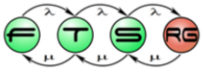
Intelligens rendszerfelügyelet (VIMIA370)

IT rendszerek modellezése

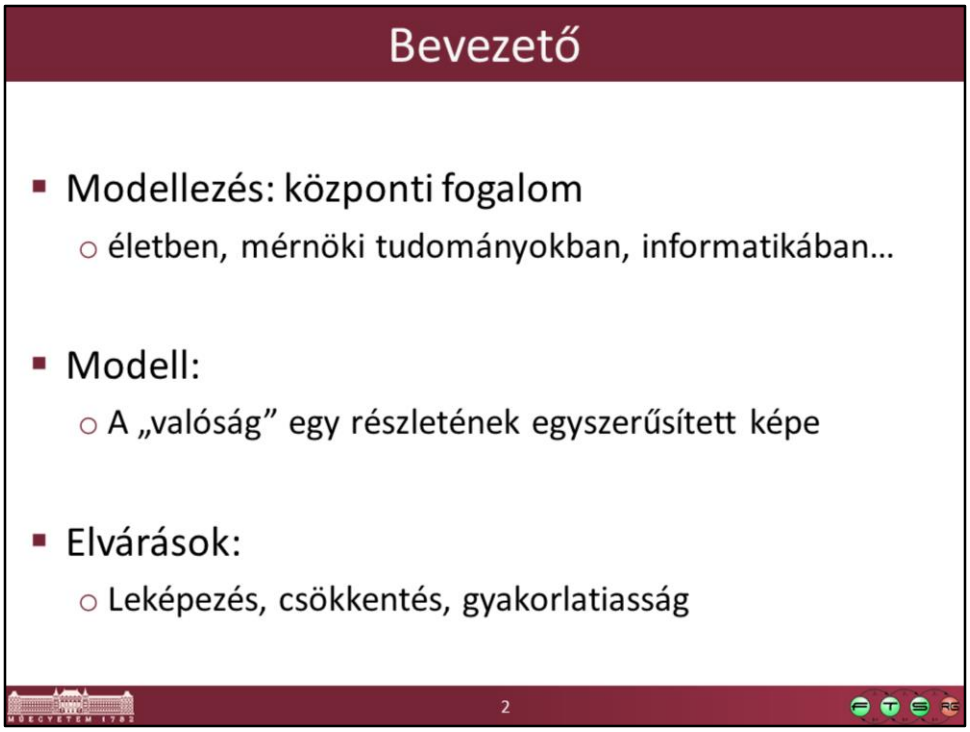
Micskei Zoltán

http://mit.bme.hu/~micskeiz



Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Utolsó módosítás: 2014. 02. 12.



Bonyolult rendszerekkel csak úgy tudunk dolgozni, hogy először egyszerűbb modelleket építünk, és ezeknek a segítségével megvizsgáljuk a rendszert különböző szempontokból.

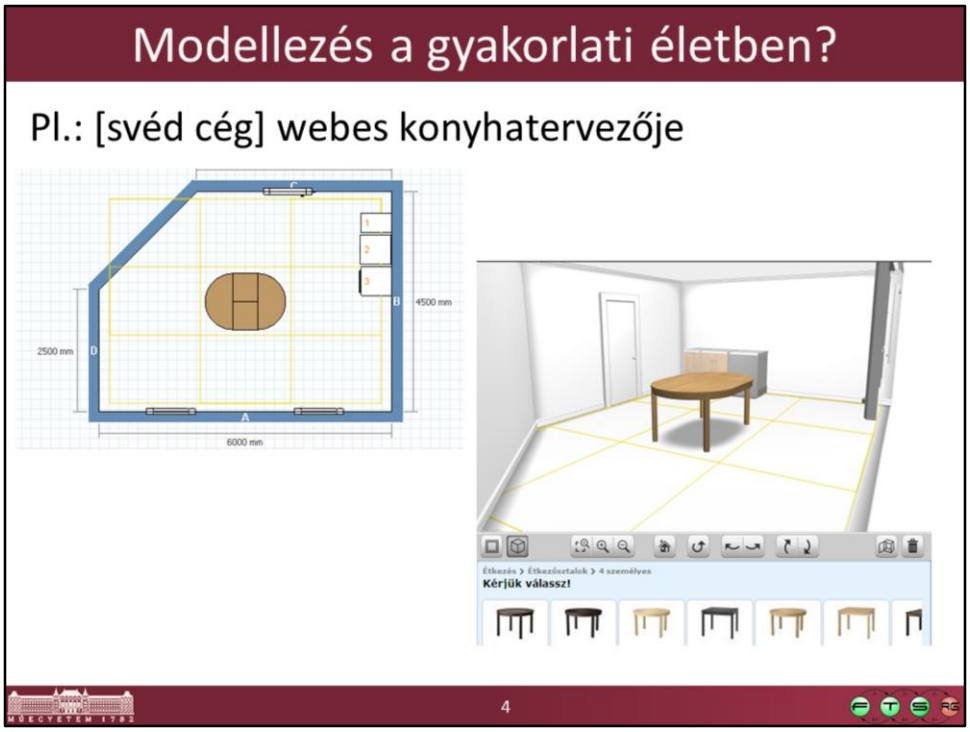
A modellezés nagyon általános fogalom, majd mindenki használja, és természetesen teljesen eltérő módokon, úgyhogy nehéz egyértelmű szóhasználatot találni (modell egy fizikai egyenletrendszer, modell egy épületről készített makett, modell egy térkép…).

A modellezésnek a tantárgyban csak egy kis szeletét érintjük, ami a későbbi előkerülő adatmodellek megértéséhez szükséges. Bővebben majd például a Rendszermodellezés (VIMIA401) vagy későbbi MSc tantárgyakban kerül elő a téma.

Absztrakció

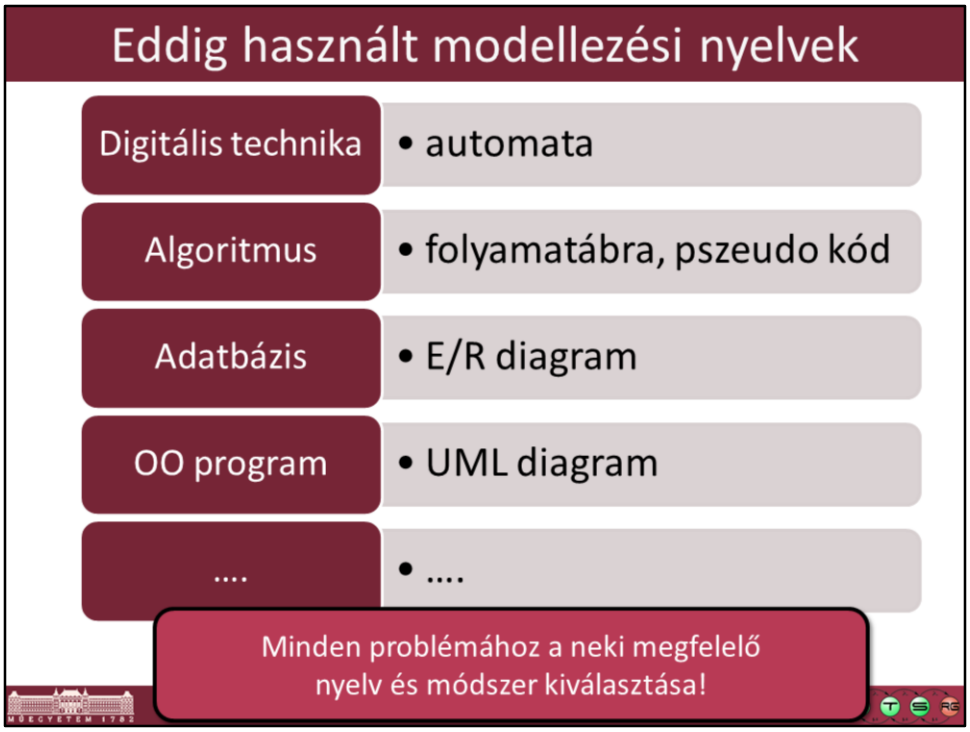
Modell készítésekor absztrakciót használunk

