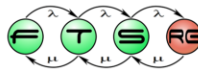


IT infrastruktúra modellezése

Micskei Zoltán

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



Utolsó módosítás: 2011. 02. 15.

Bevezető

- Modellezés: központi fogalom az informatikában
- Modell:
 - „a valóság egy részletének egyszerűsített képe”
- Cél: komplexitás kezelése

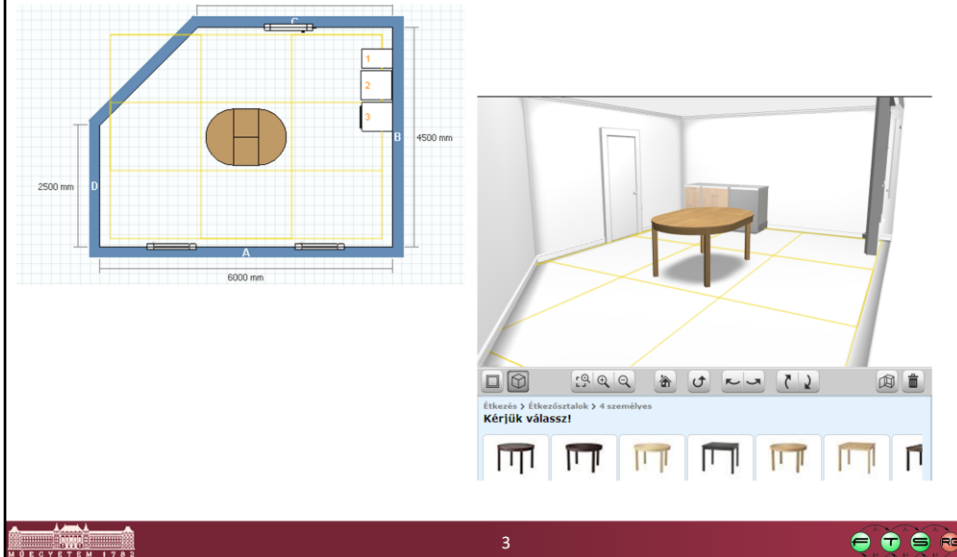


Bonyolult rendszerekkel csak úgy tudunk dolgozni, hogy először egy egyszerűbb modellt építünk, megvizsgáljuk a rendszert különböző szempontokból.

A modellezés nagyon általános fogalom, majd mindenki használja, és természetesen teljesen eltérő módokon, úgyhogy nehéz egyértelmű szóhasználatot találni.

Modellezés a gyakorlati életben?

Pl.: [svéd cég] webes környatervezője



Ez is egy ugyanolyan 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 használt modellezési nyelvek

Digitális technika

- automata

Algoritmus

- folyamatábra, pseudo kód

Adatbázis

- E/R diagram

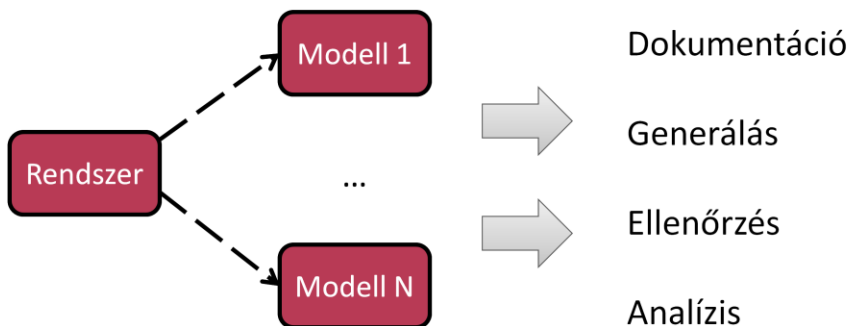
OO program

- UML diagram

Minden problémához a neki megfelelő nyelv és módszer kiválasztása!

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

Modellek lehetséges felhasználása

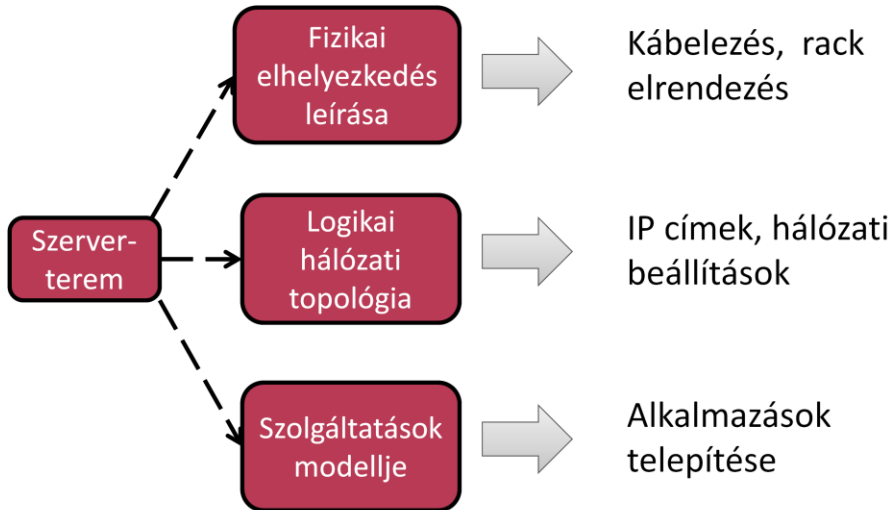


Nagyon sokféle cél miatt építhetünk modelleket, ha van egy modellünk, akkor azt felhasználhatjuk:

