

Vállalati információs rendszerek szervezési kérdései

<http://irh.inf.unideb.hu/user/kuki/vir.pdf>

Irodalom

- Harald Störrle: UML 2 – Unified Modeling Language
- Ian Sommerville: Szoftverrendszerek fejlesztése
- Bana István: Az SSADM rendszerszervezési módszertan
- Vég Csaba: Rational Unified Process - Áttekintés

Szoftverkrízis (példák, adatok)

- 2009 Toyota Prius „smart car embedded code”
- „Megbolondult” ellenőrző lámpák, leállítás stb.
- Más típusokat is érintett
- Nem lehet online „peccselni”
- 40 millió gépjármű - Intelligent Power Module Recall

Szoftverkrízis (példák, adatok)

- 1962 – Mariner 1 Vénusz űrszonda
- „Eltévedt” – az óceán fölött lett megsemmisítve
- Papíron helyes formula – rosszul lett programozva vagy betáplálva (R_n)
- Az egyik első, ám annál veszélyesebb bug.)

Szoftverkrízis (példák, adatok)

- Transz-szibériai gázvezeték vezérlő szoftvere
- Kanadai fejlesztés
- Szándékos bug – CIA
- Az emberiség történelmének legnagyobb, nem nukleáris eredetű robbanását okozta
- L'affaire Farewell – film is készült belőle

Szoftverkrízis (példák, adatok)

- 1985 „Therac-25 medical accelerator”
- Radioaktív (besugárzásos) alapon működő kórházi készülék: gyógyítás és terápia
- „Race Condition Bug” (Wiki)
- Legkevesebb öt páciens halt meg a túlzottan erős elektronbombázás miatt

Szoftverkrízis (példák, adatok)

- 1988 - Robert T. Morris, és a világ első számítógépes férgé (Cornell Univ.)
- Morris féreg az ARPAnet ellen
- 6000 gép, köztük kormányzati és állami szerverek megbénítása
- Egy bugra (buffer overflow) épült

Szoftverkrízis (példák, adatok)

- Kerberos véletlenszám-generátor. 1988 és 1996 között használta a Kerberos
- '80-as évek közepén kifejlesztett egyik legszélesebb körben elterjedt általános célú hitelesítési protokoll
- A véletlenszám-generátor hibája miatt a rendszer nagyon egyszerűen kizárható volt
- A hibát csak 1996-ban javították ki, dokumentált kárról nincs tudomás

Szoftverkrízis (példák, adatok)

- 1995, az AT&T hálózata. A történelem egyik legnevezetesebb „bugja”. A telefontársaság új szoftvereket telepített, majd a távolsági hívásokat kezelő vadonatúj rendszer szupergépei végleg lefagytak, amikor egy kódcsomagot kaptak a többi géptől. A dologban az a vicces, hogy ez az üzenet akkor került elküldésre, ha az egyik gép lefagyott... Ekkor szólt a másiknak, hogy vegye át a feladatokat, azonban maga az üzenet volt az, amitől lefagyott... Így megint ment tovább az üzenet, a végtelenségig.

Szoftverkrízis (példák, adatok)

- 1990 január 15-én, 114 gép fagyott le sorban, kb. hetvenezer ember nem tudott távolsági hívásokat bonyolítani. Hogyan oldották meg végül? Nos, az AT&T alig kilenc óra leforgása alatt javította a hibát, mégpedig úgy, hogy visszatelepítették a rendszer előző verzióját. A régi, jól bevált módszer...

Szoftverkrízis (példák, adatok)

- Az Intel Pentium hiba
- Az új chip lebegőpontos számítási műveletekben nem remekelt
- 475 millió dollárjába került az Intelnek a mintegy ötmillió hibás chip visszahívása.

Szoftverkrízis (példák, adatok)

- Ping of Death 1995-96
- Egy megfelelő ping paranccsal a neten keresztül kékhalált lehetett okozni a megtámadott Windowsos (néha Mac és Linux) gépben
- A bug elég mélyen volt, magában az IP-csomagellenőrző kódban rejtőzött

Szoftverkrízis (példák, adatok)

- Az Ariane 5 Flight 501 rövid repülése (37 sec).
1996
- A rakéta a start után letért a kijelölt pályáról, majd az önmegsemmisítő szerkezet beindult és tette a dolgát
- 370 millió dolláros kár, egy apró szoftverhiba miatt: a 64 bites lebegőpontos számokat 16 bites integerre átszámoló kódrészlet hibás volt.

Szoftverkrízis (példák, adatok)

- Millenium (Y2K) probléma
- Fel lehetett rá készülni
- Komoly problémákat nem okozott
- Hollywood

Szoftverkrízis (példák, adatok)

- Taurus (Transfer and Automated Registration of Uncertificated Stock)
- Londoni Tőzsde 1993 – az addigi 2 hetes regisztrációs periódust 2-3 napra csökkentette volna.
- Eleve rossz utat választottak akkor, amikor 1988 szeptemberében úgy döntöttek, nem hoznak létre egy központi számítógépes regisztert a részvény-tranzakciók követésére

Szoftverkrízis (példák, adatok)

- Taurus (Transfer and Automated Registration of Uncertificated Stock)
- A széles körű tesztelés januári beindításának idejére viszont bebizonyosodott, hogy a rendszer így kezelhetetlenül bonyolult.
- A tőzsde 75 millió fontot fektetett a rendszerbe, míg a tőzsdeügynökségek, illetve a tőzsdén jegyzett nagyvállalatok további mintegy 350 millió fontot költöttek arra, hogy majdan csatlakozzanak a rendszerhez.
-

Szoftverkrízis (példák, adatok)

- London Ambulance CAD (Computer assisted dispatch) system 1992 október 26- 27,
- Becslések szerint 20-30 beteg halálát okozhatta a rendszer miatti késlekedés
- <http://www.lond.ambulance.freeuk.com/cad.html>

Szoftverkrízis (példák, adatok)

- Denver Baggage Handling System

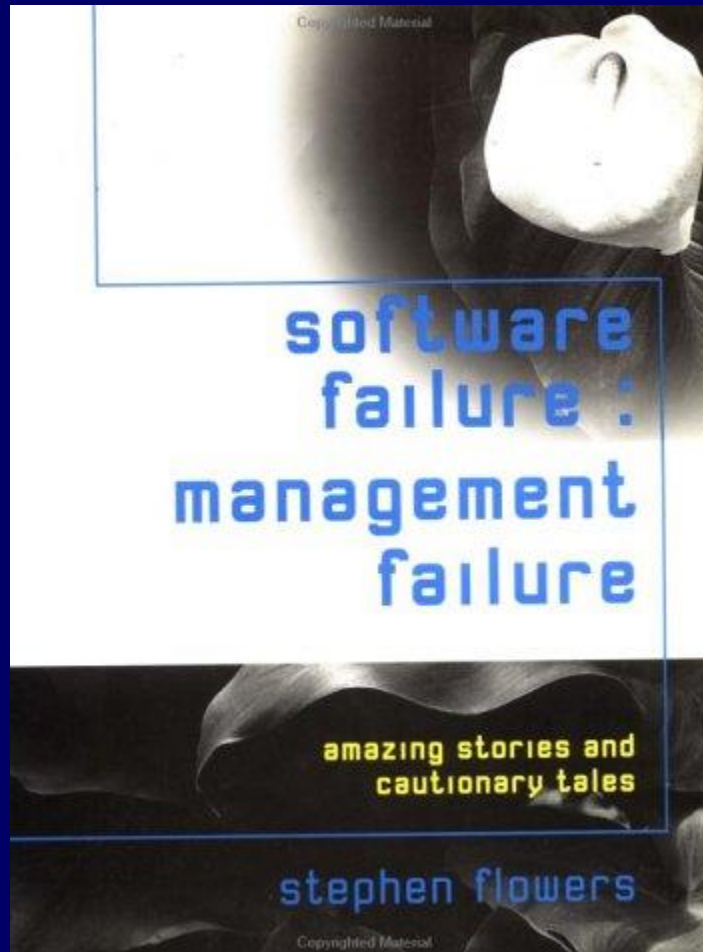
http://calleam.com/WTPF/?page_id=2086



Szoftverkrízis (példák, adatok)

- Denver Baggage Handling System
- Az új repülőtér 16 hónapig nem tudott működni a hiba miatt
- Fő oka a rendszer bonyolultságának alulbecslése volt
- További 560M \$-t tett hozzá a repülőtér költségeihez.
- „Iskolapélda lett”

Szoftverkrízis (példák, adatok)



Alapfogalmak 1

- *RENDSZER: egymással kapcsolatban lévő elemek körülhatárolt csoportja*
 - Szerkezete van
 - statikus: hogyan kapcsolódnak az elemek
 - dinamikus: időben hogyan kapcsolódnak a folyamatok
 - Rendszer környezete, határai
 - Erőforrások
 - Input, output
 - Vezérlés, irányítás

Alapfogalmak 2

- *Adat*: egymással kapcsolatban nem álló, önmagukban nem értelmezett tények vagy elképzelések rögzített formájú megjelenítése, amely alkalmas arra, hogy egy folyamat során műveleteket végezzünk rajta.
- *Információ*: az az ISMERET, melyet az adatok összegyűjtése, szervezése és elemzése által valamilyen logikai vagy matematikai folyamat eredményeképpen kapunk.

Alapfogalmak 3

- *Szervezet*: egy olyan rend, melynek emberek és intézmények alávetik magukat, amikor valamilyen feladatot el kell végezniük.
Szervezetet kell létrehozni, ha
 - Mindig sok az elvégzendő feladat
 - A feladatok ismétlődnek => azonos feladatot azonos módon kell elvégezni
 - Több résztvevő => mindenki tudja a feladatát
- *Szervezeti felépítés*
 - a feladatok,
 - a felelősség és
 - a jogkör helyekhez rendelése
- *Megjegyzés*: nagyvállalatnál a helyeket
 - csoportokká
 - osztályokká
 - főosztályokká fogják össze és a vezetés alá rendelik

Alapfogalmak 4

- *Munkafolyamat*: valamilyen tevékenységsorozat, amit különböző helyeken végeznek, meghatározott sorrendben és valamilyen cél elérésének érdekében.
- *Munkafolyamat –szervezés*: feladata meghatározni
 - milyen lépésekben
 - mely helyeken
 - milyen sorrendben kell elvégezni a munkát, és ehhez
 - milyen eszközöket kell használni
- *Az információs folyamatszervezés feladata*
 - az egyes tevékenységekhez szükséges adatok meghatározása,
 - az adatok tárolásának meghatározása
 - az adatok továbbításának meghatározása
 - az adatfeldolgozás módjának meghatározása
 - kiírandó adatok meghatározása
- *Szervezés*: olyan tevékenység, amely a struktúrát (szervezeti felépítést) és a folyamatokat meghatározza, bevezeti, (esetleg) megváltoztatja

Alapfogalmak 5

- *Szervező*: speciális tevékenységet végző munkatárs, aki az előbb említett feladatokat elvégzi.
- *Rendszerszervező*: meghatározza, ill. módosítja az információs folyamatokat, meghatározza az ehhez szükséges segédeszközöket és kidolgozza a munka irányelveit
- *Feladatai*
 - leírja és elemzi a meglévő folyamatokat és eljárásokat
 - alternatívákat dolgoz ki a gazdaságos adatfeldolgozás és a folyamatok hatékonysága érdekében
 - együttműködve a programozókkal meghatározza a számítógépes eljárásokat és az azokhoz szükséges programokat
 - meghatározza a számítógépes eljárásokhoz szükséges berendezéseket
 - bevezeti a számítógépes rendszert,
 - betanítás
 - próbaüzem
- *Folyamatszervező*: a fentiek közül részfeladato(ka)t old meg

Alapfogalmak 6

- *A számítógépes információs rendszer célja: a nagy mennyiségű adaterőforrást feldolgozva **gazdasági eredményt** hozzon a rendszer üzemeltetője (felhasználója) számára.*
- Az információs rendszer fejlesztésének lépései
 - Érdemes-e változtatni? (átvilágítás)
 - Mit kell tartalmaznia a rendszernek? (elemzés)
 - Hogyan valósítjuk meg a rendszert? (tervezés)
 - A rendszer megvalósítása (számítógépes rendszer leírása, létrehozása, tesztelése)
 - A rendszer bevezetése
 - Értékelés
- *A szervezési módszertan feladata: összefoglalja azokat az általános elveket, amelyek meghatározzák a szervezők szemléletmódját és módszereit*
Megjegyzés: nincs általánosan elfogadott módszertan
- Példák
 - SSADM – Structured System Analysis and Design Method
 - RUP – Rational Unified Process

Alapfogalmak 7

- *Programtervező feladatai*
 - az eljárások részekre bontása
 - az egyes programok sorrendjének meghatározása
 - programok együttműködésének biztosítása (közös a programozóval és a rendszerszervezővel)
- *Programozó: a rendszerszervező tevékenységét programokká ülteti át. Egy eljárást több program valósít meg.*
Feladatai
 - programok megírása (kódolás)
 - tesztelés (modul, integrált)
 - kezelési utasítás készítése
 - programdokumentáció készítése

Módszertanok nélkül

- Személyfüggő fejlesztések
- Örökös viták
- Dokumentációs gondok
- Karbantartási gondok
- Munkahelyváltás nehézségei
- Vezetési nehézségek (projektszervezési módszertan)

A biztos bukásra ítélt projekt életciklusa

- Iszonyú lelkesedés
- Kiábrándulás
- Kétségbeesés
- Zűrzavar
- Bűnbakkeresés
- Az ártatlanok megbüntetése
- A kívülállók megbüntetése
- Célok meghatározása!!!! Itt kell kezdeni!!!!

Információs rendszerek jelenleg is érvényes gondjai

1. Hosszú, a tervezettet meghaladó fejlesztési idő
 - Nem normázható tevékenységek
 - A projekt megkezdése előtt nem világos az elérendő cél
2. Magas fejlesztési költségek, ezzel szemben kevés kimutatható eredmény
 - Elhúzódó fejlesztés
 - A várható eredmények számszerűsítése nehéz
3. Az elkészült rendszerek nehezen módosíthatók, rugalmatlanok
 - Hiányos dokumentáció
 - Rendszer funkcióinak helytelen megválasztása
 - Rossz programtervezés

Információs rendszerek jelenleg is érvényes gondjai

4. Kielégítetlen felhasználói követelmények

- Követelmények nem megfelelő leírása a felhasználói oldalon

5. Gyakori a redundáns adattárolás

- Azt jelzi, hogy baj van a rendszer integráltságával

6. Új eszközökön régi rendszer valósul meg

- A régi rendszer megoldásait automatikusan átviszik az újra, az új eszközök sajátosságait figyelmen kívül hagyják

Módszertanok

■ Módszertannal

- Technologizált fejlesztés (Engineering)
- Áttekinthetőség
- Szabványos dokumentáció
- Projektvezetési támogatás
- Csereszabátosság
- CASE

Ellenállás a szabványos módszertanok bevezetésével szemben

Módszertanok – strukturált

- A strukturált módszertanok jellemzői
 - Termékszemlélet
 - Minőségkezelés (ISO, TQM)
 - Életcikluson alapuló technológia (megvalósíthatóság - elemzés – tervezés – kivitelezés – bevezetés)
 - Projektvezetési támogatás
 - Testre szabhatóság
 - IR: Felhasználó – Folyamatok - Adatok
 - Elemzés és tervezés szétválasztása

Módszertanok – áttekintés 4

- A strukturált módszertanok jellemzői
 - „Felülről lefelé” szemlélet az elemzésben
 - „Alulról felfelé” szemlélet a tervezésben
 - Logikai és fizikai szemlélet szétválasztása
 - Számítógépes támogatás

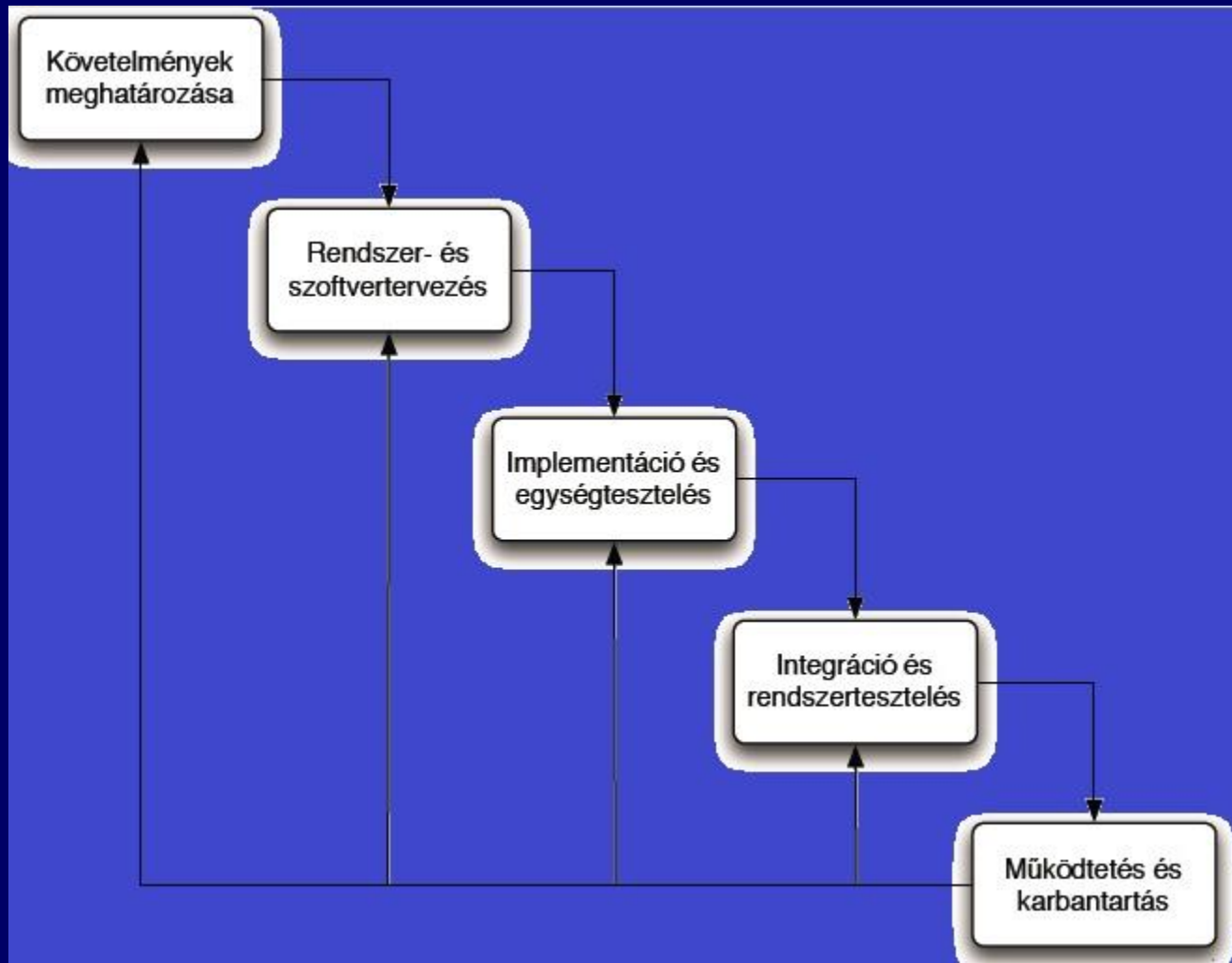
Életciklus-modellek: Vízesés (waterfall)

- 1970 W.W. Royce, első modell
- Elkülönülő fázisokra bont:
 - Követelmény-meghatározás
 - Követelmény-elemzés
 - Rendszertervezés
 - Implementáció, egységteszt
 - Integráció, rendszerteszt
 - Működtetés, karbantartás
- Máig használják, sok más modellnek ez az alapja

Vízesés modell



Vízesés modell



A fázisokról röviden

- Követelmények: A felhasználó igényeinek analízálása, a szoftverrel szemben támasztott követel-mények összegyűjtése (*Requirements Specification Document*)
- Tervezés:
 - System Design: hardver, szoftver környezet, architektúra (blokkok, interfészek) (*System Architecture Document*)
 - Software Design: a rendszerarchitektúra fő szoftver blokkjainak továbbbontása modulokra (*Software Design Document*)
- Implementáció, kódolás: A rendszer egységeinek kódolása
- Verifikáció: modulok egyesítése, tesztelése
- Karbantartás: átadás a felhasználónak, esetleges módosítások

Életciklus-modellek: Vízesés 2

- Feltételezi, hogy az implementáció előtt ismertek a követelmények, ezek jól leírhatóak
- A követelmények nem tartalmaznak feloldatlan, nagykockázatú következményeket
- A követelmények természete nem változik
- A kulcsfelhasználók
- A megfelelő implementációs architektúra mindenki számára jól van értelmezve
- Lineárisan megy végig az életcikluson. Van elég idő a soros haladáshoz

A vízesés-modell előnyei 1

- Nemcsak a szoftverfejlesztésnél használható, hanem egyéb végfelhasználói projektekénél is.
- A jól érthető és kevésbé komplex projektek esetén szabályozottan rögzíti a komplexitást, és jól megbirkózik azzal.
- Könnyű megérteni.
- Egyfázisú fejlesztéseknél egyszerű a használata.
- Strukturáltságot biztosít még a kevésbé képzett fejlesztők számára is.
- Biztosítja a követelmények rögzítését, és azok nem is változnak a fejlesztés során.

A vízesés-modell előnyei 2

- Meghatározott sablont biztosít arra vonatkozóan, hogy mely módszereket használjuk az analízis, tervezés, kódolás, tesztelés és üzemeltetés során.
- Ha jól alkalmazzuk, a hibák már valamely korai fázisban kiderülnek, amikor javításuk még lehetséges, és kevésbé költségigényes.
- A projekt menedzser számára könnyű a tervezés és a szereplők kiválasztása.
- Ha valaki készen van az adott fázisban rá kiosztott munkával, másik projekten dolgozhat, míg a többiek is elérik a fázis lezárásához szükséges állapotot.
- A mérföldkövei könnyen érthetőek.
- Könnyű ellenőrizni a projekt aktuális állapotát.

