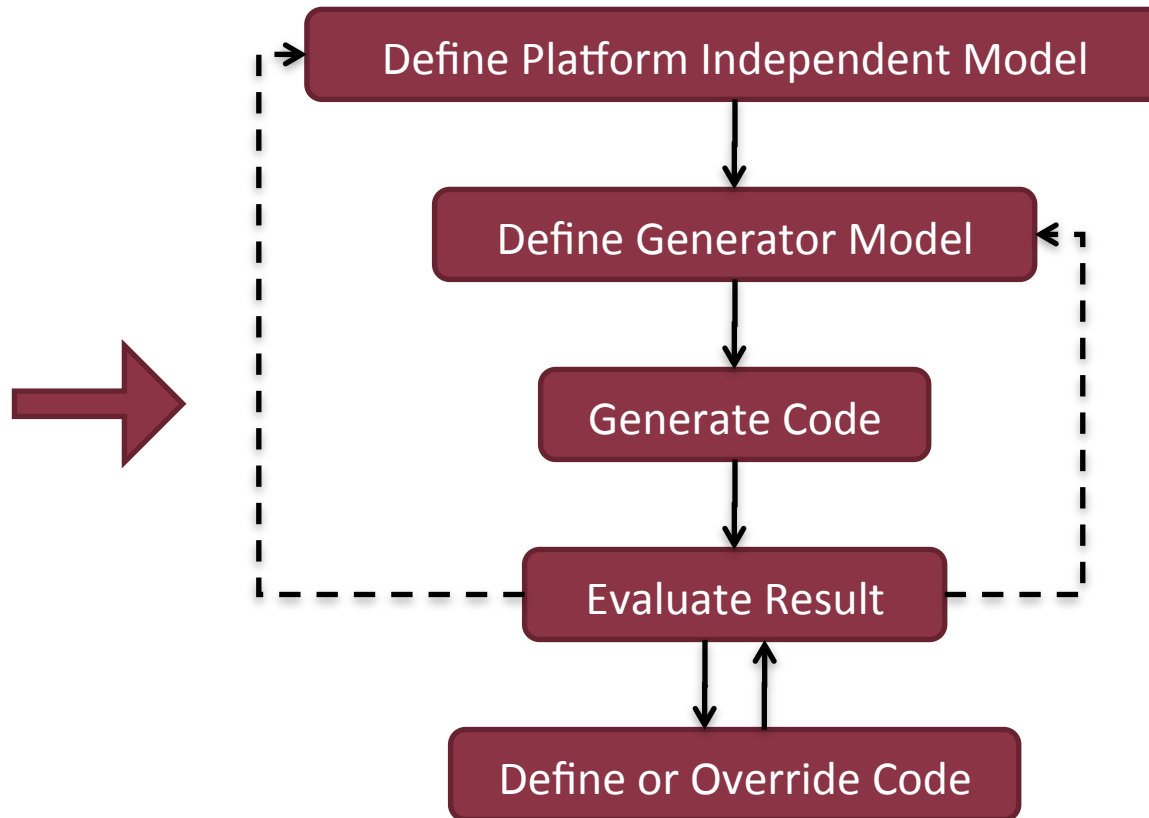
- SocialNetwork
  - Person -> SocialEntity
  - Community -> SocialEntity
  - SocialEntity
  - Acquaintance
  - SocialNetwork
  - Sex
  - AcquaintanceType

The right-hand side shows the 'Properties' view for the selected project, listing various properties and their values:

Property	Value
Bundle Manifest	true
Compliance Level	5.0
Copyright Fields	false
Copyright Text	
Language	
Model Name	Social Network
Non-NLS Markers	false
Runtime Compatibility	false
Runtime Jar	false
Runtime Version	2.5
Color Providers	false
Creation Commands	true
Creation Icons	true
Edit Directory	/hu.optxware.eclipseCourse.socialNetwork.edit/src
Edit Plug-in Class	socialNetwork.provider.SocialNetworkEditPlugin
Edit Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.edit
Edit Plug-in Variables	
Font Providers	false
Optimized Has Children	false
Provider Root Extends Class	
Table Providers	false
Creation Sub-menus	false
Editor Directory	/hu.optxware.eclipseCourse.socialNetwork.editor/src
Editor Plug-in Class	socialNetwork.presentation.SocialNetworkEditorPlugin
Editor Plug-in ID	hu.optxware.eclipseCourse.socialNetwork.editor
Editor Plug-in Variables	
Rich Client Platform	false
Array Accessors	false
Binary Compatible Reflective Methods	false
Class Name Pattern	

Modellspecifikus  
paraméterek

# Az EMF felhasználása



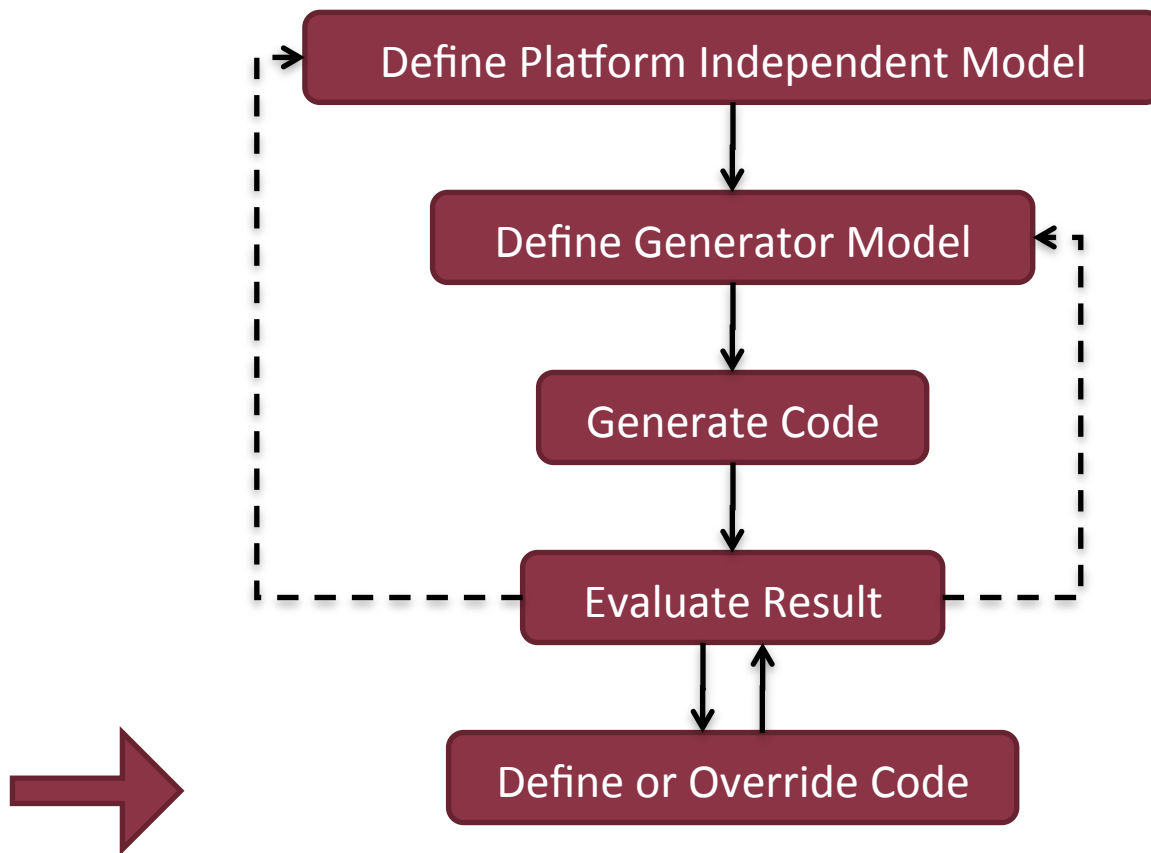
# Kódgenerálás

- ECore modellből forráskód generálható
  - Modell perzisztencia
  - Modell menedzsment
  - Modell editor
  - Modell tesztelés
- Ezek testre szabhatóak
  - Generátor modell
  - Kód sablonok

# Kódgenerálás

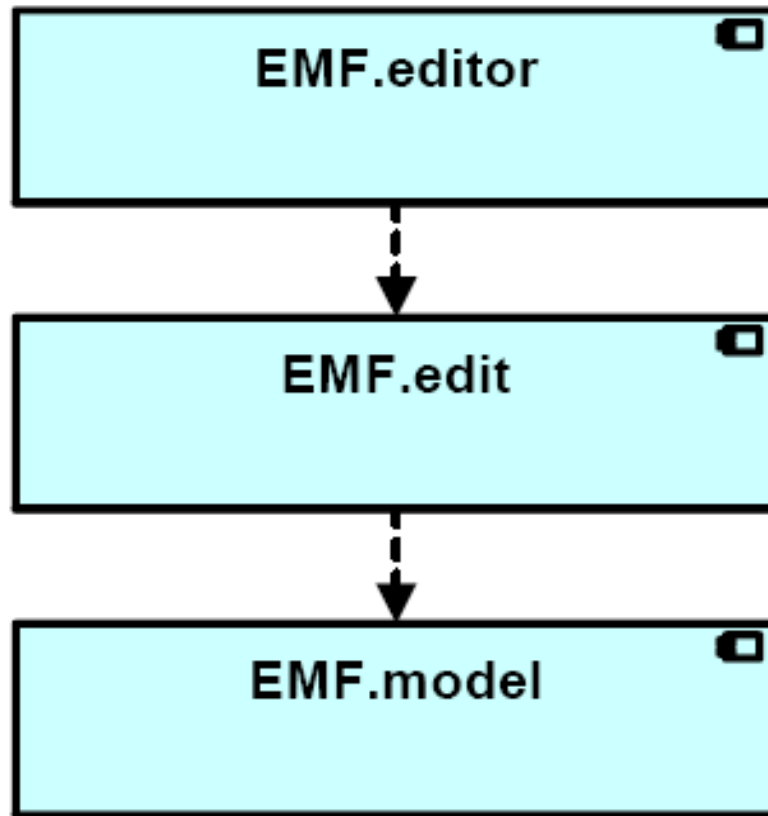
- A generálás alapja egy genmodel fájl
  - Platform-specifikus
  - Az ECore modell alapján készül
  - Részletes
- Létrejön néhány Eclipse plug-in
  - Alapértelmezett editor
  - Perzisztencia kezelés
  - Modell manipulációs

