Struktúra alapú modellezés

Kritikus Rendszerek Kutatócsoport

2022

Tartalomjegyzék

1. A strukturális modellezés alkalmazásai 2

1.1. Hálózatok 2

1.2. Hierarchikus rendszerek . 5

1.3. Tulajdonságok 6

1.4. Típusok 8

2. A strukturális modellezés elmélete 11

2.1. Tulajdonságmodell 12

2.2. Gráfmodellek 13

2.3. Hierarchia 15

3. Nézetek 15

3.1. Szűrt nézet 15

3.2. Vetített nézet 17

4. Strukturális modellezési technikák 18

4.1. Hierarchia modellezése . . 18

5. Gyakorlati alkalmazások 19

5.1. Számítógéphálózat modellezése 19

5.2. Grafikus felhasználói felület 20

6. Összefoglalás 21

7. Kitekintés: technológiák és technikák* 21

7.1. Technológiák 21

7.2. Haladó strukturális modellezési technikák 23

7.3. Struktúramodellező eszközök és vizualizáció . . . 23

8. Elméleti kitekintés* 24

8.1. Bináris relációk tulajdonságai 24

9. Ajánlott irodalom 25

Irodalomjegyzék 28

Tárgymutató 28

Bevezetés

Hogyan épül fel egy rendszer? Milyen részekre bontható és ezek között milyen kapcsolat van? Ahhoz, hogy a rendszerünkkel kapcsolatos problémákat megválaszoljuk, fontos, hogy ezekre a kérdésekre tudjuk a választ. Ebben a fejezetben az egyes rendszerek *struktúrájának* jellemzésével foglalkozunk. Bemutatjuk a strukturális modellezés motivációját, a leggyakrabban alkalmazott formalizmusokat és azok matematikai alapjait.

1. A strukturális modellezés alkalmazásai

Mind a természetben, mind az ember alkotta rendszerekben fellelhetők bizonyos szabályszerűségek: egyesek a rendszer elemei közötti kapcsolatokat, míg mások magukat az elemeket jellemzik. Bár sok mindenben eltér egy közlekedési hálózat, egy számítógép-hálózat, egy épület vagy egy város felépítése, a strukturális modellezés eszköztárával olyan „nyelvet” kapunk, amivel ezeket hasonlóan modellezhetjük.

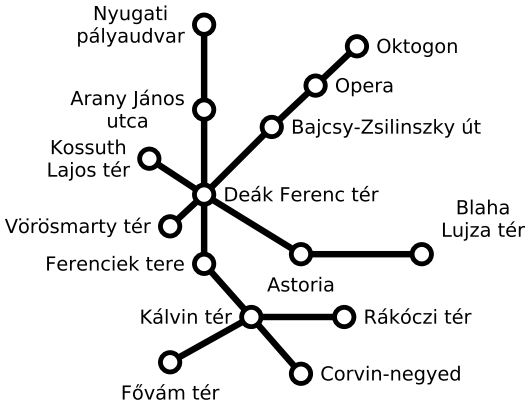
1.1. Hálózatok

Egy rendszert gyakran úgy jellemezhetünk a legjobban, ha bizonyos elemeit megkülönböztetjük és leírjuk az ezek közötti *kapcsolatokat*.

Példa. Egy nagyváros közlekedése az autóutak és sínhálózatok, a százezernyi járműnek és a rajta utazó embereknek szövevényes rendszere. A közlekedők folyamatosan mozgása mellett az infrastruktúra is rendszeresen változik a különböző fejlesztések, átalakítások és karbantartások miatt.
Egy ilyen rendszer precíz modellezése lehetetlen vállalkozás lenne, ezért helyette tipikusan olyan absztrakciókkal dolgozunk, amik az adott probléma megoldásához szükséges információkat tartalmazzák. Például az útvonalkereső alkalmazások ismerik a helyi tömegközlekedés járatait és segítséget nyújtanak abban, hogy az indulási pontunktól a célállomásig közlekedő járműveket és a közöttük lehetséges átszállásokat listázzák.
Vizsgáljuk meg például Budapest metróhálózatát! Az egyszerűség kedvéért a példában a hálózatnak csak a Nagykörúton belüli részével foglalkozunk.

A metróhálózat egyszerűsített térképe az 1(a) ábrán látható. A térképből is látható, hogy a hálózat könnyen ábrázolható egy *gráffal*, ahol a gráf minden *csomópontja* egy-egy metróállomást jelöl. A csomópontok címkéje a metrómegálló neve. Két csomópont között akkor fut *él*, ha a két megálló közvetlenül össze van kötve metróval (azaz nincs közöttük másik megálló). A metróhálózatot modellező gráf az 1(b) ábrán látható.

(a) A metróhálózat térképe [6]



(b) A metróhálózat egyszerű gráfként ábrázolva

1. ábra. Budapest metróhálózata a Nagykörúton belül

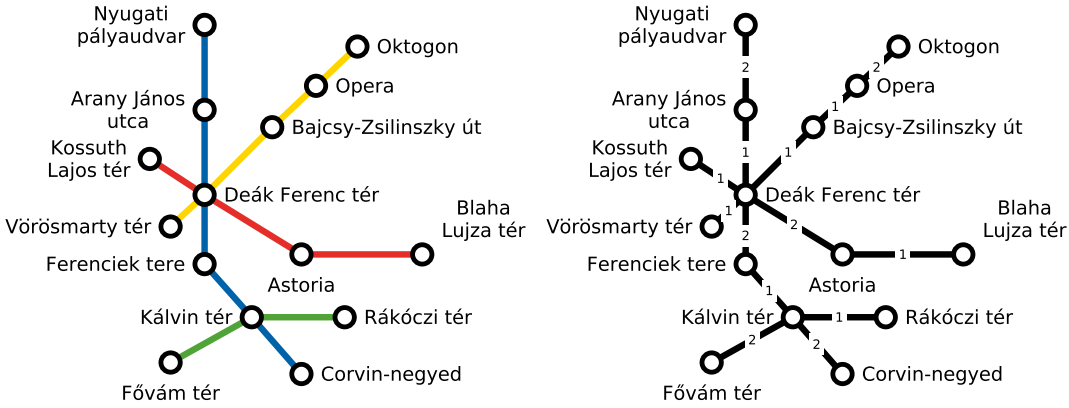
A modellünk segítségével választ kaphatunk például a következő kérdésekre:

1. Milyen megállók érhetők el a Vörösmarty térről indulva?
   Vizsgáljuk meg, hogy a Vörösmarty teret reprezentáló csomópontból kiindulva milyen csomópontok érhetők el. Ehhez elemi gráfbejáró algoritmusok használunk.
   Megjegyzés. Elemi útkereső algoritmusok pl. a *szélességi keresés* vagy *mélységi keresés*.
2. Legalább hány megállót kell utaznunk a Kossuth Lajos tér és a Kálvin tér között?
   A legrövidebb utat szintén kereséssel határozhatjuk meg.
   Megjegyzés. Például egy *szélességi kereséssel*.

Vannak azonban olyan metróközlekedéssel kapcsolatos kérdések, amelyek megválaszolásához a modell nem tartalmaz elég információt:

1. Milyen megállók érhetők el a Fővám térről indulva *legfeljebb egy átszállással*?
2. A menetrend szerint *hány percig tart* az út a Kossuth Lajos tér és az Astoria között?

Ezeknek a kérdéseknek megválaszolásához egészítsük ki a gráfot! Az első kérdéshez szükséges, hogy az egyes metróvonalakat meg tudjuk különböztetni, amit például az élek címkézésével érhetünk el. A 2(a) ábrán színekkel jelöltük a különböző élcímkéket. Induljunk ki a Fővám térről: átszállás nélkül a Kálvin tér és a Rákóczi tér megállókat érhetjük el, míg egy átszállással elérhetjük az M3-as metró vonalán található megállókat is.



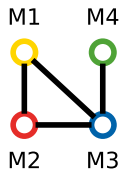
(a) A gráf élcímkékkel kiegészítve

(b) A gráf élsúlyokkal kiegészítve

2. ábra. A metróhálózatot ábrázoló gráf kiterjesztései

A második kérdés megválaszolásához az egyes megállók közötti utazási időt kell jellemeznünk. Ehhez vegyünk fel *élsúlyokat* a gráfba. A 2(b) ábrán élsúlyokkal jelöltük az egyes megállók között menetidőt. Ezek ismeretében meghatározható a Kossuth Lajos tér és a Kálvin tér közötti út menetrend szerinti időtartama. Ez a modell arra is alkalmas, hogy meghatározzuk a legrövidebb utat a két csomópont között, például a *Dijkstra algoritmus* segítségével.

Példa. Melyik metróvonalak között tudunk átszállni? A kérdésre választ kaphatunk a fenti modell segítségével. Nagyméretű hálózat esetén azonban már sokáig tarthat a válasz meghatározása, ezért érdemes olyan modellt készíteni, ami a válaszhoz nem szükséges részeket *absztrahálja*.