- dokumentáció készítése: pl. UML modellből diagramok generálása
- generálás: pl. működést leíró modellből forráskód készítése
- analízis: pl. rendszer felépítését leíró modellből teljesítményjellemzők számolása, mennyi kérést tudunk kiszolgálni, elég lesz-e egy adott méretű processzor
- ellenőrzés: 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 melyik részek párhuzamosíthatóak.

Példa: modellek felhasználása



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.

Modellezési nyelv

- Milyen elemeket használhatunk a modellben?
→ **metamodel** (modellezési nyelv modellje)

Típusa – példánya kapcsolat

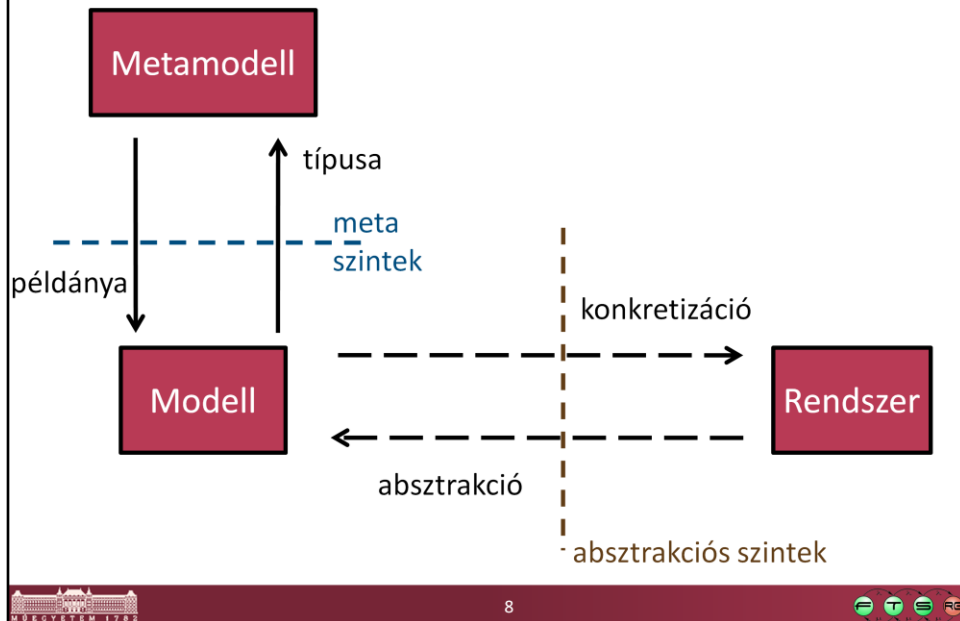
- Sablon definiálása
- Kényszerek, összefüggések



Az egyes metaszintek között típusa – példánya kapcsolatok vannak. Egy metamodel mindig egy sablont ad meg, hogy milyen fogalmi elemekből épül fel az alatta lévő szinten lévő modell, azok között milyen kapcsolatok és kényszerek vannak.

Angol elnevezés: `typeOf` - `instanceOf`

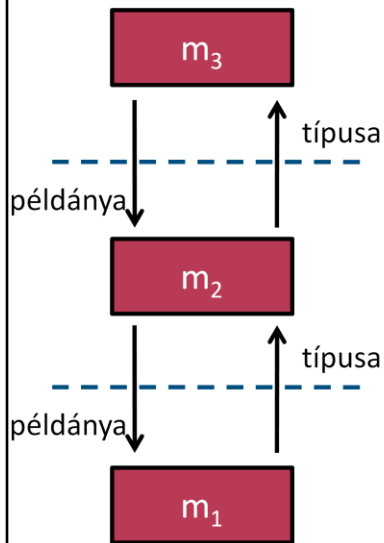
Kapcsolatok az egyes szintek között



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 metamodelnek is lehet definiálni a metamodeljét