Student
name : String neptun : String inHostel : Boolean

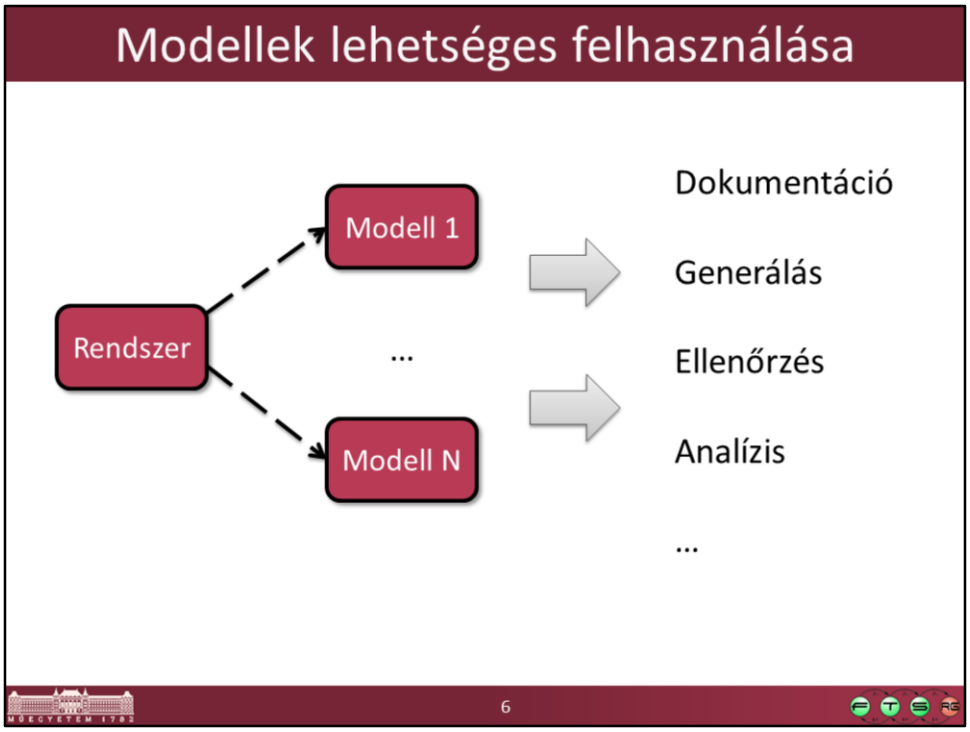


Ez is egy ugyanolyan modell és modellezési nyelv, mint amiket a későbbiekben fogunk használni.

- Jól definiált elemkészlete van, speciális megjelenítési szimbólumokkal, definiált jelentéssel.
- A célja az, hogy egy komplex feladatot könnyebben meg tudjunk oldani.
- Lehet koncepciótervhez használni, lehet dokumentációra használni, lehet bevásárlási listát generáltatni belőle…

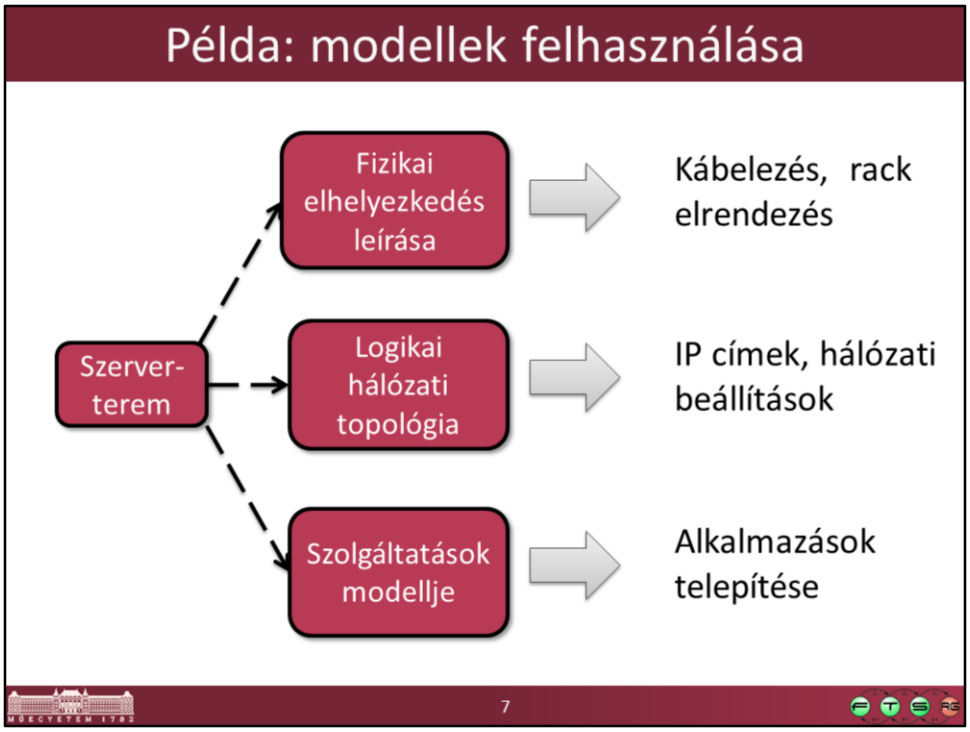


Eddig is rengeteg modellezési nyelvet használtunk már a tanulmányaink során, mindegyik problémához megvan az annak megfelelő nyelv, amivel a legkényelmesebben/leghatékonyabban/legkezelhetőbben le lehet írni a kérdéses rendszert.

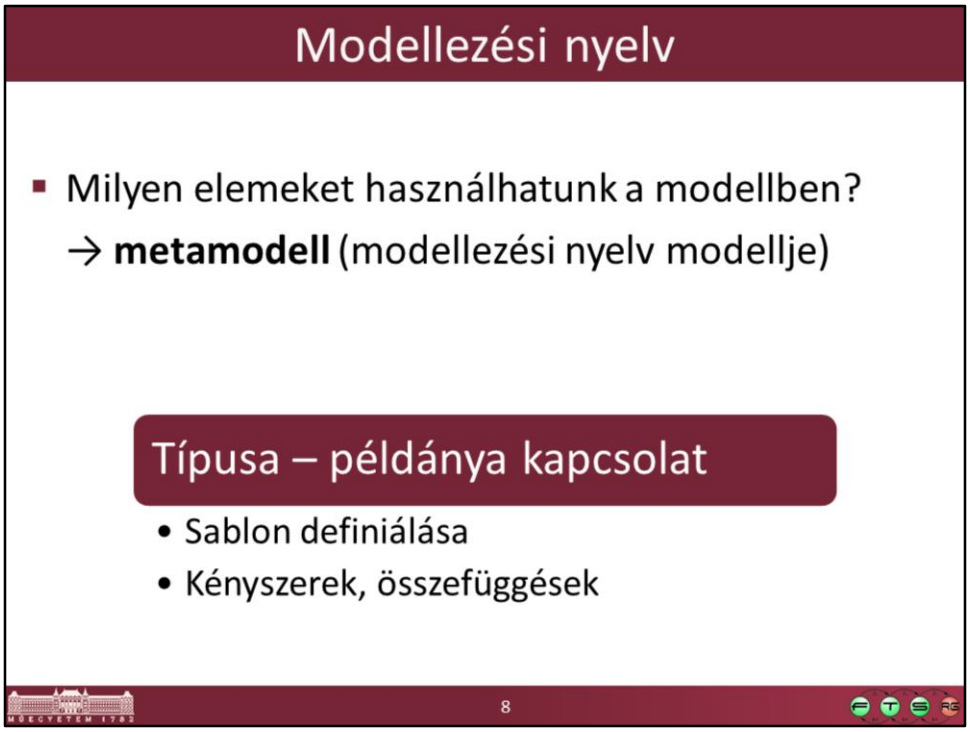


Nagyon sokféle cél miatt építhetünk modelleket. Ha van egy modellünk, akkor azt felhasználhatjuk:
- dokumentáció készítésére: pl. UML modellből diagramok generálása
- generálásra: pl. működést leíró modellből forráskód készítése
- analízisre: pl. rendszer felépítését leíró modellből teljesítményjellemzők számolása, mennyi kérést tudunk kiszolgálni, elég lesz-e egy adott méretű processzor
- ellenőrzésre: helyességi, biztonsági követelmények ellenőrzése, pl. jók-e a rendszer időzítési viszonyai, párhuzamos szálak között nem alakulhat-e ki versenyhelyzet, van-e olyan változó, amit nem szabadítottunk fel, stb.