# Az EMF felhasználása



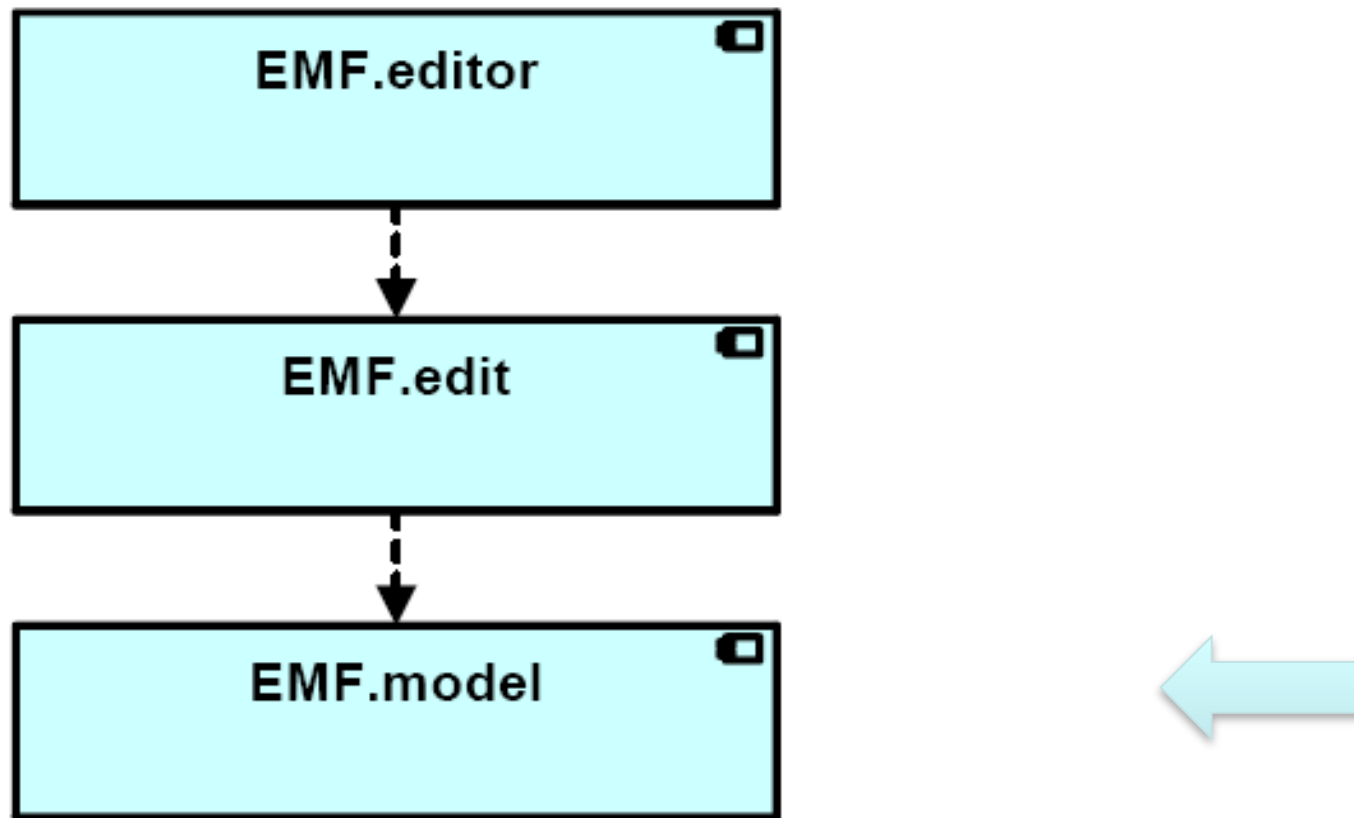


# A generált EMF komponensek



- ❖ 3. Tree editor
- ❖ 2. Modellmanipuláció
- ❖ 1. Modell tárolása

# A generált EMF komponensek



# EMF.Model

- Az ECore modell teljes implementációja
- Hatékony perzisztenciakezelés
  - XMI technológia
- A modell és a kód közel van egymáshoz
  - Előre tudjuk, mit kapunk
  - Általában nem szükséges módosítani

# EMF.Model

- Lehetséges kiegészítések
  - Saját fájlformátum támogatás
    - Parser
    - Okos szerkesztő
    - Xtext projekt így működik
  - Extra információk beszúrása a generált forrásfájlokba
    - Kerülendő
    - Inkább legyen az ecore modell része

# Az ECore keretrendszer

- Főbb képességek
  - Reflexió
  - Értesítés (notification)
  - Perzisztencia
  - Asszociációk kezelése
  - Többszörös öröklődés

# Reflexió

- eClass()
  - Minden üzleti objektum megkaphatja a saját EClass reprezentációját
  - Hasonló a Java getClass() hívásához
    - eGet/eSet/elsSet/eSet/eUnset()
  - Tulajdonságok generikus kezelése
- eContainer/eContents()
  - Hierarchia
  - Szülő/gyerekek visszaadása tartalmazások mentén

# Reflektív API

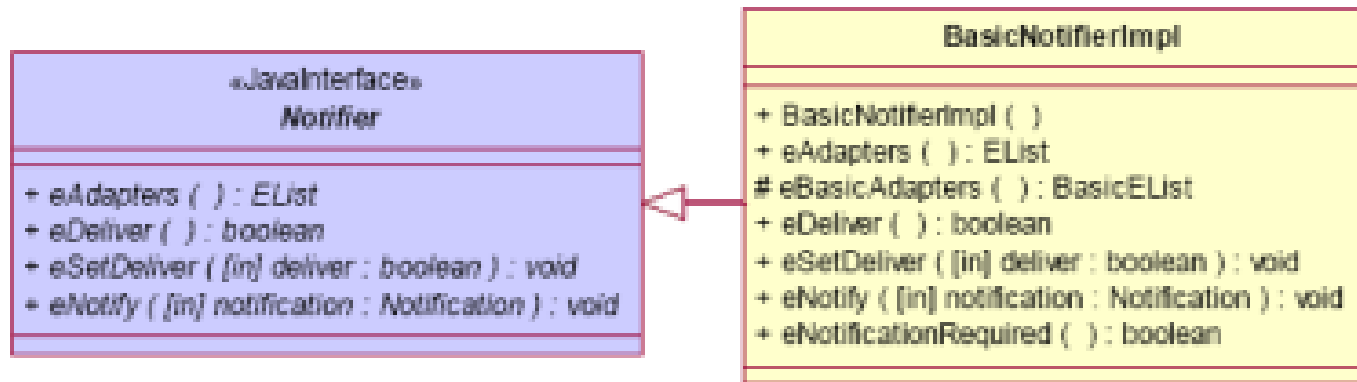
- Platform biztosítja
- Általános szolgáltatásokhoz
  - Sorosítás
  - Értesítés
  - Switch megoldásokhoz

# Generikus vs generatív

- Generikus megoldás
  - Általános, minden esetre kiterjedő
    - Pl. sorosítás, Reflective Tree Editor
- Generatív megoldás
  - Kódot generálunk a modell alapján
  - Minden esetre lehet eltérő kódot használni
    - Pl. modell-specifikus tree editor, később GMF editor



# Értesítés



- Minden modellobjektum támogatja az értesítésküldést
  - Observer minta
  - Event objektumok az értesítésben
  - Genmodellben állítható, hogy mi vált ki értesítést

# Értesítés

- Feliratkozás
  - `eAdapters () .add (Adapter)`
- Értesítésküldés
  - `eNotify (Notification)`
- A megvalósítást nem részletezzük
  - EMF biztosítja
  - Tipikusan sose akarjuk módosítani...

# Perzisztencia

- EMF modellek tárolása: erőforrásokban (resource)
- Egy objektumhoz rendelhetünk egy Resource példányt
  - `eResource()` adja vissza
- Beépített implementáció: `XMIResourceImpl`
  - Betöltés/mentés XMI formátumú fájlokból/fájlokba

# EPackage implementáció

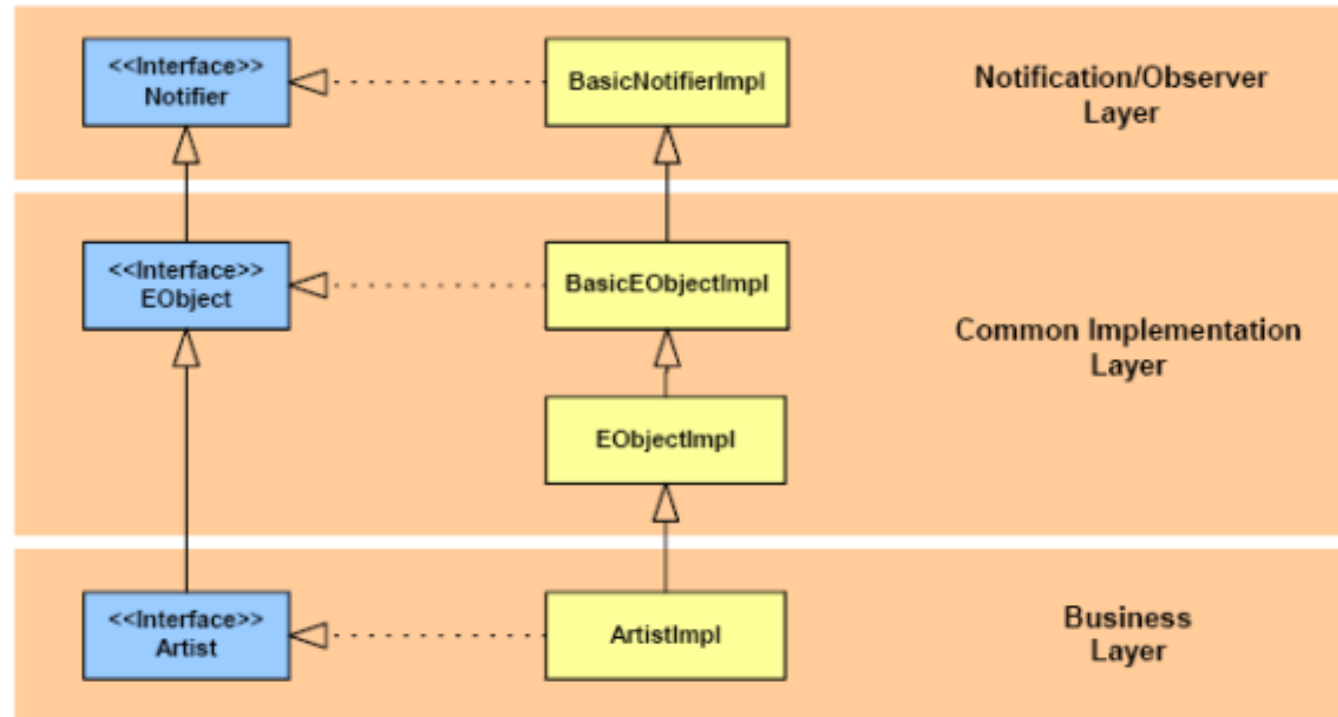
- Példány elérése:  
`<csomagnév>Package.eINSTANCE`
- Minden osztályhoz metódus
  - `EClass` `get<osztálynév>`
- Minden osztály minden tulajdonságához metódus
  - `EStructuralFeature`  
`get<osztálynév>_<tulajdonságnév>`