Több metaszint használata



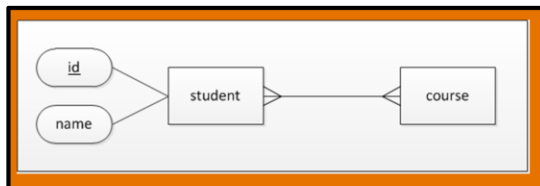
Mindegyikre „modellként” hivatkozunk

m_2 m_1 -hez képest metamodell

De m_2 m_3 -hoz képest példány modell

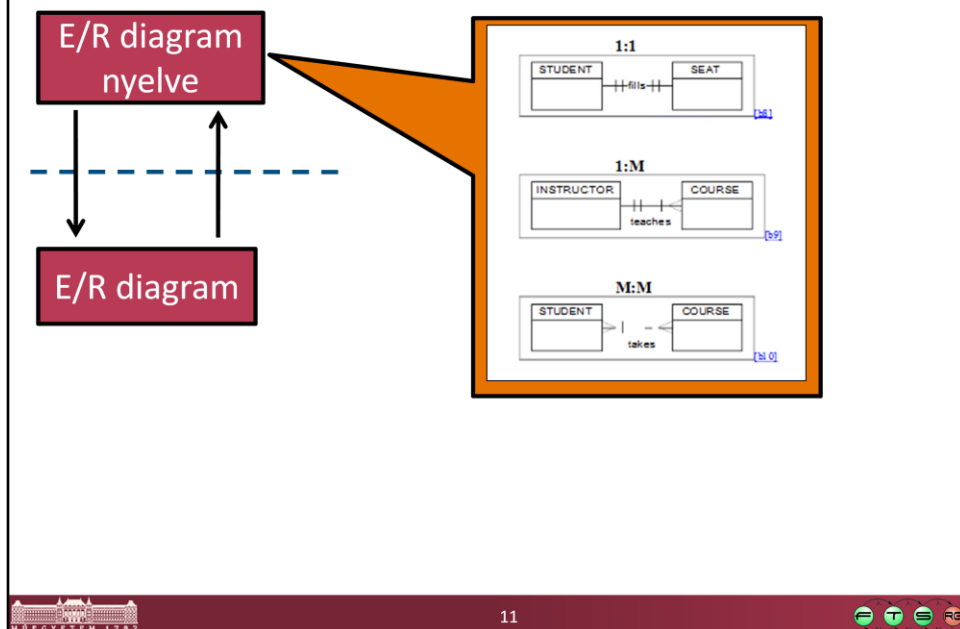
Lehet többszintű a modellezés, ilyenkor az egyik leírás, ami az alsóbb szint metamodellje, az ugyanakkor egy felsőbb szint példánya is.