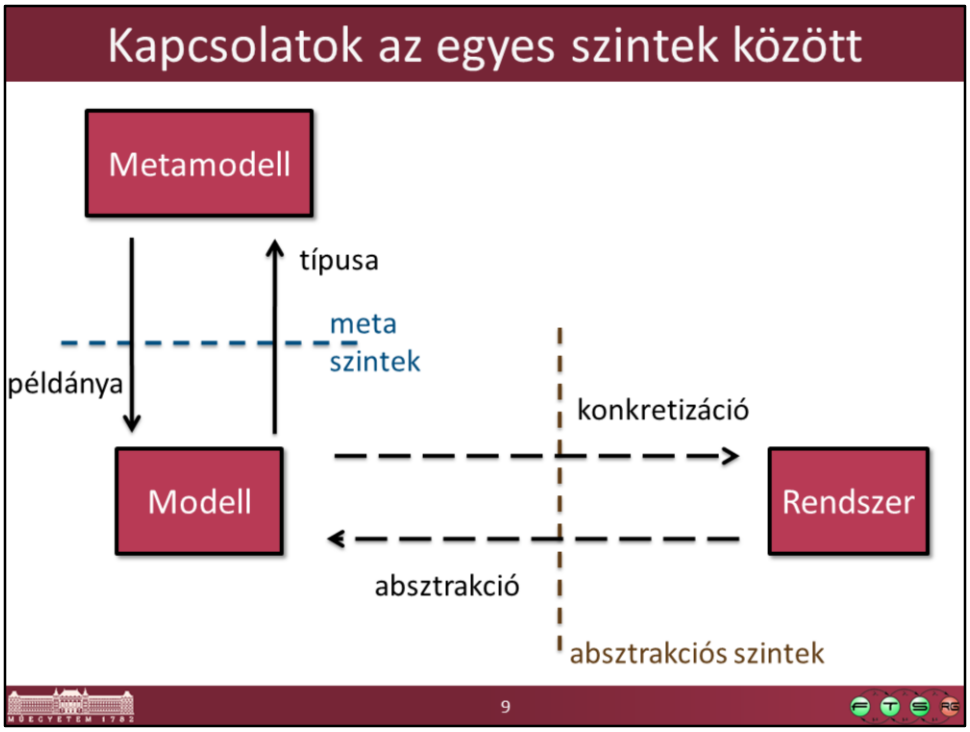
Egy modellt természetesen több célra is felhasználhatunk, de sokszor van olyan, hogy a különböző célokra különböző modellt építünk. Például egy program esetén mondjuk UML osztálydiagramon ábrázoljuk az osztályok közötti kapcsolatokat, és ebből generálunk kód vázakat, ezen kívül pedig felrajzoljuk egy precedencia-gráfot a részegységek kommunikációjáról, hogy megvizsgáljuk, hogy melyik részek párhuzamosíthatóak.



Példa arra, hogy van egy rendszerem, és ahhoz a különféle igényeknek megfelelően különböző modelleket készítek. A rendszert leíró rendkívül sok információból mindegyik modell csak azokat tartalmazza, ami az adott célhoz szükséges, a többit elhanyagolja. A fizikai elhelyezéshez fontos, hogy milyen széles, milyen nehéz egy szerver, ez a tulajdonság azonban nem releváns a hálózati topológiában. A modellező feladata az, hogy megtalálja, hogy mik azok a részletek, amiket az egyes modelleknek tartalmaznia kell.

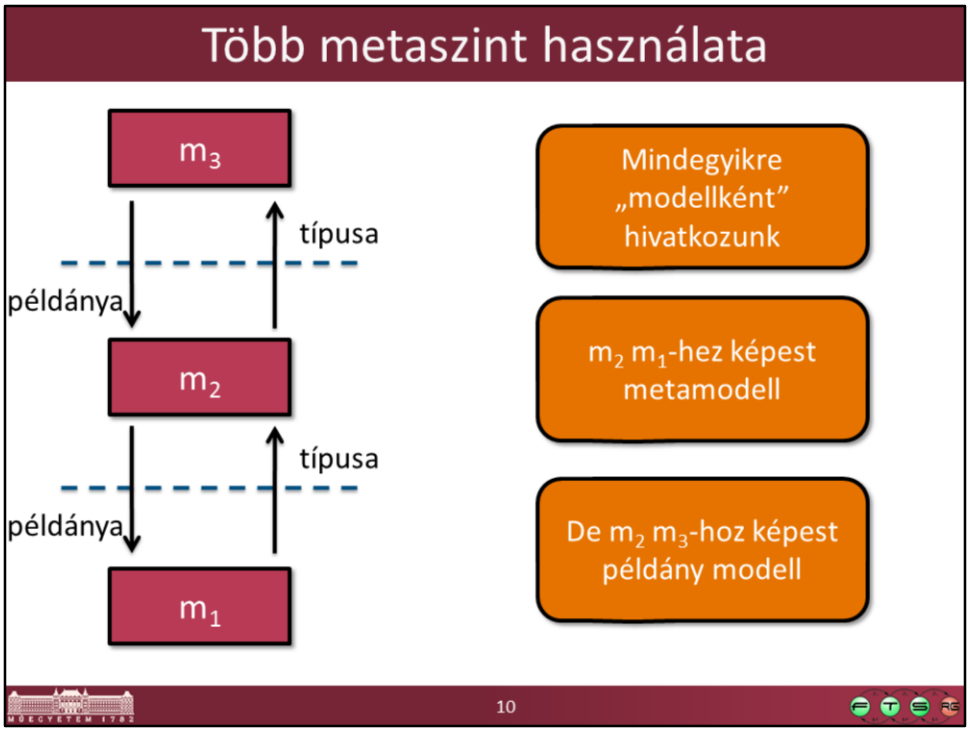


Ezt a kapcsolatot most nevezzük típusa – példánya (angol elnevezés: typeOf – instanceOf), kapcsolatnak (bár van, aki máshogy hívja).



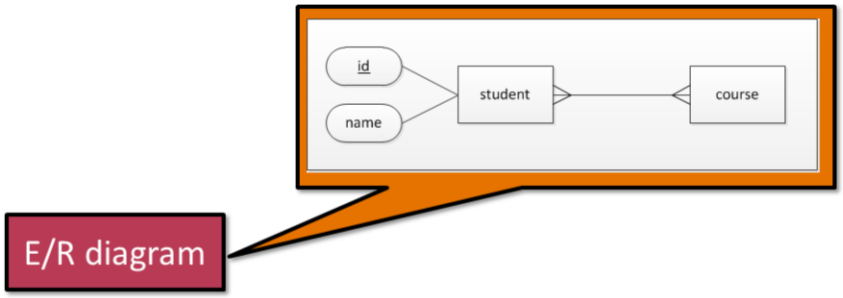
Mindkét irány állhat több, mint két szintből:
- Absztrakciós szintek: több, különböző részletességű modellt építhetünk, az absztrakciós szintek között lépkedve mindig valamilyen részletet elhanyagolunk, leegyszerűsítjük az alacsonyabb absztrakciós szinten található rendszerünket.
- Metaszintek: a metamodellnek is lehet definiálni a metamodelljét

Figyelem: az ábra csak illusztráció, nem szabványos rajzjeleket használ!

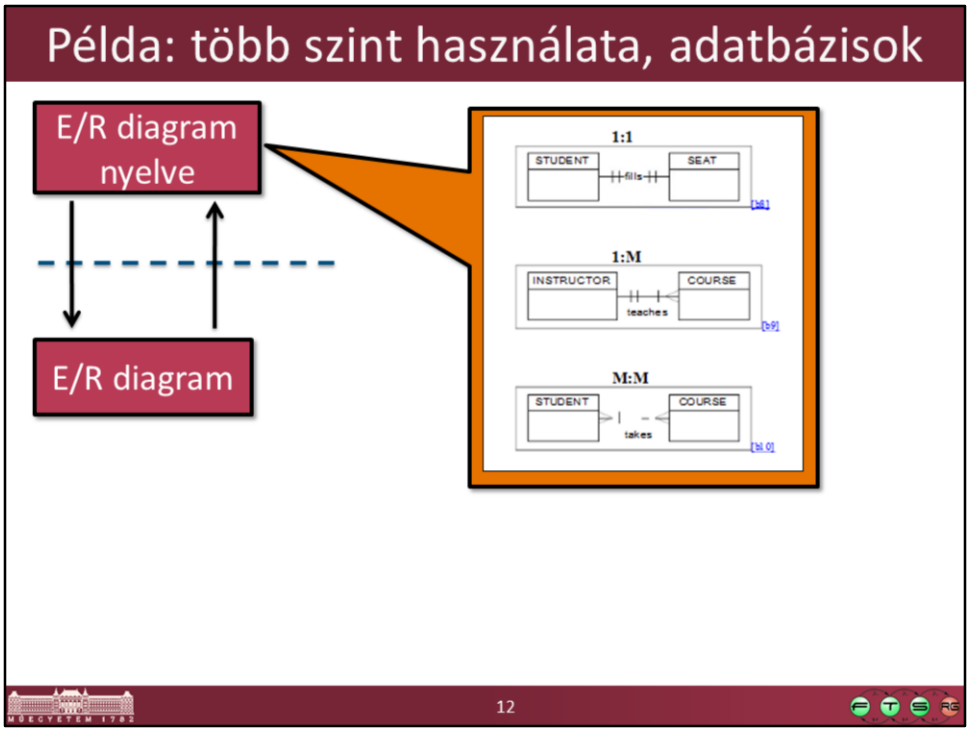


Lehet többszintű a modellezés, ilyenkor az egyik leírás, ami az alsóbb szint metamodellje, az ugyanakkor egy felsőbb szint példánya is.