A vízesés-modell hátrányai 1

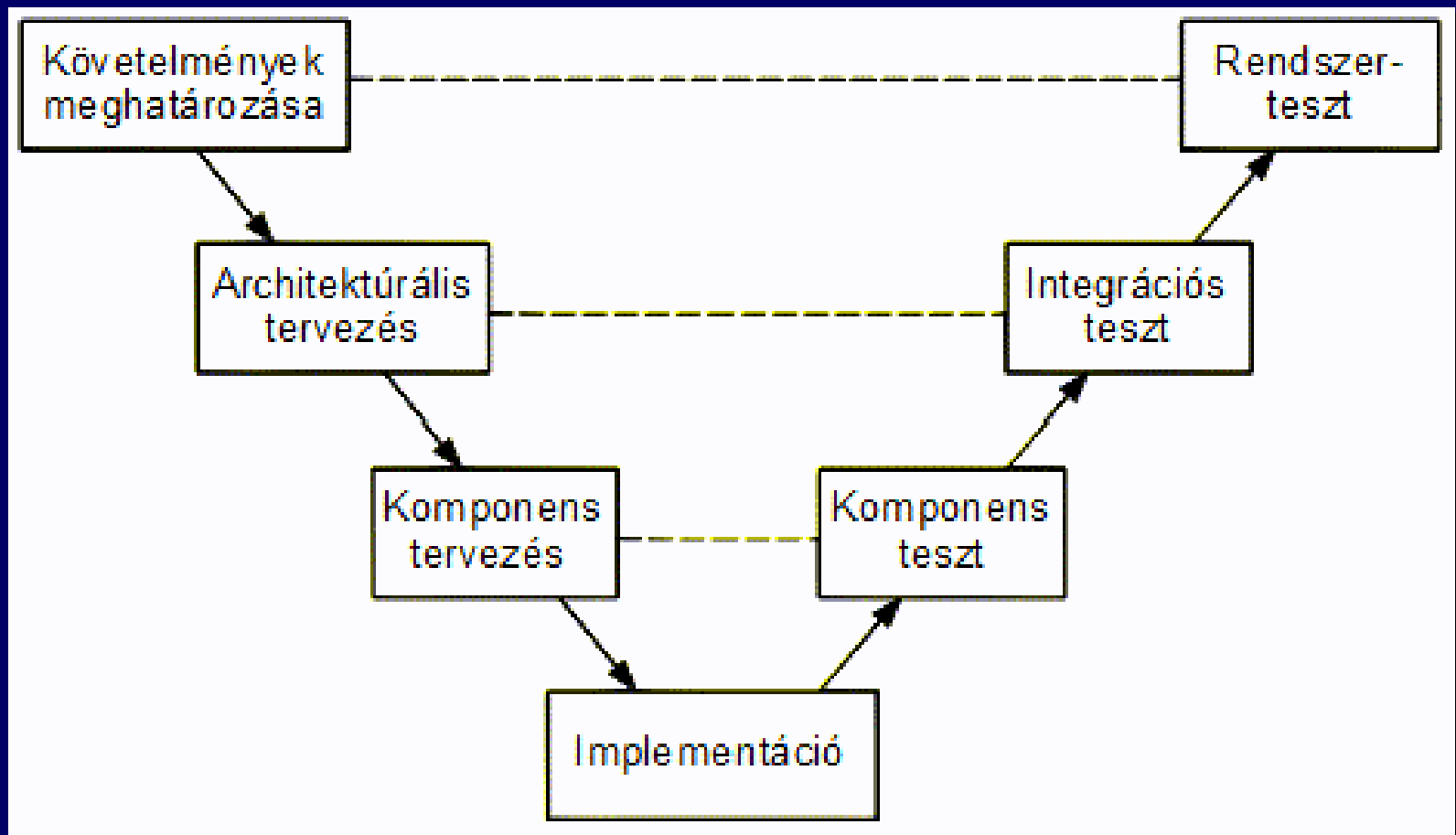
- Lineáris, így nehéz a visszalépés a felmerül problémák megoldására, és ez jelentősen megnöveli a javítás mind költség-, mind időigényét.
- Nem kezeli a szoftverfejlesztésben elterjedt iterációkat.
- Nincs összhangban a szoftverfejlesztés problémamegoldó természetével.
- Az integráció a teljes folyamat végén, egyben, robbanásszerűen történik. A korábban fel nem fedezett hibák ilyenkor hirtelen, együttesen jelennek meg, így felderítésük és javításuk egyaránt nehezebb feladat.
- A megrendelő csak a folyamat végén láthatja a rendszert, menet közben nincs lehetősége véleményezni azt.
- A minőség szintén csak a folyamat utolsó fázisában mérhető.

A vízesés-modell hátrányai 2

- Minden egyes fázis az előző fázis teljes befejezésére épít, ezzel jelentősen megnő a kockázat.
- A fejlesztés során a követelmények nem módosíthatók, hiszen már az életciklus elején befagyasztjuk őket.
- Már a fejlesztés kezdetén ismernünk kell valamennyi követelményt, azok későbbi módosítására vagy bővítésére nincs lehetőség.
- Elképzelhető, hogy bár a végtermék megfelel valamennyi specifikációnak, mégsem működik (pl. mert az elvárásokban vannak ellentmondások).
- Dokumentum-vezérelt, és túlzott dokumentálási követelményeket állít fel.
- Az egész szoftvertermék egy időben készül, nincs lehetőség kisebb részekre osztására.

V-modell

- Ezt a változatát a vízésésmodellnek a német védelmi minisztérium fejlesztette ki, és tette a német hadsereg szoftver-fejlesztési modell szabványává 1992-ben.
- Minden tervezési fázisok van egy párja a tesztelési fázisok között
- A tesztelés hangsúlyozása



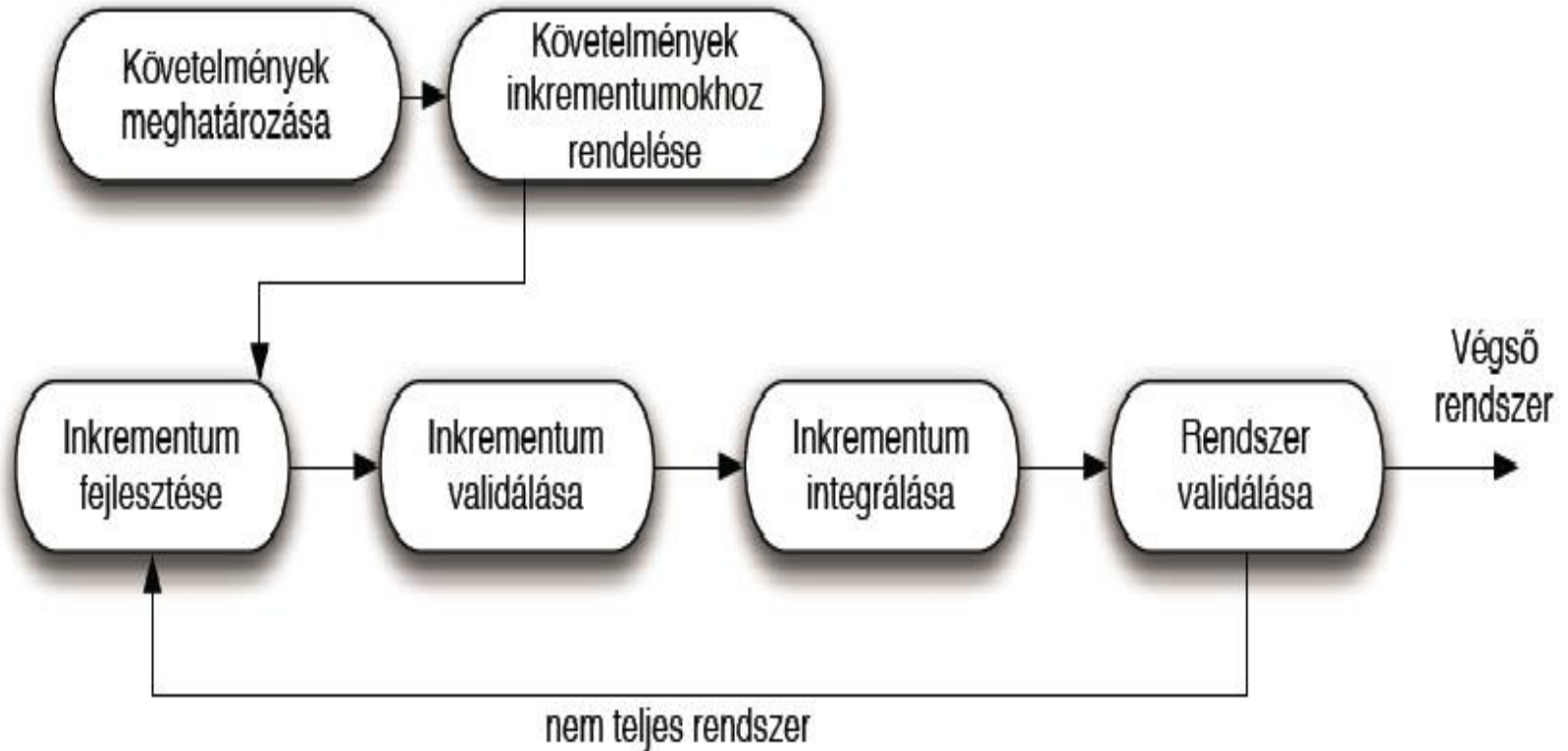
Inkrementális fejlesztés

- Mills, 1980
- A vízesés modellt követi egy korai tervezési fázisig, majd
- Kis részekben implementálja az architektúrát
- Különösen jó rosszul definiált projekteknel
- Több szervezést igényel
- Megköveteli a moduláris tervezést
- Rögzíti az architektúrát
- Az implementálandó részeket gondosan kell kiválasztani (hogyan önállóan működjenek)
- Nem mindig megfelelő

Inkrementális fejlesztés menete

- Követelmények prioritás alapú felosztása
- Fejlesztés felbontása „inkrementumokra”
 - Követelmények befagyasztása
 - Kívánt inkrementum létrehozása
 - Első inkrementum – core product

Inkrementális fejlesztés folyamata



Inkrementális fejlesztés folyamata

- A fejlesztés indításakor a megrendelő vázlatosan közli a követelményeit, illetve meghatározzák mely követelmények a fontosabbak, s melyek kevésbé fontosak.
- Amikor egy inkremens elkészül, azt integrálni kell a már elkészült rendszerbe, s utána azt a felhasználó használatba is veheti. Az üzemeltetéssel kapcsolatos tapasztalatok visszacsatolhatóak a további inkremensek tervezésébe.

Inkrementális fejlesztés folyamata

- Az inkrementális fejlesztés előnye, hogy
 - A megrendelőnek nem kell megvárnia a teljes rendszer elkészültét, hanem már az első inkremens átadása után használhatja a rendszer legfontosabb szolgáltatásait.
 - A korábban kifejlesztett inkremensek tekinthetők prototípusnak, a használatuk során szerzett tapasztalatok beépíthetők a fejlesztés folyamatába.
 - Csökkenti a kockázatot. Az egyes inkremensekben ugyan lehetnek hibák, azonban nem valószínű, hogy ezek az egész projekt kudarcát okozzák.
 - A magas prioritású inkremenseket szállítjuk le először, így azok lesznek a legjobban kitesztelve.
- Az egyes inkremenseknek viszonylag kicsinyeknek kell lenniük (Sommerville szerint 20 ezer sornál kisebbek), minden egyes inkremensnek szolgáltatni kell valamilyen rendszerfunkciót. Időnként nehézségeket okozhat a rendszer ilyen méretű elemekre való particionálása.

Inkrementális fejlesztés előnyei

- Tolerálja a követelmények megváltozását
- Csökkenti a kockázatot
- Biztonságosabb, hibatűrőbb alkalmazást eredményez
- Lehetővé teszi a résztvevők folyamatos tanulását, a módszertan finomítását
- Lerövidíti a fejlesztés időtartamát
- Az iteratív fejlesztés terv szerű

Egyéb iterációs módszerek

- ❑ Spirál modell
- ❑ Evolúciós modell
- ❑ Rational Unified Process

Döntési folyamat

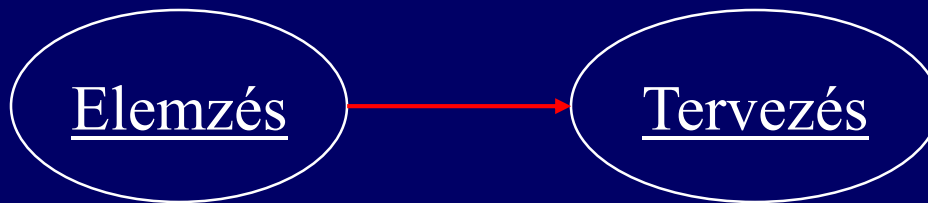
- A problémamegoldás 3 lépését tekinthetjük :

Elemzés

Szervezeti célok
Probléma besorolása
Probléma megfogalmazása
Probléma állításai

Döntési folyamat

- A problémamegoldás 3 lépését tekinthetjük :



Modellalkotás
Választási kritériumok
Kivitelezés előkészítése
Eszközök összegyűjtése

Döntési folyamat

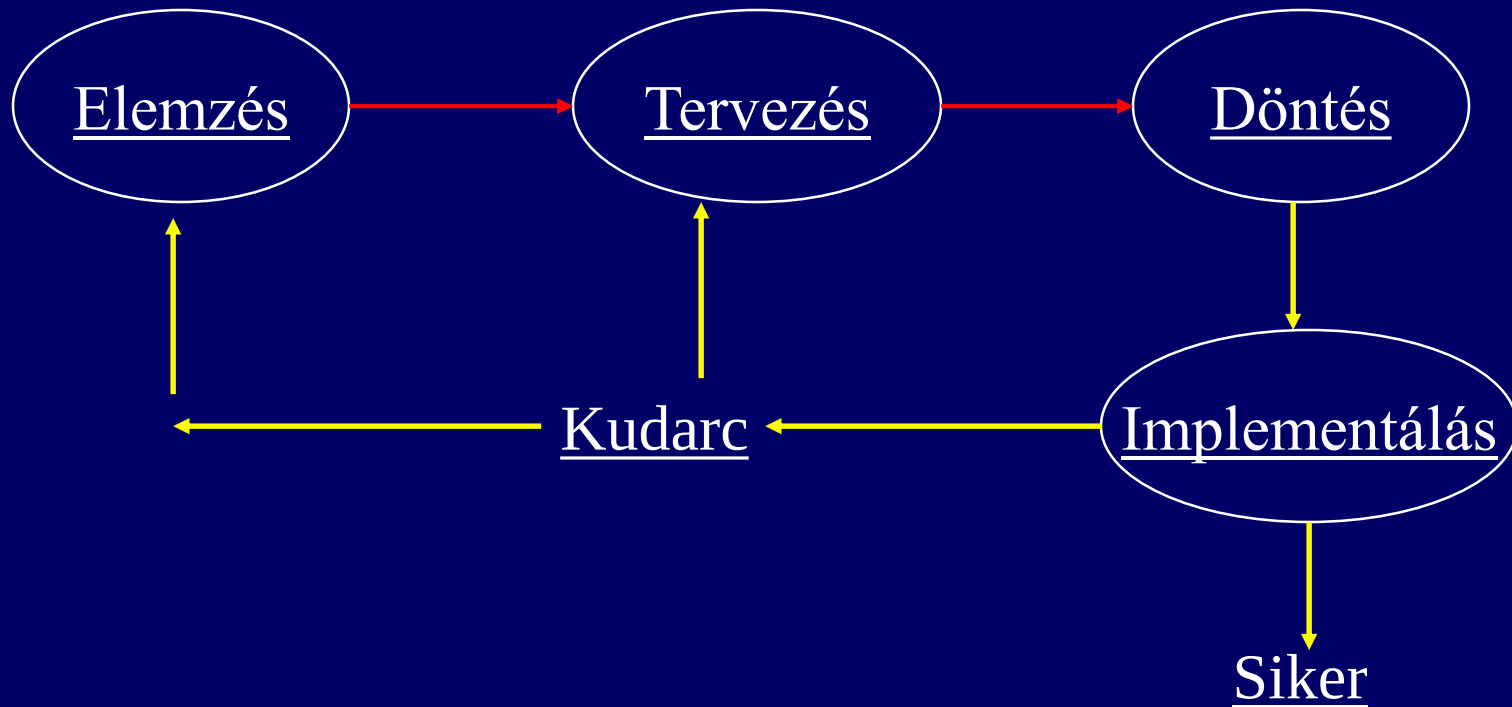
- A problémamegoldás 3 lépését tekinthetjük :



Modell megoldása
Jó (legjobb) alternatíva kiválasztása
Érzékenységi analízis
Implementálás megtervezése

Döntési folyamat

- A problémamegoldás 3 lépését tekinthetjük :



Döntési folyamat – más közelítés

1. Az inicializálás

A feladat közvonalazása, a probléma felvetése.

2. Probléma definiálás-modell specifikáció

A számszerűsíthető adatok ismeretében (a változók) a matematikai modell felállítása, a módszer kiválasztása.

3. Optimalizálás

A célfüggvénnyel felállított matematikai modell számítógépes megoldása egy döntéstámogató program segítségével

4. Értékelés, javaslattétel

Különböző paraméterekre elvégezve a számítást a kapott eredmény alapján a javaslat megfogalmazása.

5. Döntés realizálás

A döntés megvalósítása.

Döntési problémák csoportjai

- Struktúrált: Mindhárom fázis programozható
- Félig struktúrált: Nem minden fázis programozható
- Nem struktúrált: Egyik fázis sem programozható

Példák döntési feladatokra

	Operatív	Manager	Stratégiai
Struktúrált	Számlázás	Költségvetés elemzése	Terjesztési hálózat
Félig struktúrált	Készletgazdálkodás	Költségvetés előkészítése	Hosszabb távú előrejelzések
Nem struktúrált	Folyóirat címlapterv	Szoftverbeszerzés	K+F tervezés

Az üzleti informatika alapismeretei



Információs rendszerek

- Transaction Processing System (TPS)
- Management Information System (MIS)
- Office Automatization System (OAS)
- Decision Support System (DSS)
- Expert System (ES)
- Executive Information System (EIS)

Az IT és a vállalati információs rendszerek fejlődése

1950-	Elektronikus adatfeldolgozó rendszerek (EDP)
1960-	tranzakció feldolgozó rendszerek (TPS)
	Vezetői információs rendszerek (MIS)
1970-	döntéstámogató rendszerek (DSS, GDSS)
	irodaautomatizálás
1980-	felsővezetői információs rendszerek (EIS)
	szakértői rendszerek (ES)
	vállalatirányítási információs rendszerek (ERP)
1990-	Stratégiai információs rendszerek
	üzleti intelligencia (BIS)
2000-	e-business

Gazdasági rendszerek csoportosítása

- IT rendszerek korszakai alapján
- szervezeti struktúra szerint
- működési terület szerint

IT rendszerek korszakai

1960- Adatfeldolgozó rendszerek

cél: működés hatékonyságának javítása, az élő munka kiváltása

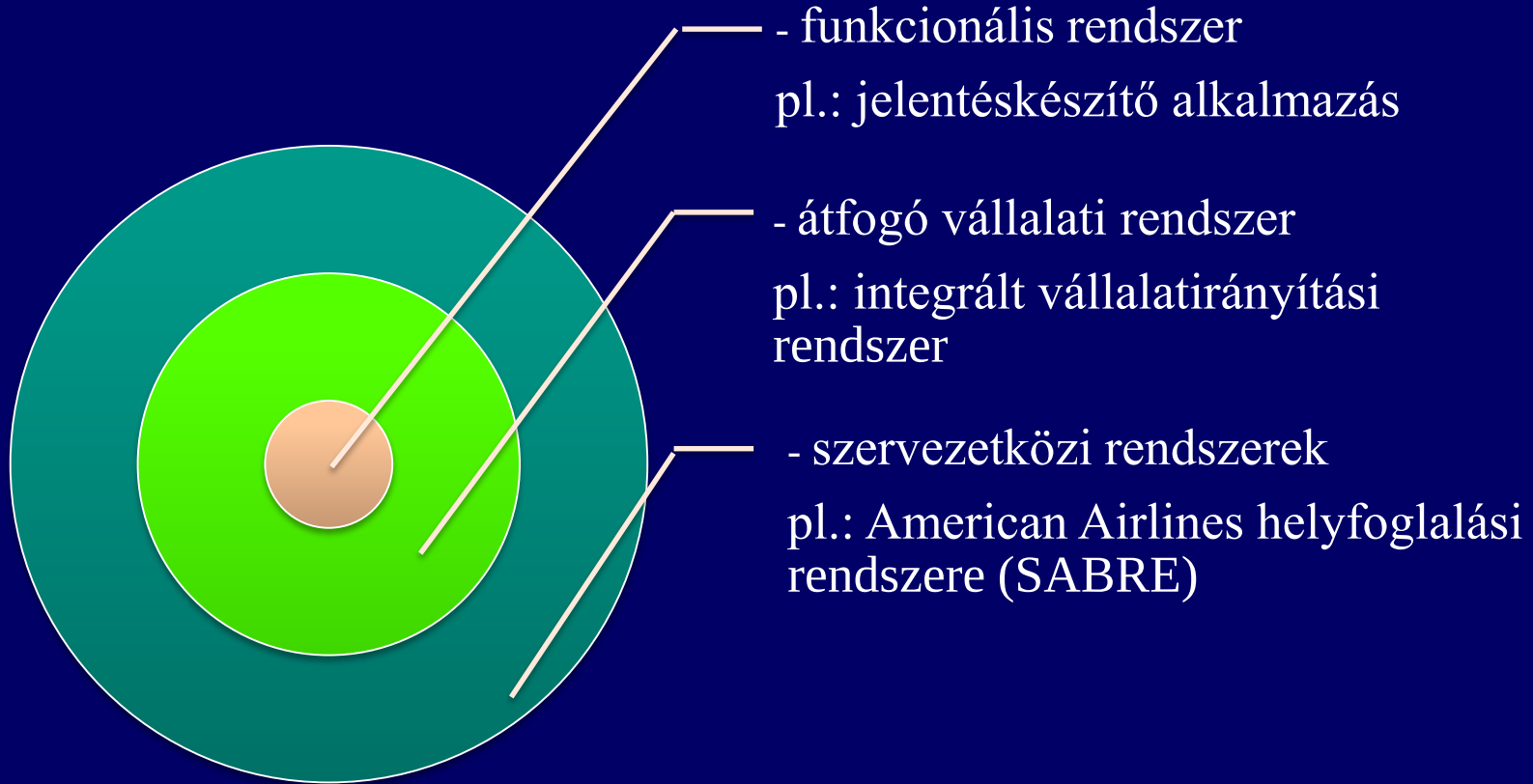
IT
fejlődése

1970- Vezetői információs rendszerek
cél: vezetői információszolgáltatás

Üzleti
fejlődése

1980- Stratégiai információs
rendszerek (főként DSS, EIS)
cél: versenyképesség javítás, a munka optimalizálása

Szervezeti struktúra szerint



Működési terület szerint

TPS

OAS

ISS

KMS

MIS

EIS

DSS

GSS

ES
(KBS)

CAD/CAM

üzleti
intelligencia

TPS

- Transaction Processing System (TPS)
- tranzakciókezelő rendszer
- tranzakció: egy ügymenettel kapcsolatos esemény
- a napi ügymenethez tartozó, ismétlődő feladatok ellátása



TPS

TPS vázlatos felépítése



TPS jellemzői

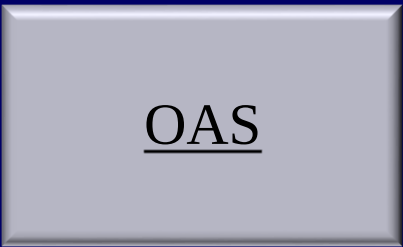
- fontosabb funkcionális területei: a marketing, a gyártás, a pénzügy, a könyvelés, az emberi erőforrások.
- tranzakció feldolgozásának módjai:
 - kötegelt, pl.: fizetések számítása
 - eseményvezérelt on-line vagy valós idejű, pl.: helyfoglaló rendszerek
 - hibrid, pl.: ATM (azonosítás, pénzkiadás valós időben, de a számla módosítása később történik meg)



TPS

OAS

- Office Automation System (OAS)
- iroda-automatizáló rendszer
- feladata támogatni:
 - az irodai dokumentumkezelési munkákat
 - a csoportmunkát
 - a kommunikációt



OAS

IOS

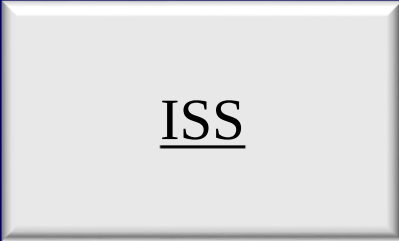
- iroda feladata: gazdasági és üzleti folyamatok regisztrálása, feljegyzése és az így létrejött dokumentumok tárolása, feldolgozása és nyilvántartása
- az integrált irodai rendszer (IOS) feladatai:
 - dokumentumfeldolgozás
 - dokumentumarchiválás
 - munkafolyamat-szabályozás
 - csoportmunka-támogatás
 - kommunikációtámogatás



OAS

ISS

- Intelligent Support System (ISS)
- intelligens támogató rendszer
- a szellemi munkát végző alkalmazottak támogatása



ISS

KMS

- Knowledge Management System (KMS)
- tudásmenedzsment rendszer
- a tudással, mint vállalati erőforrással kapcsolatos tevékenységek támogatása



KMS

MIS

- VIR (vállalati információs rendszer)
 - Management information system (MIS)
 - Komplex rendszerek, a vállalati információs folyamatok lefedésére