Példa: több szint használata, adatbázisok



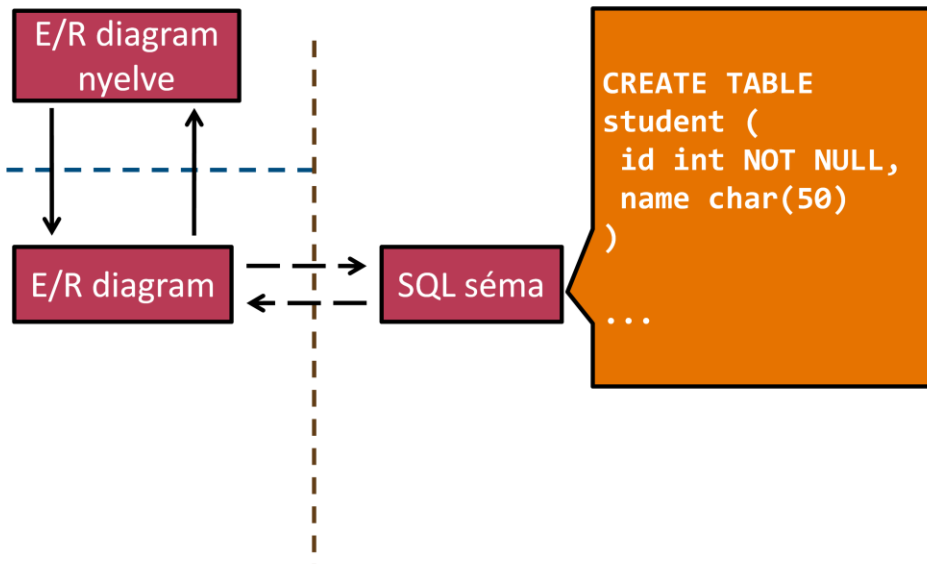
E/R diagram

Példa: több szint használata, adatbázisok

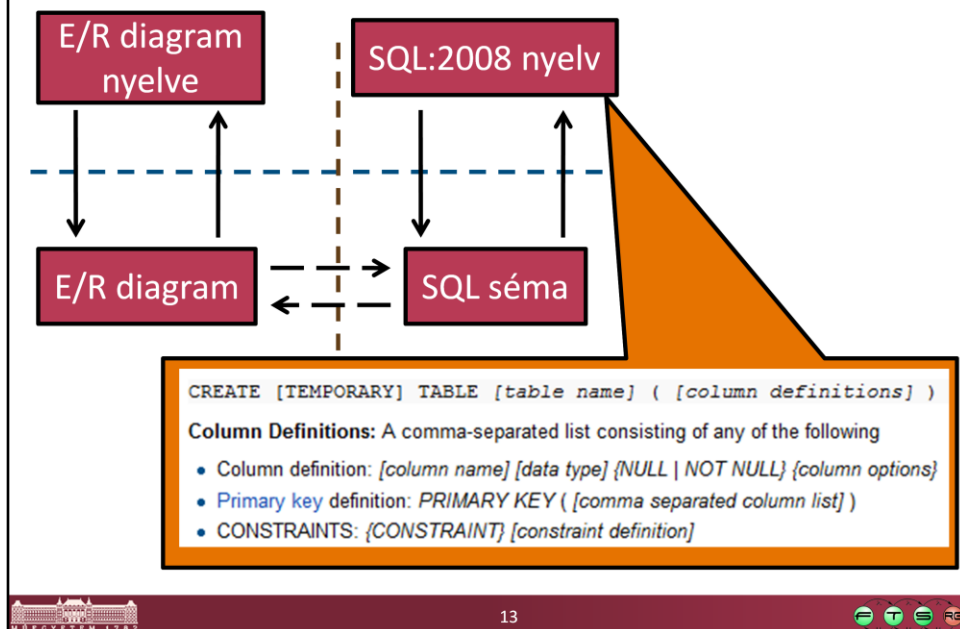


(Ez itt még nem az E/R diagram modellezési nyelv pontos definíciója, csak egy részlet a lehetséges nyelvi elemek példáiról.)

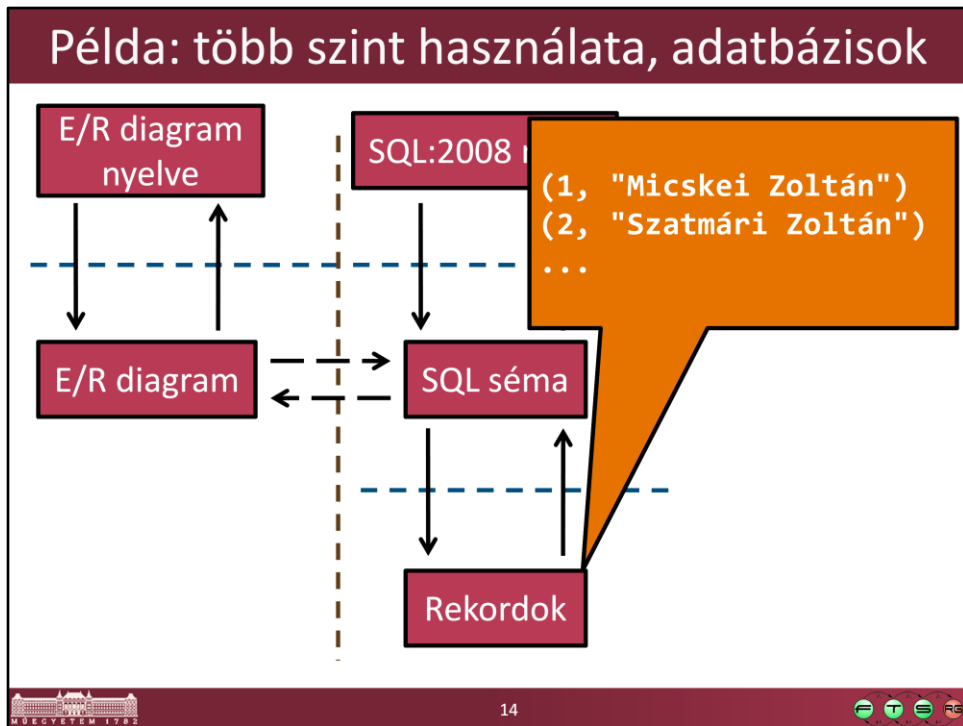
Példa: több szint használata, adatbázisok