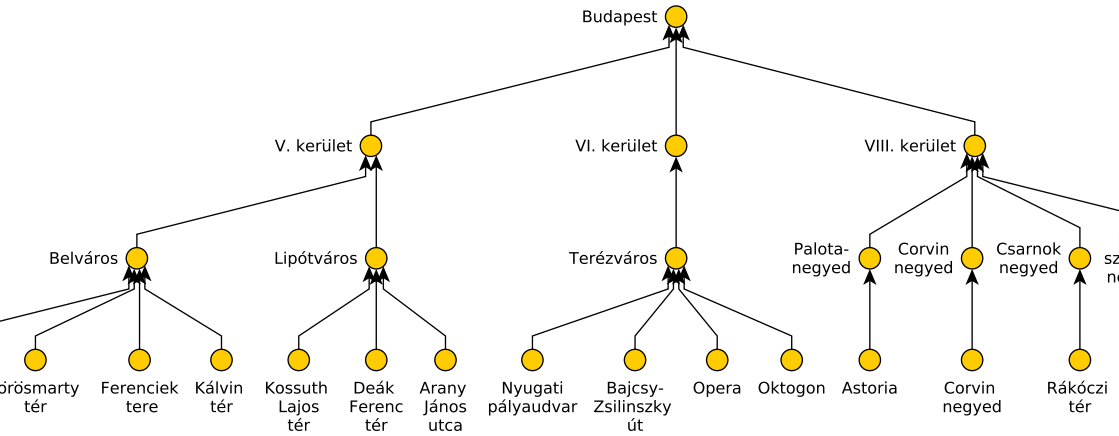
3. ábra. Átszállási lehetőségek a metróvonalak között a Nagykörúton belül

A 3. ábrán látható modellből azonnal kiderül, hogy mely metróvonalak között van átszállási lehetőség. A modell segítségével tehát hatékonyabban tudunk válaszolni erre a kérdésre, de a korábbi kérdésekhez szükséges információt elvesztettük az absztrakció során.

A közlekedési hálózathoz hasonlóan sok rendszer jól modellezhető gráffal: az élőlények táplálkozási lánca, a közösségi hálók, az úthálózat, telekommunikációs hálózatok stb.

1.2. Hierarchikus rendszerek

Példa. Budapestnek 23 kerülete van, amelyek további városrészekből állnak. Melyik városrészben van az Opera metrómegállója? Melyik városrészben van a legtöbb metrómegálló? Ezekhez hasonló kérdésekre úgy tudunk hatékonyan válaszolni, ha készítünk egy *hierarchikus modellt* a problémáról.



4. ábra. Budapest kerületei, városrészei és metrómegállói (részleges modell)

Készítsünk modellt, amely ábrázolja Budapest, a kerületek, a városrészek és a metrómegállók viszonyát! A 4. ábra modellje négy szintet tartalmaz:

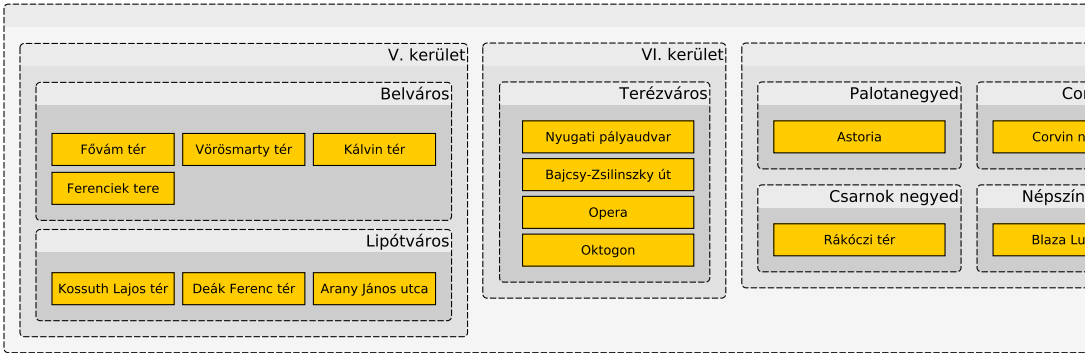
1. A hierarchia legfelső szintjén Budapest szerepel.
2. A második szinten a város kerületei találhatók.
3. A harmadik szinten az egyes városrészek vannak.
4. A negyedik szinten a metrómegállók találhatók.

Látható, hogy a hierarchikus modellt is ábrázolhatjuk gráfként. A csomópontok a modell különböző szintű elemeit reprezentálják, míg az élek a *része* viszonyt fejezik ki, például az Opera megálló a VI. kerület része. A gráf *gyökér csomópontja* a hierarchiában legmagasabban szereplő elem, Budapest.

Amennyiben egy rendszert hierarchikusan részekre bontunk, nem fordulhat elő, hogy egy elem tartalmazza a szülő elemét, ezért a hierarchikus modelleket reprezentáló gráfok körmentesek. A gyökér elemet leszámítva minden elemnek van szülője, tehát a gráf összefüggő is, így a hierarchikus modellek gráfjai egyúttal *fák*.

Megjegyzés. A fa struktúrában a gráf élei *explicit módon* jelölik a tartalmazási hierarchiát. A gyakorlatban nem mindig jelenítjük meg explicit módon a tartalmazási viszonyokat.

A tartalmazási hierarchia ábrázolható a tartalmazási viszonyok *implicit megjelenítésével* is:



5. ábra. A tartalmazási viszonyok implicit módon jelölve

Az 5. ábrán egy olyan diagramot látunk, amelyben az egyes síkidomok közötti bennfoglaló ábrázolása reprezentálja a modellben szereplő tartalmazási viszonyt.

Láthattuk tehát, hogy mind a modellelemek közötti kapcsolat, mind a modellhierarchia hatékonyan ábrázolható gráfként. A 6. ábrán látható modell a 2(a) és a 4. ábrákon található metróhálózatot és a területek hierarchiáját is tartalmazza.

1.3. Tulajdonságok

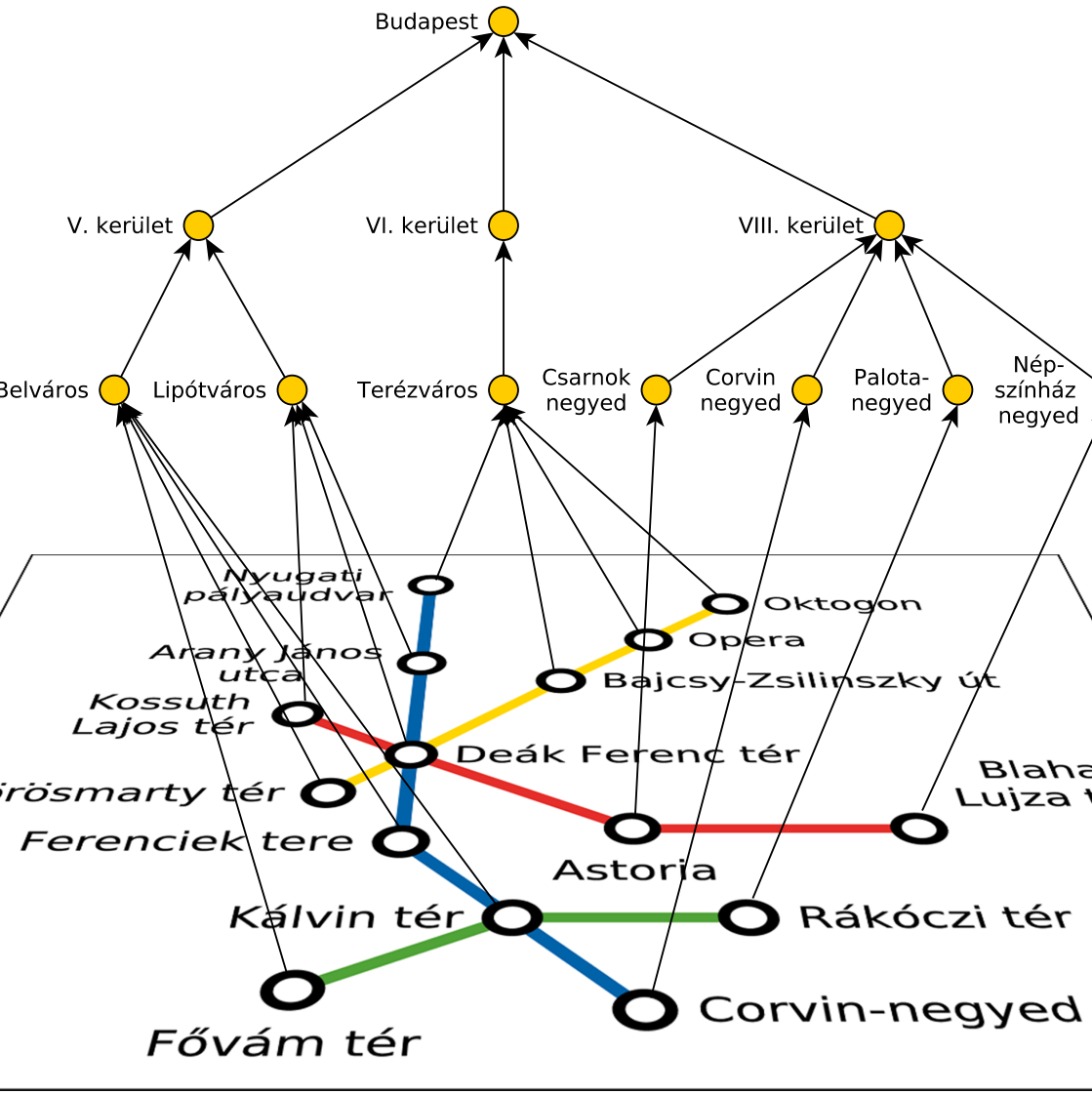
Példa. A közlekedési vállalat üzemen kívüli járművei telephelyeken parkolnak és itt végzik rajtuk a szükséges karbantartásokat. A metrók telephelyeinek neve metró *járműtelep*, a buszoké az *autóbuszgarázs*.
Mennyi üzemanyag tárolható a legnagyobb autóbusz garázsban? Milyen hosszú a Kőér utcai metró járműtelep vágányhálózata? Ezek a kérdések a modellünk elemeinek tulajdonságaival kapcsolatban merülnek fel. Építsünk modellt a problémára!