Példa: több szint használata, adatbázisok

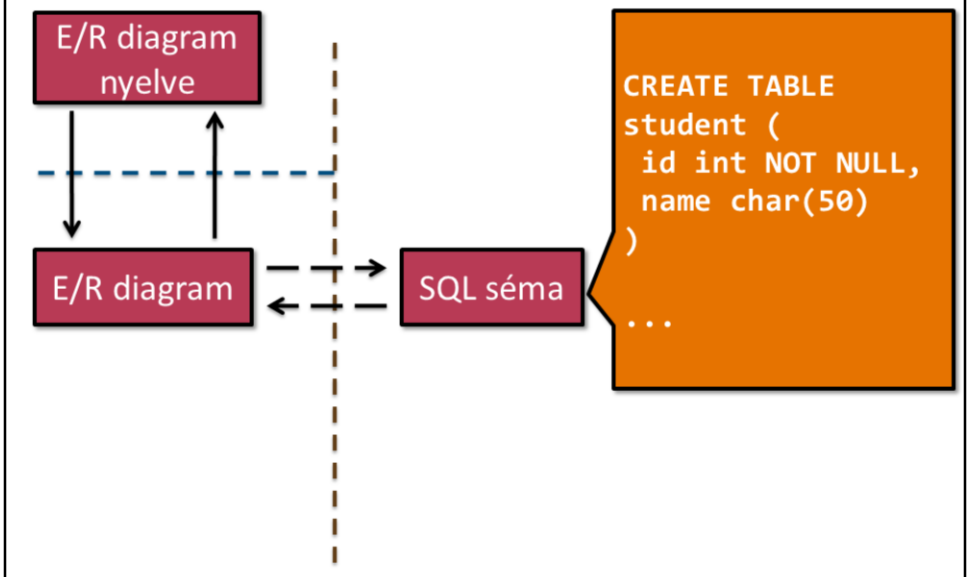


E/R diagram

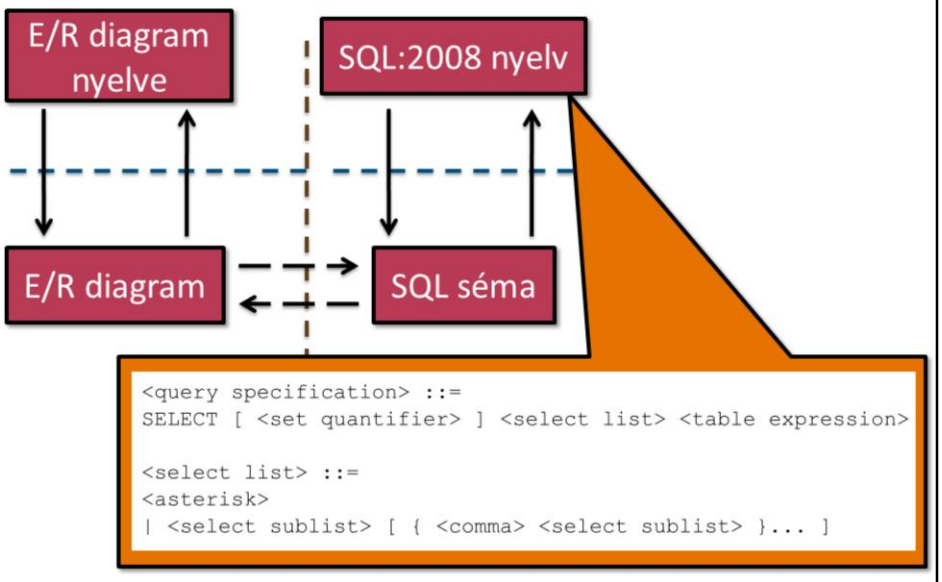


(Ez itt még nem az E/R diagram modellezési nyelv pontos definíciója, csak egy részlet a lehetséges nyelvi elemek példáiról.)

Példa: több szint használata, adatbázisok

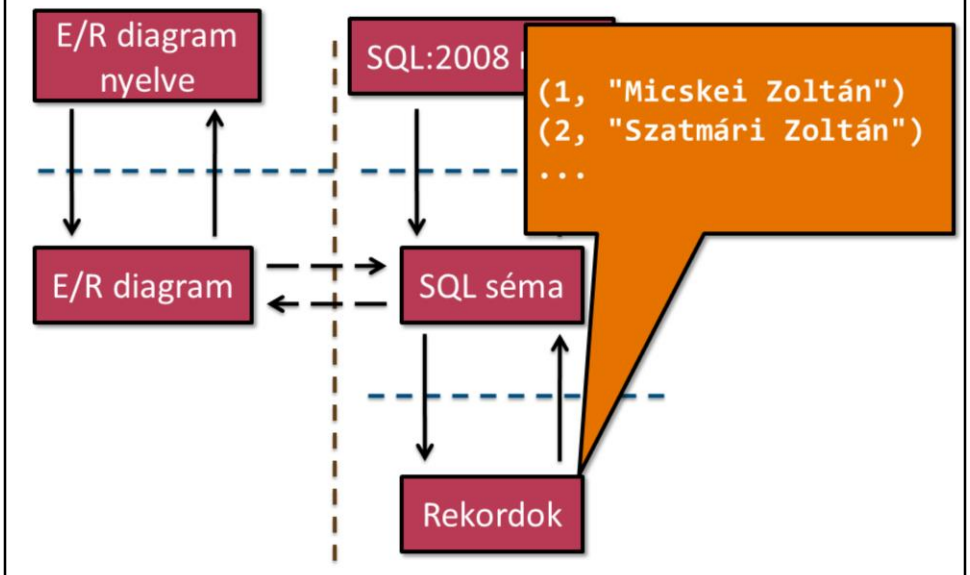


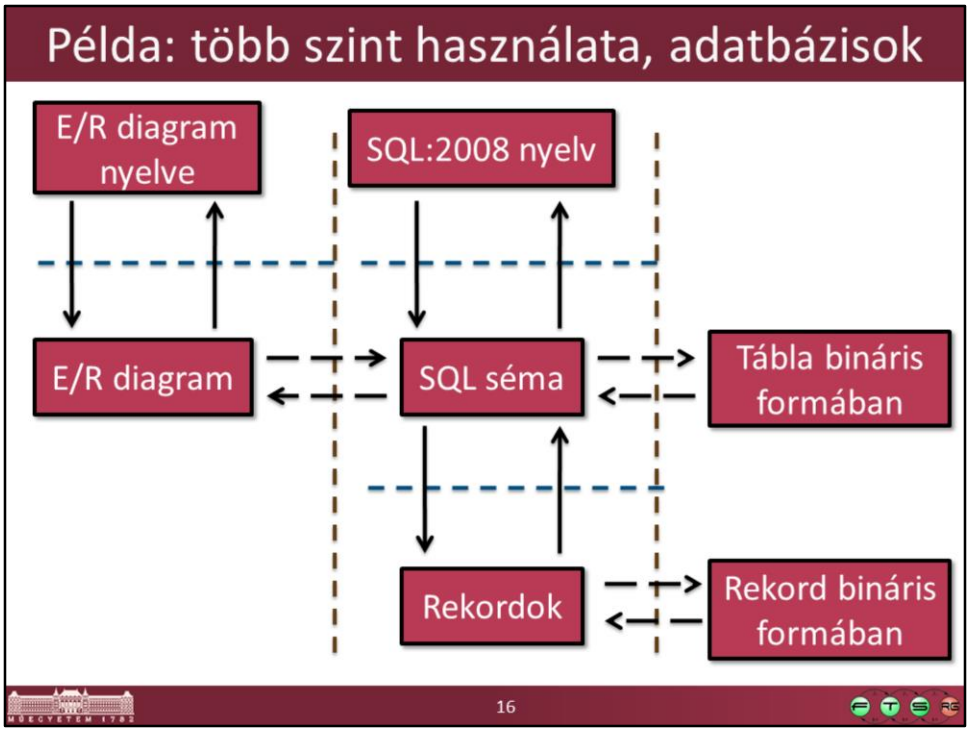
Példa: több szint használata, adatbázisok



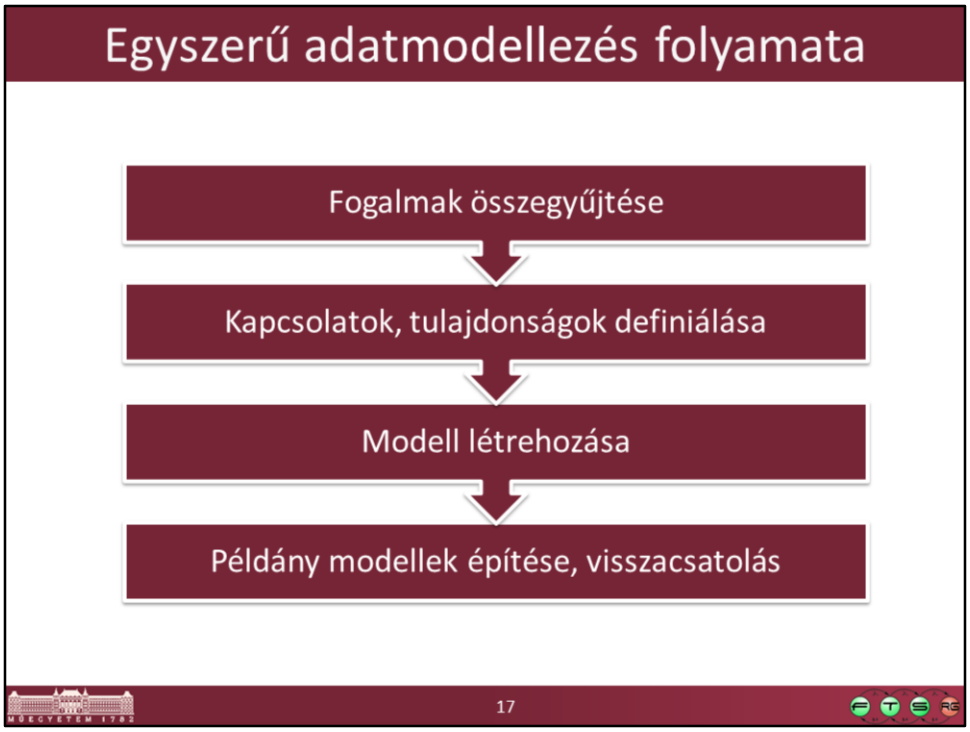

```
<query specification> ::=
SELECT [ <set quantifier> ] <select list> <table expression>

<select list> ::=
<asterisk>
| <select sublist> [ { <comma> <select sublist> }... ]
```