Példa: több szint használata, adatbázisok

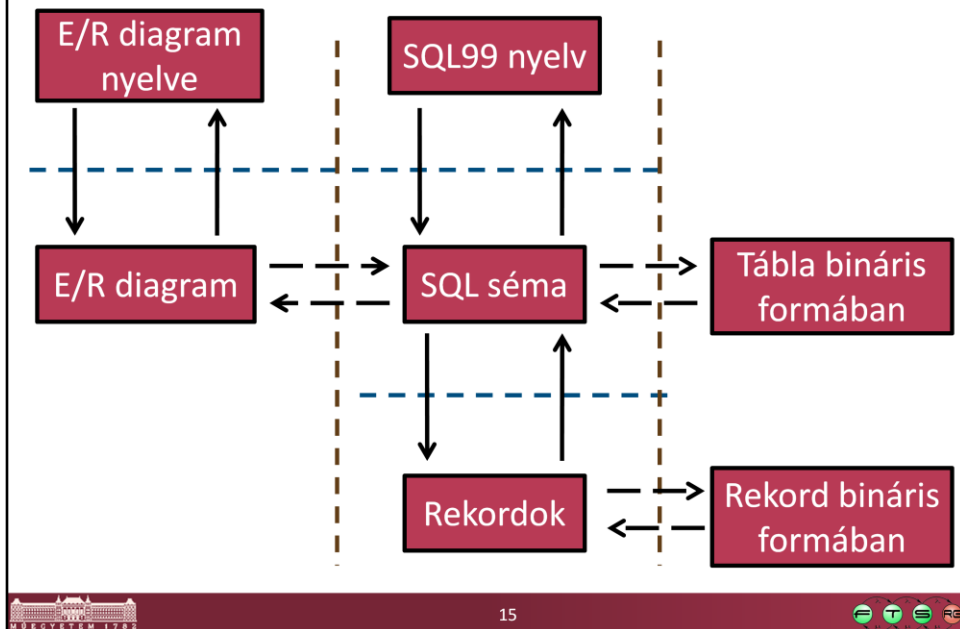


Kép forrása: http://en.wikipedia.org/wiki/Create_%28SQL%29#Create_statements



Kép forrása: http://en.wikipedia.org/wiki/Create_%28SQL%29#Create_statements

Példa: több szint használata, adatbázisok

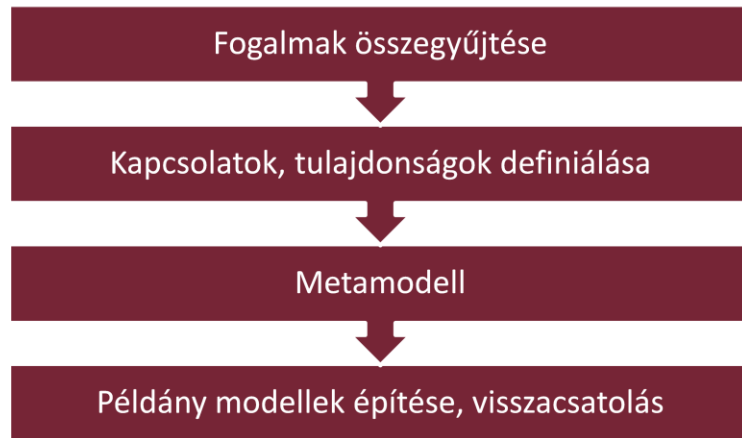


E/R diagram: entitások, attribútumaik és kapcsolataik

SQL séma: egy CREATE TABLE ... utasítás már konkrétabb ennél, ott benne vannak a konkrét adattípusok, elsődleges és idegen kulcsok expliciten 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.

Egyszerű adatmodellelés folyamata

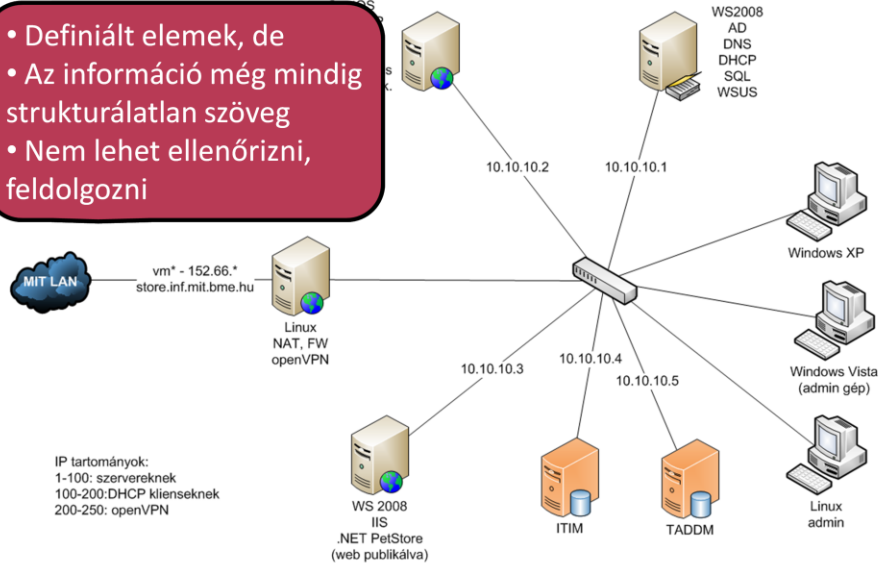


Példa: IT topológia, rendszerterv

- Hogyan írjunk le egy IT infrastruktúrát?
- Fogalmak: gépek, eszközök, hálózati kapcsolatok...