A telephelyeket például az alábbi modellel írhatjuk le:

azonosító	*helyszín*	*funkció*	*kapacitás*	*vágányhossz*	*max. üzemany*
T1	Mexikói út	metró járműtelep	24	8 500 m	
T2	Kőér utcai	metró járműtelep	60	16 512 m	
T3	Cinkota	autóbuszgarázs	265		250 000 lit
T4	Kelenföld	autóbuszgarázs	322		200 000 lit

1. táblázat. A BKK telephelyei (részleges modell)

A modellünk elemei a táblázat sorai. A táblázat fejlécében definiált *tulajdonságokból* (pl. helyszín, vágányhossz) az egyes modellelemek eltérő értékeket vehetnek



6. ábra. Budapest metróhálózata és a városrészek, kerületek hierarchiája

fel. Látható, hogy a táblázatnak nem minden celláját töltöttük ki, mivel az egyes jellemzők nincsenek mindenhol értelmezve.

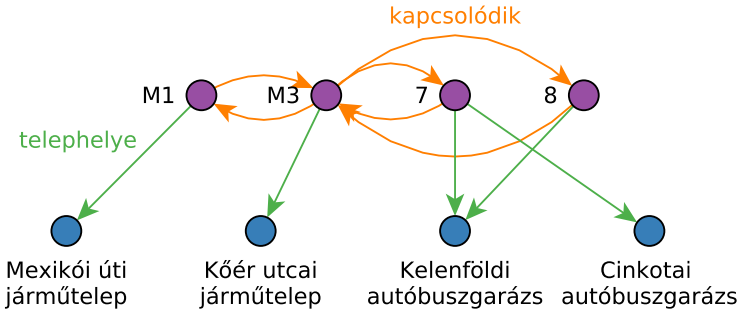
Megfigyelhetjük azonban, hogy az azonos funkcióval rendelkező modellelemek azonos tulajdonságokra vesznek fel értéket. Mindkét típusú telephelynek van *azonosító*, *helyszín*, *funkció* és *kapacitás* attribútuma. Az autóbuszgarázs esetén a *max. üzemanyag*, a metró járműtelep esetében a *vágányhossz* attribútumot rögzítjük.

1.4. Típusok

Az 1. táblázatban láthattuk, hogy a *funkció* jellemzővel megkülönböztethetjük az egyes a telephelyeket. Ha a *funkció* jellemzőt a modellelemek *típusának* tekintjük, akkor úgy is fogalmazhatunk, hogy a *vágányhossz* jellemző csak a metró járműtelep típusú elemekre értelmezett, a *max. üzemanyag* jellemző csak az autóbuszgarázs típusra. A típusokat felhasználhatjuk a gráfok jellemzésére is.

Példa. A korábbi metróhálózatot szeretnénk kiegészíteni a buszhálózatra vonatkozó információval. Szeretnénk tárolni azt is, hogy az egyes vonalakon közlekedő járművek melyik telephelyen parkolnak. Az autóbuszvonalon közelekedő járművek autóbuszgarázsban, a metróvonalon közlekedő járművek metró járműtelepen parkolnak.

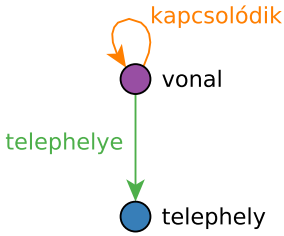
Készítsünk egy példánygráfot, amelyen ábrázoljuk az M1 és M3-as metróvonal, valamint a 7-es és a 8-as buszvonalak kapcsolódásait (*kapcsolódik*) *élek*, valamint azt, hogy az egyes vonalak melyik telephelyhez tartoznak (*telephelye*) *élek*.



7. ábra. Közlekedési vonalak és telephelyek

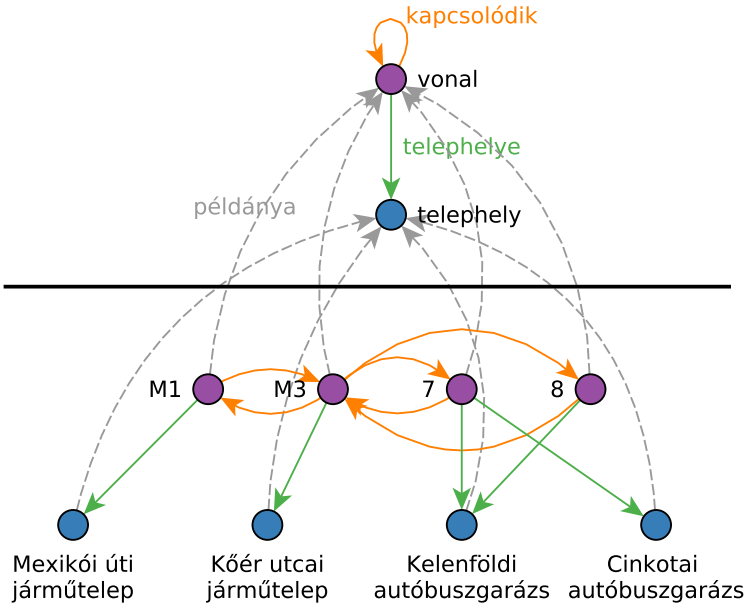
A példánygráfot követve a problémához tartozó *típusgráfban* meg kell jelennie a *vonal* és a *telephely* fogalmaknak. Az egyes vonalak kapcsolódhatnak egymáshoz a *kapcsolódik* él mentén, míg a vonal telephelyét a *telephelye* él jelzi.

Megjegyzés. Az egyes telephelyek tulajdonságait nem ábrázoltuk az ábrán.



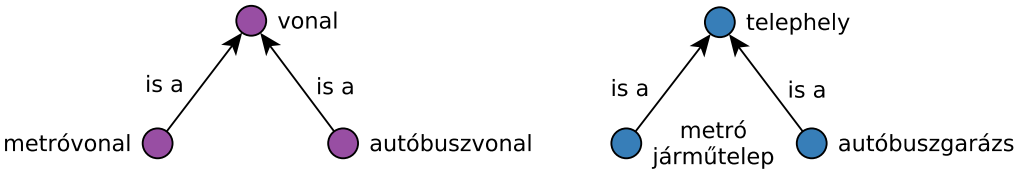
8. ábra. Közlekedési vonalak és telephelyek típusgráfja

A példánygráfot és a típusgráfot egy gráfban is jelölhetjük. Ekkor az egyes példányok és a típusok között egy *példánya* él fut.



9. ábra. Közlekedési vonalak és telephelyek példány- és típusgráfja

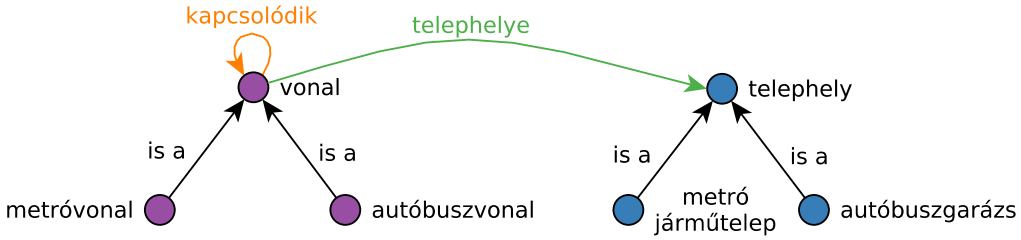
A 10. ábra bemutatja a problémához tartozó *típushierarchiát*. A típushierarchia egy hierarchikus modell, amely az egyes típusok leszármazási (*is a*) viszonyait ábrázolja.