MIS

MIS

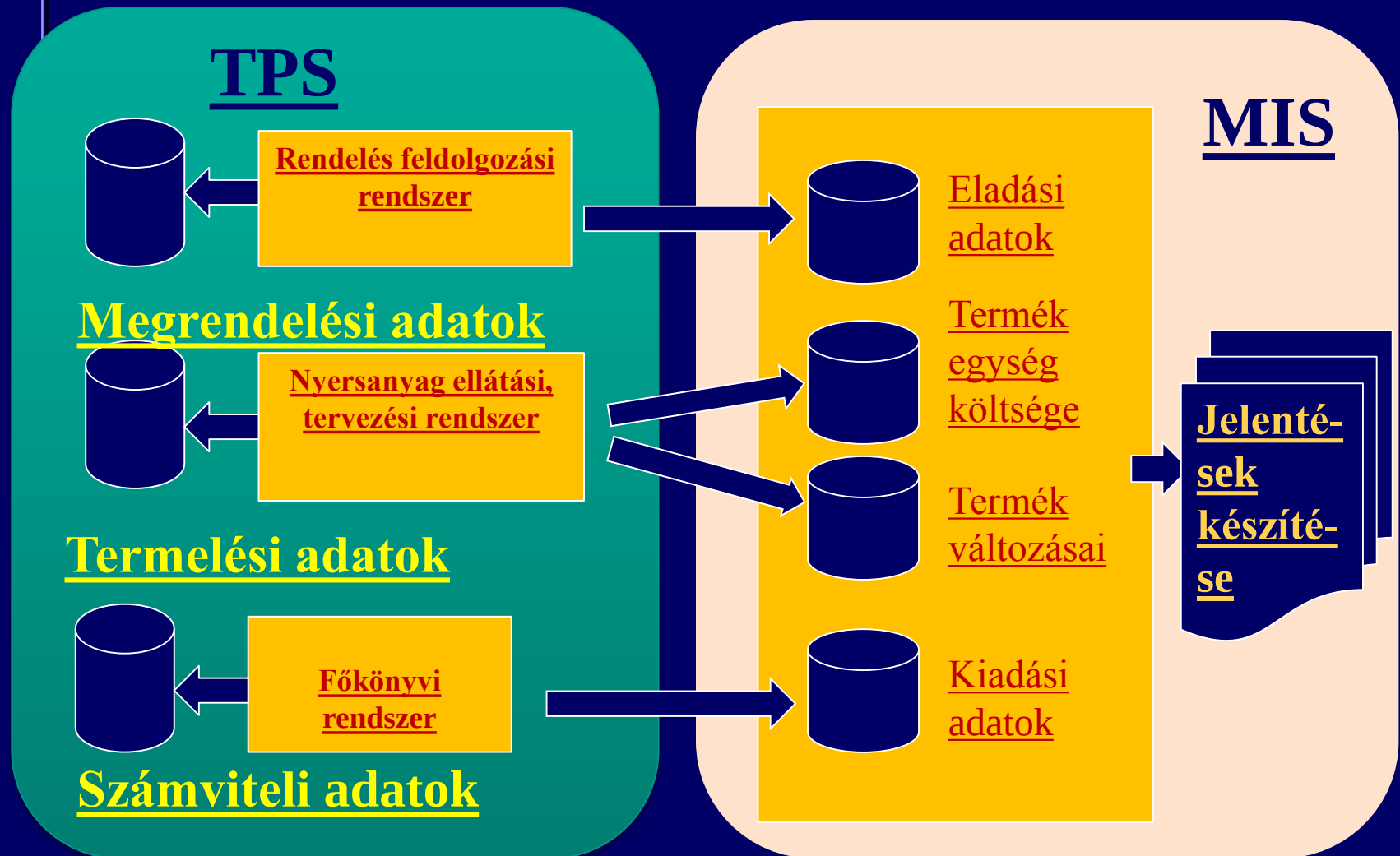
- **rendszeres jelentések készítése**
(napi, heti, havi rendszerességgel)
- **kivételes, különleges események jelentése**
(időnként, vagy bekövetkezésekor)
- **igény szerinti jelentések**

Jellemzői

- jól meghatározott, strukturált problémák megoldásához nyújt segítséget
- főként operatív, esetleg taktikai szinten hatékony

MIS

A TPS és a MIS kapcsolata



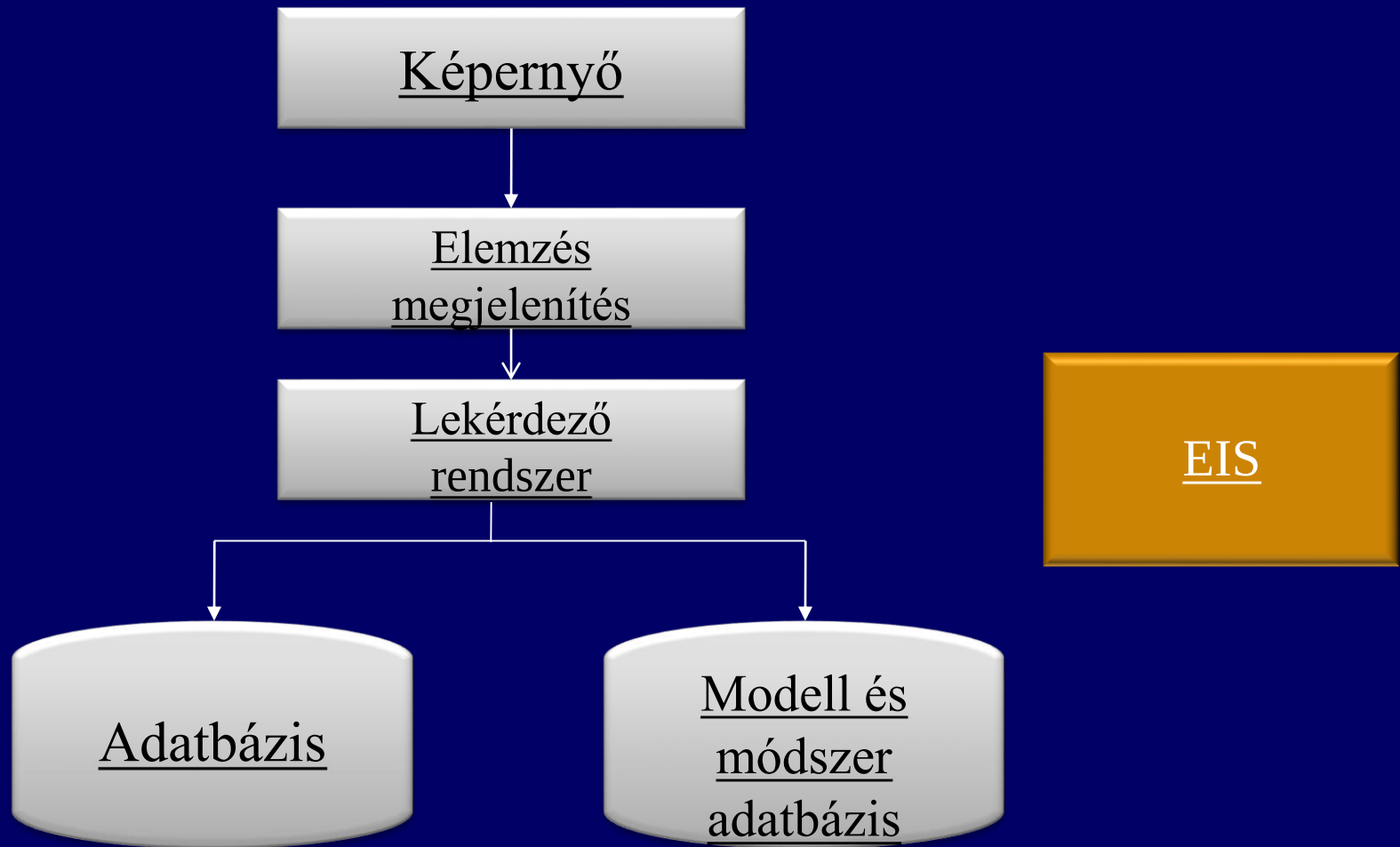
EIS

- Executive Information System (EIS),
Felsővezetői információs rendszerek
- Felső szintű vezetők részére
 - döntést nem hoz, csak annak megtalálásához látja el a menedzsert információval
 - tömör, összegzett információk, grafikus elemmel kiegészítve
 - eltérések kiemelése
 - „prezentációs” jellegű kimenet
 - múlt–jelen–jövő (modellezés)
- Kész szoftver – egyedi fejlesztés



EIS

Az EIS modell



ESS

- Executive Support System (ESS),
Felsővezetői döntéseket támogató rendszerek
- Bizonyos funkciókkal kibővített EIS-t takar.
- A hagyományos EIS funkciókon túl tartalmaz
 - analitikus,
 - modellezési lehetőségeket ad (DSS),
 - esetleg mesterséges intelligenciát és szakértői képességeket is jelenthet (ES),
 - valamint kommunikációs lehetőségeket is támogat (GDSS)



EIS

DSS

- Decision Support Systems (DSS)
- Döntéstámogató rendszerek
- vezetők döntéshozatalát támogatják
 - csoportos döntéstámogató rendszer (GDSS)
 - szakértői rendszerek (KBS)



DSS

DSS jellemzők

- alapvető cél: a döntések minőségének javítása
- Számítógépes eljárásokkal, döntési modellekkel rendelkeznek
- több adatbázist használhatnak
- interaktivitást, ad hoc lekérdezést tesz lehetővé
- jellemzője a felhasználóbarát felület és a rugalmasság
- MIS „utódok”



DSS

DSS erőforrásai

- **hardver:** interaktivitás lehetősége; nagy számítási teljesítmény
- **szoftver:** DSS generátor – DSS szoftver
- **adatbázisok:**
Hozzáférési lehetőség $\Leftrightarrow$ adatvédelem, adatbiztonság
- **modellbázisok:**
Matematikai, statisztikai, egyéb modellek
- **emberi erőforrások:** fejlesztők, üzemeltető felhasználók, menedzserek



DSS

DSS felépítése

1. Kommunikációs alrendszer

- hardver
- szoftver

2. Modellező alrendszer

- A modell nem más, mint valamilyen fizikailag létező objektumnak az absztrakciója.
- A modell tehát mindig egy jelenséget, egy tárgyat vagy egy tevékenységet reprezentál.

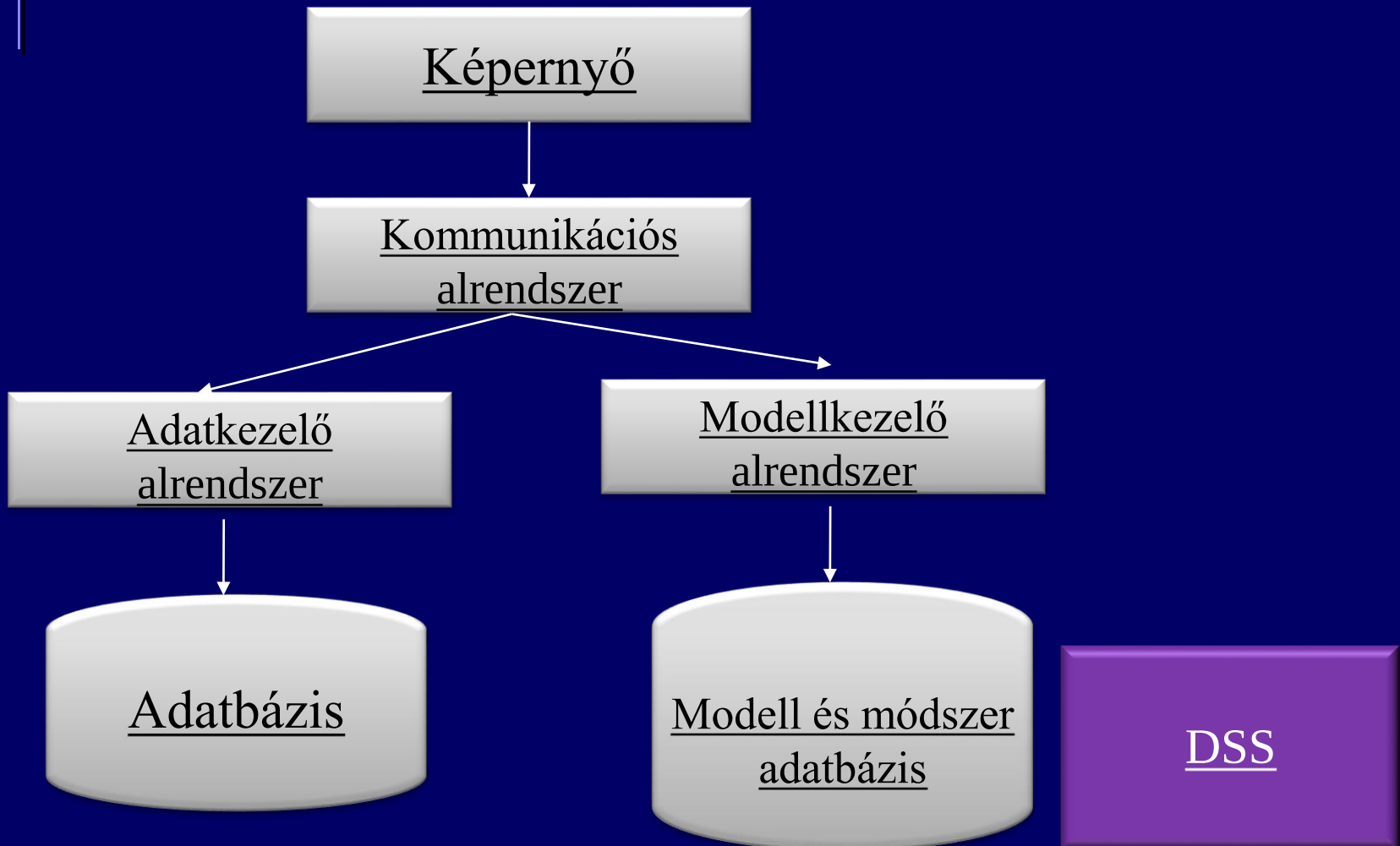
3. Adatbáziskezelő-alrendszer

- adatok tárolása
- adatok rendelkezésre állása
- adatok elérése
- adatelérés kontrollja