# EFactory implementáció

- Példány elérése:
  - `<csomagnév>Factory.eINSTANCE`
  - `<csomagnév>Package.eINSTANCE`  
`.get<csomagnév>Factory`
- Minden osztály példányosítására metódus
  - `<osztálynév> create<osztálynév>`

# EClass implementáció

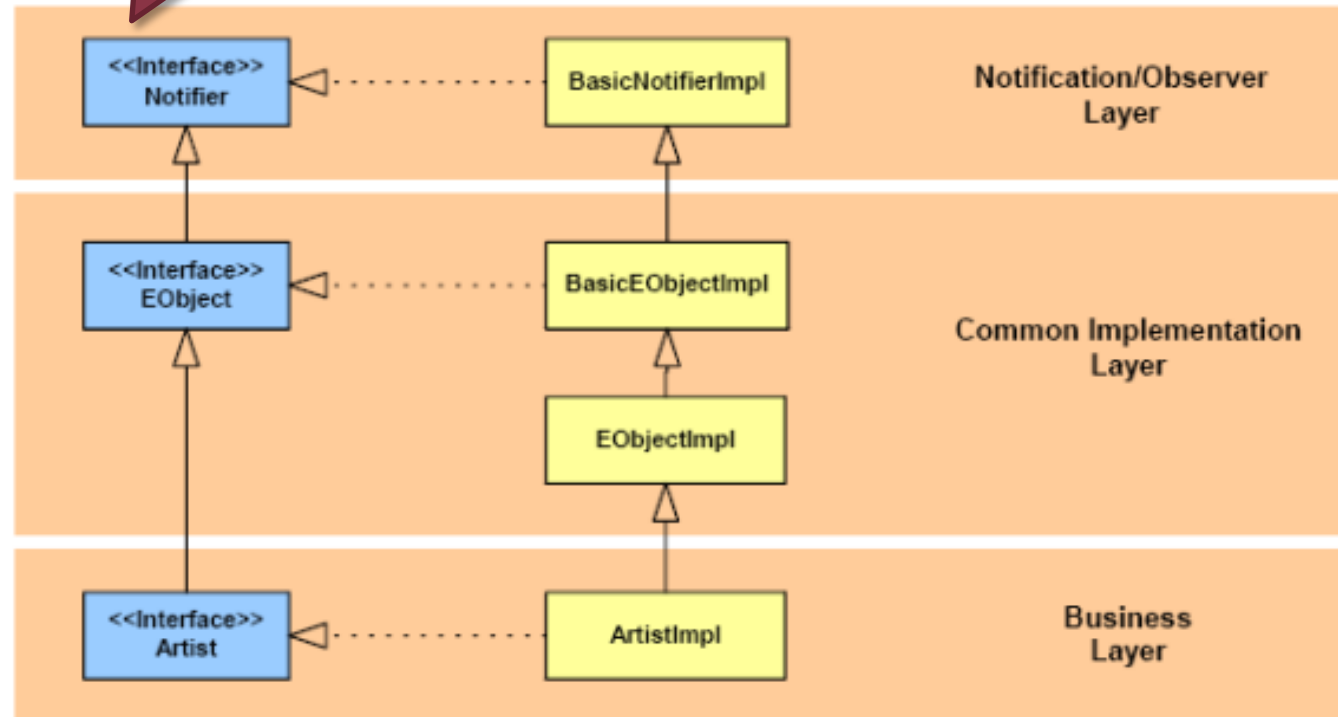
- A generált kód egy előre definiált keretrendszerterjeszt ki