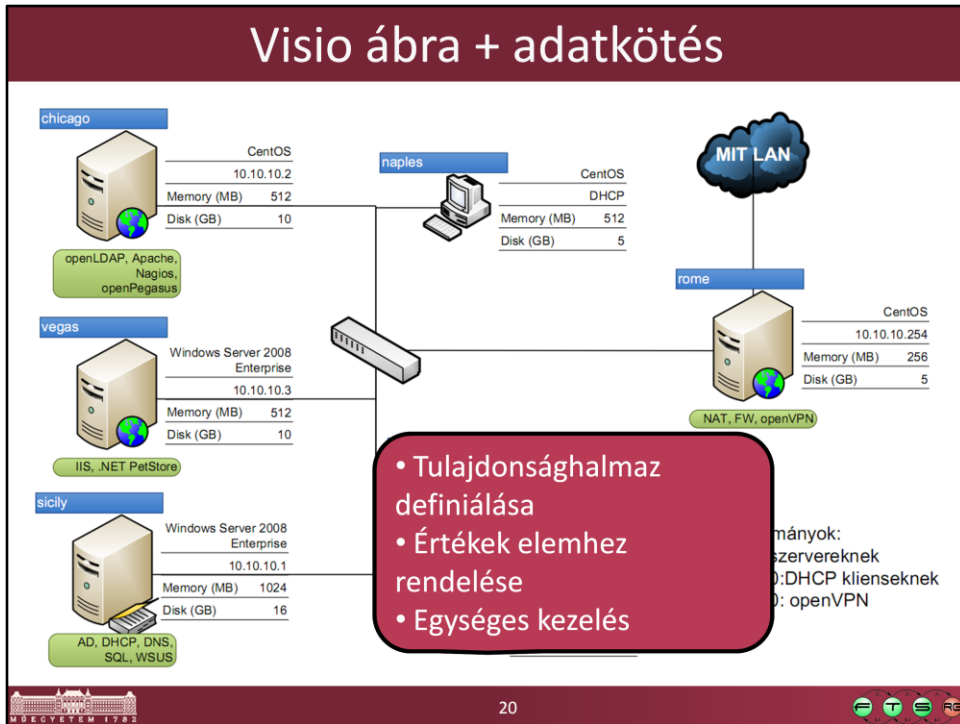
Visio ábra

- Definiált elemek, de
- Az információ még mindig strukturálatlan szöveg
- Nem lehet ellenőrizni, feldolgozni



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

Visio ábra + adatkötés



Az egyes elemekhez tulajdonságokat adunk meg, megmondjuk azoknak mi a típusa → definiáljuk a metamodellt. Pl. egy számítógéphez most a név, OS, IP cím, memória, lemez méret, 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.

DEMO Visio + adatkötés

- Tulajdonságok megadása elemekhez
 - Séma: adott elemtípushoz tartozó tulajdonságok

- Tárolt és megjelenített adatok szétválasztása
 - Megjelenítési stílusok, különböző nézetek

- Külső adatforrás kötése
 - Szinkronizáció



Visio Professional: New / Getting started / Samples / IT Asset Management
További példa: <http://visiotoolbox.com/en-US/trainings.aspx?trainingid=1&aid=132>

UML (Unified Modeling Language)

Kibocsájtó:	Object Management Group
Megalkotók:	Rational, IBM, Oracle, HP, Unisys...
Verziók:	UML 1.0 – 1997, aktuális: UML 2.3 – 2010
Cél:	vizuális modellező nyelv

Szabványos modellezési nyelvek

- „Egy közös nyelvet beszéljünk”
- **Definiált:**
 - elemkészlet (absztrakt szintaxis)
 - ábrázolásmód (konkrét szintaxis)
 - jelentés (formális szemantika)
 - további kényszerek (jólformáltsági szabályok)
- Példa: UML (szoftverfejlesztés), SDL (telekom)...



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.

Unified Modeling Language (UML)

- Korábbi OO módszerek egyesítése
 - UML 1.x: OO rendszerek modellezése
 - UML 2.0: általános, testreszabható nyelv
- Struktúra:
 - osztály, objektum, komponens, telepítés
- Viselkedés:
 - használati eset, állapotgép, aktivitás, interakció
- Diagram ↔ Modell



24



Angol elnevezések:

Structural: class, object, component, deployment

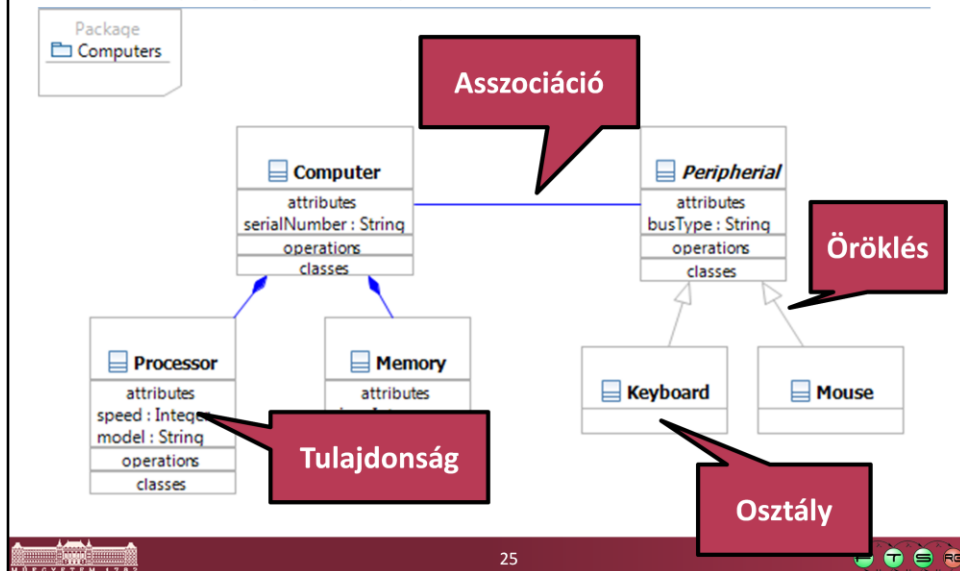
Behavioral: use case, state machine, activity, interaction

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.