10. ábra. Közlekedési vonalak és telephelyek típushierarchiája

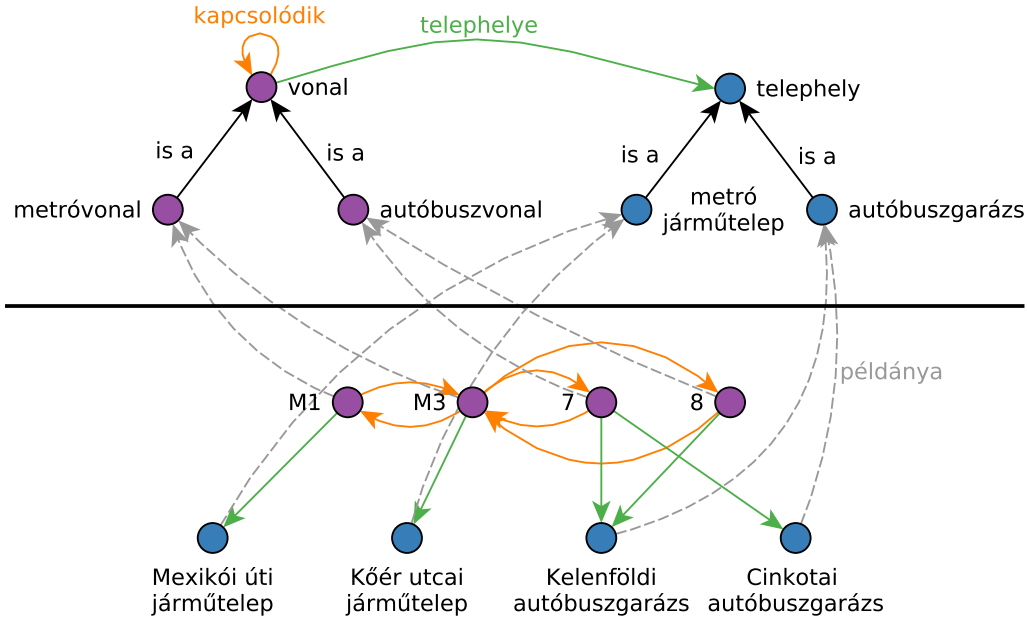
A rendszer fejlesztésekor gyakran fontos, hogy a típusgráfot és a típushierarchiát együtt lássuk. A *metamodell* tartalmazza a típusgráfot, a típusok hierarchiáját, a tulajdonságmodellt (ill. további, itt nem részletezett szabályokat, pl. multiplicitási kényszerek, jólformáltsági kényszerek).

A típusgráfban és a típushierarchiában tartalmazott információt, valamint tulajdonságmodellt együtt modellezve megkapjuk a terület metamodelljét. A metamodell részlete a 11. ábrán látható. Bár az ábrán a csomópontok elrendezése alapján következtethetünk arra, hogy melyik vonalakat látjuk, ezt az információt elvesztettük: ebben a reprezentációban az egyes metróvonalakat nem tudjuk megkülönböztetni.



11. ábra. Közlekedési vonalak és telephelyek (részleges) metamodellje

A példánymodellt és a metamodellt ábrázolhatjuk egyszerre is, a két modell elemei között a *példánya* viszony teremt kapcsolatot (szürke szaggatott nyíl).



12. ábra. Közlekedési vonalak és telephelyek példánymodellje és (részleges) metamodellje egy gráfon ábrázolva

Feladat. Az egyes járműtelepek is valamelyik városrészhez tartoznak. Készítsünk olyan metamodellt, amelyben ábrázolható az egyes járműtelepek és városrészek kapcsolata is!

2. A strukturális modellezés elmélete

Ahogy az eddigi példákban láttuk, a strukturális modellezés célja, hogy a rendszer felépítését jellemezze, beleértve az egyes elemek típusát, a közöttük lévő kapcsolatokat és hierarchiát, valamint az elemek tulajdonságait. Egy rendszer strukturális modellje tehát alkalmas arra, hogy az alábbi kérdésekre (nem feltétlenül kimerítő) választ nyújtson:

- Milyen elemekből áll a rendszer?
- Hogyan kapcsolódnak egymáshoz az elemek?
- Milyen hierarchia szerint épül fel a rendszer?
- Milyen tulajdonságúak a rendszer elemei?

A strukturális modellre az alábbi tömör definíciót adhatjuk.

Definíció. A *strukturális modell* a rendszer felépítésére (*struktúrájára*) vonatkozó tudás. A strukturális modell a rendszer alkotórészeire, ezek tulajdonságaira és egymással való viszonyaira vonatkozó statikus (tehát változást nem leíró) tudást reprezentál.

Megjegyzés. Fontos, hogy maga a strukturális modell változhat az idő során (pl. a metróhálózat fejlődik), de maga a modell nem ír le időbeli változásokat (pl. hogy miként mozognak a szerelvények).

A következőkben a korábbi példákra építve bemutatjuk a strukturális modellezés elméleti hátterét és precízen definiáljuk a szükséges fogalmakat.

2.1. Tulajdonságmodell

Definíció. A *jellemző* egy, a modell által megadott *parciális függvény*, amelyet a modellelemeken értelmezünk.

Megjegyzés. A H halmazon értelmezett *parciális függvény* nem jelent mást, mint a H valamely (nem megnevezett) részhalmazán értelmezett *függvény*. Konkrét esetünkben az egyes jellemzőket a modellelemek egy-egy részére értelmezzük (nem feltétlenül az összesre).

A jellemzőket gyakran táblázatos formában ábrázoljuk. Vizsgáljuk meg például az 1. táblázat jellemzőit:

azonosító	*helyszín*	*funkció*	*kapacitás*	*vágányhossz*	*max. üzemany*
T1	Mexikói út	metró járműtelep	24	8 500 m	
T2	Kőér utcai	metró járműtelep	60	16 512 m	
T3	Cinkota	autóbuszgarázs	265		250 000 lit
T4	Kelenföld	autóbuszgarázs	322		200 000 lit

Megjegyzés. A tulajdonságmodell mögötti matematikai struktúra az ún. *reláció*. A reláció pontos halmazelméleti definíciójával és a relációkon értelmezett műveleteket definiáló *relációalgebrával* bővebben az *Adatbázisok* tárgy foglalkozik.

A modellelemeket az *azonosító* jellemzőjükkel különböztetjük meg egymástól. A *kapacitás* jellemzőhöz tartozó függvény $\mathit{kapacitás}(e) \to n$, ahol e egy modellelem azonosítója, n egy nemnegatív egész szám. Például

$$\mathit{kapacitás}(\mathsf{T2}) = 60, \quad \mathit{kapacitás}(\mathsf{T3}) = 265.$$

A *funkció* jellemzőhöz tartozó függvény *funkció*$(e) \to t$, ahol e egy modellelem azonosítója, t a {metró járműtelep, autóbuszgarázs} halmaz eleme. Például

$$\textit{funkció}(\mathsf{T2}) = \text{metró járműtelep}, \quad \textit{funkció}(\mathsf{T3}) = \text{autóbuszgarázs}.$$

A fentiekhez hasonlóan, a *vágányhossz* jellemzőhöz tartozó parciális függvény *vágányhossz*$(e) \to n$, ahol e egy modellelem azonosítója, n egy nemnegatív egész szám. Fontos különbség viszont az eddigiekhez képest, hogy ezt jellemzőt csak bizonyos modellelemekre értelmeztünk, másokra nem: a metró járműtelepek esetén vesz fel értéket, az autóbuszgarázsok esetén viszont nem. Vagyis látható, hogy a *vágányhossz* jellemző csak akkor vesz fel értéket, ha a *funkció* attribútum értéke metró járműtelep.

Ha tehát a *funkció* attribútumot a telephely *típusának* tekintjük, akkor úgy is fogalmazhatunk, hogy a *vágányhossz* jellemző csak a metró járműtelep típusú elemekre értelmezett. Általánosan:

Definíció. A *típus* egy kitüntetett jellemző, amely meghatározza, hogy milyen más jellemzők lehetnek értelmezettek az adott modellelemre, illetve milyen más modellelemekkel lehet kapcsolatban. A többi jellemzőt *tulajdonságnak* hívjuk.

Megjegyzés. Jelen anyagban az egyszerűség kedvéért feltételezzük, hogy a modellelemeknek pontosan egy típusa van.

A modellünkben két típus van: az autóbuszgarázs és a járműtelep. Például a cinkotai telephely az autóbuszgarázs, míg a mexikói úti telephely a metró járműtelep típus példánya.

Definíció. Egy adott t típus *példányainak* nevezzük azon modellelemeket, amelyek típusa t.

A modellezési gyakorlatban elterjedt (de nem univerzális) megkötés, hogy az egyes elemek típusát állandónak tekintjük. Ha az elem típusa a rendszer működése során nem változhat meg, akkor olyan típust kell hozzárendelni, amely az elem teljes életciklusa során előforduló összes lehetséges jellemzővel, kapcsolattal összhangban van.

2.2. Gráfmodellek

Formális definíciók (∗). A definíciók támaszkodnak a gráfelmélet alapjaira, ezért röviden összefoglaljuk a felhasznált definíciókat.

A *gráf* egy olyan $G = (V, E)$ struktúra, ahol V halmaz a csomópontok, E az élek halmaza. Az élek csomópontok között futnak, *irányítatlan gráf* esetén az E halmaz csomópontok

rendezetlen $\{v_1, v_2\}$ párjaiból áll (tehát nem különböztetjük meg a $\{v_1, v_2\}$ és a $\{v_2, v_1\}$) párokat, míg *irányított gráf* esetén csomópontok rendezett (v_1, v_2) párjaiból.

Címkézett gráf esetén a gráf elemeit (csomópontjait és/vagy éleit) *címkékkel* láthatjuk el. A címkézés célja lehet *egyedi azonosító* hozzárendelése vagy bizonyos tulajdonság leírása (pl. a csomópontok kapacitása, élek típusa). Ha precízen szeretnénk jellemezni a gráfot, a következő terminológiát használhatjuk: ha csak a csomópontokhoz rendelünk címkéket, *csúcscímkézett* gráfról beszélünk, míg ha csak a gráf éleihez rendelünk címkéket, *élcímkézett* gráfról beszélünk.

A típusok gráf jellegű modellek esetén is fontos szerepet játszanak. A gráfmodell elemeit (külön-külön a csomópontokat és az éleket is) típusokba sorolhatjuk. A típusok meghatározzák, hogy az egyes csomópontok milyen élekkel köthetők össze.