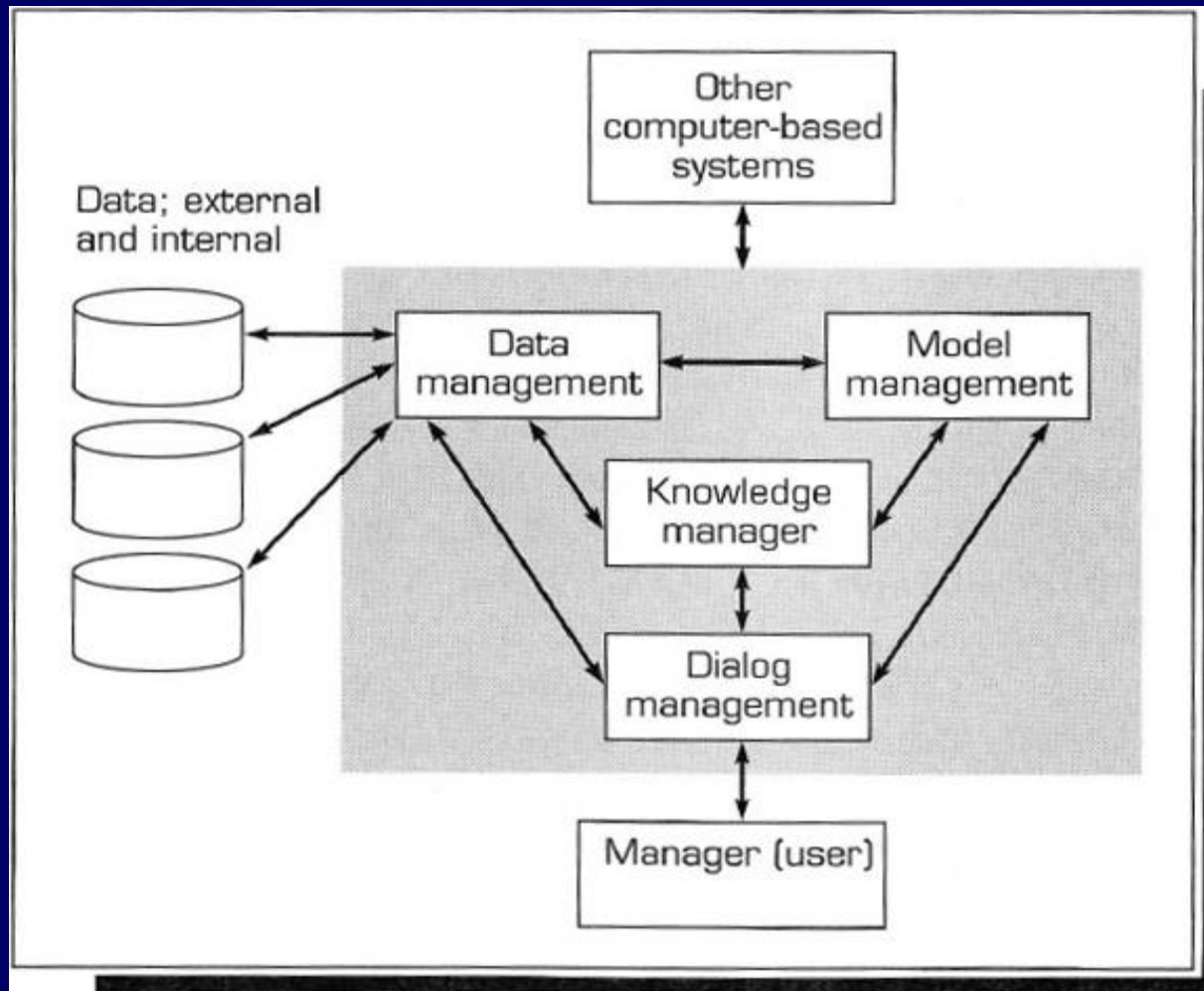


DSS

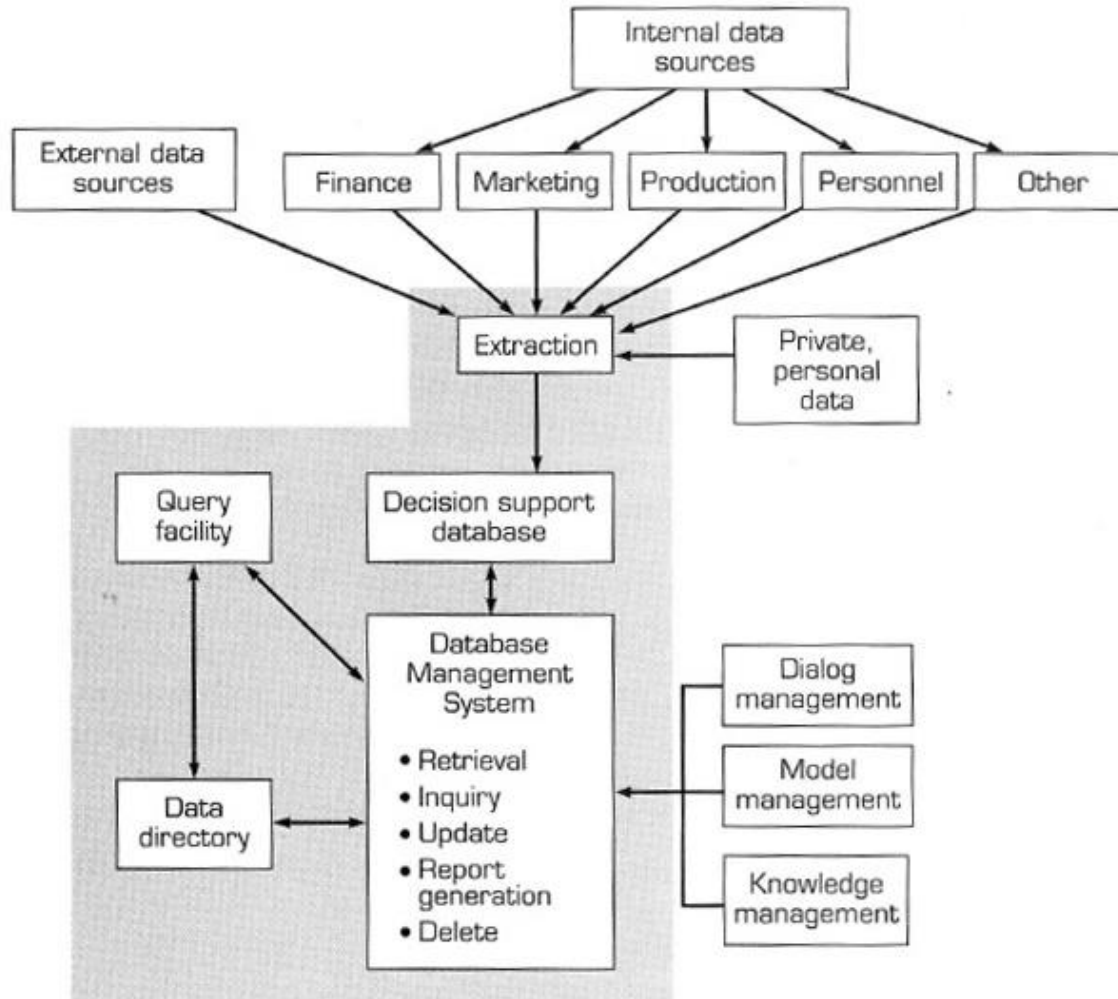
A DSS modell



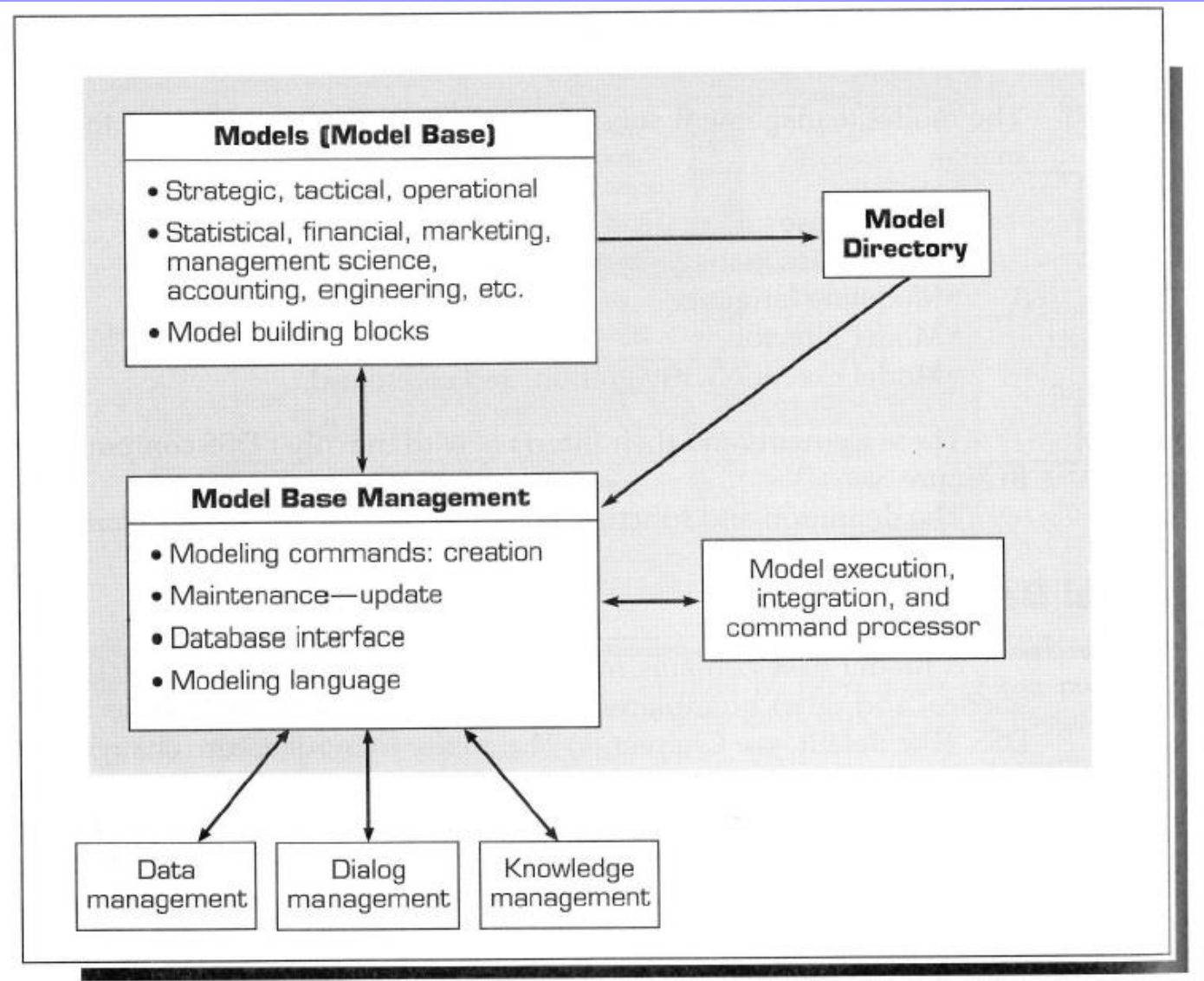
A DSS modell - Áttekintő



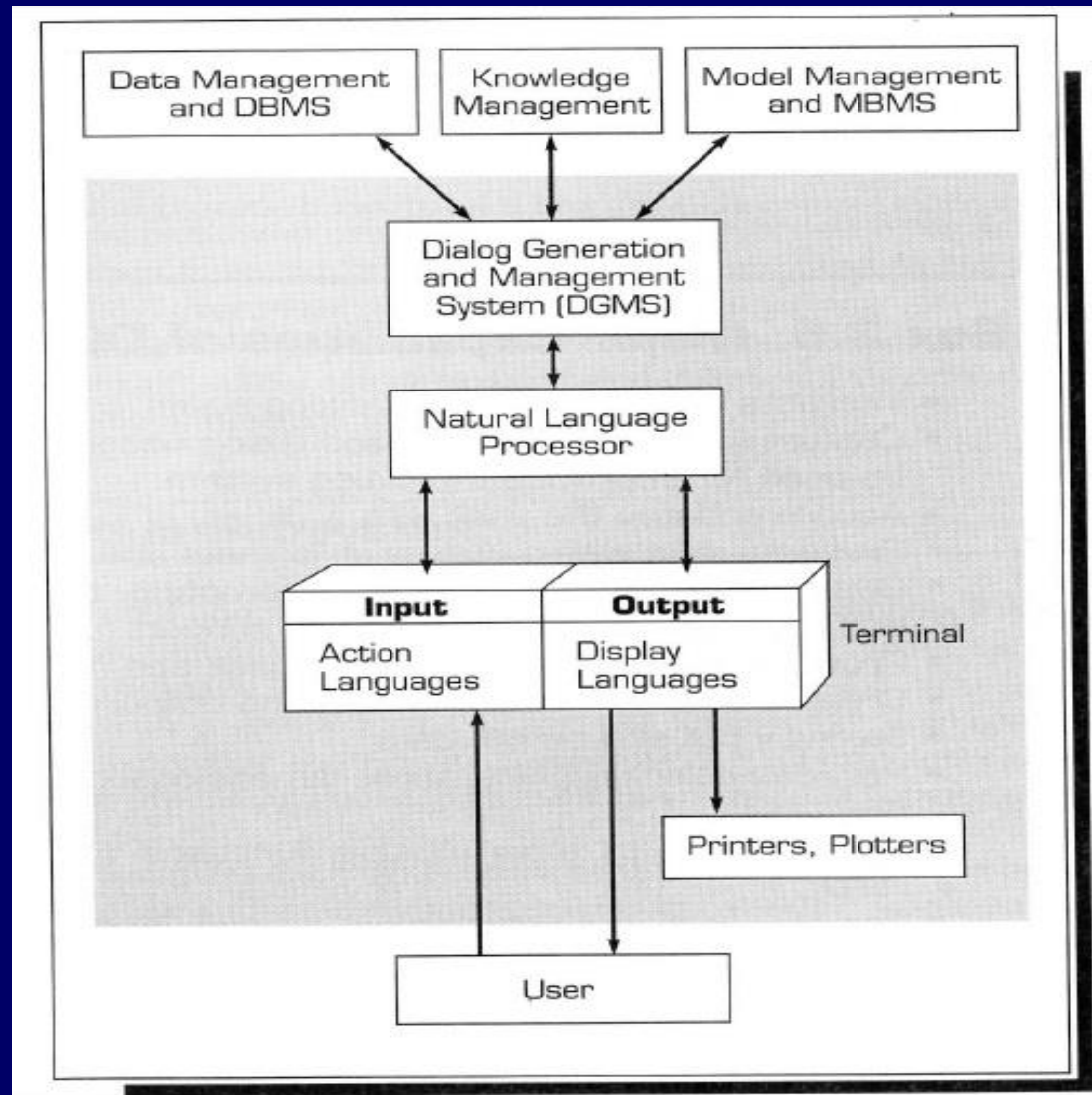
A DSS modell - Adatkezelő



A DSS modell - Modellkezelő



A DSS modell - Jelentéskezelő



DSS szoftverek típusai

- **Modell-alapú DSS**

A rendszer különböző módszereket kínál egy döntési modell összeállítására.

- **Adat-alapú DSS**

A rendszer adatok, legtöbbször idősorok kezelésével, elemzésével támogatja a döntéshozatalt.

- **Dokumentum-alapú DSS**

Ez a típus különböző dokumentumok feldolgozásánál, a dokumentumokban való keresési lehetőségekkel nyújt döntéstámogatást.



DSS

DSS szoftverek típusai

- **Tudás-alapú DSS**

A rendszerek a problémát leíró tudás segítségével nyújtanak megoldási módszert.

- **Kommunikáció-alapú DSS**

Ez a típus a kommunikáció és a hálózati technika lehetőségeit használja fel döntéstámogatáshoz (pl. csoportos döntéshozatalhoz).

- **Táblázat vagy Web-alapú DSS**

A rendszerek egy táblázat (pl. Excel), vagy Web felületen keresztül kommunikálnak a felhasználóval. A háttérből pedig, pl. egy modell-alapú DSS szoftver nyújtja a szolgáltatásokat.



DSS

GSS vagy GDSS

- Group (Decision) Support System (GDSS),
Csoportos döntéstámogató rendszerek
- A DSS továbbfejlesztése, nem egyszemélyes,
hanem egy kisebb csoport által közösen meghozott
döntéseket támogat.
- Nagy hangsúly van a kommunikáción: e-mail,
közös hozzáférésű állományok, videokonferencia
lehetősége.

GSS

GSS típusok

- Egy időben – egy helyen
 - hálózatra kötött személyi számítógépek
- Egy időben – különböző helyeken
 - videokonferencia
- Különböző időben – azonos helyen
 - hirdetőtáblák
- Különböző időben – különböző helyeken
 - e-mail



GSS

Információs rendszerek jellemzése - Összefoglalás

■ TPS

Korai programcsomag, nagy mennyiségű számítás, listakészítés stb. végrehajtására.

Feladathoz készített adathalmaz

Numerikus számítások

Nincs döntéshozási/támogatási kapacitás

Információs rendszerek jellemzése - Összefoglalás

■ MIS

Adatbázis alapú vállalati információs rendszer.

Szakemberek által kezelt adatbázis

Numerikus számítások

Nincs döntéshozási kapacitás, kismértékű támogatás

Strukturált feladatok

Információs rendszerek jellemzése - Összefoglalás

■ DSS

Beépített modellező rendszer

Felhasználóbarát felület, DBMS

Numerikus számítások

Erőteljes döntéstámogatási kapacitás

Félig strukturált feladatok

Információs rendszerek jellemzése - Összefoglalás

■ ES

Adott szűk alkalmazási terület
problémamegoldására

A területet lefedő tudásbázis

Numerikus, szimbolikus számítások

Teljes döntéshozási kapacitás

Nem strukturált feladatok

Információs rendszerek jellemzése - Összefoglalás

■ EIS

Felső vezetést kiszolgáló rendszer.

Teljes belső, valamint külső adatforrások

Numerikus, szimbolikus számítások

Naprakészség, nyomkövetés, 'Drill Down'

Információs rendszer

- Adatok (információk)
 - Rájuk vonatkozó információs események
 - Az információkon végrehajtott tevékenységek
 - az előzőekkel kapcsolatos felhasználók és erőforrások
 - valamint a mindezeket szabályozó eljárások
- szervezett együttese

- Az IR nem csak az információk világát öleli fel, hanem a felhasználókat és a fizikai erőforrásokat is.
- Az IR által kiszolgált környezetet valós (alkalmazási) rendszernek nevezzük.
- Tervezés előtt ki kell jelölni az alkalmazási rendszer határait.
- IR nem egyenlő a szoftverrel: szoftver az IR része

Adatok és információk

- **Adat:** a valóság nem értelmezett, de értelmezhető (észlelhető, érzékelhető, felfogható és megérthető) tükörképe (ismeret)
- „Géza”:
 - itt van -> **észlelhető** (vak nem látja),
 - le van írva -> **érzékelhető**
 - ismert abc-vel írtuk le -> **felfogható** (analfabéta észlel, de nem fog fel)
 - nem idegen -> **megérthető**

Adat

- Nem feltétlenül tényszerű, a tényen kívül utalhat elképzelésre, tervre, szándékra, kitalációra
- A megjelenítési forma nem kötött

Adat

- Példa:
„Ez a kocszi zöld”
- 4 dimenzió:
ismerni a szóban forgó tárgyat **általában** (kocsi)
és **konkrétan** (ez)
tulajdonság, jellemző megadása **általában** (szín)
és **konkrétan** (zöld)
- A mindennapi életben a 4 tényezőből több jöhet a szöveggörnyezetből, egy szg-rendszerben meg kell mindet adni.

Adat

Az USA-ban készült egy tartalom alapján lekérdező rendszer. Feltették neki a kérdést, hogy mi köze Marilyn Monroe-nak Ann Arbor városához. A válasz meglepő volt: az alma. A neves színésznő almafából készíttetett bútort. Ann Arbor egyetemén meg volt egy Alma ... kollégium.

Tanulság: mindig gondosan meg kell határozni az adat négy dimenzióját. Egyenlőre nincsenek emberien gondolkodó számítógépek

Adat

- **Szintaktika:** a közlésre vonatkozó jelölési szabályok
- **Szemantika:** a négy dimenzió helyes összefüggéseit szabályozza
- **Pragmatika:** a közölt ismeret gyakorlati használhatósága (lásd információ)
- Egy IR tervezőnek az adatokat a 4 dimenzió szerint kell elrendeznie:
 - nem lépheti át a formális kommunikáció korlátait (jelkészlet, „ennek a kocsinak a színe kerek”)

Információ

- (Az ember által) új ismeretté értelmezett adat
- Az adat személytelen, objektív
- Az információ az adatot fogadó szubjektumához kötődő lényeg
- A rendszerszervező alapvető feladata, hogy ne zúdítsion adattömeget a felhasználóra. Az adatfeldolgozást úgy kell megtervezni, hogy a produkált adatösszefüggéseket a felhasználó könnyen és gyorsan tudja információvá értelmezni.
- Az ismeretet mindig úgy kell tálni, hogy a megcélzott felhasználó által könnyen értelmezhető legyen

Adat vagy információ?

- „A RSZ előadás előadója X.Y”
- Adat, mert nem tartalmaz új ismeretet
- „Az előadónak 123 rendszámú, Z típusú kocsija van”
- Adat, mert nem tartalmaz új ismeretet
- „A termésátlag 35 q/ha ezen a parcellán”
- Információ, új ismeretet tartalmaz

Információs események és tevékenységek

- Azon információs tevékenységek melynek a feladata a felhasználók által igényelt adatösszefüggések biztosítása: **adatfeldolgozás**
- Az információs tevékenységet valamilyen **információs esemény** váltja ki.
- Az esemény valamilyen változáshoz kötődik.
- A jelenségek ún. életciklussal rendelkeznek (születnek, módosulnak, elhalnak), ennek megfelelően kell adatokat létrehozni stb.

Felhasználók, erőforrások

- IR: fejlesztők és végfelhasználók
- A végfelhasználó olyan személy, aki a maga ismeretei szerinti adatösszefüggéseket kíván, melyeket könnyen tud információvá szervezni.

IR-ek vetületei (horizontális bontás)

- System Engineering
- adat-vetület (alap adatszerkezet)
- feldolgozás-vetület (változékonyság, rugalmasság - **logikai adatfüggetlenségre** vigyázni kell)
- környezet-vetület (felhasználók, eszközök képességeivel számolnia kell, eszközváltásra képesnek kell lenni: **fizikai adatfüggetlenség**)

IR-ek vetületei (horizontális bontás)

- System Engineering
- adat-vetület (alap adatszerkezet)
- feldolgozás-vetület (változékonyság, rugalmasság - **logikai adatfüggetlenségre** vigyázni kell)
- környezet-vetület (felhasználók, eszközök képességeivel számolnia kell, eszközváltásra képesnek kell lenni: **fizikai adatfüggetlenség**)

Vetületek példa

- Munkaszüneti nap nem veszünk fel megrendelést
- Adat, Környezeti
- A PC-ket hálózatba kötöttük
- K
- A béreket másodikán utaljuk
- Feldolgozási
- Ehhez az adathoz csak az osztályvezetők vagy magasabb pozícióban lévők férhetnek hozzá
- A, K

Szintek (vertikális)

- Fogalmi szint
(fogalmak az AR jelenségeiről)
Az AR fogalmait és összefüggéseit elemzik
- Fizikai szint
Adatábrázolás, eszközön való elhelyezkedés
program belső szervezése
- Logikai szint
Adatok, események, tevékenységek tartalmát és
összefüggéseit elemzik

Szintek, példa

- A bér nem lehet 57000 forintnál kevesebb
- FOGALMI
- A bér egész típusú adat
- FIZIKAI
- A bér a DOLGOZÓ rekord része
- LOGIKAI
- A bér az alapbér és a prémium összege
- FOGALMI

Módszertani hiányosságok

Eszköz-kötöttség

- Fizikai (vertikális) függőség: A terv tartalmát és a megvalósítás részleteit együtt tervezik,
- a tervező összekapcsolja a megoldandó feladatot a választott megoldási formával, hogy az előbbi elveszti saját jellegét:
- Hordozhatatlan, szuboptimális
- Elemzési hiány – a rendszer elemeit és összefüggéseit lehet és kell elemezni eszköztől függetlenül is
- Munkatöbblet

Tervezési szintek tudatos szétválasztása

Feladat-szemlélet

- A felhasználó igényeiből a *szükséges átalakítások nélkül* vezetik le az IR belső struktúráját
- Azt a jelenséget, hogy a RT belső elemei és általános szerkezete a külső használat jegyeit viseli logikai (horizontális) függőségnek nevezzük
- A rendszer struktúrájára rányomja a képét a bemenet/kimenet.

Logikai függőség problémái

- Egyoldalúság: ha a gépesítés főnöke főkönyvelő, akkor pénzügyi, ha főmérnök, akkor műszaki beállítottságú lesz a rendszer.
- Redundancia: (saját rendszereket épít a raktárgazda) → inkonzisztencia
- Rugalmatlanság: A tervezőnek fel kell készülnie, hogy a az igények rohamosan változni fognak. A felhasználó nem lát általános képet, csak mai összefüggésekben gondolkodik.

A tervezési vetületek tudatos szétválasztása

vetület-orientáció (Adat, feldolgozás, környezet)

- Az alapul választott vetület a tervezési folyamatban és a tervben rányomja bélyegét a többi vetületbe tartozó tényezőre is.
- Például a feldolgozástervből vezetik le az adatszerkezetet, nincs az utóbbinak önálló elemzése

- **Esemény-orientáció:** IR mozgatója az esemény → először a bekövetkező eseményeket kell feltárni, nyomon követni ezekből következnek a tevékenységek és az adatok (de lehetnek ezen túlmenően is adatok). Az IR feladata nem csak a tükrözés.
- Egy esemény több adategységet érintő tevékenységsorozatot is kiválthat.

Vetület-orientáció folyt.

- **Adat-orientáció:** Az adat az IR legstabilabb vetülete, de nem feltétlenül határozza meg a rá vonatkozó események és tevékenységek struktúráit. Feldolgozásra is szükség van, a 'SELECT' nem old meg mindent
- **Tevékenység-orientáció:** A tervező az aktuális információellátási funkcióban gondolkodik (megrendelések feldolgozása). Ez fizikai ill. logikai függőségekhez vezethet, mert a feldolgozáshoz szorosan kapcsolódóan határozzák meg az adatokat

Adatbázis

- Az adatbázis adatok olyan osztott használatú rendszere, melyben a különböző jelenségekre vonatkozó ismeretek egymással alkotott természetes összefüggéseik szerint szervezettek, és amely független az őt feldolgozó programoktól.
- → Eszközfüggetlen (fizikai kezelés részletei nem érdekesek)
- → Összetett, különböző jelenségek összefüggéseit tükrözi
- → adatprogram-független: az adatváltozás nem vált ki, legfeljebb csak minimális, programváltozást

Relációs modell - Tartomány

Definíció

Egy D tartomány (domain) atomi értékek egy halmaza.

Jellemzői:

név

adattípus

formátum

korlátozás

további információk az értelmezéshez

Tartomány

Példák

- Mobiltelefonszámok: 11 decimális számjegy +dd-dd-ddddddd formátumban.
- Személyi számok: 11 decimális számjegy d dddddd dddd formátumban.
- Nevek: tetszőleges hosszúságú karaktersorozat.
- Tömegek: nemnegatív valós szám mértékegységgel (font vagy kg).

Relációséma

Definíció

Relációséma alatt az $R(A_1;A_2;:::;A_n)$ jelölést értjük, ahol R a relációséma neve, $A_1;A_2;:::;A_n$ pedig **attribútumok**.

Minden A_i attribútum egy szerepkör neve, amelyet valamely D tartomány játszik. D -t az A_i attribútum tartományának nevezzük, és $\text{dom}(A_i)$ -vel jelöljük.

Példák

HALLGATÓ(Név, Személyi_szám, Lakcím, Szak, Évfolyam,
Neptun_kód)

TANSZÉK(Tanszék_név, Tanszékvezető_neve, Kar)

AUTÓ(Márka, Típus, Gyártási_év, Motorszám, Rendszám)

Reláció

Definíció

Az $R(A_1; A_2; \dots ; A_n)$ relációséma egy r **relációja** – amit szokás r (R)-rel is jelölni – **elem n-eseknek** egy halmaza:

$$r = \{t_1; t_2; \dots ; t_m\}.$$

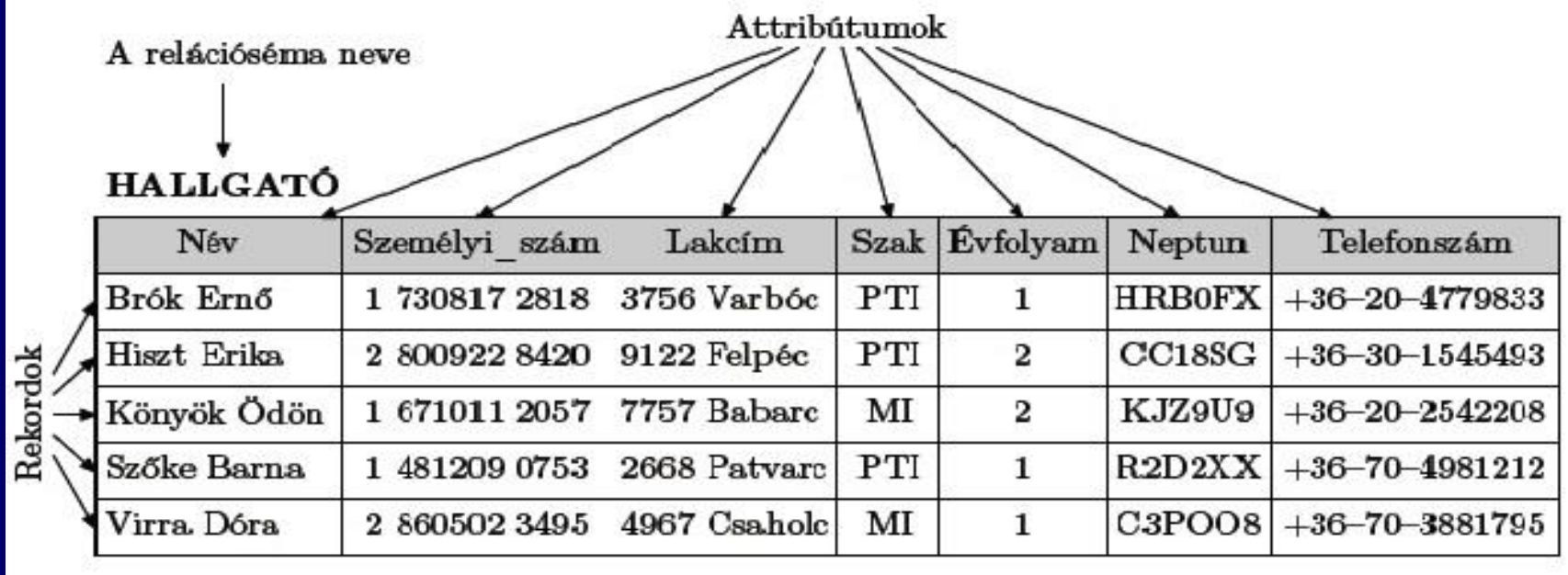
Minden t_i elem n -es ($1 \leq i \leq m$) n darab értéknek egy rendezett listája:

$$t_i = \langle v_1; v_2; \dots ; v_n \rangle,$$

ahol minden v_j érték ($1 \leq j \leq n$) vagy $\text{dom}(A_j)$ -nek az eleme, vagy egy speciális NULL érték.

A definícióban említett elem n -eseket **rekordoknak** is nevezzük.

N-fokú relációk



Megjegyzések

Az összes lehetséges kombináció közül a reláció egy adott pillanatban csak azokat a rekordokat tartalmazza, amelyek a valós világ pillanatnyi állapotát tükrözik: ez a **reláció aktuális állapota**. Ahogyan a valós világ változik, úgy változik a reláció (állapota) is.

A **reláció sémája** – az előzőekkel ellentétben – viszonylag statikus, **nem változik**, néhány ritka esetet leszámítva.

Megszorítások *****

Definíció

A **tartománymegszorítás** kimondja, hogy minden rekordban minden egyes A attribútumhoz tartozó értéknek a $\text{dom}(A)$ tartományból kell származnia, és ezen $\text{dom}(A)$ tartományok minden elemének atomi értéknek kell lennie.

Definíció

Az R relációsémának létezik egy olyan **szuperkulcsnak** nevezett **attribútumhalmaza**, amely olyan tulajdonságú, hogy tekintve R bármelyik r relációját, az adott relációban nincs két olyan rekord, amelynek az értékei azonosak lennének ezen attribútumokra vonatkozóan.

Megszorítások - Kulcs

Definíció

Egy R relációséma K kulcsa R -nek egy olyan superkulcsa, amelyből egy A attribútumot elhagyva, az így kapott K' attribútumhalmaz már nem superkulcsa R -nek.

Egy kulcs kielégíti a következő két feltételt:

- Bármilyen relációt tekintve, a reláció két különböző rekordjának nem lehetnek azonosak a kulcsban szereplő attribútumokhoz tartozó értékei.
- Minimális superkulcs, azaz egy superkulcs, amelyből nem tudunk úgy eltávolítani egyetlen attribútumot sem, hogy az egyediségre vonatkozó feltétel továbbra is fennálljon.

Egy K kulcs egyszerű, ha egyetlen attribútum alkotja, egyébként összetett.

Kulcsjelöltek

Definíció

Egy relációsémának egynél több kulcsa is lehet. Ilyen esetben a kulcsok mindegyikét kulcsjelöltnek hívjuk.

Definíció

A modellező feladata, hogy a kulcsjelöltek közül kiválasszon egyet a relációséma elsődleges kulcsául. Ez a kulcsjelölt lesz az, amelynek az értékeit a relációkban szereplő rekordok azonosítására fogjuk használni. A kulcsmegszorítás szerint a relációsémának mindig rendelkeznie kell elsődleges kulccsal.