Példa: több szint használata, adatbázisok





- *E/R diagram*: entitások, attribútumaik és kapcsolataik
- *SQL séma*: egy CREATE TABLE … utasítás már konkrétabb ennél, ott benne vannak a konkrét adattípusok, elsődleges és idegen kulcsok explicite megjelennek, több-több kapcsolatokat kapcsolótáblával valósítjuk meg stb.
- *Fizikai tárolás*: az SQL tábla definíció pedig végül valami bináris adatszerkezetre fordul le, amit az adatbázis-kezelő a merevlemezen el tud tárolni.

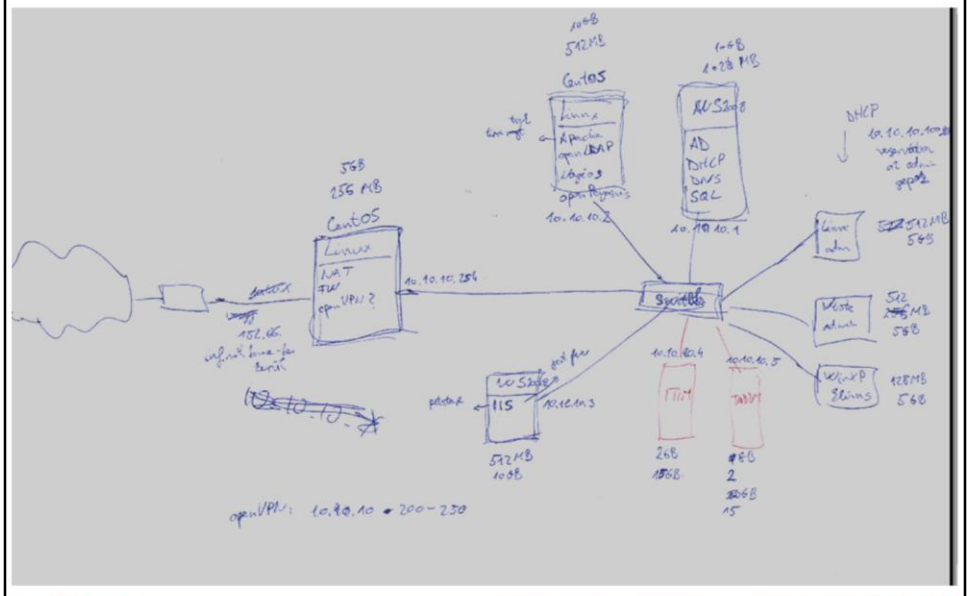


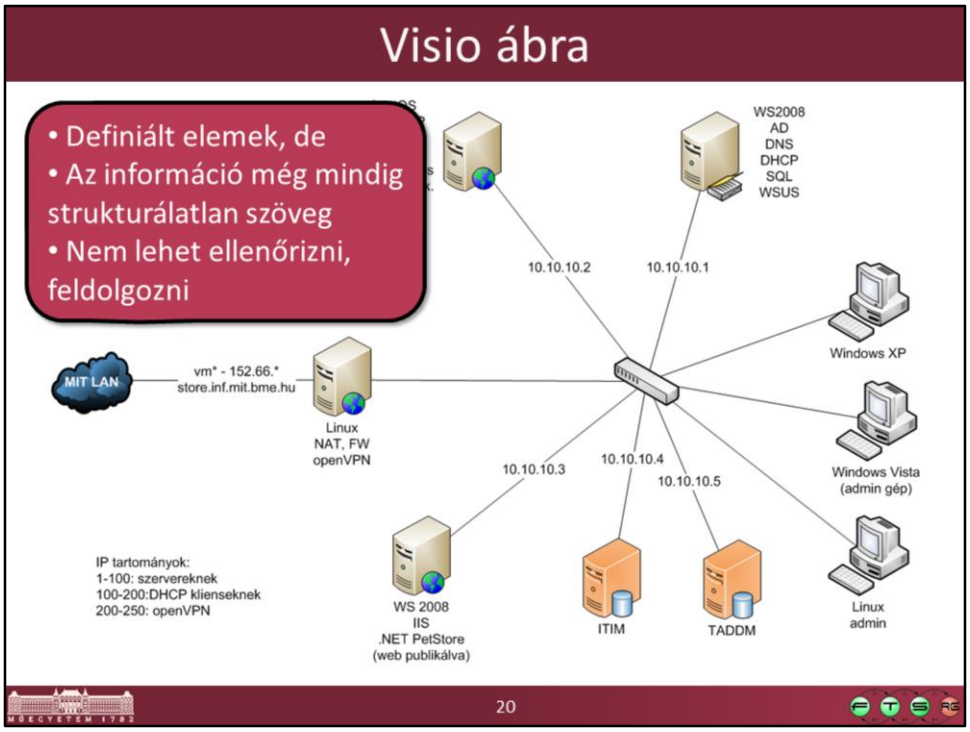
(Az ilyen modellekre szoktak még domain model vagy concept model néven is hivatkozni, mi most adatmodellnek nevezzük a tárgyban.)

Példa: IT topológia, rendszerterv

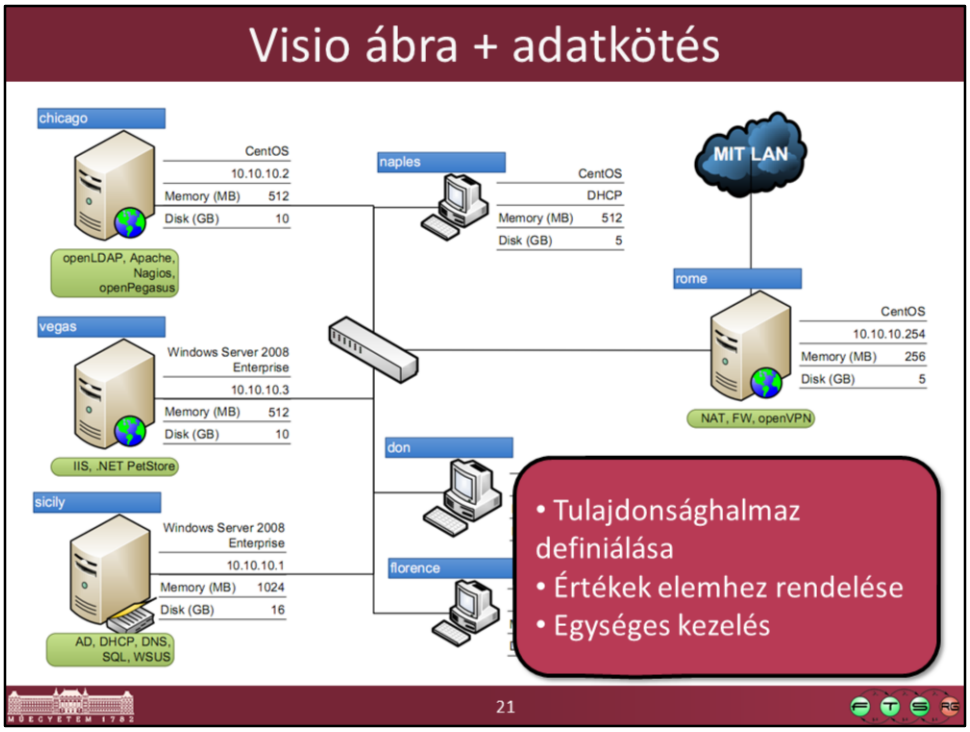
- Hogyan írjunk le egy IT rendszert?
- Fogalmak: gépek, hálózatok, alkalmazások…