# EClass implementáció

- A generált kód a már előre definiált keretrendszerterjeszt ki

Publikus  
interfészek

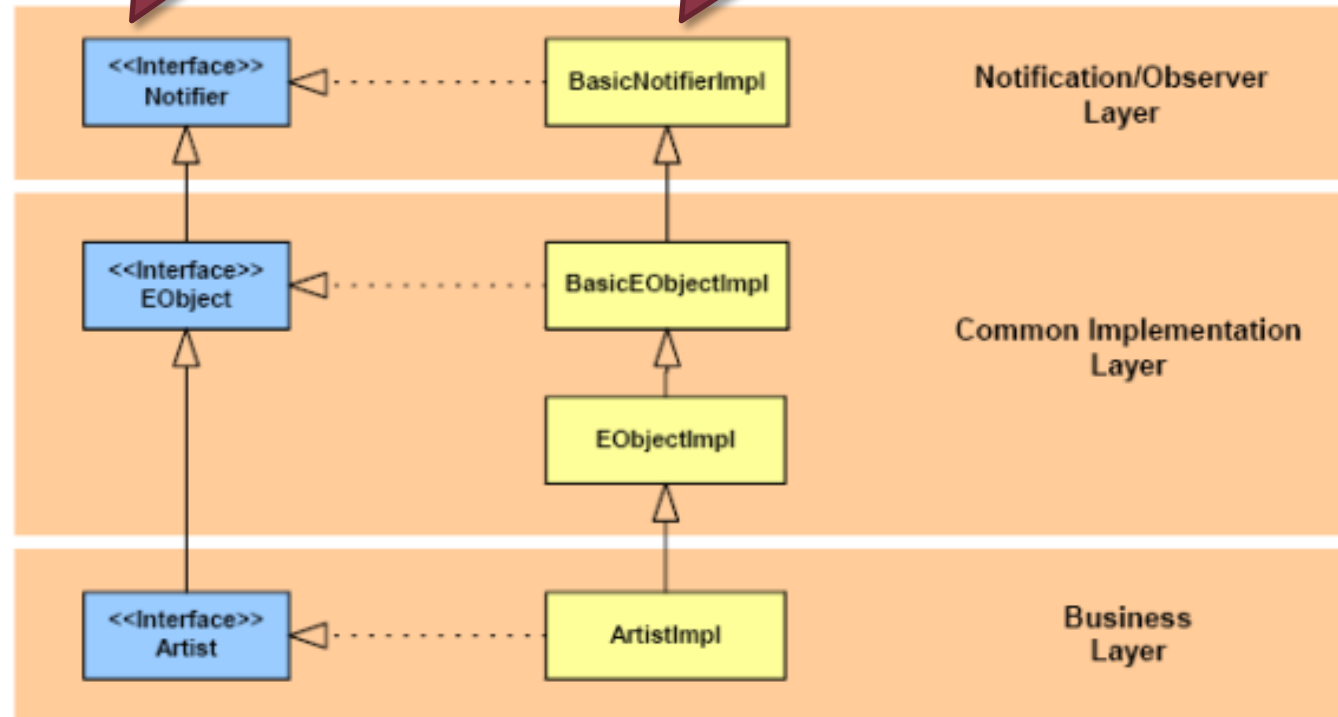


# EClass implementáció

- A generált kód az előző definiált keretrendszer terjeszt ki

Publikus  
interfészek

Implementációs  
osztályok





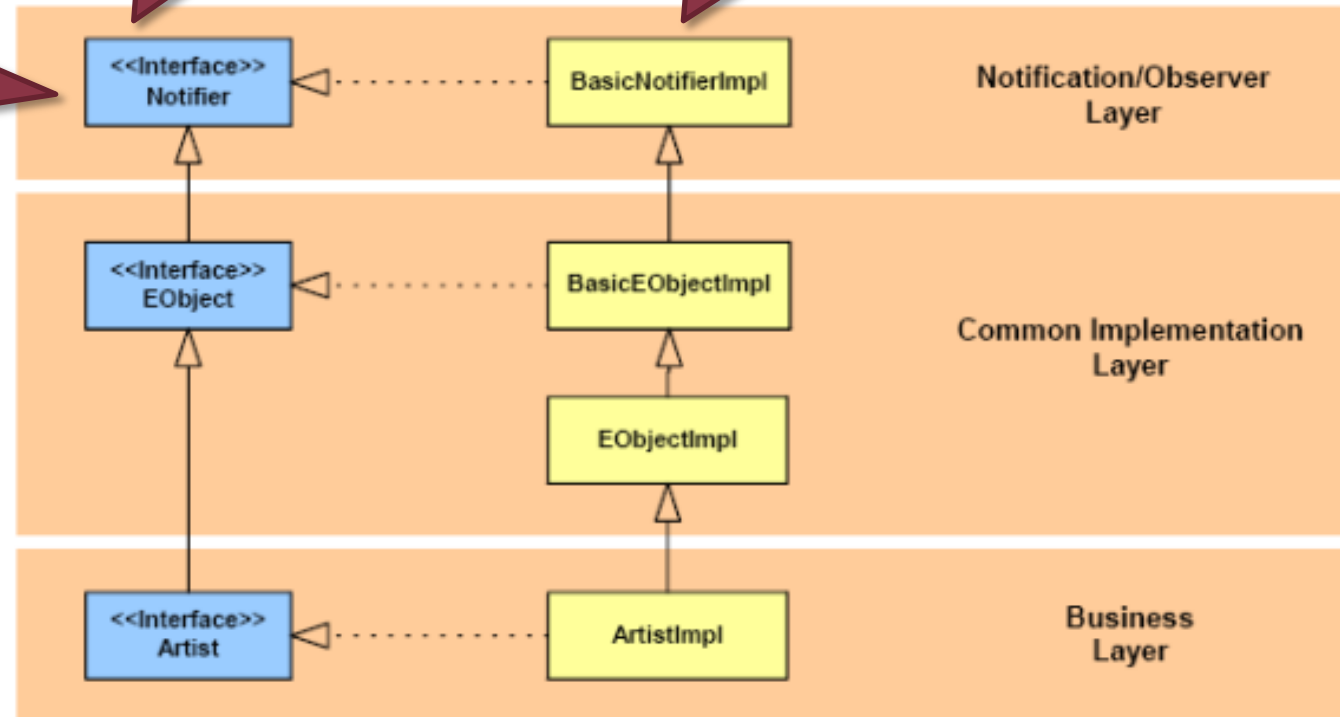
# EClass implementáció

- A generált kód az előző definiált keretrendszerterjeszt ki

Publikus  
interfészek

Implementációs  
osztályok

Általános  
értesítési  
mechanizmus



# EClass implementáció

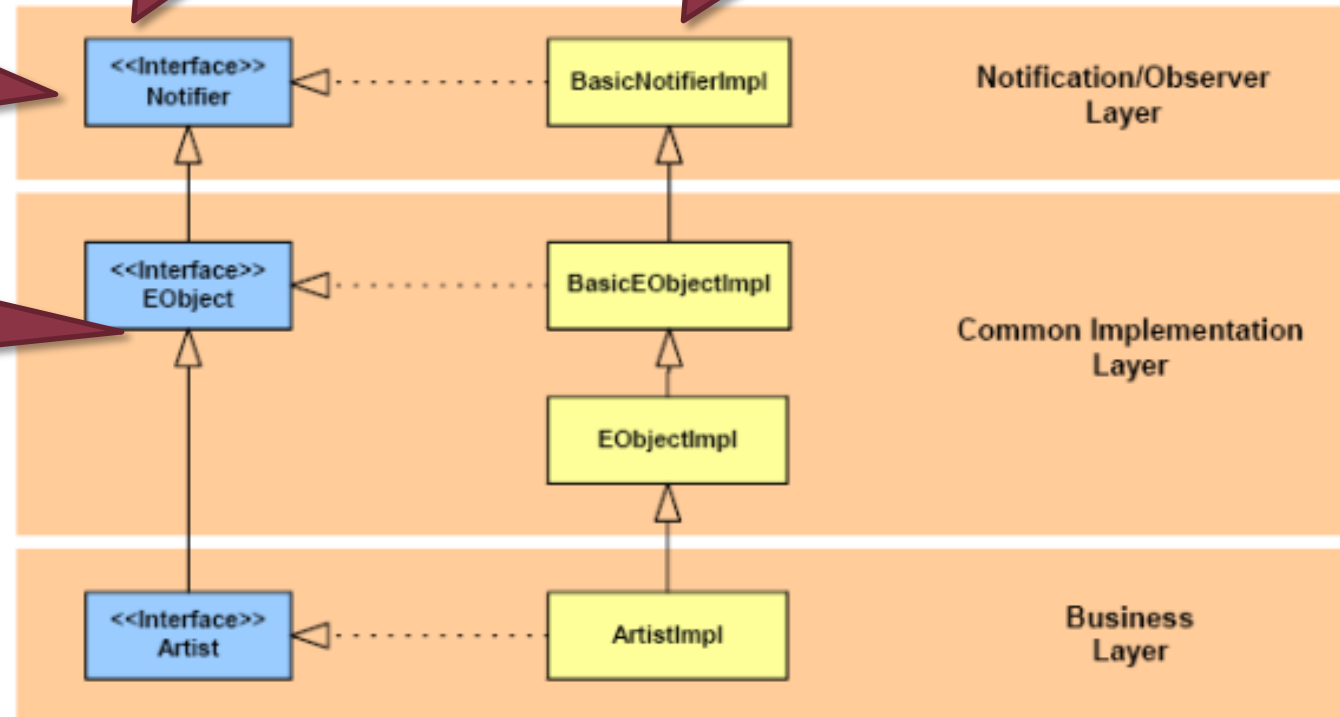
- A generált kód az előző definiált keretrendszer terjeszt ki

Publikus  
interfészek

Implementációs  
osztályok

Általános  
értesítési  
mechanizmus

EMF  
megvalósítás



# EClass implementáció

- A generált kód a szülőben definiált keretrendszerterjeszt ki

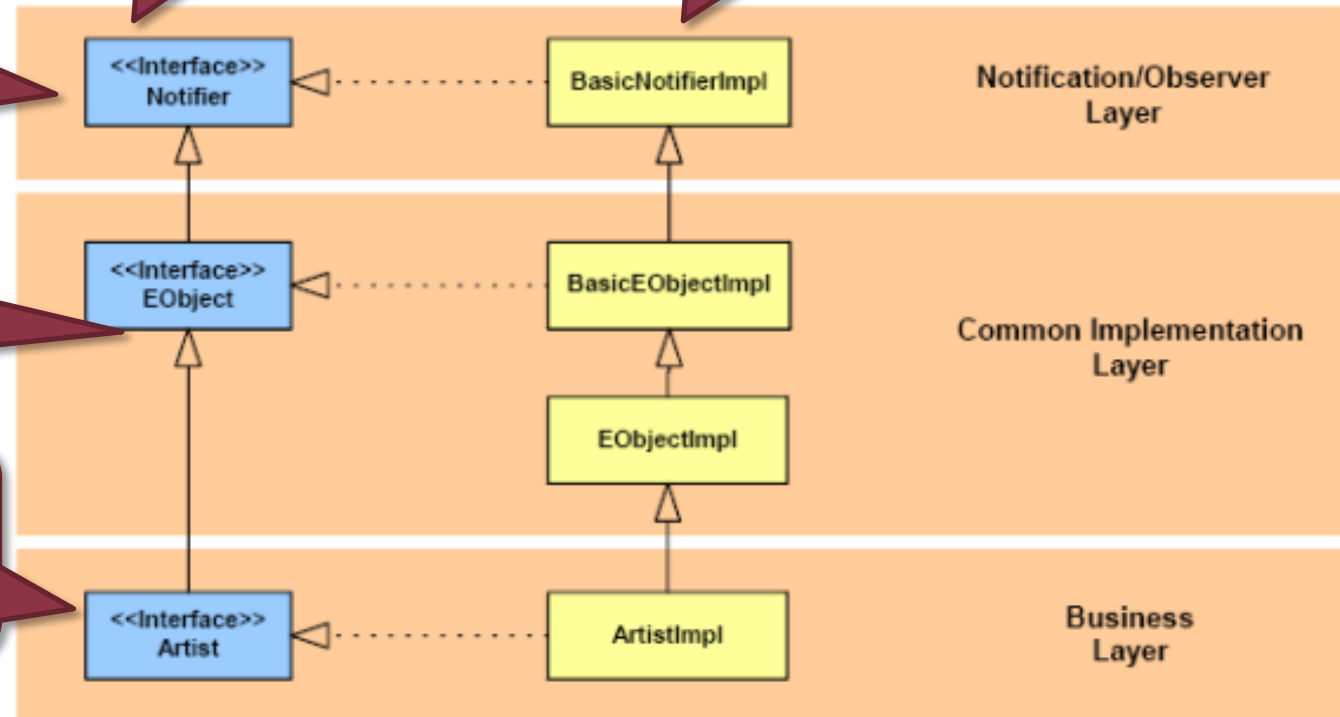
Publikus  
interfészek

Implementációs  
osztályok

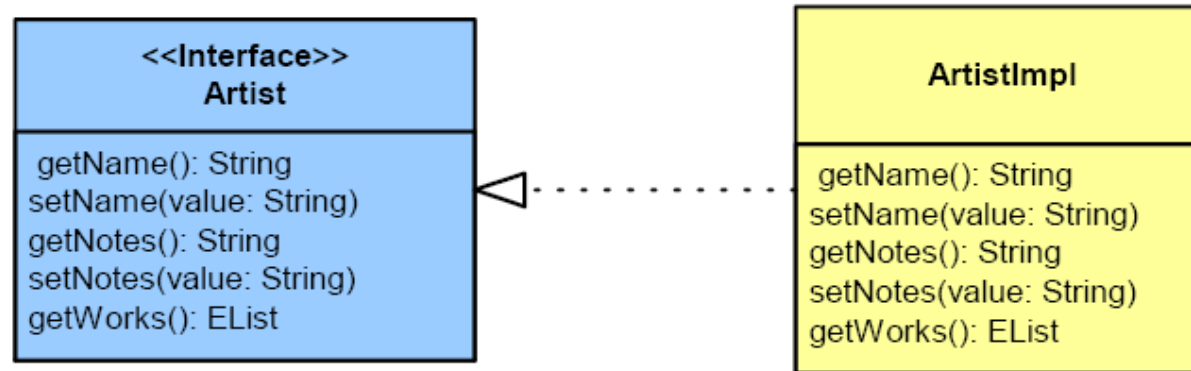
Általános  
értesítési  
mechanizmus

EMF  
megvalósítás

Saját meta-  
modell elemei



# EClass implementáció



- **Attribútumok és referenciák kezelése**
  - EAttribute: getter és setter metódusok
  - EReference: multiplicitásfüggő
    - “many”: szerkeszthető kollekciónak, hozzá egy getter metódus
    - “one”: getter metódus az értékre
- **Kezdőértékek tárolása és inicializálása**

# EAttribute implementáció

```
protected static final String NAME_EDEFAULT = null;

protected String name = NAME_EDEFAULT;

public String getName() {
    return name;
}

public void setName(String newName) {
    String oldName = name;
    name = newName;
    if (eNotificationRequired()) {
        eNotify(new ENotificationImpl(...));
    }
}
```

- Általában nem akarjuk módosítani az implementációt
  - Kivéve pl. származtatott tulajdonságok

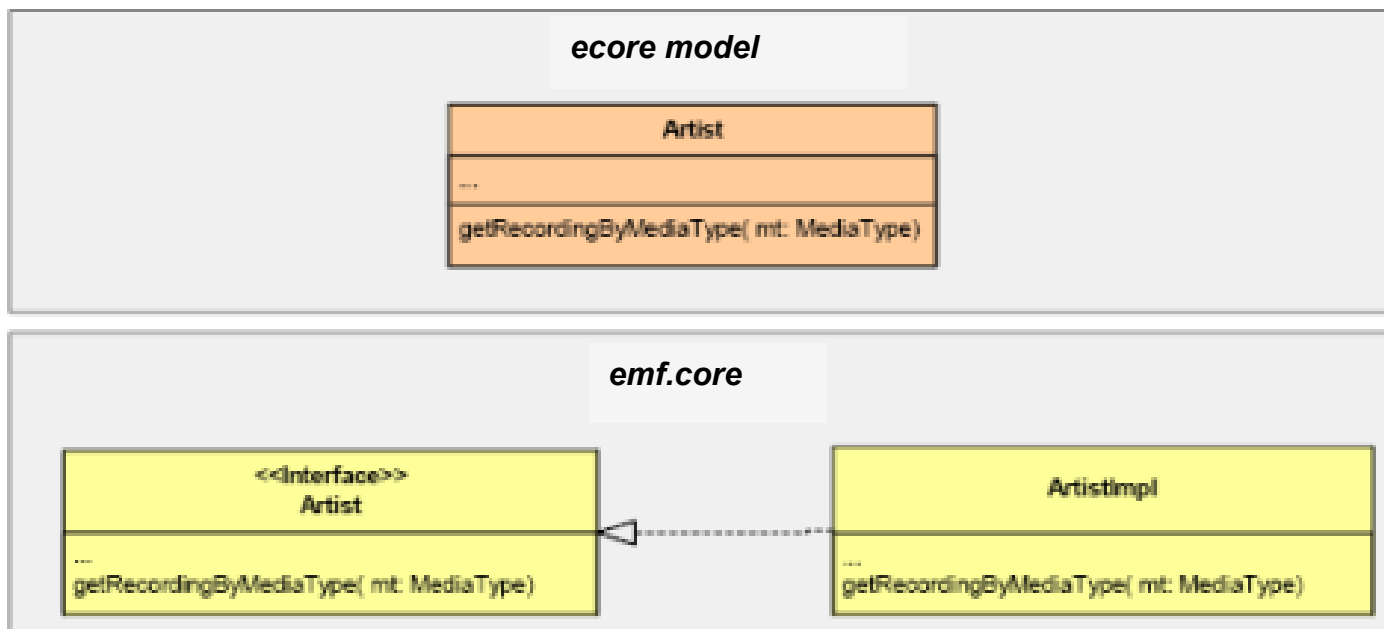
# EReference implementáció

- Hasonló az EAttribute-hoz
- A típus itt egy másik objektum lesz
- Néhány kiegészítés szükséges, ha nem tartalmazás jellegű reláció van
  - Az objektum más tárban is lehet
    - Referencia-feloldás
  - Ellenőrizni kell a kétirányú referenciák integritását
    - Ellenkező oldal frissítése