Az egyes csomópont- és éltípusok viszonyát gráfként is ábrázolhatjuk.

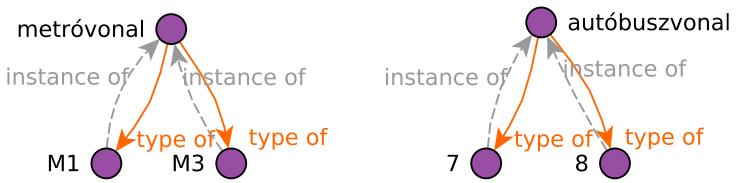
Definíció. A *típusgráf* egy olyan gráf, amelyben minden csomóponttípushoz egy típuscsomópont, minden éltípushoz egy típusél tartozik.

A gráfmodelleken is értelmezzük a *példány* fogalmát.

Definíció. Egy adott típusgráf *példánygráfja* alatt olyan gráfmodellt értünk, amelynek elemei a típusgráf csomópont- és éltípusainak példányai, valamint minden él forrása és célja rendre az éltípus forrásának és céljának példánya.

A típusgráfot és példánygráfot egy gráfon ábrázolva a típus-példány viszonyok is megjeleníthetők: a példánygráf csomópontjaiból a típusgráf csomópontjaira *instance of* (*példánya*) élek mutatnak. Szintén *instance of* viszony áll fenn a példánygráf élei és a típusgráf élei között.

Megjegyzés. Az *instance of* viszony helyett gyakran annak inverzét, a *type of* (x *típusa* y-nak) viszonyt ábrázoljuk. Gráfban ábrázolva az *instance of* és a *type of* élek adott csomópontok között egymással ellentétes irányúak.



13. ábra. Példány és típusgráf *type of* élekkel

Definíció. Egy rendszer *metamodellje* tartalmazza a típusgráfot, az egyes típusok közötti kapcsolatokat, ill. további megkötéseket is.

Megjegyzés. A további megkötések között szerepelhetnek jólformáltsági kényszerek, multiplicitási kényszerek stb. Ezekkel most nem foglalkozunk részletesen.

2.3. Hierarchia

Formális definíciók (∗). A *séta* szomszédos csúcsok és élek váltakozó sorozata, mely csúccsal kezdődik és csúcsban végződik. Az *út* olyan séta, amely nem metszi önmagát, valamint első és utolsó csúcsa különbözik. A *kör* olyan séta, amely nem metszi önmagát, valamint első és utolsó csúcsa megegyezik. A körmentes, összefüggő gráfokat *fának* nevezzük. (A körmentes gráfokat *erdőnek* nevezzük.) A fák esetén gyakran kiemelt szerepet tulajdonítunk egy csomópontnak: a *gyökér csomópont* a fa egy megkülönböztetett csomópontja. A *gyökeres fa* olyan fa, ami rendelkezik gyökér csomóponttal. *Gyökeres, szintezett fa* esetén a fa csomópontjaihoz hozzárendeljük a gyökértől vett távolságukat is.

A hierarchikus modellezésben kiemelt szerepet játszanak az irányított körmentes gráfok. Egy gráf DAG (*directed acyclic graph*), ha nem tartalmaz irányított kört.

Egy rendszer hierarchiája a rendszer dekompozíciójával állítható elő.

Definíció. A *dekompozíció* (*faktoring*) egy rendszer kisebb *komponensekre* bontása, amelyek könnyebben érthetők, fejleszthetők és karbantarthatók.

A rendszer így kapott egyes komponensei (részrendszerei) gyakran további dekompozícióval még kisebb részekre bonthatóak. Természetesen a dekompozíció során ügyelnünk kell arra, hogy az egyes részekből visszaállítható legyen az eredeti rendszer, különben a kapott strukturális modellünk hiányos.

Definíció. Egy dekompozíció *helyes*, ha a dekompozícióval kapott rendszer minden elemének megfeleltethető az eredeti rendszer valamelyik eleme, és az eredeti rendszer minden eleméhez hozzárendelhető a dekompozícióval kapott rendszer egy vagy több eleme.

3. Nézetek

A struktúramodellekből különböző *nézeteket* állíthatunk elő.

Tulajdonságmodelleken a leggyakrabban használt műveletek a *szűrés* és a *vetítés*. Ezek során úgy *absztraháljuk* a modellt, hogy bizonyos modellelemeket és/vagy azok egyes jellemzőit elhagyjuk.

Megjegyzés. A relációalgebrában a *szűrés* művelet neve *szelekció*, a *vetítés* művelet neve *projekció*.

3.1. Szűrt nézet

Ismét idézzük fel az 1. táblázat telephelyeket tartalmazó tulajdonságmodelljét.

azonosító	*helyszín*	*funkció*	*kapacitás*	*vágányhossz*	*max. üzemany*
T1	Mexikói út	metró járműtelep	24	8 500 m	
T2	Kőér utcai	metró járműtelep	60	16 512 m	
T3	Cinkota	autóbuszgarázs	265		250 000 lit
T4	Kelenföld	autóbuszgarázs	322		200 000 lit

Definíció. A *szűrés* művelet során a modell elemein kiértékelünk egy feltételt és azokat tartjuk meg, amelyek megfelelnek a feltételnek.

Tulajdonságmodellek esetén a szűrés az elhagyott modellelemek a modell sorai lehetnek, gráf jellegű modellek esetén a gráf csúcsai vagy élei.

3.1.1. Tulajdonságmodell szűrése

Példa. Szeretnénk megtudni, hogy mely telephely alkalmas legalább 100 jármű befogására.

Azokra az elemekre szűrünk, amelyeknél a kapacitás jellemző értéke 100-nál nagyobb vagy egyenlő.

Az eredmény:

azonosító	*helyszín*	*funkció*	*kapacitás*	*vágányhossz*	*max. üzemanyag*
T3	Cinkota	autóbuszgarázs	265		250 000 liter
T4	Kelenföld	autóbuszgarázs	322		200 000 liter

Példa. Szeretnénk megtudni, hogy mely metró járműtelep alkalmas legalább 50 jármű befogására.

Végezzünk szűrést azokra a modellelemekre, ahol a *funkció* attribútum értéke metró járműtelep, a *kapacitás* attribútum értéke nagyobb vagy egyenlő 50-nél.

Az eredmény:

azonosító	*helyszín*	*funkció*	*kapacitás*	*vágányhossz*	*max. üzemany*
T2	Kőér utcai	metró járműtelep	60	16 512 m	

3.1.2. Gráfmodell szűrése

Egy gráfmodellből a szűrés a modell egy *részgráfját* állítja elő.

Példa. Soroljuk fel az M2-es és az M4-es metró megállóit.

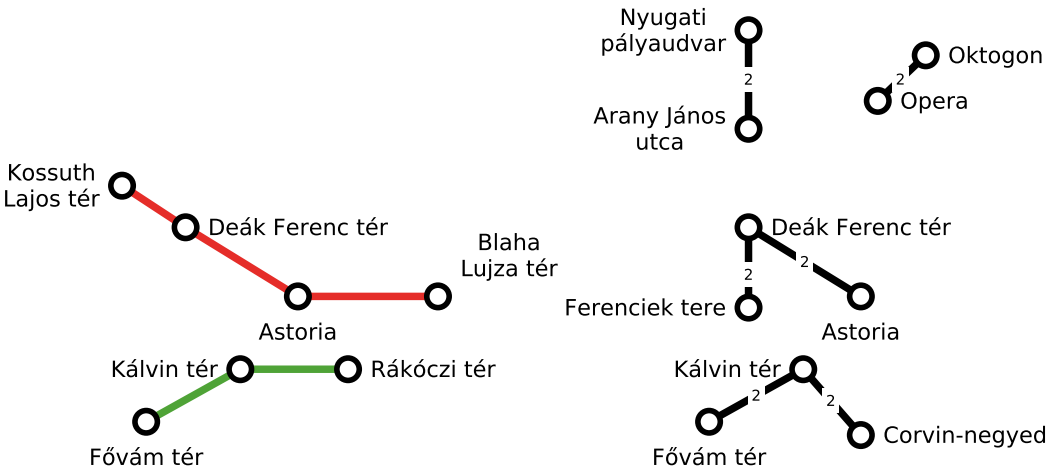
A modellen szűrést végzünk, ami csak azokat a csomópontokat tartja meg, amelyekhez tartozik olyan él, amelynek a címkéje M2 vagy M4. A kapott részgráf a 14(a) ábrán látható.

Példa. Határozzuk meg, hogy mely szomszédos metrómegállók között hosszabb egy percnél a menetidő.

A modellen szűrést végzünk, ami

- csak azokat a csomópontokat tartja meg, amelyekhez tartozik 1-nél nagyobb súlyú él,
- csak az 1-nél nagyobb súlyú éleket tartja meg.

A kapott részgráf a 14(b) ábrán látható.



(a) Az M2 és az M4 metróvonalak megállói (b) Az 1 percnél távolabbi metrómegállók

14. ábra. Szűrések a metróhálózatot tartalmazó gráfon

3.2. Vetített nézet

Definíció. *Vetítés* során a modell egyes jellemzőit kiválasztjuk és a többit elhagyjuk a táblázatból.