UML elemkészlet (ismétlés)

▪ Osztálydiagram alap elemkészlet



Angol elnevezések:

Osztály – Class

Öröklés – Generalization

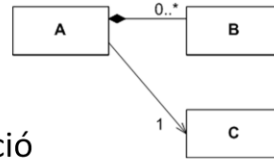
Tulajdonság – Property

Asszociáció - Association

UML elemkészlet (ismétlés)

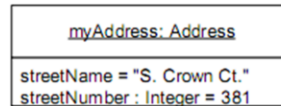
▪ Asszociáció

- Navigálhatóság
- Multiplicitás
- Tartalmazás: Kompozíció / Aggregáció



▪ Példány

- InstanceSpecification
- Slot



▪ Interfész

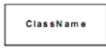

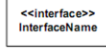
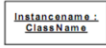

- Szerződés (elvárt működés)
- Javaslat: metódusokat adjon meg








▪ 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 példány 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	
Interface	 
InstanceSpecification	
Package	

PATH TYPE	NOTATION
Aggregation	
Association	
Composition	
Dependency	
Generalization	
InterfaceRealization	
Realization	



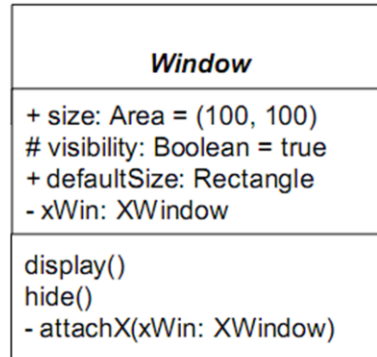
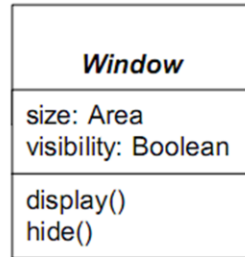
UML specifikáció: <http://www.omg.org/spec/UML/2.2/Superstructure/PDF/>

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



Mi tipikusan ezen a szinten mozgunk most!

Tipikus hibák adatmodellek esetén

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

DEMO UML osztálydiagram Eclipse-ben

- Eclipse UML2 Tools
- UML2 modell létrehozása
 - absztrakt szintaxis
- Osztály diagram rajzolása a modellhez
- Tulajdonságok, kapcsolatok, öröklődés



32

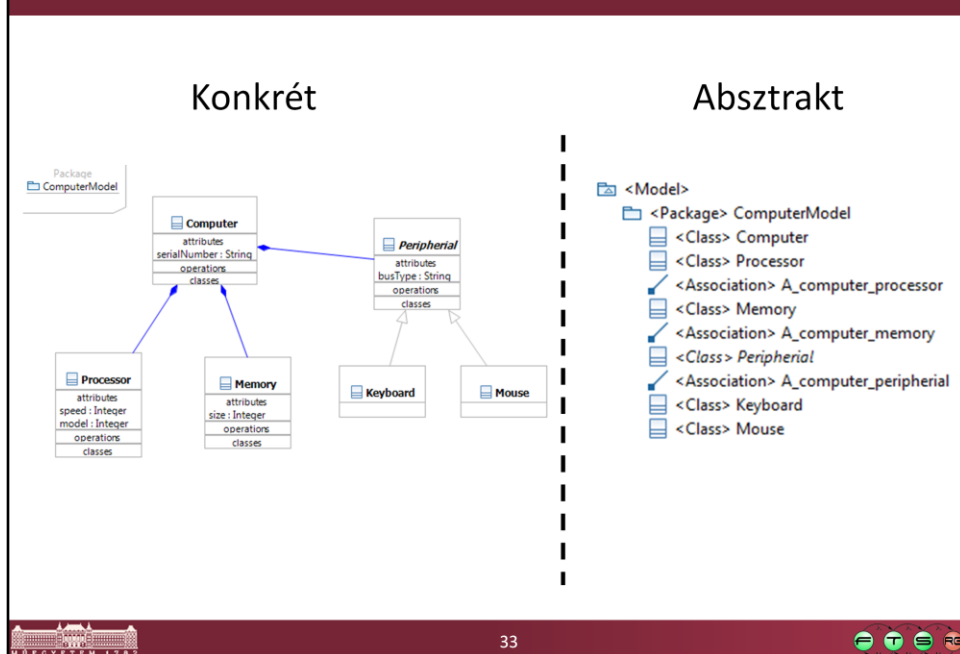


Screencast: <http://www.inf.mit.bme.hu/edu/irf/files/03-IRF-2009-DEMO-eclipse-uml2-tools.avi>