# EOperation implementáció

- Az ECore modellben metódusokat is megadhatunk
- Nincs támogatás a szemantika definiálására
- Ötlet
  - Az ECore modellben definiáljuk a metódus
    - Nevét
    - Paramétereit, típusukat
    - Visszatérési értékének típusát
  - Implementáljuk Java-ban
- A kódgenerátor csak kódvázat generál

# EOperation implementáció



- Ha definiáltunk egy metódust
  - Hozzáadja az interfészhez
  - Egy üres implementációt készít az implementáló osztályban

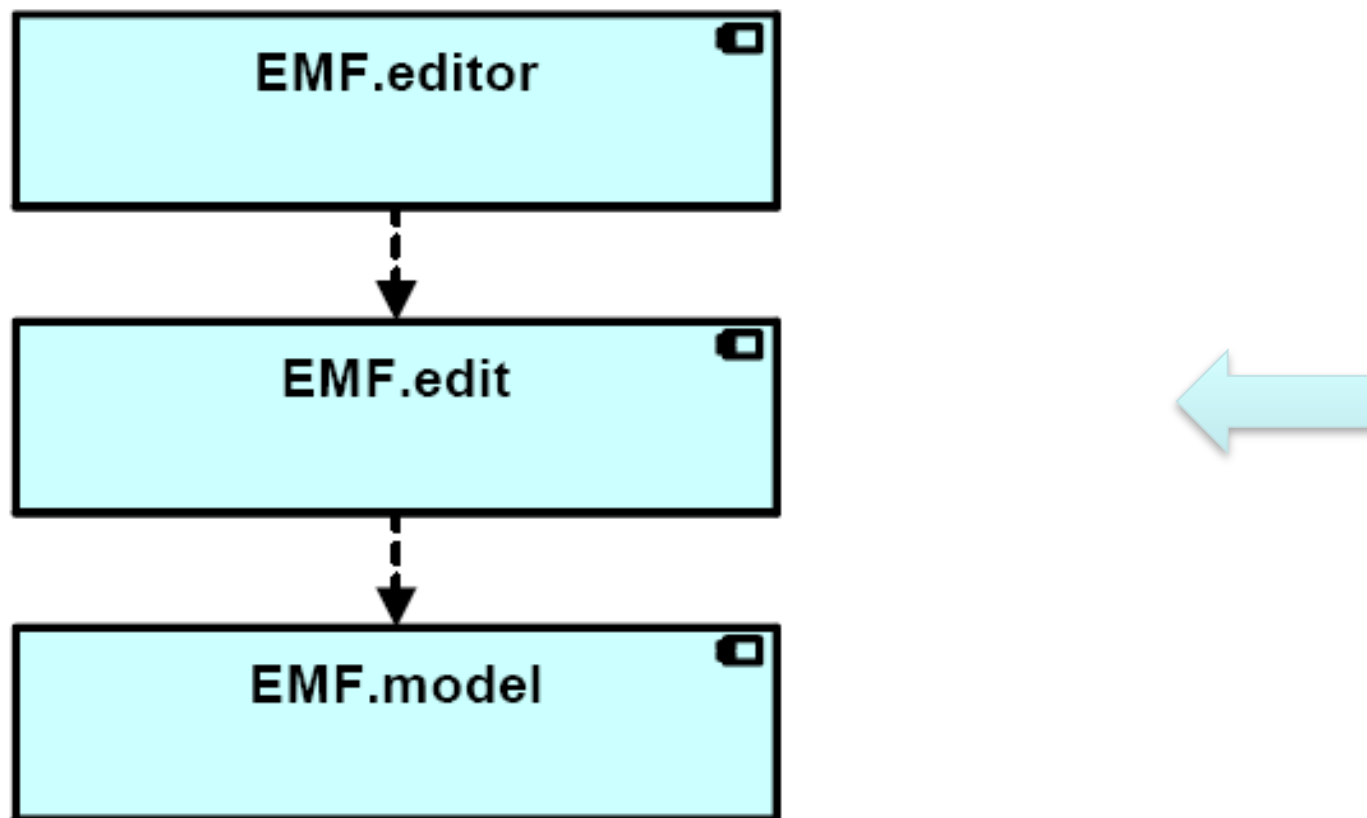


# EOperation implementáció

```
public class Ximpl extends EObjectImpl implements X {  
  
    /**  
     * @generated NOT  
     */  
    void f() {  
        // Provide the implementation  
    }  
}
```

- Első generálásnál az EMF egy Exception-t dobó implementációt készít
- Át kell venni a felügyeletet
  - Beállítjuk a @generated taget NOT-ra
  - Implementáljuk a metódust

# A generált EMF komponensek



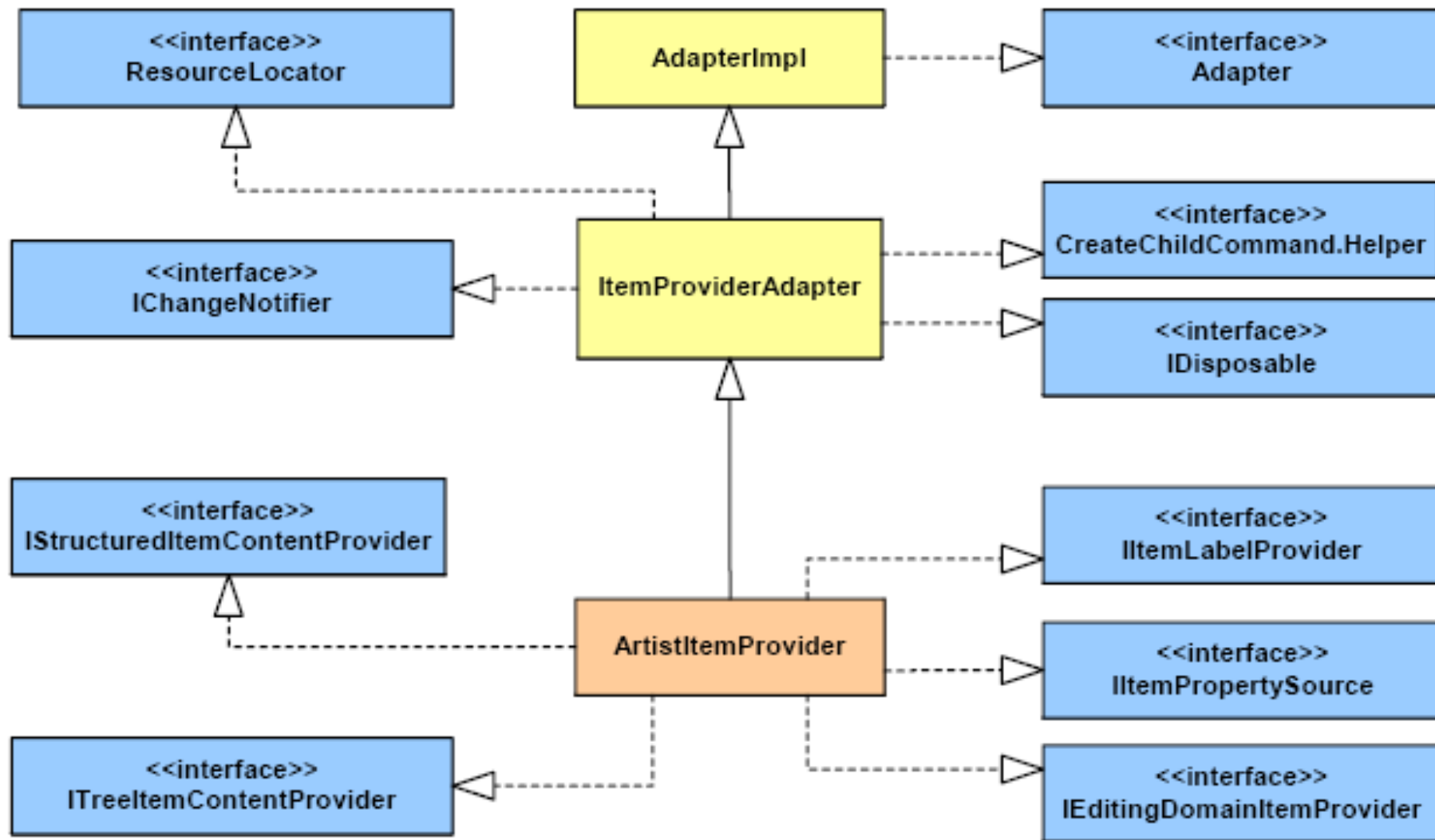
# EMF.Edit

- Szerepe
  - A GUI és a modell szétválasztása
  - GUI független akciók implementálása
- Nagyobb eséllyel módosítjuk
  - Módosítjuk az element providert
  - Új parancsokat adunk hozzá