Megjegyzés. Érvényes vetítés művelet az is, ha a tulajdonságmodell összes jellemzőjét megtartjuk.

3.2.1. Tulajdonságmodell vetítése

Példa. Olyan kimutatásra van szükségünk, ami csak az egyes telephelyek helyszínét, funkcióját és kapacitását tartalmazza.

Végezzünk szűrést a tulajdonságmodelleken a *helyszín*, *funkció* és a *kapacitás* attribútumokra.

helyszín	*funkció*	*kapacitás*
Mexikói út	metró járműtelep	24
Kőér utcai	metró járműtelep	60
Cinkota	autóbuszgarázs	265
Kelenföld	autóbuszgarázs	322

4. Strukturális modellezési technikák

A modellezési formalizmusok után bemutatunk néhány strukturális modellezési technikát.

4.1. Hierarchia modellezése

Korábban láttuk, hogy a modellelemek közti szigorú hierarchia kifejezhető fa (ill. erdő) gráfokkal. Ezek a fajta modellek képesek kifejezni a (rész)rendszerek és alkotóelemeik közti tartalmazási viszonyt, akár többszintű tartalmazással is (a részrendszerek is további részeket tartalmaznak). Azonban a gyakorlatban a modellnek sokszor ennél jóval több információt kell tartalmaznia; egy-egy adott modellelemmel kapcsolatban nem csak a tartalmazó és tartalmazott komponenseivel való tartalmazási viszonyát kell ismerni, hanem egyéb modellelemekkel való kapcsolatait.

Ilyenkor a strukturális modell (gráf jellegű része) két rétegre tagozódik: egyrészt a modell szerkezeti vázát alkotó tartalmazási hierarchiára (fa/erdő), amely az alkotóelemek rész-egész viszonyait reprezentálja, másrészt ezen felüli *kereszthivatkozás* élekre, amelyek a tartalmazási rendtől függetlenül, a körmentesség korlátozása nélkül köthetnek össze elemeket. Ennek megfelelően egy metamodell megmondhatja, hogy mely éltípusok példányait fogjuk az említett szerkezeti vázat alkotó tartalmazási éleknek tekinteni.

A hierarchikus modellek megalkotása, illetve az összetett rendszerek tervezése során többféleképp választható meg az egyes elemek elkészítésének sorrendje. Jól illusztrálja a választási szabadságot két polárisan ellentétes megközelítés: a *top-down* és a *bottom-up* modellezés.

Definíció. *Top-down* modellezés során a rendszert felülről lefelé (összetett rendszerből az alkotóelemei felé haladva) építjük. A modellezés alaplépése a *dekompozíció*.

Egy top-down modellezési / tervezési folyamatot úgy kell tehát elképzelni, hogy a kezdetektől fogva az egész rendszer modelljét építjük; azonban eleinte hiányoznak még a részletek. Idővel a modellt finomítjuk: tartalmazott alkotóelemekre bontjuk a rendszert, megadva azok tulajdonságait és kereszthivatkozásait is; majd később magukat az alkotóelemeket ugyanúgy strukturális dekompozíciónak vetjük alá.

A top-down modellezés fontos jellemzői:

- ⊕ Részrendszer tervezésekor a szerepe már ismert
- ⊖ „Félidőben” még nincsenek működő (teljes részletességgel elkészített) részek
- ⊖ Részek problémái, igényei későn derülnek ki

Definíció. *Bottom-up* modellezés során a rendszert alulról felfelé (elszigetelt alkotóelemekből az összetett rendszer felé haladva) építjük. A modellezés alaplépése a *kompozíció*: az egész rendszer összeszerkesztése külön modellezett vagy fejlesztett részrendszerekből.

Egy bottom-up modellezési / tervezési folyamatot úgy kell tehát elképzelni, hogy a kezdetektől fogva részletes modelleket építünk; azonban eleinte csak a rendszer izolált, egyszerű komponenseivel foglalkozunk. Ahogy fokozatosan egyre több komponens készül el, nagyobb részrendszerekké foglalhatjuk őket össze, az egymáshoz való kapcsolódásukat is tisztázva. Idővel az így kapott összetettebb részrendszereket is további kompozíciónak vethetjük alá.

A bottom-up modellezés fontos jellemzői:

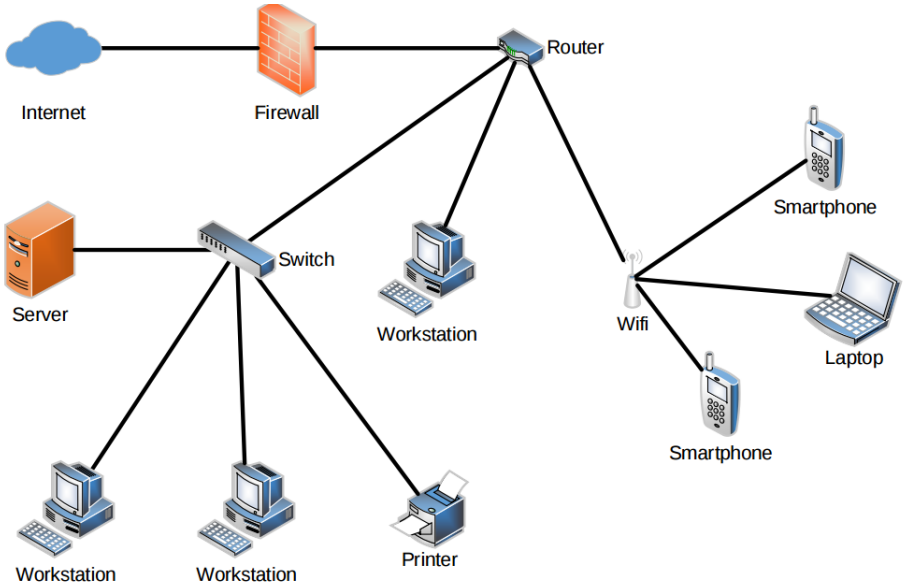
- ⊕ A rendszer részei önmagukban kipróbálhatók, tesztelhetők
- ⊕ Részleges készültségnél könnyebben előállítható a rendszer prototípusa
- ⊖ Nem látszik előre a rész szerepe az egészben

Természetesen a gyakorlatban a kétféle szélsőséges megközelítés közti kompromisszum is elképzelhető.

5. Gyakorlati alkalmazások

5.1. Számítógéphálózat modellezése

Számítógép-hálózatok kiválóan modellezhetők gráfokkal, amelyben a gráf csomópontjai a hálózat elemei (pl. számítógép, router, tűzfal), a kapcsolatok pedig ezek összeköttetései.



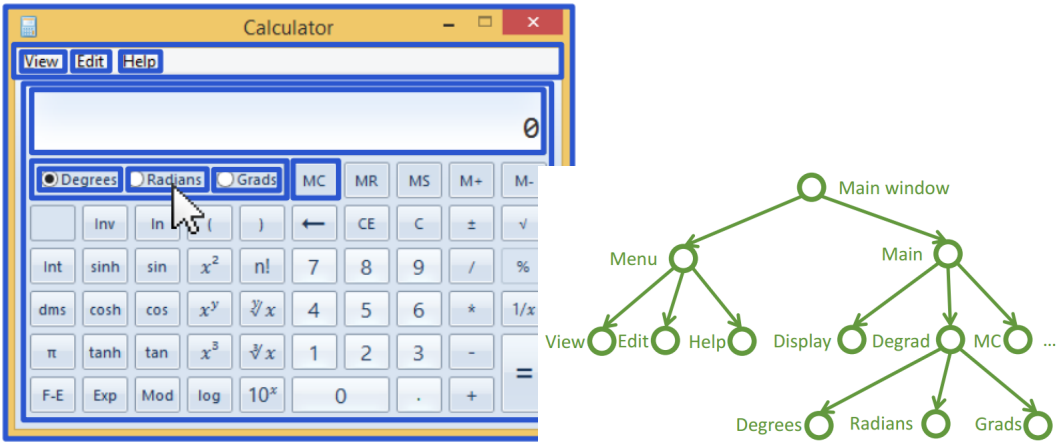
15. ábra. Hálózat

- Milyen elemekből áll a rendszer, milyen kapcsolatok lehetségesek?
- Van-e egyszeres hibapont a rendszerben?
- Milyen hosszú úton, milyen típusú elemeket érintve lehet elérni az internetet?
- Hány gép van a wifi hálózaton?
- Egy elem hibája meddig terjedhet?
- Elérhető-e az internet?

Feladat. Milyen típushierarchiát készíthetünk egy számítógép-hálózathoz?

5.2. Grafikus felhasználói felület

Egy szoftver alkalmazás *grafikus felhasználói felülete* (GUI) is egy hierarchikus modell.



(a) A számológép alkalmazás grafikus felülete

(b) A komponensek hierarchiája

16. ábra. Grafikus felhasználói felület és komponenseinek hierarchiája

Feladat. Mi történik, amikor egy alkalmazás ablakán kattintunk – hogyan határozza meg a rendszer, hogy melyik komponensre kattintottunk?

6. Összefoglalás

A fejezetben bemutattuk *struktúra alapú modellezés* motivációját, a használt formalizmusukat és azok alkalmazásait. Ismertettük *típusok* fontosságát és a típusrendszer ábrázolásának lehetőségeit.

A következő fejezetekben a *viselkedés alapú modellezést* és annak formalizmusait mutatjuk be.