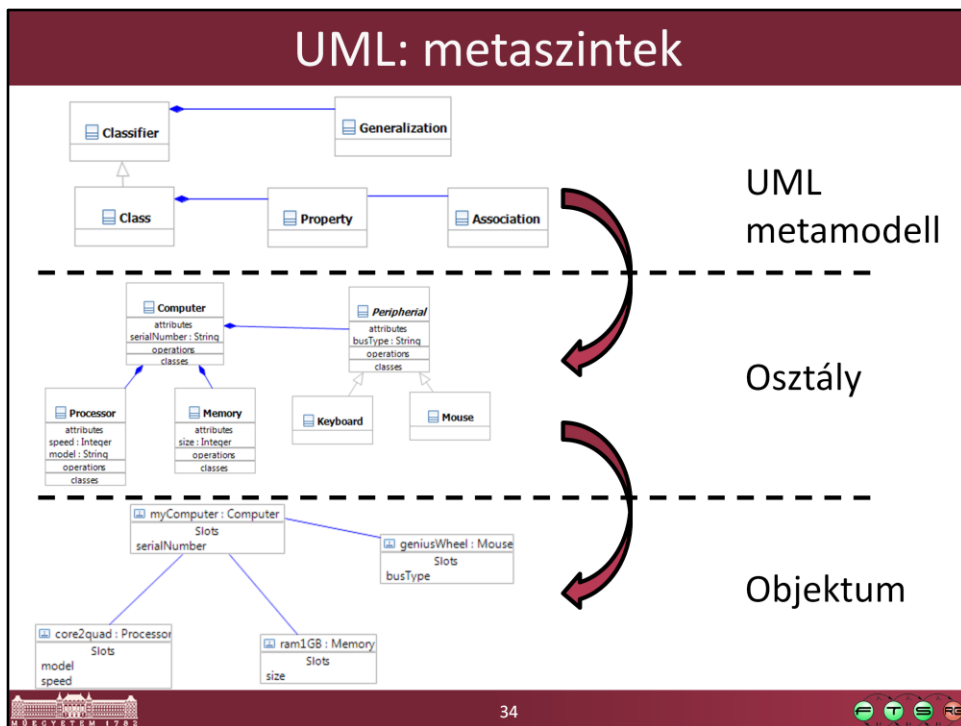
UML modellezés Eclipse-ben (nem a legkönnyebben használható eszköz, de jó látszik benne a modell absztrakt szintaxisa is):

- JDK telepítése: <http://java.sun.com>
- Eclipse letöltése: <http://www.eclipse.org/downloads/>, [Eclipse IDE for Java Developers](#)
- Eclipse elindítása, UML2 Tools csomag telepítése
 - Help / Install new software.. / --All Available Sites-- / Keresés: UML2 Tools, UML2 Tools SDK → 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
- Grafikus szerkesztés:
 - uml fájl jobb gomb, Inicialize Class diagram

UML: absztrakt és konkrét szintaxis



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



Az UML metamodel adja meg pl., hogy egy osztálydiagramon milyen elemeket használhatunk, mit jelent pontosan az öröklés, stb. Az itt feltüntetett metamodel csak egy kis része a teljes metamodelnek, az UML esetén ezt próbálták olyan részletesen kidolgozni, hogy az alapján könnyen lehessen más, specializált modellezési nyelveket kidolgozni.

(Megjegyzés: Az UML szabvány az osztálydiagramok szintjére viszont modell szintként hivatkozik, az objektumok szintjére pedig példányként. Ez a mi személyes véleményünk szerint azért kicsit zavaró, mert az objektumdiagramok szintje az, ami ténylegesen modellezi a valóságot, az osztálydiagramok pedig ennek metamodellje. Az osztálydiagramok a valós rendszer fogalmainak a modellje, és nem magának a valós rendszernek.)

XML (Extensible Markup Language)

Kibocsájtó:	Word Wide Web Consortium (W3C)
Megalkotók:	Sun, Netscape, Microsoft...
Verziók:	XML 1.0 – 1998, aktuális: XML 1.1 – 2006
Cél:	struktúrált adatok leírása

XML (ismétlés)

- Szabványos adatcsere nyelv
- Jólformált XML (well-formed)

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Felhasználók listája -->
<users>
  <user login="mizo">Micskei Zoltán</user>
  <user login="xmi">Tóth Dániel</user>
</users>
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- Felhasználók listája -->
<users>
  <user login="mizo">Micskei Zoltán</user>
  <user login="xmi">Tóth Dániel</user>
</users>
```

XML Séma (ismétlés)

- Helyes XML (valid) sémának megfelel
- Séma nélküli XML-ben Tetszőlegesen sokszor szerepelhet ez az elem XML!

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="users">
    <xs:complexType>
      <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element name="user">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:string">
                <xs:attribute name="login" type="xs:string"
                  use="required" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Attribútum használata kötelező

Egy XML dokumentumot akkor lehet igazán használni, ha sémát is adunk hozzá. Pusztán egy jólformált dokumentummal nem tudunk még sok mindent kezdeni: meg kell-e máskor is adni minden elemet, ami szerepelt benne, csak ez lehet a sorrend, mi az adott elemek számossága?

Összefoglalás

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

További információ

- Gyorstalpalók:
 - w3schools.com: [XML Schema Tutorial](#)
 - Scott Ambler: [Introduction to UML2 Class Diagrams](#)
- [Rendszermodellezés](#): elágazó tárgy köv. félév
- Kitekintés
 - Pataricza András (szerk.): *Formális módszerek az informatikában*, Typotex, 2004
 - 1. Bevezetés, 7. Metamodellezés
 - OMG: UML 2.2 [Superstructure Specification](#)
 - 7. Classes – osztály diagramok definíciója