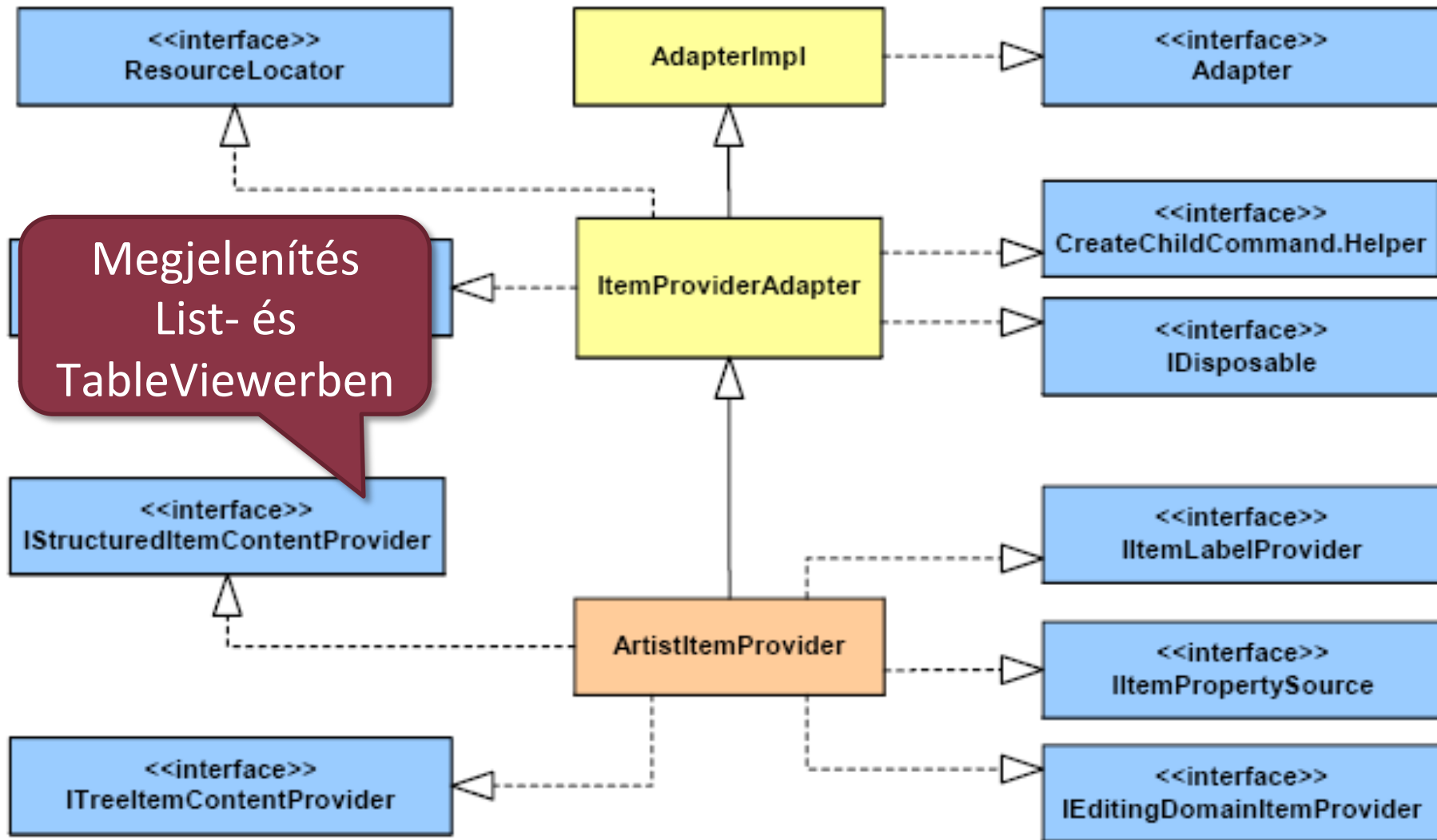
# Generator minta

- Minden modell objektumhoz
  - Egy adapter jön létre
    - Neve: ItemProvider
    - Pl. ArtistItemProvider
- A providerek közös őse:
  - `org.eclipse.emf.edit.provider.ItemProviderAdapter`
    - Alapértelmezett implementáció alap funkcionalitáshoz
    - Egy részét definiáljuk felül