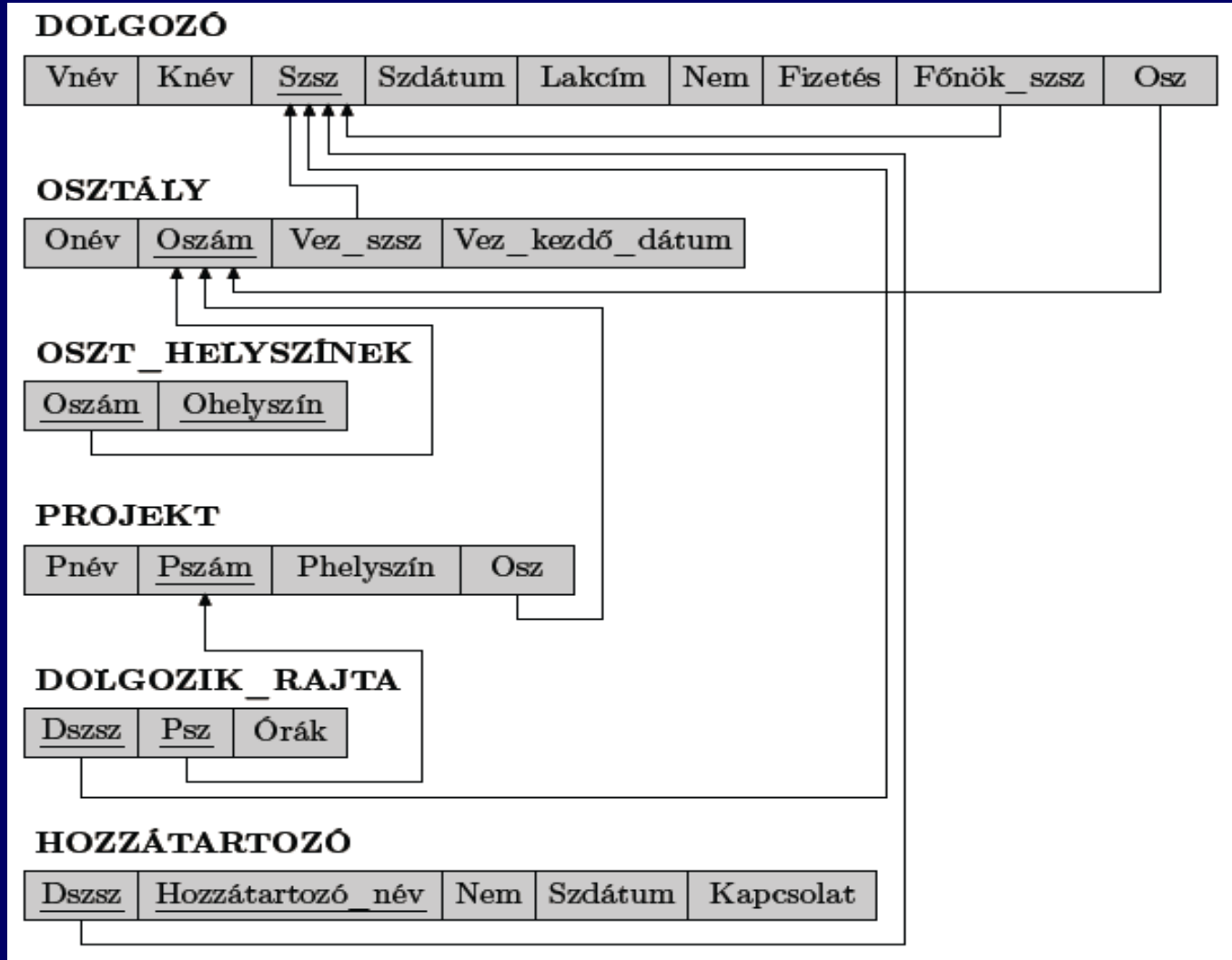
Megjegyzés

Egy relációséma elsődleges kulcsát alkotó attribútumokat aláhúzással szoktuk jelölni. Amikor egy relációsémának több kulcsjelöltje is van, tetszőlegesen lehet közülük elsődleges kulcsot választani.

Relációs adatbázisséma és relációs adatbázis

Egy relációs adatbázis rendszerint számos relációt tartalmaz, a relációkban rekordokkal, amelyek különböző módokon vannak egymással kapcsolatban.

Példa



Példa

DOLGOZO

Vnév	Knév	Szsz	Szdátum	Lakcím	Nem	Fizetés	Főnök_szsz	Ösz
Kovács	László	1 650109 0812	1965. január 9.	4033 Debrecen	F	390000	2 551208 2219	5
Szabó	Mária	2 551208 2219	1955. december 8.	1097 Budapest	N	520000	1 371110 4519	5
Kiss	István	1 680119 6749	1968. január 19.	1172 Budapest	F	325000	1 410620 4902	4
Takács	József	1 410620 4902	1941. június 20.	4027 Debrecen	F	559000	1 371110 4519	4
Horváth	Erzsébet	2 620915 3134	1962. szeptember 15.	1092 Budapest	N	494000	2 551208 2219	5
Toth	János	1 720731 2985	1972. július 31.	6726 Szeged	F	325000	2 551208 2219	5
Fazekas	Ilona	2 690329 1099	1969. március 29.	3535 Miskolc	N	325000	1 410620 4902	4
Nagy	Zoltán	1 371110 4519	1937. november 10.	1061 Budapest	F	715000	NULL	1

OSZTÁLY

Onév	Osztám	Vez_szsz	Vez_kzdő_dátum
Kutatás	5	2 551208 2219	1988. május 22.
Humán erőforrás	4	2 690329 1099	1995. január 1.
Központ	1	1 371110 4519	1981. június 19.

DOLGOZIK_RAJTA

Dszsz	Psz	Órák
1 650109 0812	1	32.5
1 650109 0812	2	7.5
2 620915 3134	3	40.0
1 720731 2985	1	20.0
1 720731 2985	2	20.0
2 551208 2219	2	10.0
2 551208 2219	3	10.0
2 551208 2219	10	10.0
2 551208 2219	20	10.0
1 680119 6749	30	30.0
1 680119 6749	10	10.0
2 690329 1099	10	35.0
2 690329 1099	30	5.0
1 410620 4902	30	20.0
1 410620 4902	20	15.0
1 371110 4519	20	NULL

OSZT_HELYSZÍNEK

Osztám	Ohelyszín
1	Budapest
4	Kecskemét
5	Vác
5	Tiszaftired
5	Budapest

PROJEKT

Pnév	Pszám	Phelyszín	Ösz
X termék	1	Vác	5
Y termék	2	Tiszaftired	5
Z termék	3	Budapest	5
Komputerizáció	10	Kecskemét	4
Reorganizáció	20	Budapest	1
Új fejlesztések	30	Kecskemét	4

HOZZÁTARTOZÓ

Dszsz	Hozzá tartozó_név	Nem	Szdátum	Kapcsolat
2 551208 2219	Anna	N	1986. április 5.	lánya
2 551208 2219	Bence	F	1983. október 25.	fia
2 551208 2219	Mate	F	1968. május 3.	házastársa
1 410620 4902	Viktória	N	1942. február 28.	házastársa
1 650109 0812	Balázs	F	1988. január 4.	fia
1 650109 0812	Anna	N	1988. december 30.	lánya
1 650109 0812	Reka	N	1967. május 5.	házastársa

Funkcionális függés

A funkcionális függés egy olyan megszorítás, amely az adatbázis két attribútumhalmaza között áll fenn.

Egy R relációsémában X akkor és csak akkor határozza meg funkcionálisan Y -t, ha valahányszor $r(R)$ két rekordja megegyezik az X értékeken, szükségszerűen megegyezik az Y értékeken is.

Abból, hogy egy R -re előírt megszorítás szerint bármely $r(R)$ relációpéldányban nem szerepelhet több, mint egy rekord egy adott X értékkel – azaz X egy superkulcsa R -nek –, következik $X \rightarrow Y$ az R attribútumainak bármely Y részhalmazára (mivel a kulcsmegszorításból következik, hogy egyetlen legális $r(R)$ állapotban sem lehet két olyan rekord, amelyeknek azonosak lennének az X értékeik).

Funkcionális függés tulajdonságai

- 1 a projektivitás szabálya: Ha X tartalmazza Y -t, akkor $X \rightarrow Y$.
- 2 az augmentivitás szabálya: ha $X \rightarrow Y$, akkor $XZ \rightarrow YZ$.
- 3 a tranzitivitás szabálya: ha $X \rightarrow Y$; $Y \rightarrow Z$, akkor $X \rightarrow Z$.
- 4 a dekompozíció szabálya: ha $X \rightarrow YZ$, akkor $X \rightarrow Y$.
- 5 az additivitás szabálya: ha $X \rightarrow Y$; $X \rightarrow Z$, akkor $X \rightarrow YZ$.
- 6 a pszeudotranzitivitás szabálya: ha $X \rightarrow Y$; $WY \rightarrow Z$, akkor $WX \rightarrow Z$:

Egy $X \rightarrow Y$ funkcionális függés triviális, ha X része Y -nak egyébként nemtriviális.

Megjegyzés

Bár $X \rightarrow A$ és $X \rightarrow B$ az additivitás szabálya miatt implikálja $X \rightarrow AB$ -t, azonban sem $X \rightarrow A$ -ból, sem $Y \rightarrow B$ -ből nem következik, hogy $XY \rightarrow AB$. Mint ahogy $XY \rightarrow A$ sem implikálja szükségképpen sem $X \rightarrow A$ -t, sem $Y \rightarrow A$ -t.

Funkcionális függés tulajdonságai

- 1 A projektivitás szabálya szerint egy attribútumhalmaz mindig meghatározza önmagát, vagy saját maga bármilyen részalmazát.
- 2 Az augmentivitás szabálya szerint egy funkcionális függés mindkét oldalának ugyanazzal az attribútumhalmazzal történő bővítése újabb érvényes funkcionális függést eredményez.
- 3 A tranzitivitás szabálya szerint a funkcionális függések tranzitívak.
- 4 A dekompozíció szabálya azt mondja, hogy egy funkcionális függés jobb oldaláról eltávolíthatunk attribútumokat.
- 5 Az additivitás szabálya szerint funkcionális függések egy $\{X \rightarrow A_1; X \rightarrow A_2; \dots; X \rightarrow A_n\}$ halmazát összevonhatjuk egyetlen $X \rightarrow \{A_1; A_2; \dots; A_n\}$ funkcionális függéssé.

Armstrong-axiómák

William Ward Armstrong 1974-ben bizonyította be, hogy a **projektivitás**, az **augmentivitás** és a **tranzitivitás** szabálya együtt helyes és teljes.

Helyesség alatt azt értjük, hogy ha adott egy R relációsémán fennálló funkcionális függéseknek egy F halmaza, akkor bármilyen függés, amely levezethető F -ből a három szabály segítségével, fenn fog állni R minden olyan r relációjában, amely kielégíti az F -beli függéseket.

Teljesség alatt azt értjük, hogy a három szabályt mindaddig ismételten alkalmazva, míg már nem kapunk újabb függéseket, előállítható az F -ből levezethető összes lehetséges függés teljes halmaza. Más szavakkal, F -ből kiindulva kizárólag a három szabály alkalmazásával meghatározható az F^+ függések halmaza, amit F lezártjának hívunk.

A reláció attribútumainak szemantikája

1. (nem hivatalos) irányelv

Egy reláció minden egyes rekordja egy egyedet vagy kapcsolat-előfordulást reprezentáljon. (Az egyes relációkra és azok attribútumaira külön-külön vonatkozik.)

- Különböző egyedek (DOLGOZÓ-k, OSZTÁLY-ok, PROJEKT-ek) attribútumai nem keverendők egyazon relációban.
- Más egyedekre való hivatkozás csak külső kulcsok használatával történjen.
- Az egyedekre és a kapcsolatokra vonatkozó attribútumokat a lehető legjobban el kell különíteni egymástól.

Tömören

Olyan sémát kell tervezni, ami könnyen magyarázható relációról relációra. Az attribútumok szemantikájának könnyen értelmezhetőnek kell lennie.

Redundáns információk a rekordokban

Amikor az információt **redundánsan** tároljuk, az tárhelyet pazarol,
karbantartási anomáliákat okoz, amelyek lehetnek
beszúrási anomáliák,
törlési anomáliák és
módosítási anomáliák.

2. (nem hivatalos) irányelv

Olyan sémát tervezzünk, amelyben nem jelennek meg beszúrási, törlési és módosítási anomáliák.

Ha mégis előfordulnak anomáliák, akkor jegyezzük fel azokat, hogy az alkalmazások számításba vehessék őket.

Relációsémák normalizációja

Definíció

A normalizáció (dekompozíció) az a folyamat, amelynek során szétbontjuk a nem kielégítő, „rossz” relációsémákat úgy, hogy az attribútumaikat több kisebb relációsémába helyezzük át.

Definíció

A normálforma a relációsémák kulcsai és a bennük fennálló funkcionális függések segítségével megfogalmazott feltétel, amellyel megállapítható, hogy a relációséma egy adott normálformában van-e.

Nem normalizált relációk (0NF)

(a)

OSZTÁLY

Onév	<u>Oszám</u>	Ovez_szsz	Ohelyszínek
			

(b)

OSZTÁLY

Onév	<u>Oszám</u>	Ovez_szsz	Ohelyszínek
Kutatás	5	2 551208 2219	{ Vác, Tiszafüred, Budapest }
Humán erőforrás	4	2 690329 1099	{ Kecskemét }
Központ	1	1 371110 4518	{ Budapest }

Normalizált relációk (1NF)

(c)

OSZTÁLY

Onév	<u>Oszám</u>	Ovez _szsz	<u>Ohelyszín</u>
Kutatás	5	2 551208 2219	Vác
Kutatás	5	2 551208 2219	Tiszafüred
Kutatás	5	2 551208 2219	Budapest
Humán erőforrás	4	2 690329 1099	Kecskemét
Központ	1	1 371110 4518	Budapest

Speciális függések

	1	2	3	4	5	6	7	8
$A \rightarrow B$	P	P	P	P	D	D	D	D
$B \rightarrow C$	P	P	D	D	P	P	D	D
$A \rightarrow C$	P	D	P	D	P	D	P	D
Triviális függés								
Lehetetlen függés								
Részleges függés								
Tranzitív függés								
Kulcstörő függés								
Hierarchikus kulcs								

P: projekció, a meghatározó tartalmazza a meghatározottat

D: nem projekció, a meghatározó nem tartalmazza a meghatározottat

Részleges függés

Definíció

Egy relációsémában részleges függés található, ha annak kulcsa összetett és van olyan leíró tulajdonsága, amely a kulcson túlmenően annak valamely részétől is függ.

$A[X+Y] \rightarrow B[X]$

P (triviális projekció)

$B[X] \rightarrow C[Z]$

D (X, a kulcs része meghatározza Z-t)

$A[X+Y] \rightarrow C[Z]$

D (X+Y kulcs meghatározza Z-t)

Részleges függés – 2NF

Definíció

Ha egy relációséma normalizált (1NF) és nem tartalmaz részleges függést, akkor a séma a második normálformában van (2NF).

Normalizált, egyszerű kulccsal rendelkező séma mindig 2NF.

Speciális függések

	1	2	3	4	5	6	7	8
$A \rightarrow B$	P	P	P	P	D	D	D	D
$B \rightarrow C$	P	P	D	D	P	P	D	D
$A \rightarrow C$	P	D	P	D	P	D	P	D
Triviális függés								
Lehetetlen függés								
Részleges függés				X				
Tranzitív függés								
Kulcstörő függés								
Hierarchikus kulcs								

Részleges függés

Tétel-1

1NF

<u>Rendelésszám</u>	<u>Cikkszám</u>	Egységár	Mennyiség
AAA	xx	20	110
AAA	yy	40	220
BBB	xx	20	330
BBB	zz	60	440

Tétel-2

<u>Rendelésszám</u>	<u>Cikkszám</u>	Mennyiség
AAA	xx	110
AAA	yy	220
BBB	xx	330
BBB	zz	440

Cikk

<u>Cikkszám</u>	Egységár
xx	20
yy	40
zz	60

2NF

Tranzitív függés

Definíció

Ha egy relációséma 2NF és nem tartalmaz tranzitív függést, akkor a séma a harmadik normálformában van (3NF).

Mj: Normalizált, egyszerű kulccsal rendelkező, egy nem kulcs attributumot tartalmazó séma mindig 3NF.

Tranzitív függés

Definíció

A tiszta tranzitivitás esetén a reláció kulcsa által nem kulcsrészként meghatározott (leíró) tulajdonságtól függ egy másik leíró, amit a kulcs szintén meghatároz.

$A \rightarrow B$ D

$B \rightarrow C$ D

$A \rightarrow C$ D

Speciális függések

	1	2	3	4	5	6	7	8
$A \rightarrow B$	P	P	P	P	D	D	D	D
$B \rightarrow C$	P	P	D	D	P	P	D	D
$A \rightarrow C$	P	D	P	D	P	D	P	D
Triviális függés								
Lehetetlen függés								
Részleges függés				X				
Tranzitív függés								X
Kulcstörő függés								
Hierarchikus kulcs								

Tranzitív függés

Rendelés-1

2NF

<u>Rendelésszám</u>	...	Vevőkód	Vevőnév
AAA		111	XXX
BBB		112	YYY
CCC		111	XXX
DDD		113	ZZZ

Rendelés-2

<u>Rendelésszám</u>	...	Vevőkód
AAA		111
BBB		112
CCC		111
DDD		113

Vevő

<u>Vevőkód</u>	Vevőnév
111	XXX
112	YYY
113	ZZZ

3NF

Adatbázis-tervezés

Egy adatbázis relációsémái jóltervezettnek nevezhetők, ha minden reláció összes nem kulcs attribútuma függ

- a kulcstól (1NF)
- a teljes kulcstól (2NF)
- és csak a kulcstól (3NF)

Triviális függés

P1. A településeket Országkód+Megyekód+Településkód együttes azonosítja olyan módon, hogy a Megyekód és a Településkód relatív sorszáma. Ekkor a kulcs egyedi részei között nincs függés. Az Országkód+Megyekód+Településkód $\rightarrow$ Országkód+Megyekód $\rightarrow$ Országkód függések projektívek és triviálisak.

$A[X+Y+Z] \rightarrow B[X+Y]$ P

$B[X+Y] \rightarrow C[X]$ P

$A[X+Y+Z] \rightarrow C[X]$ P

Lehetetlen függés

A táblázat második oszlopa egy logikailag kizárt esetet mutat.
Ha ui. fennáll az alábbi két függés

$$A[X+Y+Z] \rightarrow B[X+Y]$$

$$B[X+Y] \rightarrow C[X]$$

akkor az

$$A[X+Y+Z] \rightarrow C[X]$$

Összefüggésnek is igaznak kellene lenni. Ezt tagadja a 2. oszlop 3. bejegyzése.

Speciális függések

	1	2	3	4	5	6	7	8
$A \rightarrow B$	P	P	P	P	D	D	D	D
$B \rightarrow C$	P	P	D	D	P	P	D	D
$A \rightarrow C$	P	D	P	D	P	D	P	D
Triviális függés	X							
Lehetetlen függés		X						
Részleges függés				X				
Tranzitív függés								X
Kulcstörő függés								
Hierarchikus kulcs								

Kulcstörő függés

Egy relációsémában kulcstörő függés van, ha a séma valamely leíró tulajdonsága meghatározza az összetett kulcs részét.

$A[X+Y] \rightarrow B[Z]$

$B[Z] \rightarrow C[X]$

$A[X+Y] \rightarrow C[X]$

D (X+Y kulcs meghatározza Z-t)

D (Z meghatározza X-et, a kulcs egy részét)

P (Triviális projekció)

Speciális függések

	1	2	3	4	5	6	7	8
$A \rightarrow B$	P	P	P	P	D	D	D	D
$B \rightarrow C$	P	P	D	D	P	P	D	D
$A \rightarrow C$	P	D	P	D	P	D	P	D
Triviális függés	X							
Lehetetlen függés		X						
Részleges függés				X				
Tranzitív függés								X
Kulcstörő függés							X	
Hierarchikus kulcs								

Kulcstörő függés

Oktatás

<u>Tanuló</u>	<u>Tárgy</u>	<u>Tanár</u>
AAA	Matek	Kovács
AAA	Fizika	Szabó
BBB	Matek	Kovács
BBB	Fizika	Nagy

Ha egyszakos tanárok vannak, Tanár → Tárgy

Tanulja

<u>Tanuló</u>	<u>Tárgy</u>
AAA	Matek
AAA	Fizika
BBB	Matek
BBB	Fizika

Tanítja

<u>Tanár</u>	<u>Tárgy</u>
Kovács	Matek
Szabó	Fizika
Nagy	Fizika

Kulcstörő függés

Tanára

<u>Tanuló</u>	<u>Tanár</u>
AAA	Kovács
AAA	Szabó
BBB	Kovács
BBB	Nagy

Tanítja

<u>Tanár</u>	<u>Tárgy</u>
Kovács	Matek
Szabó	Fizika
Nagy	Fizika

Hierarchikus kulcs

Itt a kulcstörő függés azon különleges esetéről van szó, amikor Az összetett kulcs egyik része meghatározza a másik részt.

$A[X+Y] \rightarrow B[X]$	P
$B[X] \rightarrow C[Y]$	D
$A[X+Y] \rightarrow C[Y]$	P

Speciális függések

	1	2	3	4	5	6	7	8
$A \rightarrow B$	P	P	P	P	D	D	D	D
$B \rightarrow C$	P	P	D	D	P	P	D	D
$A \rightarrow C$	P	D	P	D	P	D	P	D
Triviális függés	X							
Lehetetlen függés		X						
Részleges függés				X				
Tranzitív függés								X
Kulcstörő függés							X	
Hierarchikus kulcs			X					

Hierarchikus kulcs

P1. A tervezők a rendelésadatokkal kapcsolatosan a következő viszonyokat állapították meg:

- a) $\text{Tételid}[\text{Rszám} + \text{Cikkszám}] \rightarrow \text{Rmenny}$
- b) $\text{Rszám} \rightarrow \text{Cikkszám}, \text{Rmenny}$

Matematikai alapon ne is keressünk megoldást
Jelen esetben kétféle rendelésről van szó: többtétel és egytétel.

- a) $\text{Tételid}[\text{MRszám} + \text{Cikkszám}] \rightarrow \text{MRmenny}$
- b) $\text{SRszám} \rightarrow \text{Cikkszám}, \text{SRmenny}$

Normálforma dekompozíció

Relációk egyenkénti elemzése, attribútumfüggések vizsgálata, szükség esetén relációk szétagolása. Szemantikai elemeket is tartalmazhat.

A függési anomáliát okozó (általában a meghatározott) attribútumot kiemeljük a relációból és elhelyezzük egy másikba, mely vagy létezik már, vagy most hozzuk létre.

Pl.

Rendelészszám $\rightarrow$ Vevőkód $\rightarrow$ Vevőnév tranzitív függés

A Vevőkód $\rightarrow$ Vevőnév párost kivetítjük a VEVŐ relációba

Ez azonban a következő problémákat vetheti fel:

Normálforma dekompozíció

1. Szinonim leírók: Tegyük fel, hogy már létezik a VEVŐ reláció és abban a Vevőnév-nek megfelelő adat, de Vevő_megnevezése névvel! A tervezőnek fel kell fedeznie a Vevőnév=Vevő_megnevezése azonosságot, hogy a duplikációt elkerülje. Nyilvánvaló, hogy a szinonim nevek egyikét meg kell szüntetni.

Normálforma dekompozíció

2. Homonim leírók: Tegyük fel, hogy már létezik a VEVŐ reláció és abban egy Vevőnév adat, de mást jelent, mint a kivetített tulajdonság. (pl. egy rövidített név).

A tervező hibát követ el, ha nem alkalmaz bevetítést. Ilyenkor az ortodox eljárásnak megfelelően nem elegendő csak a nevet vizsgálni, szemantikai elemzést is végezni kell.

Valamelyik Vevőnevet át kell nevezni. Ez kínos és hosszú procedúra lehet egy már igen sok helyen dokumentált adatmodellezés esetén.

Normálforma dekompozíció

3. Szinonim kulcsok: Tegyük fel, hogy már létezik a VEVŐ reláció és abban a Vevőkódnak megfelelő adat, de Vevő_azonosító névvel! A tervezőnek fel kell fedeznie a Vevőkód=Vevő_azonosító azonosságot, nehogy egy új relációt hozzon létre az „új” kulccsal.

Nyilvánvaló, hogy a szinonim nevek egyikét meg kell változtatni, és ez gondot jelent.