Kézi rajz

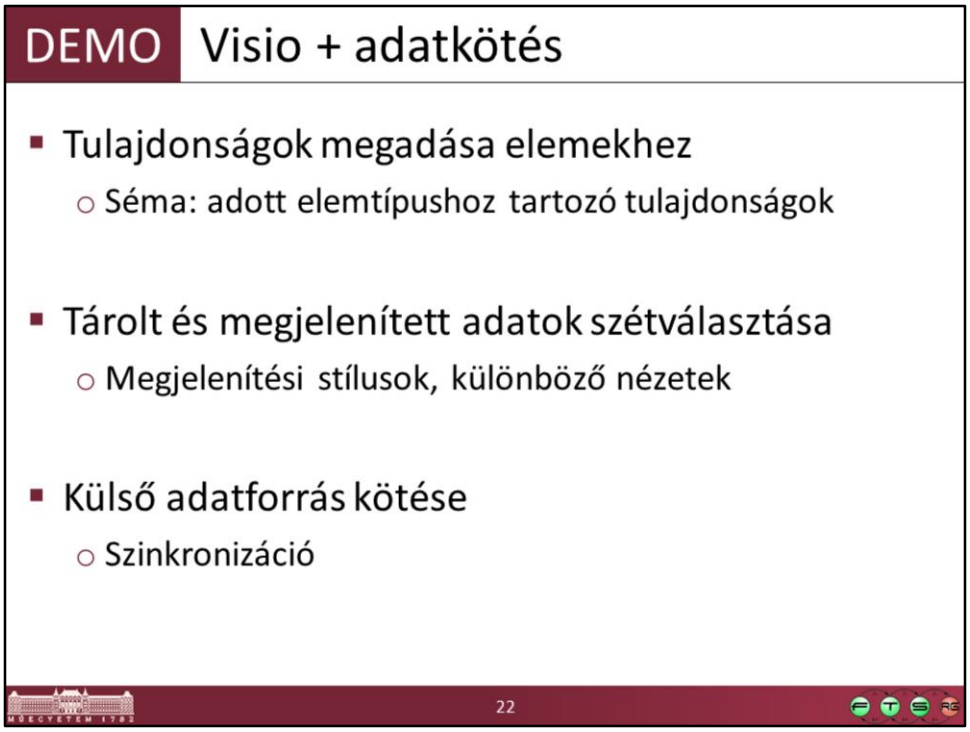

Gentoo
CentOS
AD
DHCP
DNS
SQL
DHCP
10.10.10.1
10.10.10.2
10.10.10.256
10.10.10.3
10.10.10.4
10.10.10.5
openVPN: 10.10.10.200-250



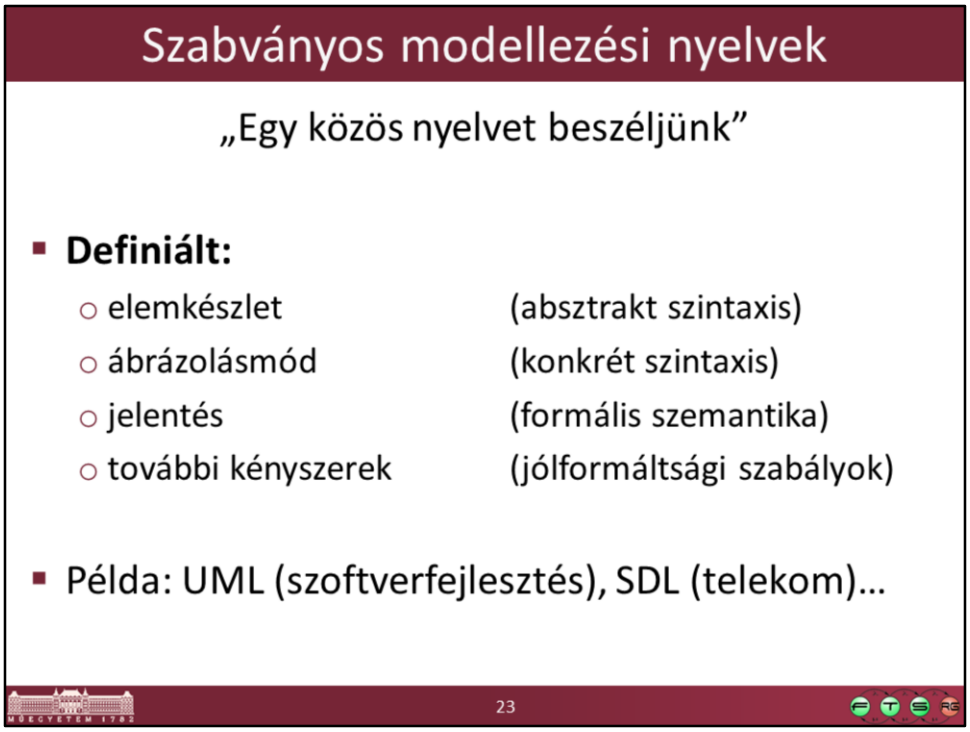
Egy modell nem feltételül vizuális, lehet csak szöveges is. A vizuális modellező nyelveknek azonban megvan az a hasznuk, hogy általában gyorsabban megérthetőek, átláthatóak.



Az egyes elemekhez tulajdonságokat adunk meg, megmondjuk azoknak mi a típusa → definiáljuk a metamodellt. Például egy számítógéphez most a név, OS, IP cím, memória, lemezméret, alkalmazások tulajdonságok tartoznak. A konkrét ábra ennek a metamodellnek egy példányát tartalmazza, ahol konkrét értékeket adunk meg. Ebből a modellből sokkal könnyebb lekérdezni információt, ellenőrizni valamit.



Visio Professional: New / Getting started / Samples / IT Asset Management
További példa:
- How to use Visio 2010 for network installations, URL: http://visio.microsoft.com/en-us/Get_Started/How_To/Pages/How-to-use-Visio-2010-for-Network-Installation.aspx

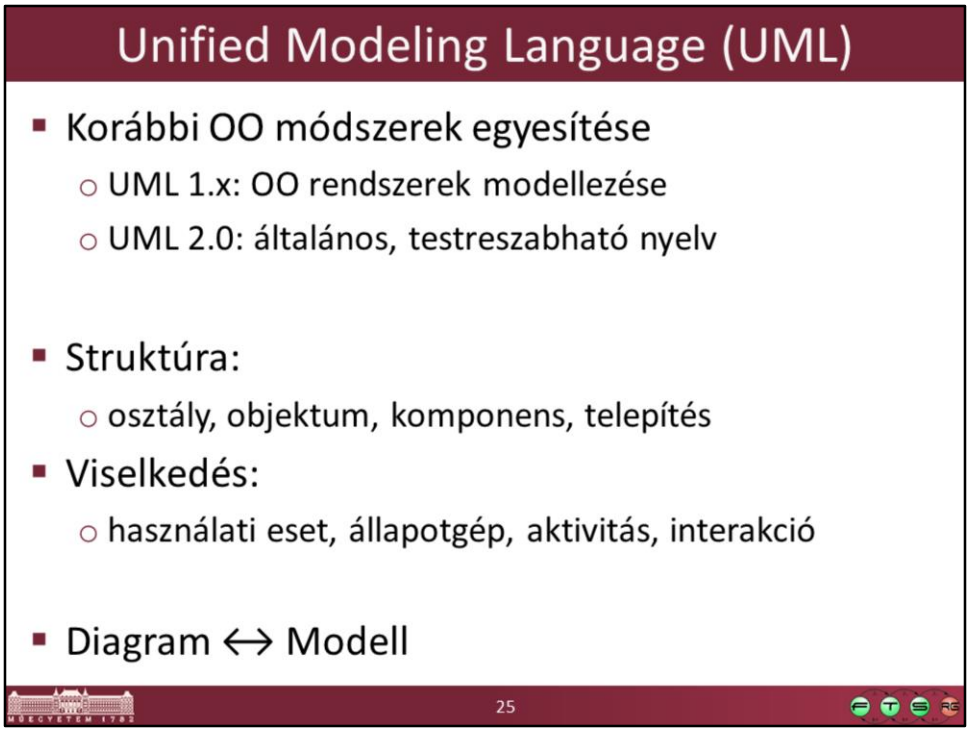


A szabványos nyelvek haszna, hogy sokkal könnyebb mással megértetni, eszközt találni hozzá, más rendszerbe átvinni az információt. Ugyanakkor nem biztos, hogy mindig az lesz a szabvány, ami a legjobb/legalkalmasabb, hanem az, amit a legtöbben használnak, amiben meg tudnak egyezni.

Modellezési nyelv esetén, ahhoz, hogy az tényleg jól definiált és használható legyen, a fenti négy aspektust meg kell adni.