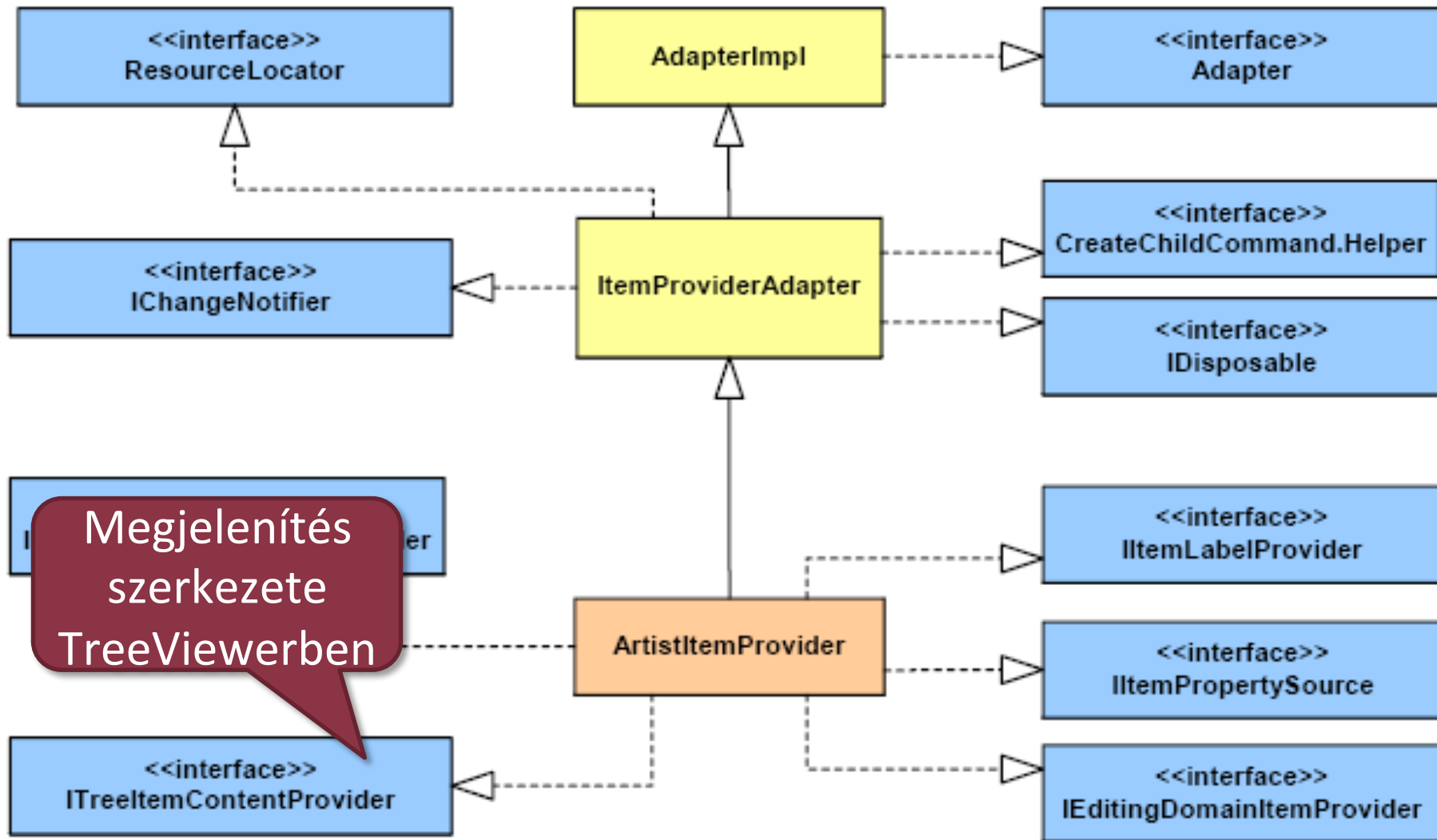
# Generator struktúra



# Generator struktúra

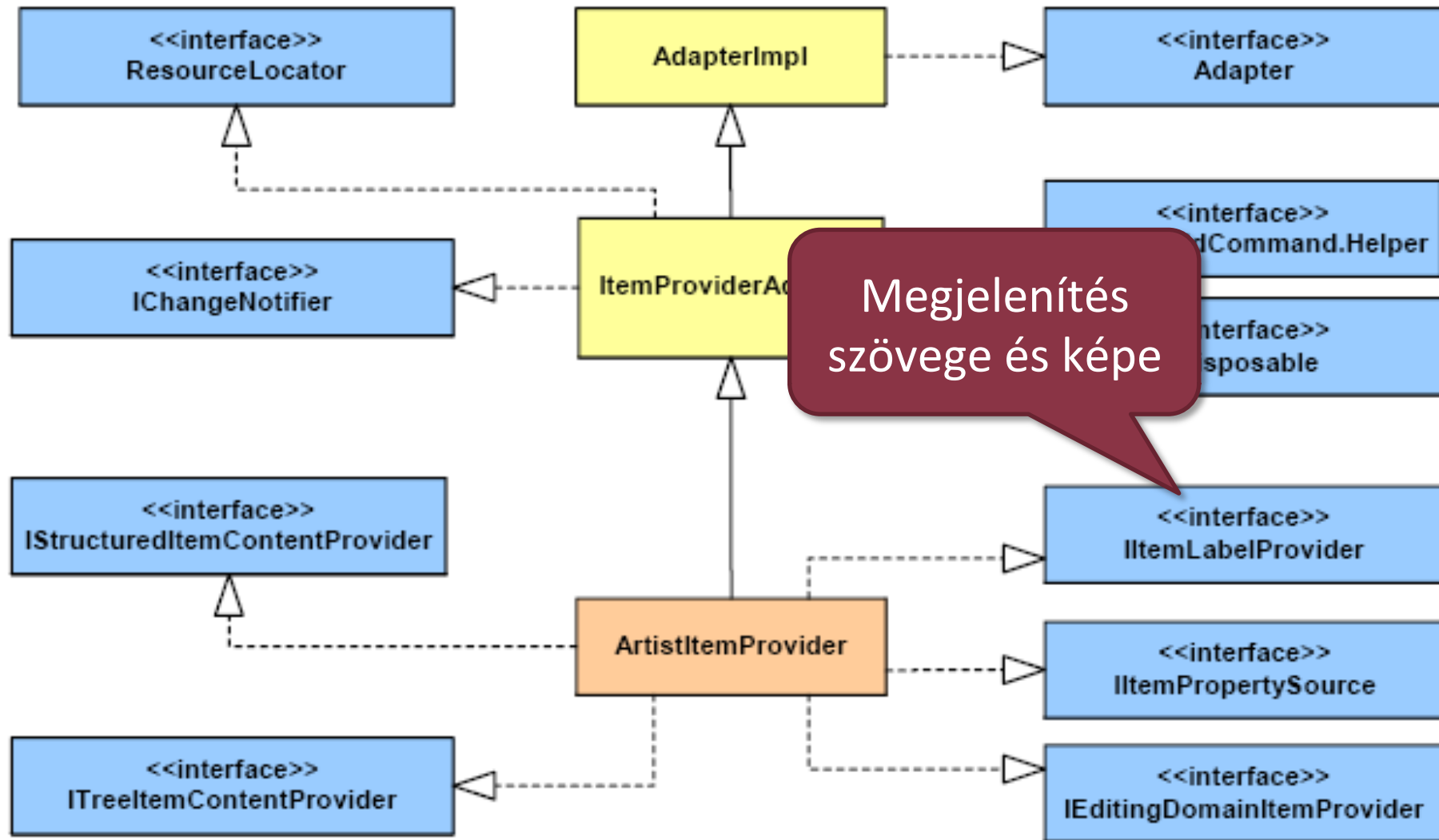


# Generator struktúra



Megjelenítés szerkezete TreeViewerben

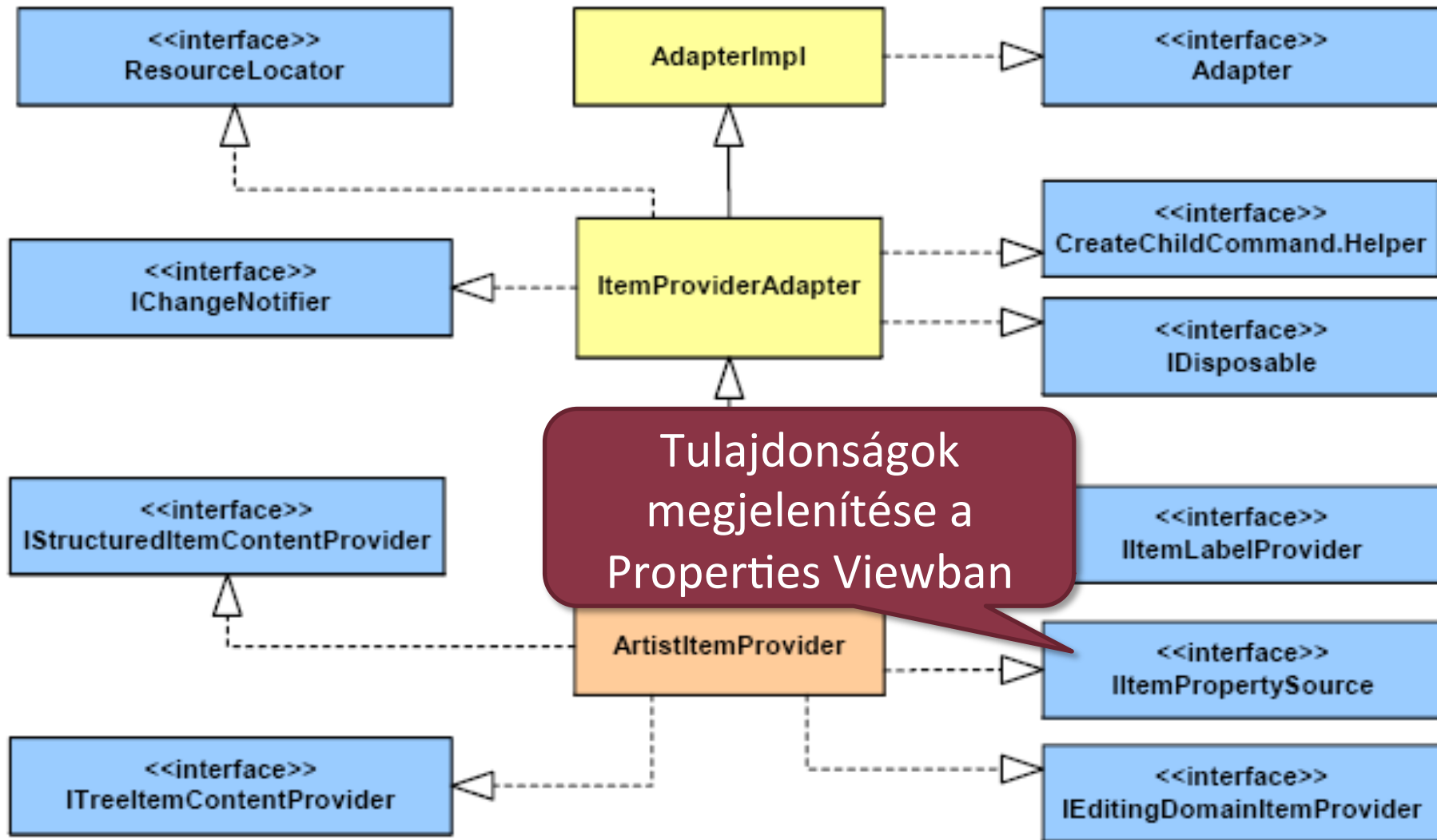
# Generator struktúra



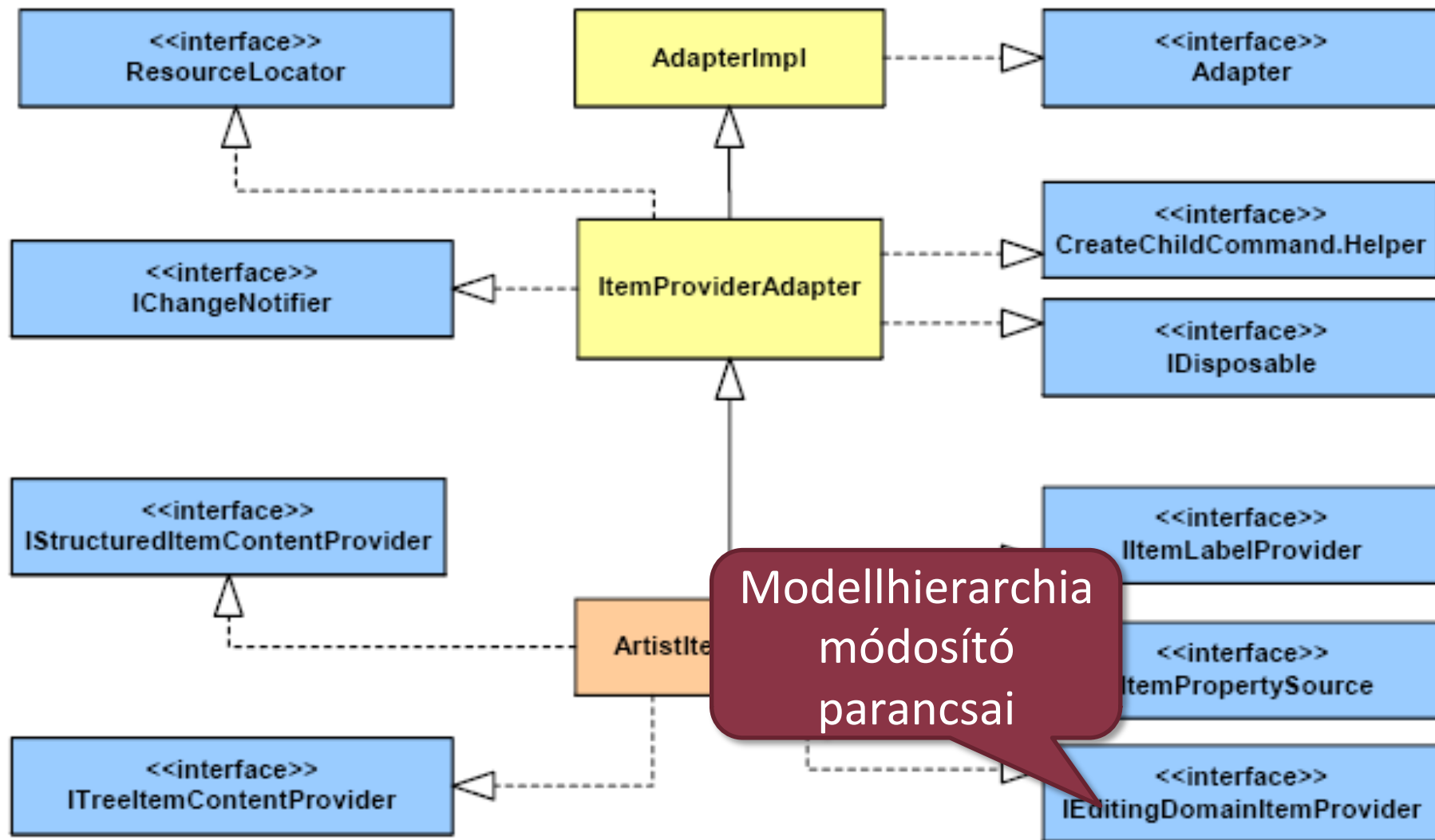
Megjelenítés szövege és képe



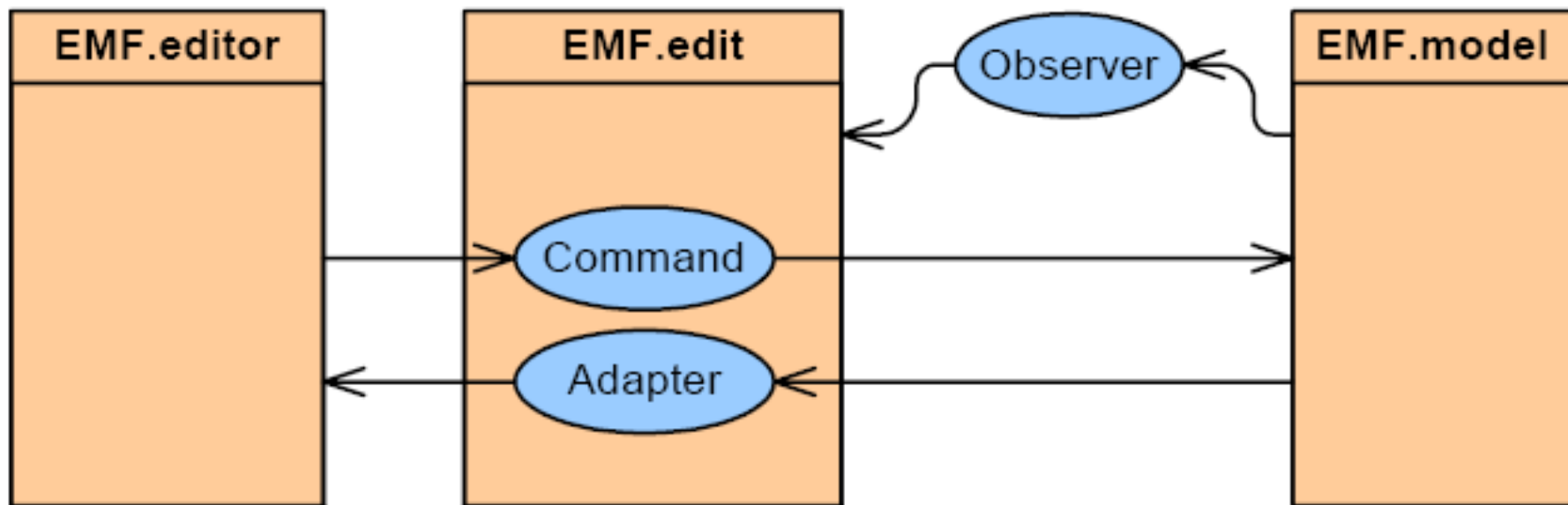
# Generator struktúra



# Generator struktúra



# EMF.Edit és a tervezési minták



# Címkék változtatása

- A testreszabás tipikus példája a címkék változtatása
  - A genmodel-ben megadhatjuk, hogy egy objektum mely attribútuma jelenjen meg címkeként
  - Mi van, ha többől akarjuk összerakni?
- Változtatni kell a `ItemProvider.getText`-en
  - Töröljük/NOT-ra állítjuk a `@generated` taget
  - Saját implementációt írunk