7. Kitekintés: technológiák és technikák*

Az alábbiakban bemutatunk néhány, a strukturális modellezés témaköréhez kapcsolódó gyakorlati technológiát, specifikációt és eszközt. Az itt felsorolt fogalmak nem részei a számonkérésnek, gyakran előkerülnek viszont a későbbi tanulmányok és munkák során, ezért mindenképpen érdemes legalább névről ismerni őket.

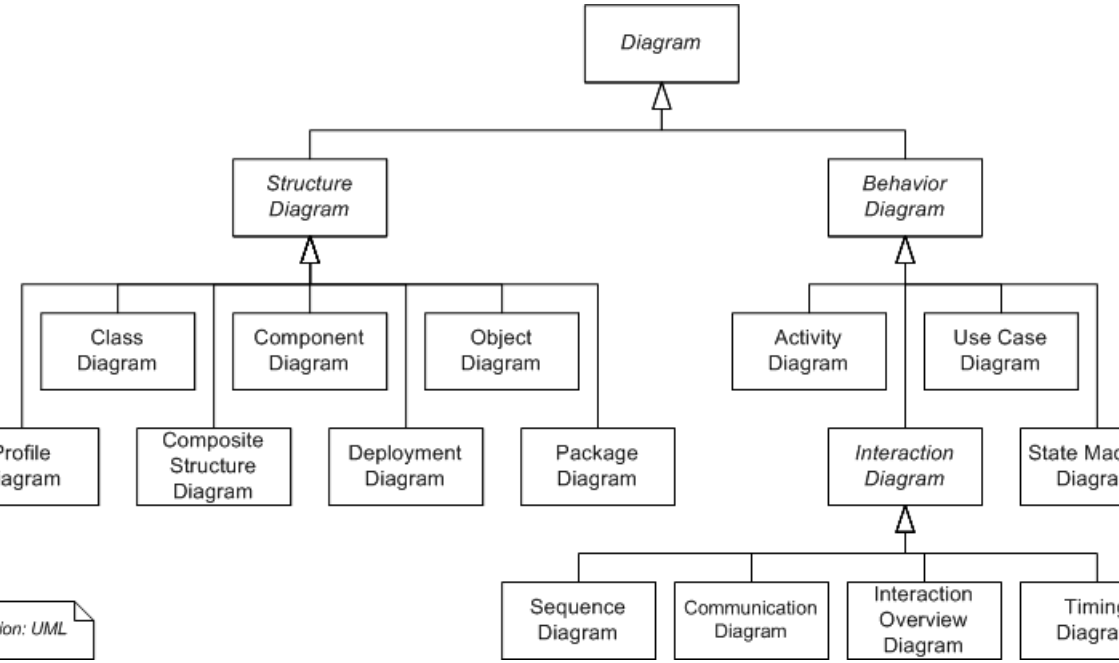
7.1. Technológiák

A gyakorlatban sokféle modellezési nyelvet és technológiát használnak. Ezek közül mutatunk be most azokat, amelyek a később tanulmányok során előkerülnek.

7.1.1. UML

Az UML (*Unified Modeling Language*) egy általános célú modellezési nyelv az [5]. Az UML három fő diagramtípust definiál:

- *Structure Diagram:* strukturális modellek leírására. A *Class Diagram* az osztályok (metamodell), míg az *Object diagram* a példányok (modell) leírására alkalmas. A *Composite Structure Diagram* egy rendszer struktúráját és a rendszer komponenseinek kapcsolatát mutatja be.
- *Behaviour Diagram:* viselkedési modellek leírására, pl. a *State Machine Diagram* segítségével állapotgépek az *Activity Diagramon* folyamatok ábrázolhatók. A *Behaviour Diagramek* között megkülönböztejük az *Interaction Diagrameket*. Ezeknek szintén a viselkedés leírása a célja, de a hangsúly a vezérlés- és adatáramlás bemutatásán van. Ilyen pl. a *Sequence Diagram* (szekvenciadiagram), amely az egyes objektumok közötti interakciót mutatja be üzenetek formájában.



17. ábra. UML diagramok típusai és a közöttük lévő viszony osztálydiagramként ábrázolva [12]

Az UML nyelvvel részletesen foglalkozik a *Szoftvertechnológia* tárgy. A nyelvről egy jó összefoglalót ad a [2] oldal.

7.1.2. EMF

Az Eclipse fejlesztőkörnyezet[1] saját modellezési keretrendszerrel rendelkezik, ez az EMF (*Eclipse Modeling Framework*). Az EMF metamodellező nyelve, az Ecore lehetővé teszi saját, ún. *szakterület-specifikus nyelv* (*domain-specific language*, DSL) definiálását. Az EMF mára több területen is *de facto* modellezési keretrendszer.

Az Eclipse fejlesztőkörnyezettel és az EMF keretrendszerrel foglalkozik az *Eclipse alapú technológiák* szabadon választható tárgy.

7.2. Haladó strukturális modellezési technikák

A struktúramodellek definiálására és fejlesztésére különböző technikák léteznek. Korábban tárgyaltuk a *top-down* és a *bottom-up* tervezést. Itt két további, általánosan alkalmazható technikát mutatunk be.

7.2.1. Tervezési minták

Az *objektum-orientált* tervezés során gyakran előforduló problémákra különböző *tervezési minták* (*design patterns*) léteznek. A tervezési minták között külön szerepet kapnak a rendszer struktúráját leíró *szerkezeti minták* (*structural patterns*). A tervezési mintákkal bővebben a *Szoftvertechnikák* tárgy foglalkozik.

7.2.2. Refaktoring

A *dekompozícióhoz*, azaz *faktoringhoz* szorosan kapcsolódik a *refaktoring* (*refactoring*) fogalma [3]. Refaktoringon egy rendszert definiáló programkód vagy modell átalakítását értjük. A refaktoring lényege, hogy az átalakítás során a rendszer megfigyelhető működése változatlan marad, de a kapott programkód vagy modell érthetőbb, karbantarthatóbb lesz. Tipikus refaktoring műveletek pl. változók átnevezése, ismétlődő programrészletek megszüntetése (pl. külön metódusba vagy osztályba kiszervezéssel).

7.3. Struktúramodellező eszközök és vizualizáció

Gráfok automatikus megjelenítésére alkalmas pl. a GraphViz[2] programcsomag. Gráfok feldolgozására gyakran alkalmazzák az igraph[3] programcsomagot. Manapság több gráfadatbázis-kezelő rendszer is elterjedt, pl. a Neo4j[4] és a Titan[5] rendszerek.

---

[1] `http://www.eclipse.org/`
[2] `http://www.graphviz.org/`
[3] `http://igraph.org/`
[4] `http://neo4j.com/`
[5] `http://thinkaurelius.github.io/titan/`

Gráfok manuális rajzolására szintén több eszköz elterjedt. Egyszerűen használható online felületet biztosít a draw.io[6] és az Arrow Tools[7]. A jegyzetben szereplő ábrák a yEd eszközzel[8] készültek. Sok információt tartalmazó gráf esetén érdemes lehet vektorgrafikus rajzoló-, ill. prezentáló eszközök, pl. a Microsoft Visio vagy Microsoft PowerPoint alkalmazás.

8. Elméleti kitekintés*

A strukturális modellezésnek komoly matematikai eszköztára is van. Az alábbiakban ezekből mutatunk be néhány részletet.

8.1. Bináris relációk tulajdonságai

Egy (D_1, D_2) halmazpáron értelmezett R bináris relációt úgy definiálhatunk, mint ezen halmazok Descartes-szorzatának egy részhalmazát: $R \subseteq D_1 \times D_2$.

Hasznos ismerni a kétváltozós relációkon értelmezett tulajdonságokat [13].

Tipp. Az elnevezések egyezése nem véletlen, a *kétváltozós reláció* egy alesete a *reláció* fogalomnak.

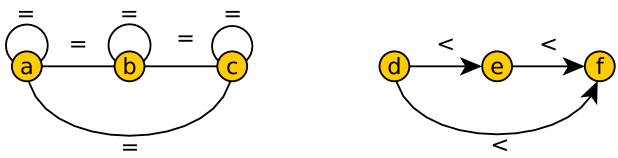
Definíció. Egy S halmazon értelmezett r kétváltozós reláció *reflexív*, ha bármely $a \in S$-re $r(a, a)$ teljesül.

Definíció. Egy S halmazon értelmezett r kétváltozós reláció *szimmetrikus*, ha bármely $a, b \in S$-re $r(a, b)$ teljesülése esetén $r(b, a)$ is teljesül. A nem szimmetrikus relációkat *aszimmetrikusnak* nevezzük.

Definíció. Egy S halmazon értelmezett r kétváltozós reláció *tranzitív*, ha bármely $a, b, c \in S$-re $r(a, b)$ és $r(b, c)$ teljesülése esetén $r(a, c)$ is teljesül.

[6] `http://draw.io/`
[7] `http://www.apcjones.com/arrows/`
[8] `https://www.yworks.com/products/yed`

Példa.



18. ábra. Az *egyenlő* (=) és a *kisebb* (<) relációk gráfon ábrázolva

Tranzitív relációk:

- Az *egyenlő* (=) és a *kisebb* (<) relációk tranzitívak, mert
  - $a = b$ és $b = c$ esetén $a = c$,
  - $d < e$ és $e < f$ esetén $d < f$.