UML (Unified Modeling Language)

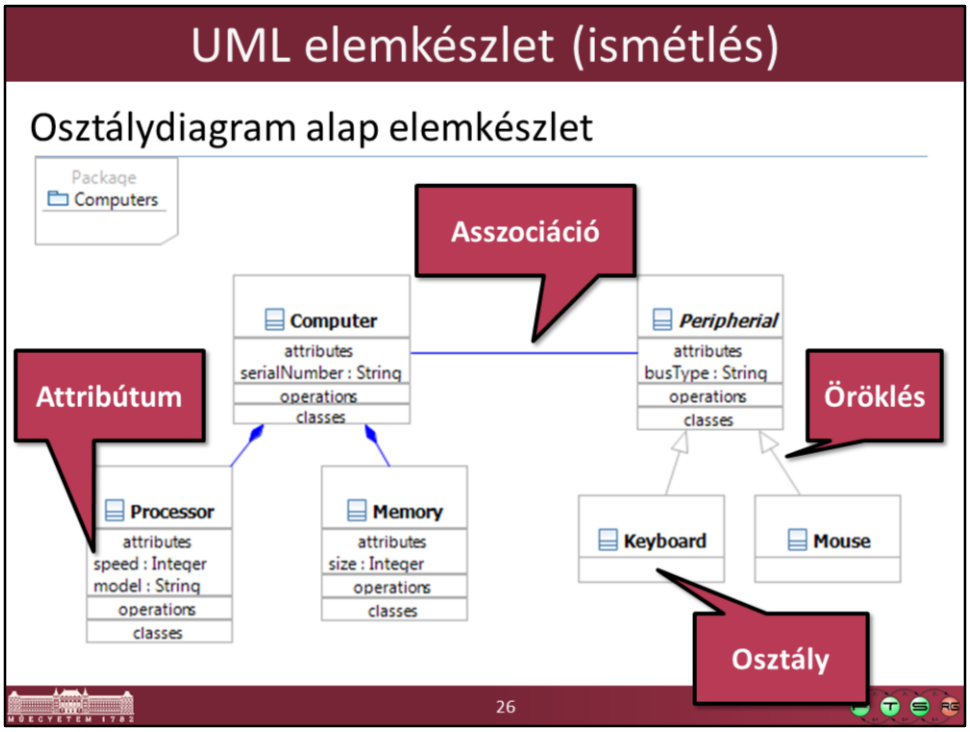
Kibocsátó:	Object Management Group
Megalkotók:	Rational, IBM, Oracle, HP, Unisys…
Verziók:	UML 1.0 – 1997, aktuális: UML 2.4.1 – 2011
Cél:	vizuális modellező nyelv



Angol elnevezések:

- Structural: class, object, component, deployment
- Behavioral: use case, state machine, activity, interaction

Diagram vs. Modell: a diagram a modellnek csak egy nézete, amikor bizonyos modell elemeket egy nézetben ábrázolunk. Egy modellhez tetszőlegesen sok diagram tartozhat, és igazából a lényegi információ a modellben van, és nem a diagramban. Mégis sokszor az egyszerűség kedvéért, ha ez nem félreérthető, a diagrammal hivatkozunk a modellre, tehát pl. az UML osztálydiagram elemei kifejezést használjuk az UML osztálydiagramon ábrázolt modell elemei kifejezés helyett.

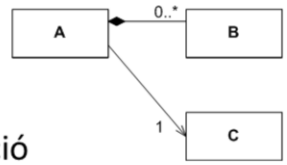


Angol elnevezések:

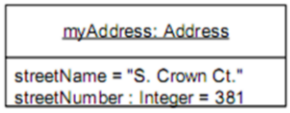
- Osztály – Class
- Öröklés – Generalization
- Attribútum – Attribute
- Asszociáció – Association

UML elemkészlet (ismétlés)

- **Asszociáció**
  - Navigálhatóság
  - Multiplicitás
  - Tartalmazás: Kompozíció / Aggregáció



- **Példány**
  - InstanceSpecification
  - Slot



- **Interfész**
  - Szerződés (elvárt működés)
  - Javaslat: metódusokat adjon meg
- **Absztrakt osztály**: nem példányosítható

- Asszociáció: valamilyen kapcsolat van a két típus között
- Navigáció: ha csak az egyik irányba navigálható, pl. A-C az ábrán, akkor az azt jelenti, hogy a C példányából a hozzá tartozó A példánya nem érhető el
- Kompozíció: az aggregáció erősebb formája, amikor csak egy kompozícióban vehet rész egy példány egyszerre

UML elemkészlet (ismétlés)

- Jelölések összefoglalása (a specifikációból):

Node Type	Notation
Class	ClassName
Interface	InterfaceName <<interface>> InterfaceName
InstanceSpecification	Instancename : ClassName
Package	PackageName

Path Type	Notation
Aggregation	
Association	
Composition	
Dependency	
Generalization	
InterfaceRealization	
Realization	

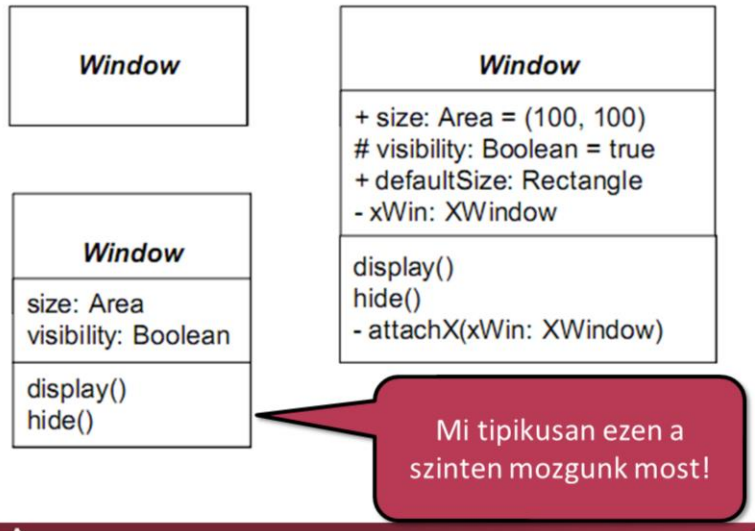
UML specifikáció: http://www.omg.org/spec/UML/2.4.1/

UML elemkészlet (ismétlés)

- Az eddigiek csak egy apró szelete az UML-nek
- A tárgyban főleg adatmodellezéssel foglalkozunk
  - Viselkedés leírása kevésbé hangsúlyos most
- Az előbbi elemkészlet jobbára elég lesz

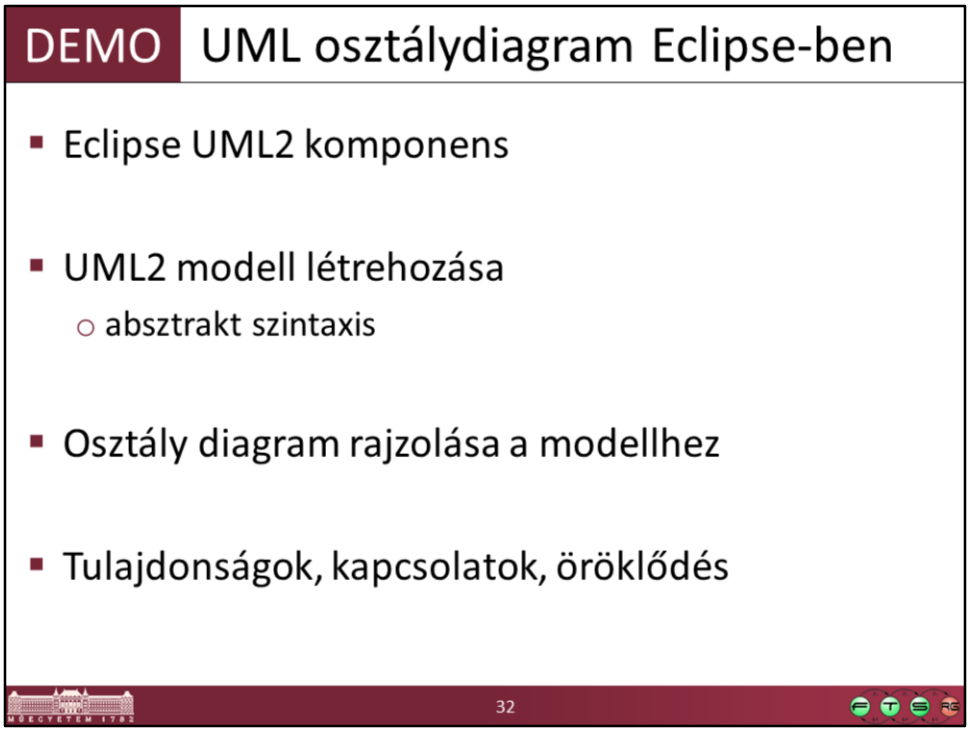
Részletek megjelenítése

Attól függően, mire van szükség, többféle nézet:



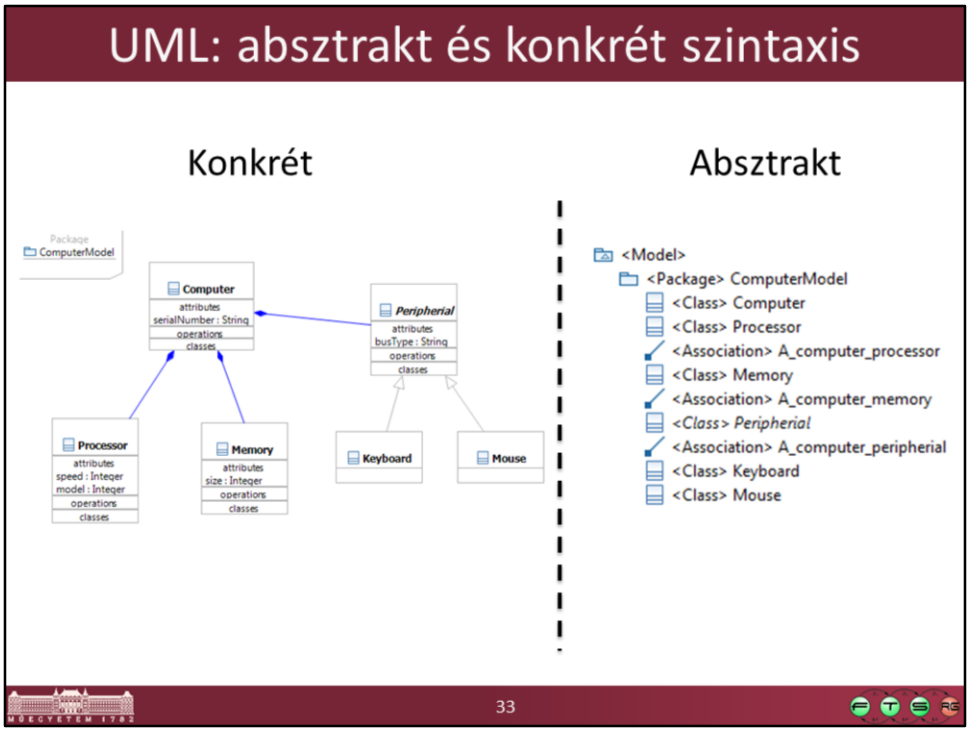
Tipikus hibák adatmodellek esetén

- Elnevezési koncepciók használata:
  - PascalCase, camelCase; objektum név inkább kis kezdőbetű, ékezet ne legyen benne
- Asszociációhoz nem kell attribútumokat felvenni, ez egy implementációs részlet
- Különböző példányoknak ne legyen ugyanaz a neve
- Példány szinten nem kell jelölni a kompozíciót
- Interfészben ne legyen attribútum

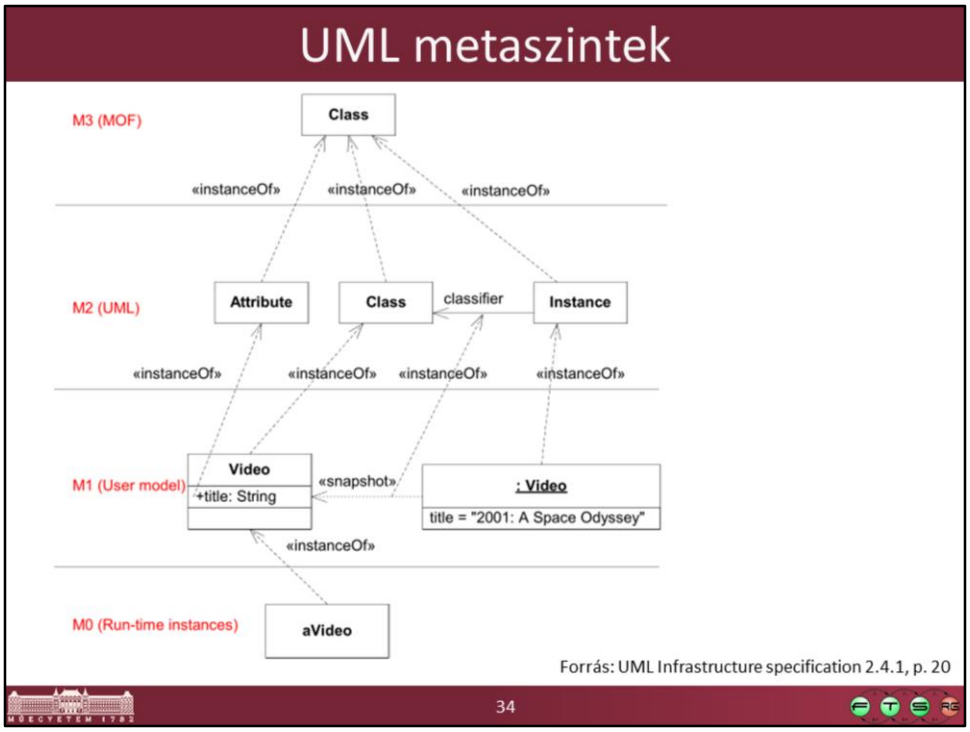


Screencast: http://static.inf.mit.bme.hu/edu/irf/

UML modellezés Eclipse-ben (nem a legkönnyebben használható eszköz, de jó látszik benne a modell absztrakt szintaxisa is):
- JDK telepítése: http://java.sun.com
- Eclipse letöltése: http://www.eclipse.org/downloads/, Eclipse IDE for Java Developers
- Eclipse elindítása, UML2 csomag telepítése
    -Help / Install new software.. / --All Available Sites-- / Keresés: UML2 → Install… (ez letölt még egy csomó egyéb szükséges komponenst is)
- File / New project / General / Project
- Window / Open Perspective / Other… / Resource
- Windows / Show View / Properties
- File / New / Other / Example EMF Model Creation Wizard / UML Model
- Modell elemek hozzáadása az absztrakt szintaxisnak megfelelően
    - UML Editor / New Child / Nested Package / Package
    - UML Editor / New Child / Owned Type / Class
    - Properties nézetben lehet elnevezni, tulajdonságait megnézni

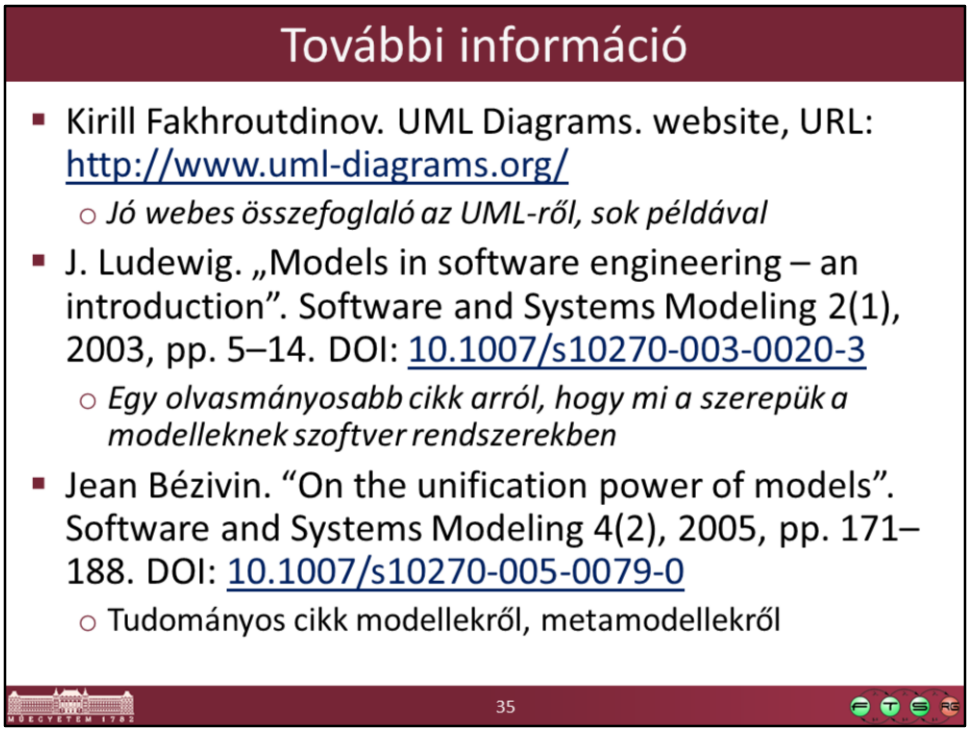


UML esetén a konkrét szintaxis a dobozok és a közöttük lévő kapcsolatokat ábrázoló vonalak. A kapcsolatok típusát különböző grafikus jelekkel azonosítjuk.



Az UML metamodell adja meg pl., hogy egy osztálydiagramon milyen elemeket használhatunk, mit jelent pontosan az öröklés, stb. Az itt feltüntetett metamodell csak egy kis része a teljes metamodellnek.

Megjegyzés: *„The instances, which are sometimes referred to as "run-time" instances, that are created at M0 from for example Person should not be confused with instances of the metaclass InstanceSpecification that are also defined as part of the UML metamodel. An instance of an InstanceSpecification is defined in a model at the same level as the model elements that it illustrates, as is depicted in Figure 7.7, where the instance specification Mike is an illustration (or a snapshot) of an instance of class Person. "* (UML Infrastructure 2.4.1)



A cikkek elérhetőek a BME belső hálózatából.

URL-ek:
- http://www.uml-diagrams.org/
- http://dx.doi.org/10.1007/s10270-003-0020-3
- http://dx.doi.org/10.1007/s10270-005-0079-0

Összefoglalás

- Modellezés, modellezés, modellezés
- Megéri először modellezni
- Adatmodellezés, metamodellezés szerepe