Ugyanis a kulcsok több relációban is szerepelhetnek *idegen kulcsként*. A relációk kapcsolatai rosszul lehetnek kijelölve, és ezen nem segít az egy relációra koncentráló dekompozíció.

Normálforma dekompozíció

4. Homonim kulcsok: Tegyük fel, hogy létezik a RENDELÉS relációnk Rendelésszám kulccsal. Valamilyen X relációból kivetítjük az ott tranzitív Rendelésszám $\rightarrow$ Rendeléstípus párost. A bevetítés a RENDELÉSBE helytelen, ha a Rendelésszám homonímia. Pl. használhatják vevői és szállítói rendelésekben egyaránt. Ha erre nem derül fény, egy relációba össze nem illő adatok kerülhetnek. Káosz.

További problémát jelent az, hogy valójában nem kapcsolt táblákat kapcsoltnak mutatnak.

Normálforma dekompozíció

5. Kulcsjelöltek: Tegyük fel, hogy egy sémában több attribútum is lehetne elsődleges kulcs. Ezekből egyet választunk. Ekkor a többi kulcsjelölt is meghatározza az egyéb leíró attribútumokat, tranzitív függést okozva. Pl.

Dolgozó(Törzsszám, Személyi_szám, TAJ-szám, Név, Cím)

Nyilván fennáll a Személyi_szám $\rightarrow$ Név, a TAJ-szám $\rightarrow$ Cím stb. függés. Kiküszöbölni nem érdemes. Megoldás: a kulcsjelölteket meg kell jelölni, így az ő függéseiket nem vesszük figyelembe a normalizálásnál.

Dolgozó(Törzsszám, Személyi_szám*, TAJ-szám*, Név, Cím)

Normálforma dekompozíció

Ezen problémák elkerülhetők az alábbi adatmodellezési irányelvek alkalmazásával:

- Minden fogalmat csak egyetlen helyen szabad leírni
- Az adatmodellezés nem megosztható egyéni feladat
- Az elemzési munka mindig folyamatos egyeztetést igényel

Normálforma szintézis

Szintaktikus eljárás, a tartományokat tekintjük, és funkcionális függéseket keresünk közöttük.

A függéseket egy halmazba helyezzük, és addig haladunk, míg valamennyi tartományt nem könyveltük meghatározóként, meghatározottként vagy mindkettőként.

Pl. Rendelésszám → Telepkód → Vállalkód

A halmazt *lefedésnek* (cover) nevezzük. Ez legyen

- Teljes: azaz tartalmazza a lehetséges tartományfüggéseket,
- Minimális: ne tartalmazza azokat a függéseket, melyek már levezethetők a meglévőekből.

Normálforma szintézis

Előnyök

P1. A dekompozíció szerint helyes a következő:

VEVŐ(Vevőkód, Vevőcím)

RENDELÉS(Rendelésszám, Vevőnév) páros.

A szintézis viszont feltárja a Rendelésszám $\rightarrow$ Vevőkód függést, ezzel kiejtjük a rejtett Rendelésszám $\rightarrow$ Vevőnév tranzitivitást.

Szintézissel a

VEVŐ(Vevőkód, Vevőcím, Vevőnév)

RENDELÉS(Rendelésszám, Vevőkód) párost kapjuk.

Normálforma szintézis

Korlátok

Vannak olyan módszerek, melyek kizárják a helytelenül ciklusnak nevezett, redundánsnak tartott *alternatív* reláció kapcsolatokat.

VEVŐ(Vevőkód, Vevőcím, Vevőnév)

TELEPHELY(Telepkód, Vevőkód)

SZERZŐDÉS(Szerződésszám, Vevőkód)

RENDELÉS(Rendelésszám, Telepkód, Szerződésszám)

A RENDELÉS reláció alternatív módon kapcsolódik a VEVŐhöz.

Normálforma szintézis

Hátrányok - szemantika

A szintézis legkomolyabb hibája, hogy mechanikus algoritmus, nem tartalmaz szemantikai megfontolásokat. Pl.

VEVŐ(Vevőkód, Vevőcím, Vevőnév)

TELEPHELY(Telepkód, Vevőkód)

RENDELÉS(Rendelésszám, Vevőkód, Telepkód)

A rendelést hol a vevő, hol annak valamelyik telephelye adja fel.
A szintézis nem figyeli az adattételek minőségét. Felfedezi a
Rendelésszám → Telepkód → Vevőkód általános függéseket,
s így nem engedi meg a Vevőkódot a RENDELÉS relációban.

Normálforma szintézis

Ezzel szemben a dekompozíció helyesen veszi észre, hogy a RENDELÉS reláción belül a Telepkód $\rightarrow$ Vevőkód attribútumfüggés nem áll fenn, mivel nem minden rendelésben található Telepkód érték. Dekompozícióval létrehozhatjuk a fenti helyes formát.

Módszertanok - SSADM ***

- Structured System Analysis & Design Method
- Angol közigazgatási rendszereknél 1980-as évektől
- Adatvezéreltség
- Logikai és fizikai lépések különválasztása
- A rendszer több nézőpontból való vizsgálata – dimenziók (adat, feldolgozás, idő)
- Top-Down és Bottom-Up megközelítések
- A felhasználó bevonása a logikai tervezésbe
- Termékszemlélet - minőségbiztosítás

Módszertanok SSADM v3 lépései

1. Célkitűzés
2. Helyzetfelmérés
3. Rendszerelemzés
4. Előzetes Rt.
5. Megvalósítás elrendelése
6. Részletes rendszerterv
7. Programrendszerterv
8. Programozás

Módszertanok SSADM v3 lépései

9. Tesztelés
10. Operátori utasítás
11. Felh. dokumentáció
12. Véleményeztetés
13. Betanítás
14. Üzemeltetési dokumentáció
15. Programdokumentáció
16. Kísérleti üzem
17. Üzemszerű üzem

AZ SSADM ALKALMAZÁSA

a. A rendszer időre történő elkészítése:

⇒ A hierarchikus felépítésű rendszerben több ellenőrzési pontot (legalább nyolcat) határoznak meg, amely pontokban végrehajtott ellenőrzés során a vezetés képet kap a projekthelyzetéről és amennyiben szükséges érvényesíti a szükséges korrekciókat.

b. A felhasználó igényeit kielégítő rendszer készül:

- ⇒ A felhasználó számára grafikus ábrákkal érhetővé, áttekinthetővé válik a fejlesztési tevékenység eredménye.
- ⇒ A követelmény központúság lehetővé teszi a felhasználó bevonását a projekt tevékenységébe. A projekt irányítás dönt az egyes döntési pontokban, szem előtt tartva a felhasználói követelmények mind teljesebb kielégítését.

AZ SSADM ALKALMAZÁSA

c. A fejlesztés eredményeként kialakuló rendszer követni tudja a környezet változásait:

⇒ A rendszer jól dokumentált, az egyes munkafázisban a célokat világosan rögzítik.

⇒ A megvalósítási dokumentációból kiderülnek a fejlesztők szándékai.

d. A meglévő szakértelem hatékony használata, átvétele:

⇒ Az SSADM átvette és használja a más programozási rendszerben sikeresen alkalmazott eszközöket (egyed, egyedtulajdonság, objektum, objektumtulajdonság, szülő objektum stb.). Eltérés csupán a "kapcsolat" fogalmának alkalmazásában észlelhető.

AZ SSADM ALKALMAZÁSA

e. A minőség javítása a hibák csökkentésével:

- ⇒ Többszemponstú megközelítés.
- ⇒ Pontosn előírtak a tesztelési követelmények.
- ⇒ A dokumentumokra vonatkozó előírások a teljességet biztosítják.

f. A hajlékonyságnak a növelése:

- ⇒ A projekt irányítás határozza meg a feladatokat és ez lehetővé teszi, hogy a kritikus részeket biztosítják.

AZ SSADM ALKALMAZÁSA

g. A termelékenység növelése:

- ⇒ A folyamat jól dokumentált. A dokumentáció jól tanítható és jól megérthető.
- ⇒ A termékközpontúságból eredően, megvéd a fölösleges fejlesztési lépésektől, és a feleslegesen felduzzasztott dokumentációtól. (Pl.: a vízesés módszertől jelentősen kisebb méretű a dokumentáció terjedelme.)

h. Az egy szállítótól való függőség csökkentése:

- ⇒ Az SSADM -ben jól elkülönített a **logikai terv**¹ és a **fizikai megvalósítási terv**².

1. A projekt idő 80% -át teszi ki

2. A projekt idő 20% -át teszi ki

Mire nem jó az SSADM?

- Nem alkalmas a szervezeti működés leírására, vagy annak tanulmányozására.
- Nem alkalmas a szervezeti erősségek és gyengeségek vizsgálatára (SWOT).
- Nem alkalmas a sikertényezők vizsgálatára.

AZ SSADM ALKALMAZHATÓSÁGA

1. Információkra vonatkozó kérdések:

⇒ A kezelendő információknak van-e szerkezet?

⇒ Lehet-e stabil logikai szerkezettel ábrázolni a kezelendő információkat?

2. Az eljárásokra vonatkozó kérdések:

■ ⇒ Az elvégzendő eljárásoknak van-e elegendő szerkezete és pontossága modelt lehessen alkotni?

■ ⇒ Lehet-e az adatfolyamra ábrát rajzolni? (Az a jó, ha a folyamatábrában nincsen egymást keresztező vonal, mert az nem kétdimenziós szerkezetre utal, hanem több dimenzióra.)

AZ SSADM ALKALMAZHATÓSÁGA

- ⇒ Melyek azok fejlesztési részek, amelyek általános és melyek azok, amelyek speciális szoftver eszközökkel kezelhetők? Cél a meglévő szoftver eszközök minél hatékonyabb alkalmazása, kihasználása.
- ⇒ Mi a terjedelme az egésznek (fejlesztési terjedelem)? Kiterjedés, alprojektek meghatározhatóak-e?

AZ SSADM JELLEMZŐI

Az SSADM a csak rendszerfejlesztés *megvalósíthatóság*, *elemzés és a tervezés* fázisait fogja át!!!

Miért?

Szükség van így az SSADM-re?

Mit lehet tenni?

- Az SSADM nagy súlyt fektet az egyes tervezési lépések eredményeinek folyamatos ellenőrzésére, valamint a leendő felhasználókkal történő konzultációkra.
- Az SSADM -ben az *adatorientált megközelítés a fontos*. A *rendszerek fix magját az adatok és azok kapcsolatai alkotják*. (Tapasztalat) szerint nagyobb eséllyel változik meg a feldolgozás módja, mint a feldolgozott anyag, adat.

AZ SSADM JELLEMZŐI

Az SSADM dimenziói:

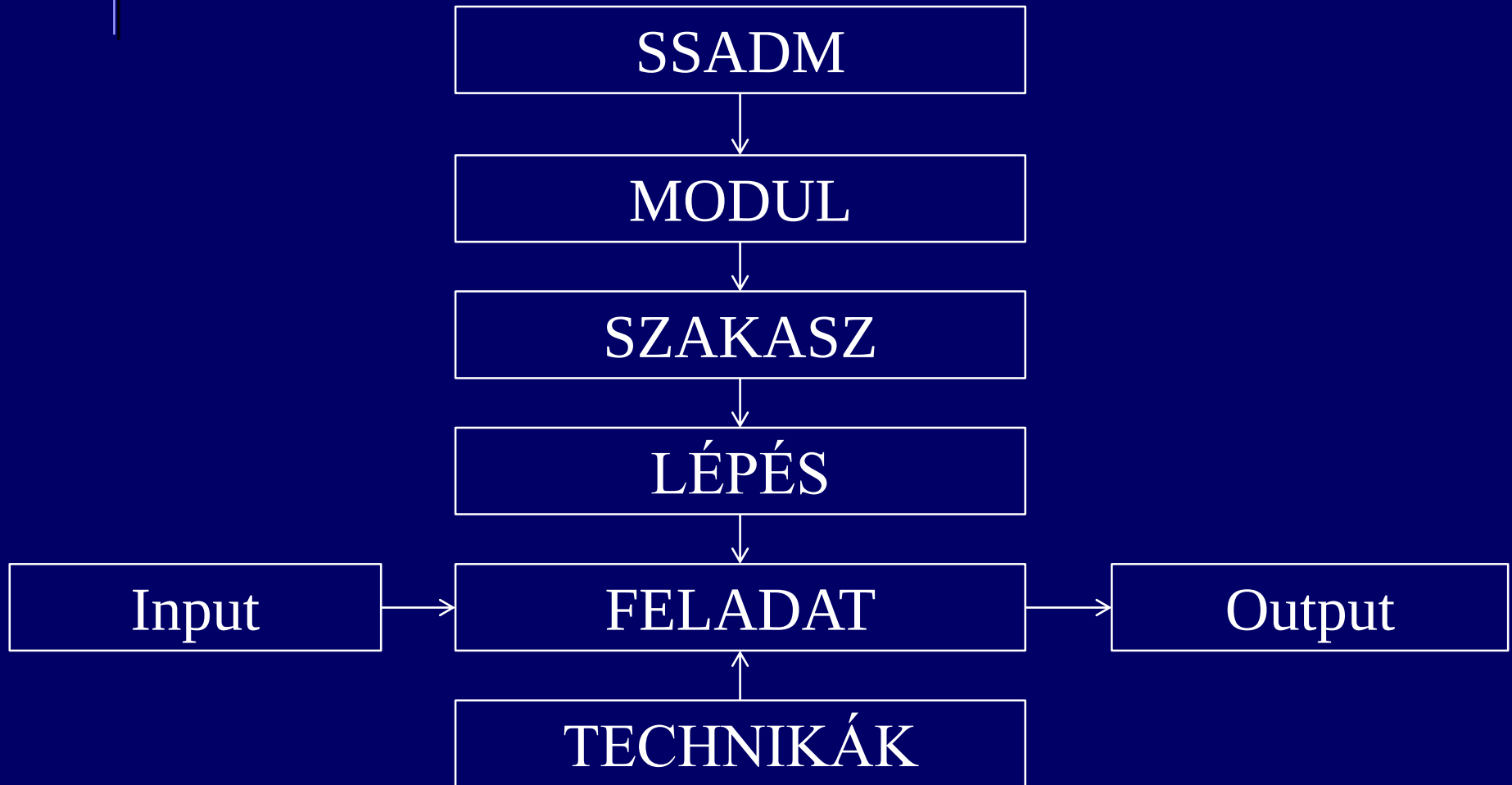
Adat

Folyamat

Idő (a környezet vetület helyett)

- Az SSADM öt modulra, ezen belül hét szakaszra tagolódik. Az **első négy szakasz** (a nulla sorszámú opcionális, de nem célszerű elhagyni) az *elemzési szakasz*, míg az **utolsó három szakasz a tervezési szakasz**.

AZ SSADM SZERKEZETE



AZ SSADM (v4) MODULJAI, SZAKASZAI

Elemzési szakaszok

Megvalósíthatóság-elemzés

⇒ **0. Megvalósíthatóság eldöntése:** Nem kötelező a végrehajtása. Annak vizsgálata, hogy érdemes-e végrehajtani a fejlesztési projektet?

Követelmény-elemzés

⇒ **1. Jelenlegi helyzet vizsgálata:** a jelenlegi (fejlesztendő) rendszer logikájának a megismerése (folyamatok, adatok feltárása, követelmények rögzítése).

⇒ **2. Rendszerszervezési változat kiválasztása:** a fejlesztők és a felhasználó végleges döntése a változatok közül.

Követelmény specifikáció

⇒ **3. Követelmények meghatározása:** adatfolyam modellezés, felhasználói interfész: prototipizálás, képernyőtervek, dialógusok.

AZ SSADM (v4) MODULJAI, SZAKASZAI

Tervezési szakaszok

Logikai rendszerspecifikáció

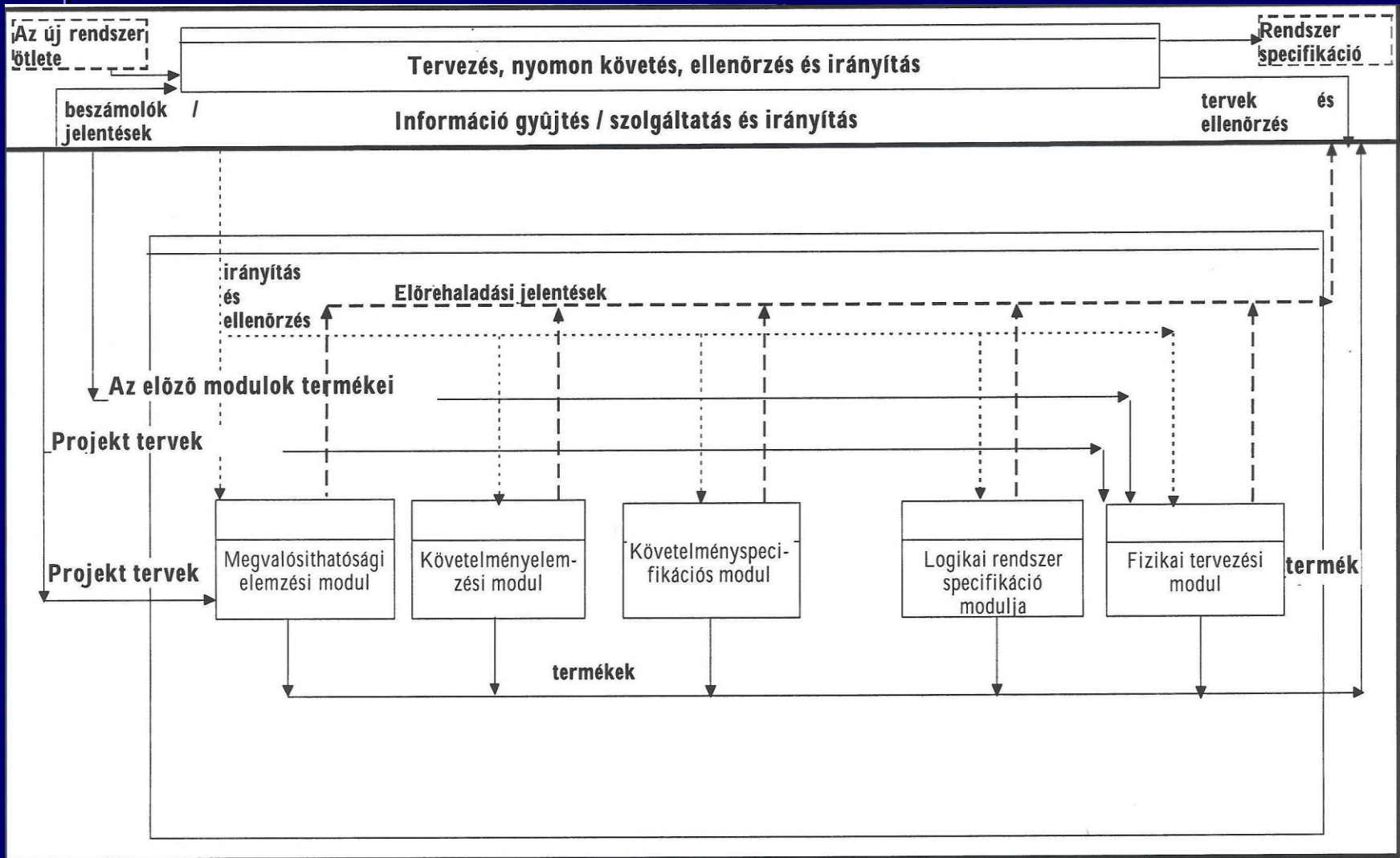
⇒4. *Rendszertechnikai változat kiválasztása*: Több változat kidolgozása, majd a célra legmegfelelőbb kiválasztása.

⇒5. *Logikai rendszertervezés*: Feldolgozások logikai szintű tervezése (lekérdezések stb.). Dialógusok tervezése.

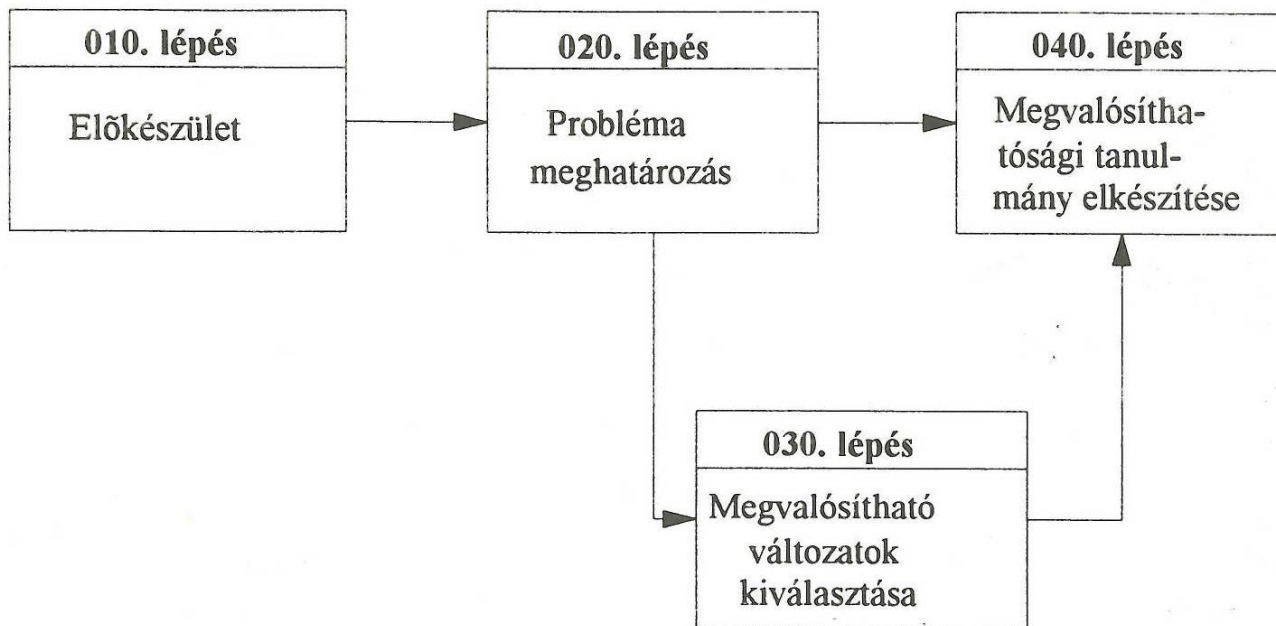
Fizikai rendszertervezés

⇒6. *Fizikai rendszertervezés*: A szakasz olyan végterméket szolgáltat eredményül, amely programozásra alkalmas. - *Adattervek*, - *Feldolgozástervek*, - *Interfésztervek*