  A 18. ábrán ábrázoltuk a fenti relációkat. Az = reláció szimmetrikus, ezért irányítatlan gráffal, a < reláció aszimmetrikus, ezért irányított gráffal reprezentálható.

Nem tranzitív relációk:

- A *nemegyenlő* reláció ($\neq$) nem tranzitív, mert
  - $a \neq b$ és $b \neq c$ esetén $a \neq c$ nem mindig áll fenn, például
  - $1 \neq 2$ és $2 \neq 1$ esetén $1 \neq 1$ nem teljesül.
- Személyek közötti *őse* reláció tranzitív, mert *őse*(a, b) és *őse*(b, c) esetén *őse*(a, c) is fennáll.
- Személyek közötti *ismerőse* reláció nem tranzitív, mert *ismerőse*(a, b) és *ismerőse*(b, c) esetén nem garantált, hogy *ismerőse*(a, c) fennáll.

Definíció. Egy S halmazon értelmezett r kétváltozós reláció *ekvivalenciareláció*, ha *reflexív*, *szimmetrikus* és *tranzitív*.

Példa. Ekvivalenciareláció az *egyenlőségreláció*, a *gráfizomorfia* és az *elérhetőség gráfokban*.

9. Ajánlott irodalom

A gráfelmélettel behatóan foglalkozik a *Bevezetés a számításelméletbe 2.* tantárgy és Fleiner Tamás jegyzete [11]. Különböző gráfalgoritmusokat mutat be – pl. *legrövidebb út* és minimális összsúlyú *feszítőfa* keresésére – az *Algoritmuselmélet* tárgy. További keresőalgoritmusok a *Mesterséges intelligencia* tárgyban szerepelnek.

Olvasmányos összefoglalót nyújt az UML nyelvről Martin Fowler „UML distilled” című könyve [4].

A tulajdonsággráfokról egy jól érthető tudományos cikk Marko Rodriguez és Peter Neubauer munkája [9]. Rodriguez a Titan elosztott gráfadatbázis-kezelő rendszer egyik fő fejlesztője, míg Neubauer a Neo4j gráfadatbázis-kezelőt fejlesztő cég alapítója (mindkét rendszert említettük a 7.3. szakaszban). Az elosztott gráfadatbázis alkalmazását, elméleti és gyakorlati kihívásait kiváló prezentációkban mutatja be [7, 8].

Barabási-Albert László magyar fizikus nemzetközileg elismert kutatója a komplex *hálózatok* elméletének. Barabási „Behálózva” című könyve közérthető stílusban mutatja be a hálózatok elemzésének elméleti kihívásait a kutatási eredmények gyakorlati jelentőségét [1]. A szerzővel több interjú is készült.[9,10,11]

Az osztályok és prototípusok közötti elvi különbséget mutatja be Antero Taivalsaari, a Nokia Research fejlesztőjének cikke [10].

[9] `http://index.hu/tudomany/bal080429/`

[10] `http://index.hu/tudomany/2010/06/02/az_elso_cikk_utan_majdnem_leharaptak_a_fejunket/`

[11] `http://index.hu/tudomany/2015/02/20/az_nsa_primitiven_hasznalta_a_begyujtott_adatokat/`

Hivatkozások

[1] Barabási Albert-László: *Behálózva – A hálózatok új tudománya*. 2013, Helikon Kiadó.

[2] Kirill Fakhroutdinov: The Unified Modeling Language, 2016.
URL `http://www.uml-diagrams.org/`.

[3] M. Fowler – K. Beck – J. Brant – W. Opdyke – D. Roberts: *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Object Technology Series sorozat. 2012, Pearson Education. ISBN 9780133065268.
URL `http://books.google.hu/books?id=HmrDHwgkbPsC`.

[4] M. Fowler – K. Scott: *UML distilled: applying the standard object modeling language*. The Addison-Wesley object technology series sorozat. 1997, Addison Wesley Longman. ISBN 9780201325638.
URL `https://books.google.hu/books?id=JdEZAQAAIAAJ`.

[5] Object Management Group: Information technology – Object Management Group Unified Modeling Language (OMG UML) – part 2: Superstructure. ISO/IEC 19505-2:2012. Jelentés, 2012, Object Management Group.
URL `http://www.omg.org/spec/UML/ISO/19505-2/PDF/`.

[6] Budapesti Közlekedési Központ: Budapest metró- és HÉV-hálózata, 2016.
URL `http://www.bkk.hu/apps/docs/terkep/metro.pdf`.

[7] Marko Rodriguez: Titan: The rise of big graph data. `http://www.slideshare.net/slidarko/titan-the-rise-of-big-graph-data`, 2012. június.

[8] Marko Rodriguez: On graph computing. `http://markorodriguez.com/2013/01/09/on-graph-computing/`, 2013. január.

[9] Marko A. Rodriguez – Peter Neubauer: Constructions from dots and lines. *CoRR*, abs/1006.2361. évf. (2010). URL `http://arxiv.org/abs/1006.2361`.

[10] Antero Taivalsaari: Classes versus prototypes: Some philosophical and historical observations. *JOOP*, 10. évf. (1997) 7. sz., 44–50. p.

[11] Fleiner Tamás: A számítástudomány alapjai. Jegyzet, 2014, Budapesti Műszaki és Gazdaságtudományi Egyetem.

[12] Wikibooks: Introduction to software engineering — wikibooks, the free textbook project, 2015.
URL `https://en.wikibooks.org/w/index.php?title=Introduction_to_Software_Engineering`. [Online; accessed 16-February-2016].

[13] Wikipédia: Reláció. http://hu.wikipedia.org/wiki/Rel%C3%A1ci%C3%B3, 2015. március.

Tárgymutató

él edge 2
élcímkézett gráf edge-labeled graph 14
élsúly edge weight 4

út path [pɑːθ] 15

absztrakció abstraction [əbˈstɹæk.ʃn̩] 4, 15
alulról felfelé bottom-up 18, 19, 23
aszimmetria asymmetry [eɪˈsɪmɪtɹi] 24

BFS szélességi keresés; breadth-first search 3

csomópont node, vertex [noʊd, ˈvɜɹtɛks] 2
csúcscímkézett gráf vertex-labeled graph 14
címke label 14
címkézett gráf labeled graph 14

DAG irányított körmentes gráf; directed acyclic graph 15
dekompozíció decomposition 15, 19, 23
DFS mélységi keresés; depth-first search 3
Dijkstra algoritmus Dijksta's algorithm [ˈdɛɪkstras ˈælɡəɹɪðm] 4
DSL szakterület-specifikus nyelv; domain-specific language 23

egyedi azonosító unique identifier 14
ekvivalenciareláció equivalence relation 25
EMF Eclipse modellezési keretrendszer; Eclipse Modeling Framework 23

erdő forest 15

fa gráf tree graph 5, 15
faktoring factoring [fæktərɪŋ] 15, 23
felülről lefelé top-down 18, 19, 23
feszítőfa spanning tree 25
függvény function 12

gráf graph [ɡɹɑːf] 2, 13
GUI [ˈɡuːi] grafikus felhasználói felület; graphical user interface 20
gyökeres, szintezett fa leveled tree 15
gyökeres fa rooted tree 15
gyökér csomópont root note 5, 15

helyes dekompozíció faithful decomposition (approximate translation) 15
hierarchikus modell hierarchical model [ˌhaɪərˈɑːkɪkəl] 5
hálózat network [nɛtwɜːk] 26

irányítatlan gráf undirected graph 13
irányított gráf directed graph 14

jellemző feature 12

kapcsolat relationship 2
kereszthivatkozás cross reference 18
komponens component [kəmˈpəʊnənt] 15
kompozíció composition [ˌkɒmpəˈzɪʃən] 19
kétváltozós reláció binary relation 24
kör cycle [ˈsaɪkəl] 15

legrövidebb út shortest path 25

metamodell metamodel [ˈmɛtəmɒdl̩] 10, 14

nézet view 15

objektum-orientált object-oriented [ˈɒbdʒɛkt ɔrɪəntɪd] 23

parciális függvény partial function 12
projekció projection [pɹəˈdʒɛkʃən] 15
példány instance [ˈɪnstəns] 13, 14
példánya instance of 14
példánygráf instance graph 14

refaktoring refactoring 23
reflexivitás reflexivity [ˌriflɛkˈsɪvɪti] 24, 25
reláció relation 12, 24
relációalgebra relational algebra 12
részgráf subgraph 16

strukturális modell structural model 12
struktúra structure [ˈstɹʌktʃə(ɹ)] 2, 12
struktúra alapú modellezés structural modeling 21

szelekció selection [səˈlɛkʃən] 15
szerkezeti minta structural pattern 23
szimmetria symmetry [ˈsɪmɪtri] 24, 25
szűrés filtering 15, 16
séta walk, chain 15

tartalmazás containment [kənˈteɪnm(ə)nt] 5
tervezési minta design pattern 23
tranzitivitás transitivity [ˌtrænsəˈtɪvəti] 24, 25
tulajdonság property 6, 13
típus type 8, 13, 21
típusa type of 14
típusgráf type graph 8, 14
típushierarchia type hierarchy 9

UML egységesített modellező nyelv; Unified Modeling Language 22

vetítés projection 15, 17
viselkedés alapú modellezés behavioural modelling 21