# Ikonok változtatása

- Másik tipikus példa az ikonok megváltoztatása
- A genmodel egy egyszerű ikont rendel minden elemhez
  - Az elemek az edit projekt icons/full könyvtárában találhatóak
    - obj16: konkrét osztályokhoz
    - ctool16: gyerekelem-létrehozási parancsokhoz
- A legegyszerűbb a fájlok lecserélése saját ikonra
- Ha logika alapján kell: `ItemProvider.getImage` felülírása

# EMF.Edit parancsok

- Minden módosítás parancsokon (command) keresztül történik
  - Menü akció
  - Attribútum változtatás
  - Drag-n-drop
- Undo/redo támogatás
- A keretrendszer generált és már létező kód keverékét használja
  - EMF Common Command Framework (CCF)
  - EMF.Edit által generált parancsok

# Parancsok használata EMF.Edit-ben

- A parancsok használata a Template Method mintára épül
- Az ItemProvider implementálja a `createCommand()` metódust, a kéréseket továbbítja `protected` metódusaiknak
  - `createAddCommand()`
  - `createRemoveCommand()`
  - ...
- A módosítás során a `protected` metódusok egyikét módosítjuk általában

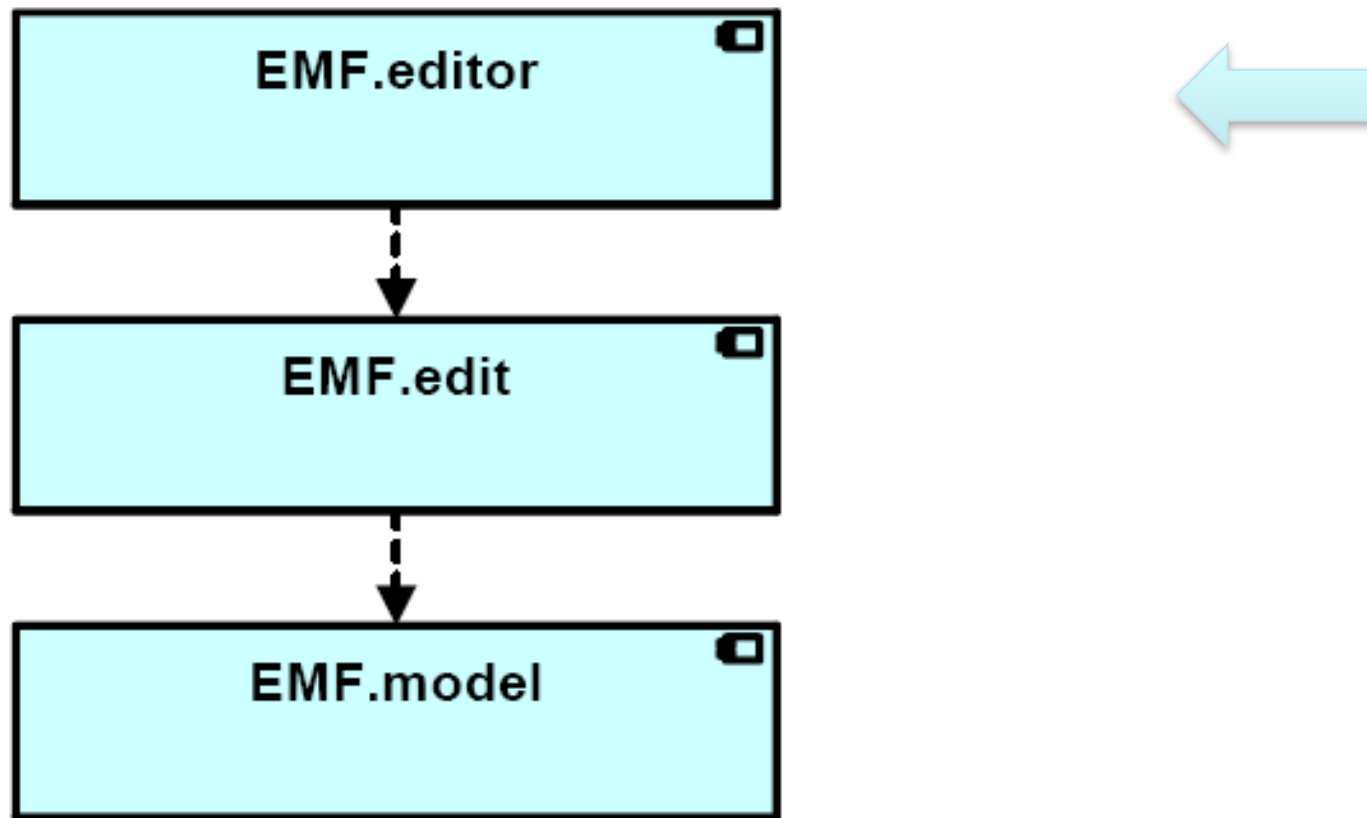
# Parancsok az EMF.Edit-ben: Példa

```
public class SetArtistNameCommand extends SetCommand {  
  
    public SetArtistNameCommand(EditingDomain domain, Eobject owner,  
        EStructuralFeature feature, Object value) {  
        super(domain, owner, feature, value);  
    }  
  
    public void doExecute() {  
        Artist artist = (Artist)this.owner;  
        Logger.log(MessageFormat.format(  
            "Name of artist changed from {0} to {1}",  
            artist.getName(), value.toString()));  
        super.doExecute();  
    }  
  
}
```

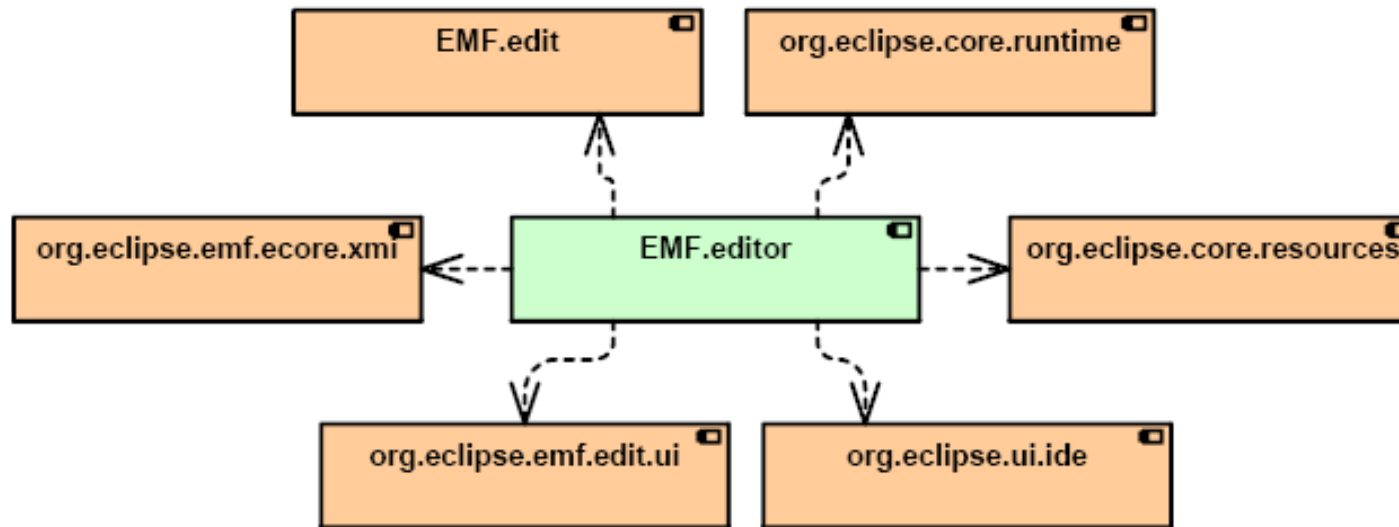
- Egyszerű példa: loggolás hozzáadása
- Bonyolultabb példák esetén az összetett parancsok módosítása is szükséges lehet



# A generált EMF komponensek



# EMF.Editor



- Az EMF.Editor generálja az SWT/JFace kódot a grafikus editorhoz
- Két fő opció
  - Meghagyjuk alapértelmezetten
  - Újraimplementáljuk teljes egészében

# Mi generálódik?

- Editor (JFace alapú)
  - Fastruktúra
  - Események – akciók összekötése
  - Workbench elemek beállítása
- Menük
- Varázsló (új modell...)
- Plugin activator

# A generált szerkesztő

The screenshot displays a software interface with three main panels:

- Resource Set:** A tree view showing a hierarchy starting with 'platform:/resource/Examples/default.socialnetwork', which contains a 'Social Network' folder. Inside 'Social Network', there are four items: 'Person John Doe' (highlighted), 'Community SpongeBob Fan Club', 'Person Jane Doe', and 'Acquaintance friendship'.
- Outline:** A tree view showing the same hierarchy as the Resource Set panel.
- Properties:** A table showing the properties of the selected 'Person John Doe' item.

Property	Value
Membership	Community SpongeBob Fan Club
Name	John Doe
Sex	male

# EMF.Editor - Összefoglalás

- A generált editor elsődlegesen tesztelésre való
  - Fastruktúra nehezen átlátható
  - Alapszintű műveletek
- Alternatív megoldások
  - GMF: GEF technológiára alapuló grafikus editor EMF modellekhez
  - Xtext: szöveges editor EMF modellekhez

# EMF - Összefoglalás

# EMF

- Általános modellező keretrendszer
  - Többféle bemenet
  - Sorosítási és szerkesztési támogatás
- Kódgenerálás
  - Testreszabható
  - Jó alapértelmezések
- Sokan használják
  - Sokféle Eclipse-es projekt alaptechnológiája
  - Jól alkalmazható különféle területeken

# További EMF alapú technológiák

- Validation
  - Kényszerek megadása és ellenőrzése
- Query
  - Magas szintű lekérdezések futtatása
- Compare
  - Modellek strukturális összehasonlítása (pl. verziókezeléshez)
- Teneo
  - EMF modellek perzisztenciája relációs adatbázisba
- SDO
  - Service Oriented Architecture megvalósítása EMF alapon
- CDO
  - Elosztott, kliens-szerver EMF modellek