AZ SSADM SZAKASZAI



AZ SSADM SZAKASZAI

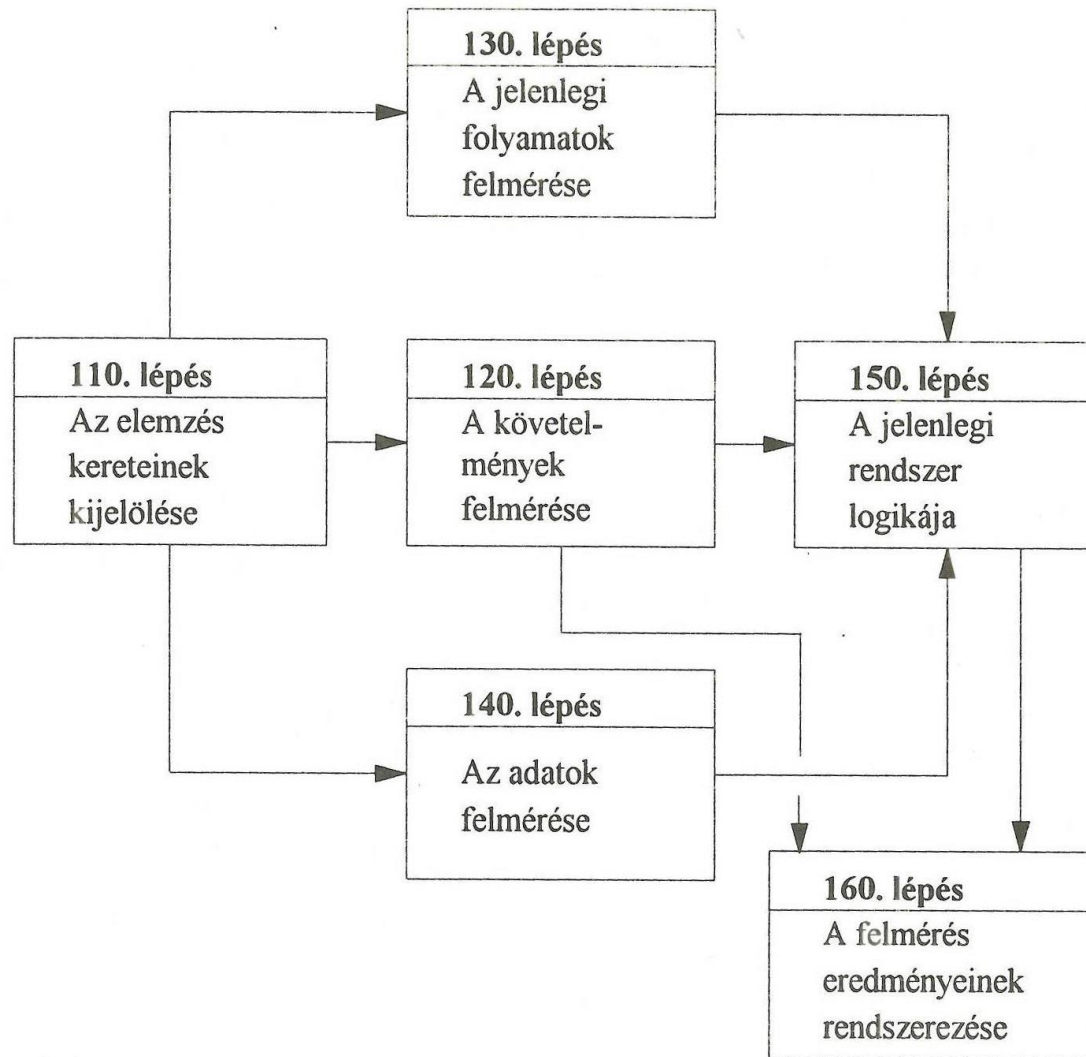


AZ SSADM SZAKASZAI

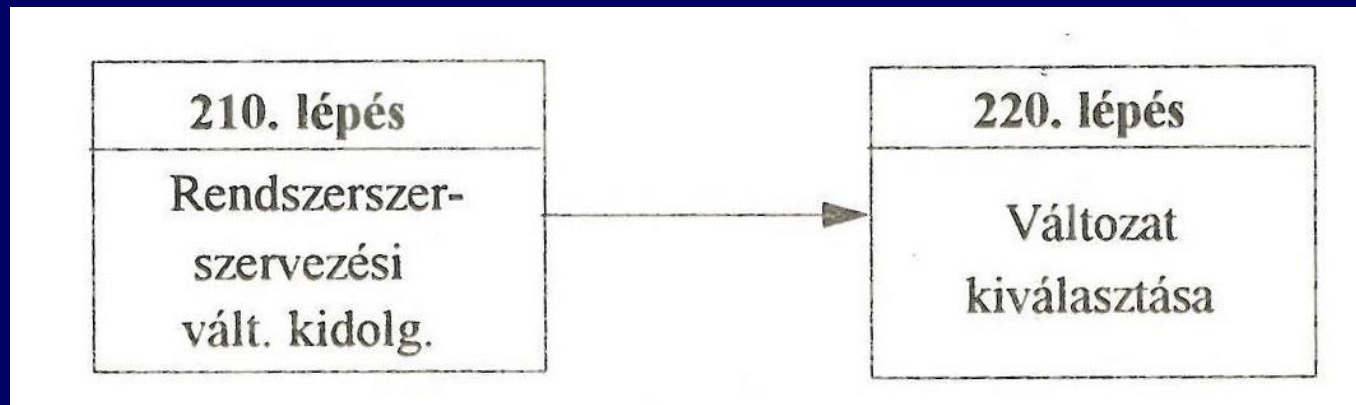
Követelmény elemzés (elemzési feladat)

1. Jelenlegi helyzet vizsgálata (adatok, folyamatok, követelmények)
2. Rendszerszervezési változat kiválasztása (alkalmatlan és drága változatok kiszűrése, az előnyös változatokat tarjuk csak meg)

AZ SSADM SZAKASZAI



AZ SSADM SZAKASZAI



AZ SSADM SZAKASZAI

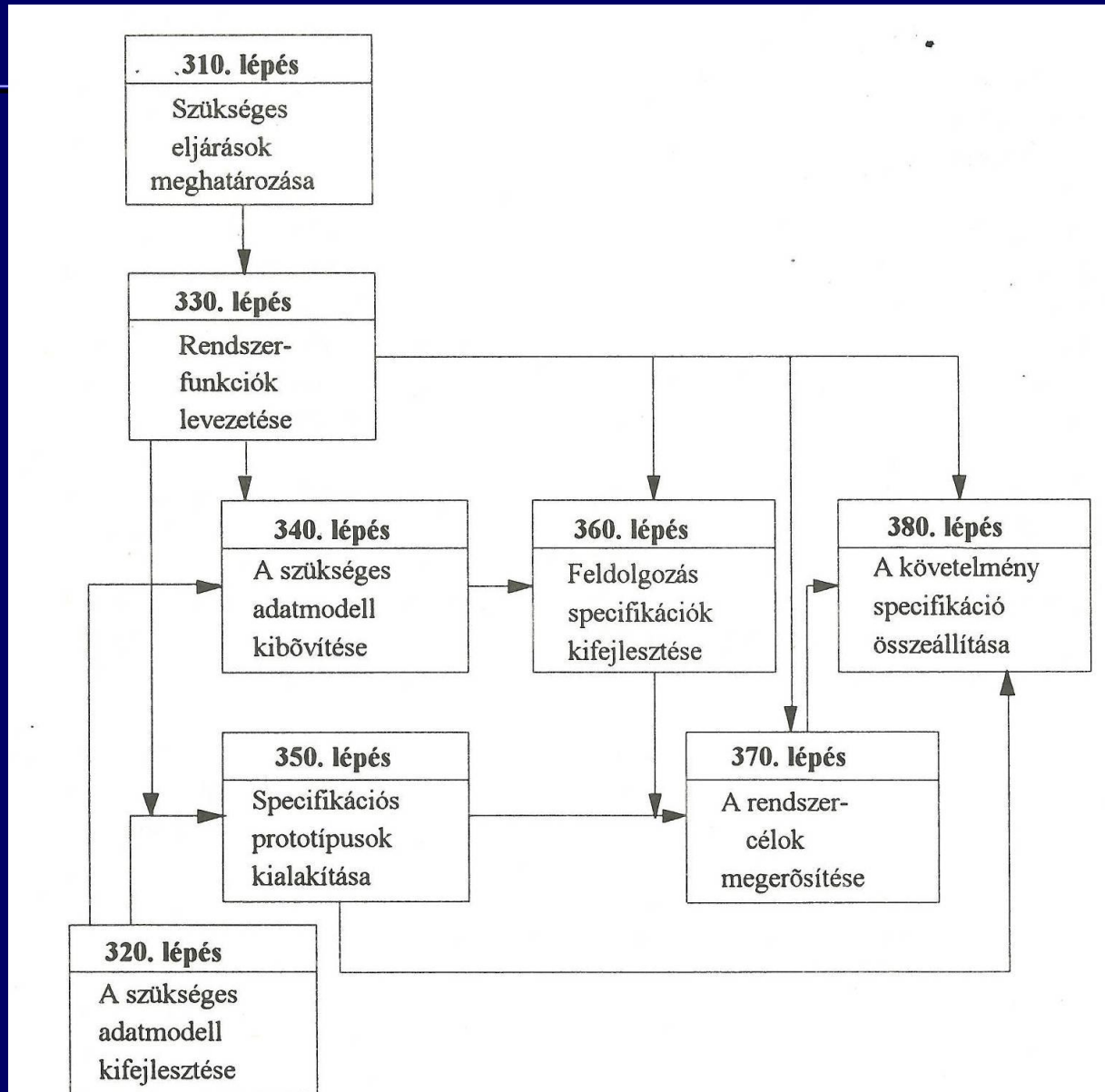
Követelmény specifikáció (elemzés és tervezés)

3. Követelmények meghatározása (EK-modell → top-down, normalizált logikai adatmodell→bottom-up)

- Meglévő rendszer fizikai folyamatai (az 1. szakaszban már elkészült)
- Meglévő rendszer logikai folyamatai (az 1. szakaszban már elkészült)
- Tervezett rendszer logikai folyamatai
- Tervezett rendszer funkciói

Eszköz: adatfolyam-modellezési technika.

AZ SSADM SZAKASZAI



AZ SSADM SZAKASZAI

Logikai rendszerspecifikáció (tervezési feladat)

4. Rendszerteknikai változat kiválasztása:

Figyelembe veendő: meglévő eszközök, hosszabb távú fejlesztési stratégia, kapacitás, költség, várható hatások. 3-6 változat kidolgozása javasolt, amely később 2-3 változatra szűkíthető.

AZ SSADM SZAKASZAI



AZ SSADM SZAKASZAI

Logikai rendszerspecifikáció (tervezési feladat)

5. Logikai rendszertervezés:

Logikai adatmodell: a korábbi szakaszokból örököljük.

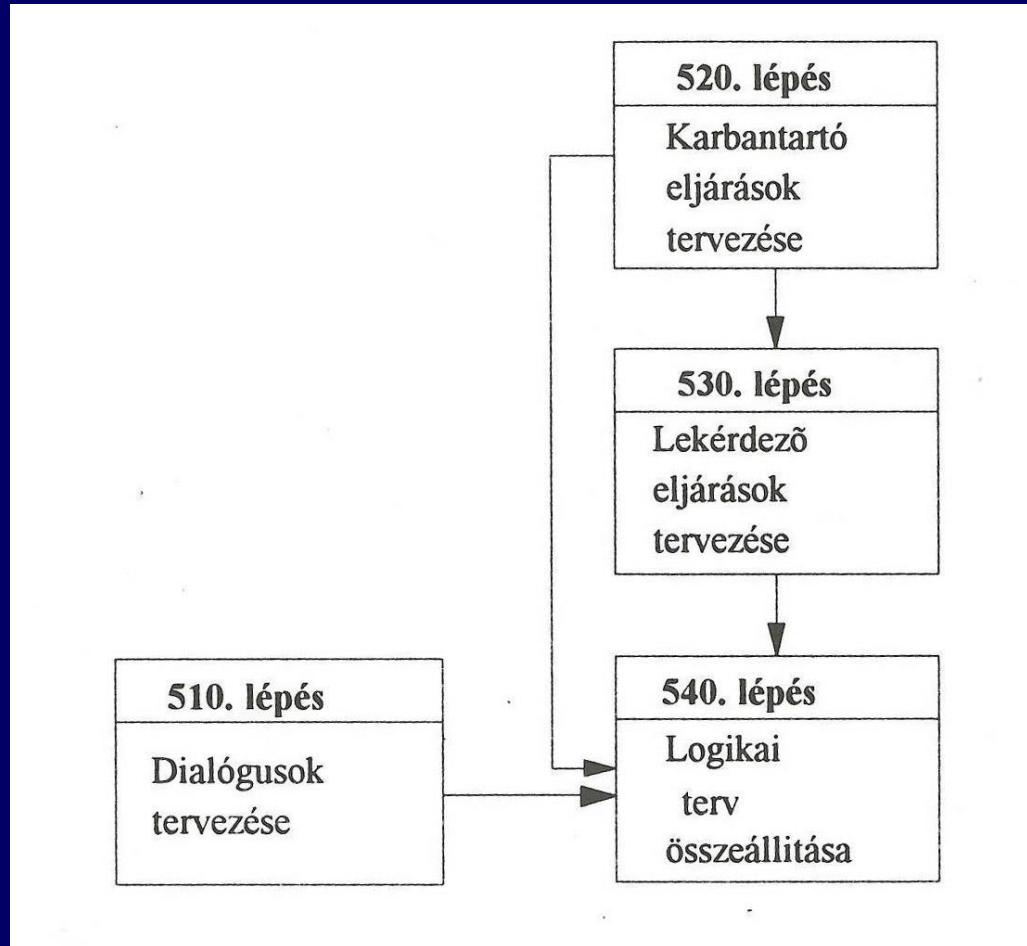
Feldolgozások (ún. Jackson struktúrával írható le):

- karbantartó (lekérdezést is tartalmazhat)
- lekérdező

Dialógusok tervezése:

- dialógusszerkezet (Jackson struktúra)
- menüszerkezet

AZ SSADM SZAKASZAI



AZ SSADM SZAKASZAI

Fizikai rendszertervezés (tervezési feladat)

6. Fizikai rendszertervezés:

Adattervek: logikai adatmodell átalakítása fizikai adattervvé.

Első közelítésű terv: egyed típusokból relációsémák.

Finomított terv: idő-követelmények figyelembe vétele.

Feldolgozástervek: funkciók áttekintése, majd részletes specifikációk elkészítése.

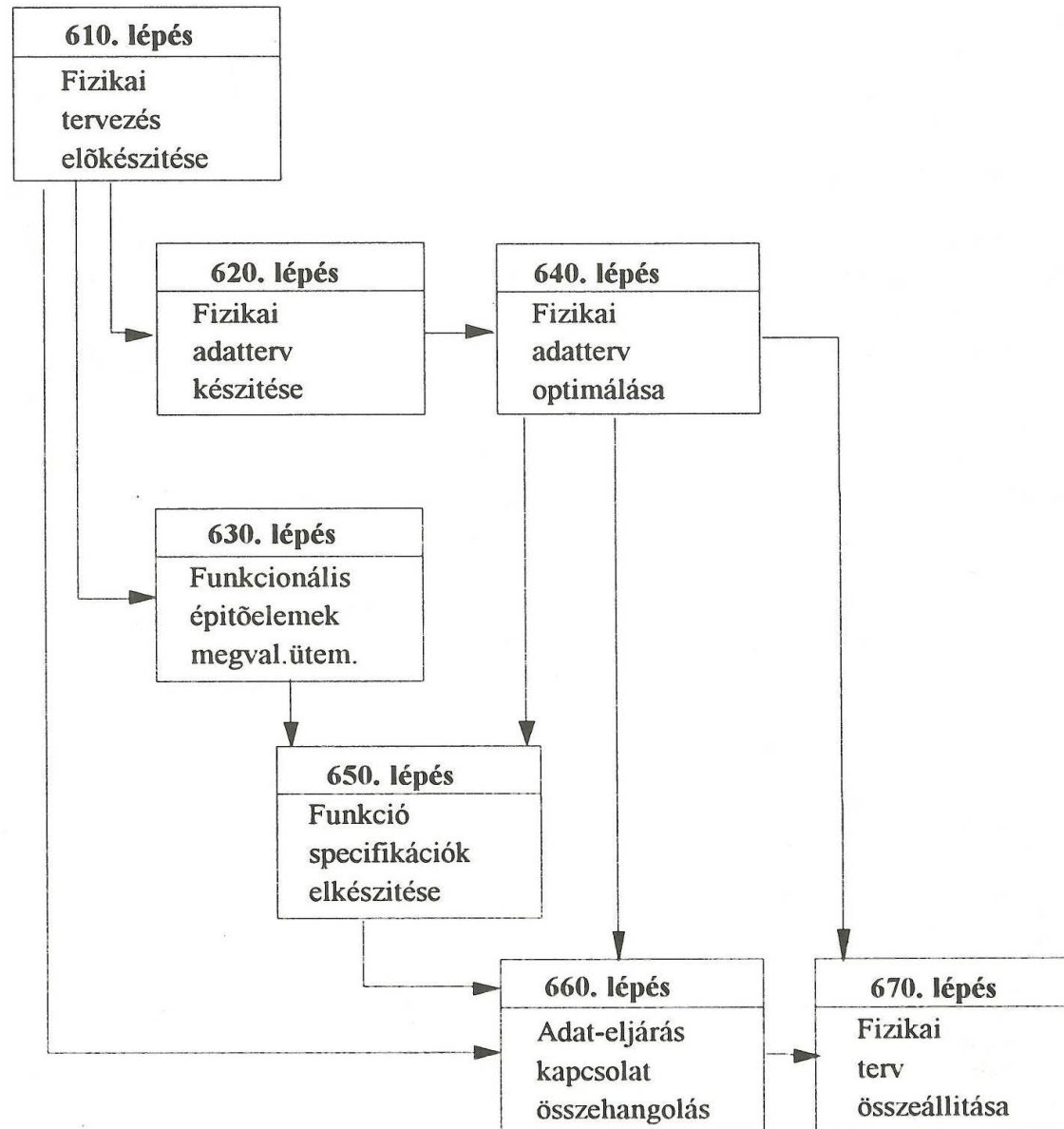
Adattervezéstől független elemek

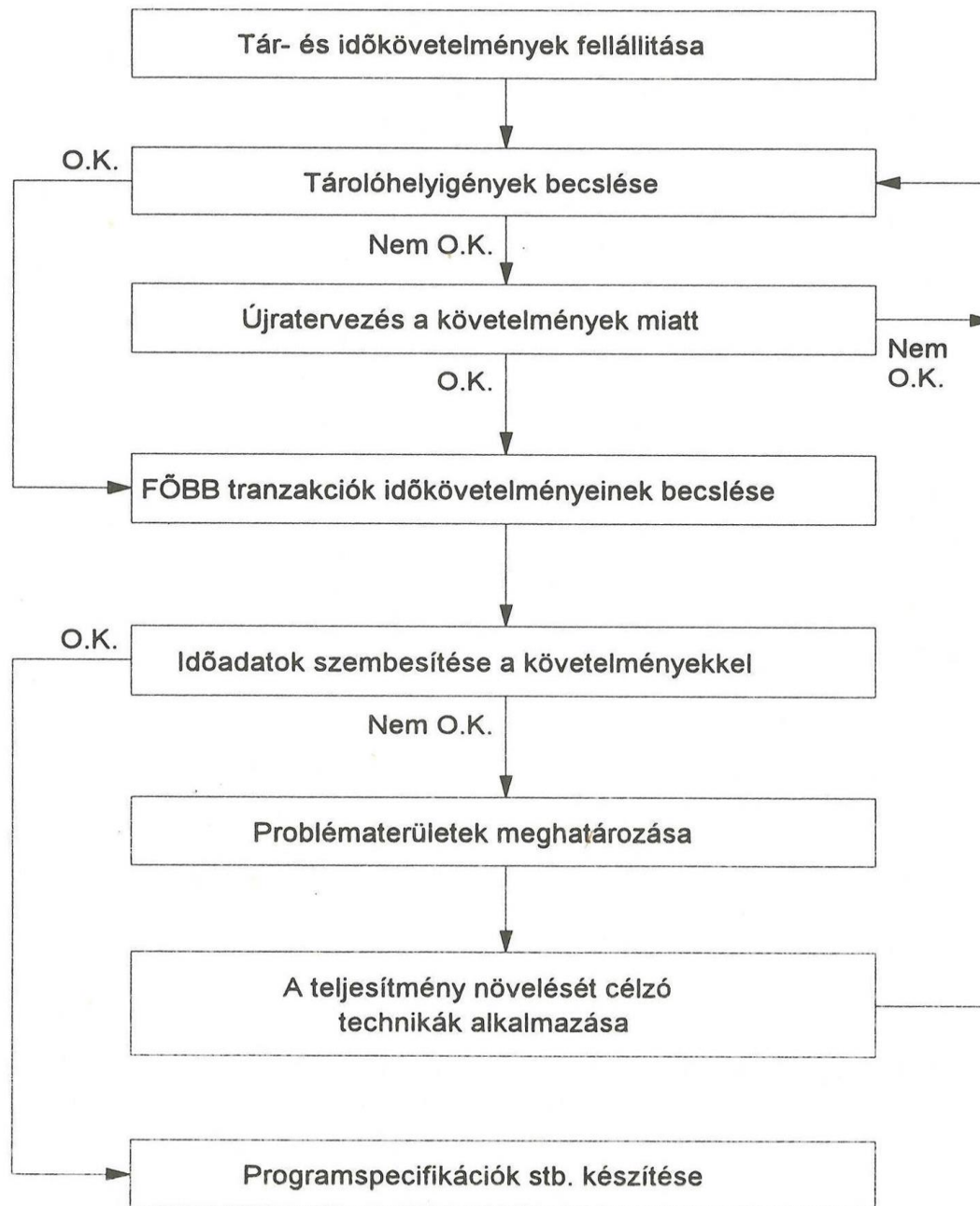
Fizikai adattervre épülő elemek

Interfész-tervek: Adat $\leftrightarrow$ feldolgozás interfészek

Felhasználói interfészek (képernyőtervek, nyomtatványok, listaformátumok) fizikai megtervezéséhez nem ad technikát az SSADM.

AZ SSADM SZAKASZAI





Az SSADM technikái

- Egyed-kapcsolat modellezés
- Visszakeresési utak
- Adatfolyam diagramm
- Egyed-élettörténet diagramm

Egyed-kapcsolat modellezés - Adat

EGYED-KAPCSOLAT DIAGRAM:

Az adatok fogalmi modellezése

A logikai adatmodellezés célja

- ↳ Segít az alkalmazási terület megértésében
- ↳ Diagrammjai révén világosan, pontosan és egyértelműen írja le a kommunikációs rendszert, s így jó kommunikációs eszköz
- ↳ Elősegíti a fejlesztők közötti egyetértést (korai szakaszban) így csökkentve a későbbi gondokat
- ↳ A file-, illetve adatbázis tervezés alapjául szolgál, ugyanakkor független minden konkrét technikától, illetve terméktől

Áttekintés

A logikai adatmodellezés során az elemző megkeresi az egyedtípusokat és a közöttük felfedezhető kapcsolatokat, valamint a legfontosabb tulajdonságtípusokat. Az eredmény dokumentálása a logikai adatmodellben történik (LAM).

A LAM elemei:

- ↳ Egy diagram: Logikai adatszerkezet
- ↳ Kapcsolódó dokumentáció:
 - Egyedtípusok
 - Kapcsolattípusok
 - Tulajdonságtípusok

A logikai adatmodellt gyakran egyedmodellnek vagy egyedkapcsolati modellnek nevezik.

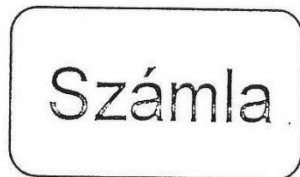
Egyed-kapcsolat modellezés

Alapfogalmak és jelölésrendszer

Egyedtípus

Konkrét vagy elvont tárgy vagy fogalom, amely fontos a vizsgált rendszerben és ezért le akarjuk írni.

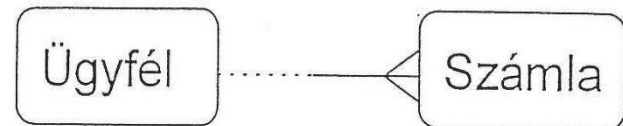
Rajzi ábrázolása lekerekített téglalappal történik, amelybe beírjuk az objektum nevét.



Kapcsolattípus

Kétoldalú. Vagy két egyedtípus, vagy egy egyedtípus saját magával való viszonya.

A kapcsolattípust a két egyedtípust összekötő vonal ábrázolja.



A kapcsolattípusok tulajdonságai

↳ A kapcsolat foka

↳ Jellege

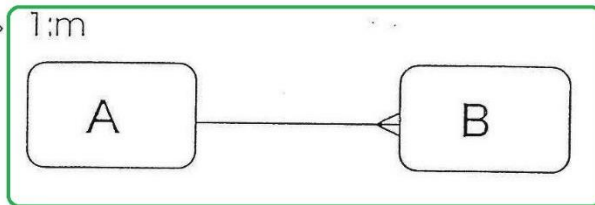
Egyed-kapcsolat modellezés

A kapcsolatok foka

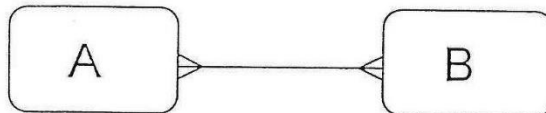
↪ 1:1



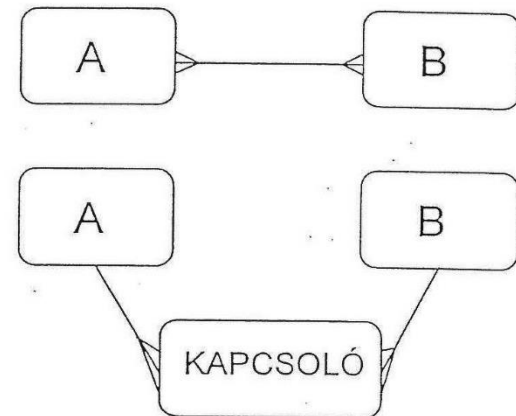
↪ 1:m



↪ m:n

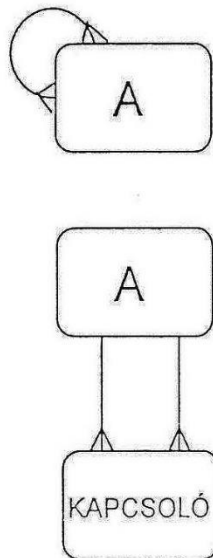


Kapcsoló egyed típus

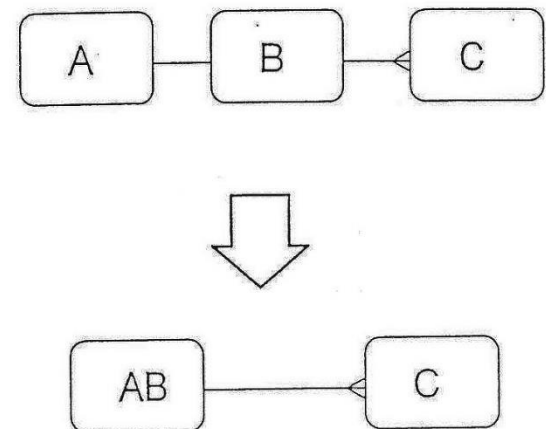


Egyed-kapcsolat modellezés

Rekurzív kapcsolat

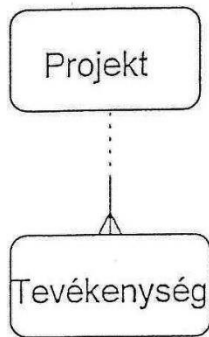


1:1 fokú kapcsolat átalakítása

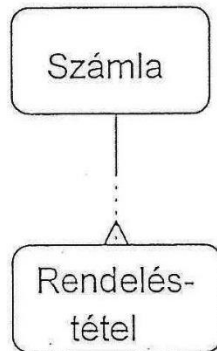


Egyed-kapcsolat modellezés

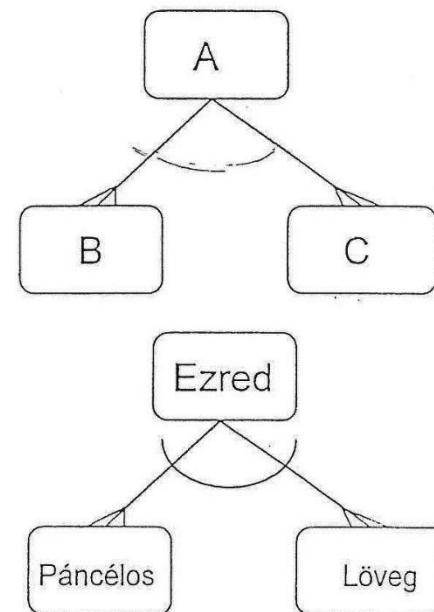
Opcionális alárendelt



Opcionálisan fölérendelt



Kizáró alárendelt



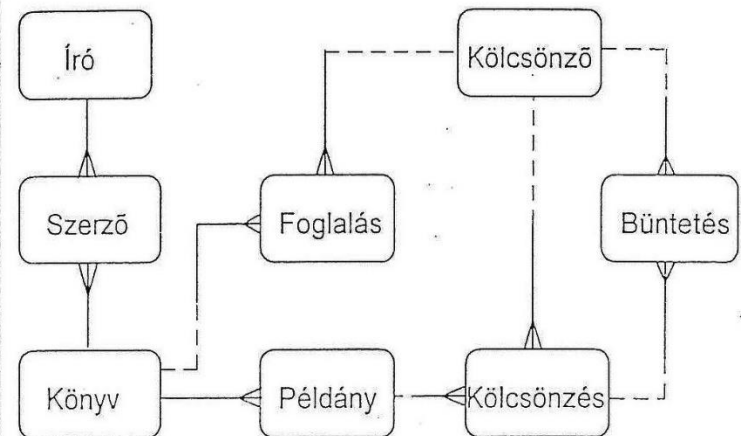
Egyed-kapcsolat modellezés

A fejlesztés menete

- Kezdő egyed típusok kiválasztása
- Közvetlen kapcsolatok meghatározása
- Kezdő diagram megrajzolása
- A kapcsolattípusok meghatározása
- Járulékos jellemzők megállapítása
- A modell ésszerűsítése
- A modell ellenőrzése

Logikai adatmodellezés

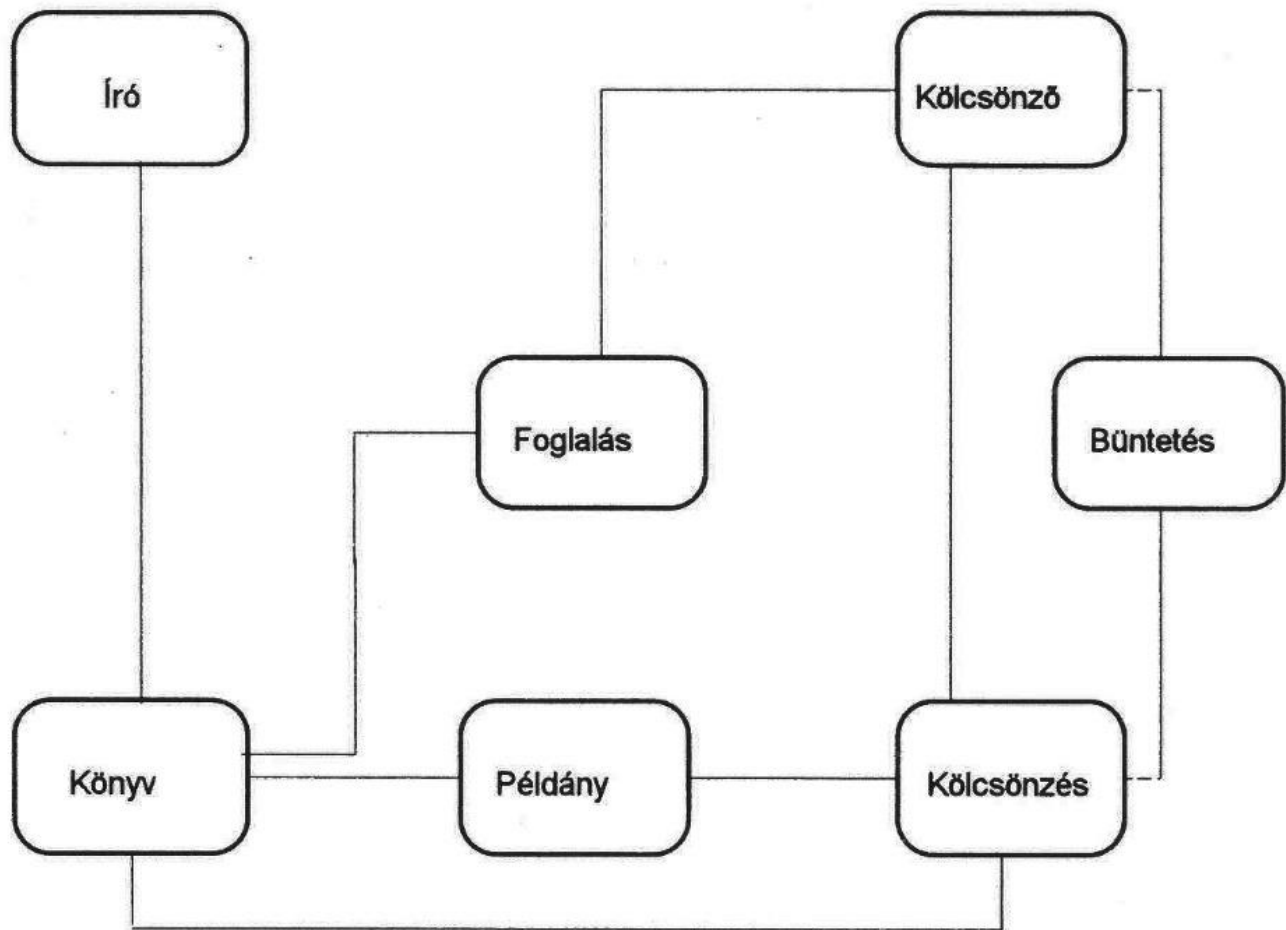
Példa: Kölcsönkönyvtár



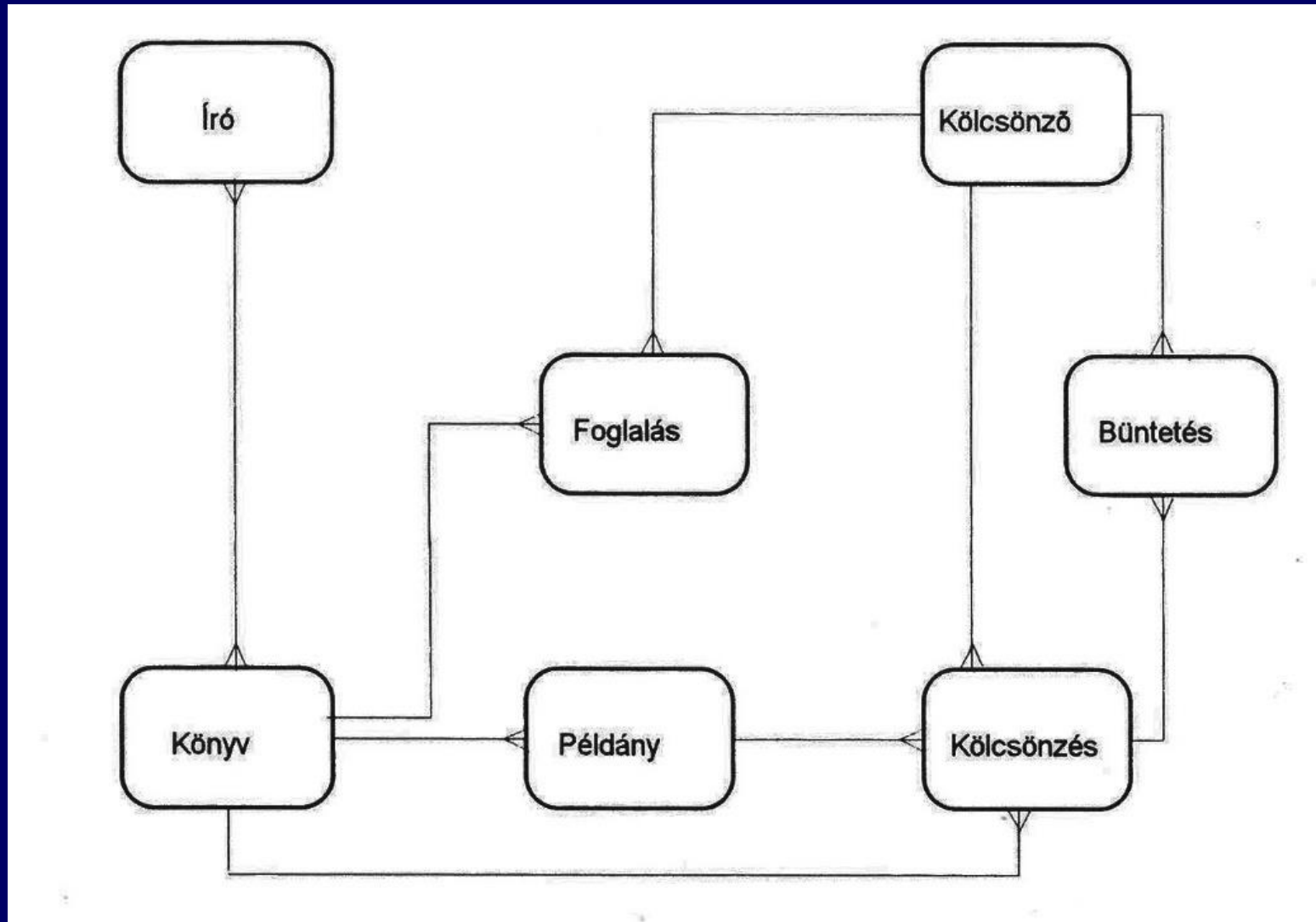
Egyed-kapcsolat modellezés

KÖNYV
ÍRÓ
PÉLDÁNY
KÖLCSÖNZŐ
FOGLALÁS
KÖLCSÖNZÉS
BÜNTETÉS

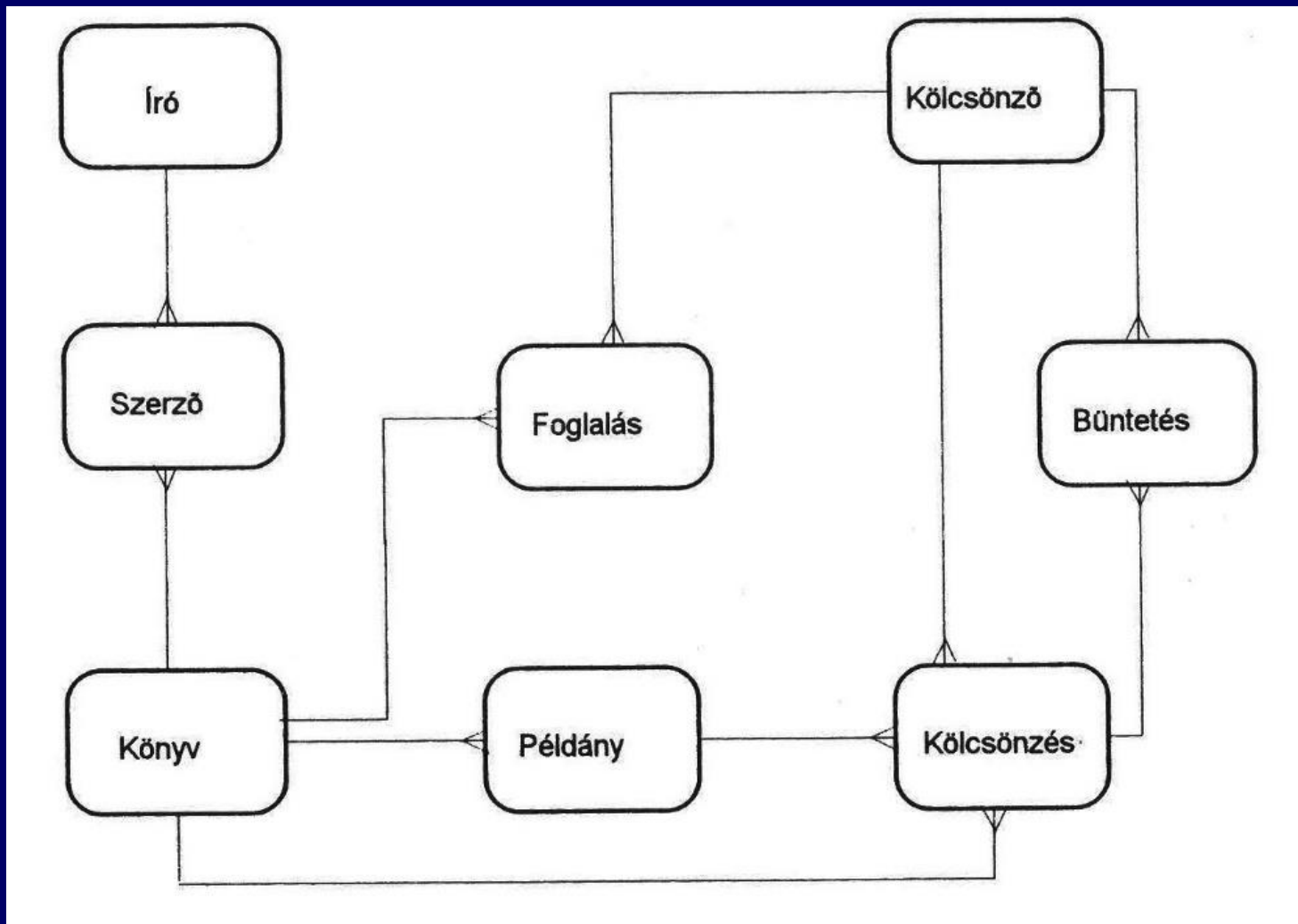
Egyed-kapcsolat modellezés



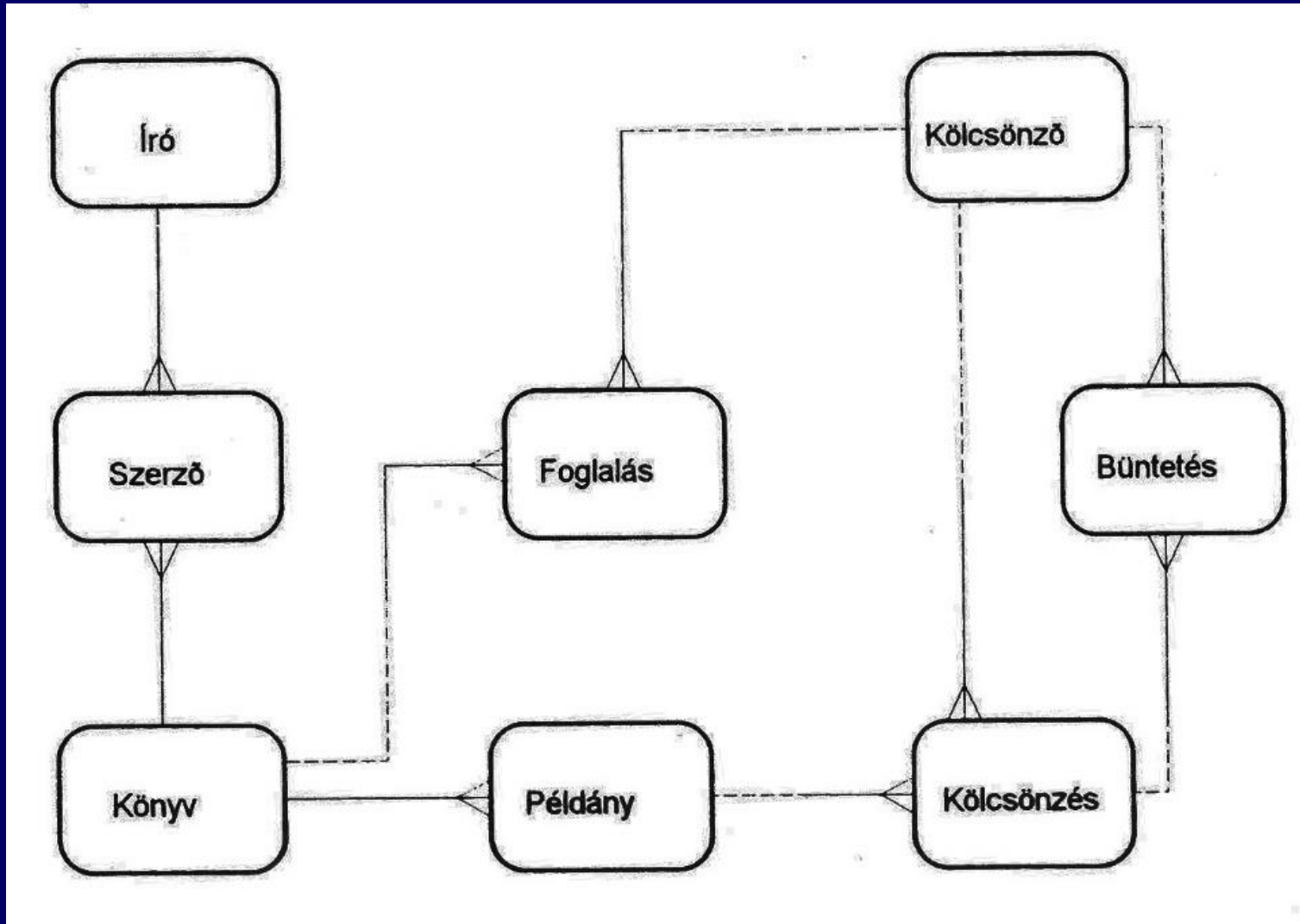
Egyed-kapcsolat modellezés



Egyed-kapcsolat modellezés



Egyed-kapcsolat modellezés



Adatfolyam diagramm - Folyamat

FUNKCIÓK SPECIFIKÁCIÓJA

Adatfolyam modellezés

Célok

- ↳ A rendszer és környezetének ábrázolása, a projekt határainak kijelölése
- ↳ A külső elemek meghatározása, amelyek az inputokat szolgáltatják, illetve az outputokat fogadják
- ↳ Azoknak az adatfolyamoknak a meghatározása, amelyek az inputokat és outputokat hordozzák a rendszer határain keresztül
- ↳ A rendszer határain belüli adatfolyamok leírása
- ↳ Az adattárak kijelölése
- ↳ Az eljárások meghatározása, amelyek feldolgozzák az adatokat, kiváltják az adatfolyamokat és az adatok tárolását

A rendszerszervezésben használt ábrák típusai (amelyek a rendszer *működését* írják le)

- ↳ Rendszer folyamatábrák
- ↳ Program folyamatábrák
- ↳ Ügyviteli folyamatábrák
- ↳ Futtatási diagramok
- ↳ Interaktív rendszerek folyamatábrái

Hátrányok

- ↳ Inkább azt írják le, hogy (fizikailag) *hogyan* történnek a dolgok, mint azt, hogy *mi* történik
- ↳ Kevésbé adnak lehetőséget különböző részletezettségi szintek alkalmazására
- ↳ Túlzottan *eljárásorientáltak*
- ↳ Nyilaiknak nincs tartalma

Adatfolyam diagramm

Az eljárás

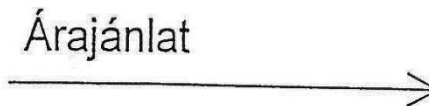


A döntéseknek nincs külön jele, ezek az eljárásokon belül foglalnak helyet

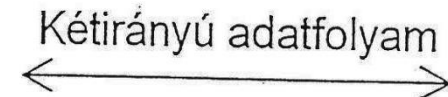
Az adatfolyam

Használjunk kifejező neveket!

Az adatáramlás iránya

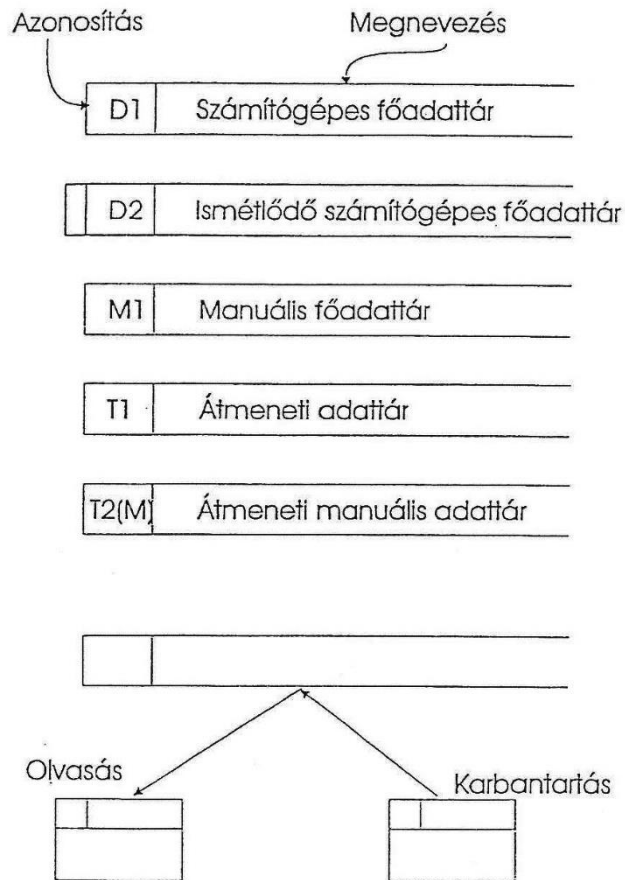


Triviális adatfolyamokat nem kell megnevezni

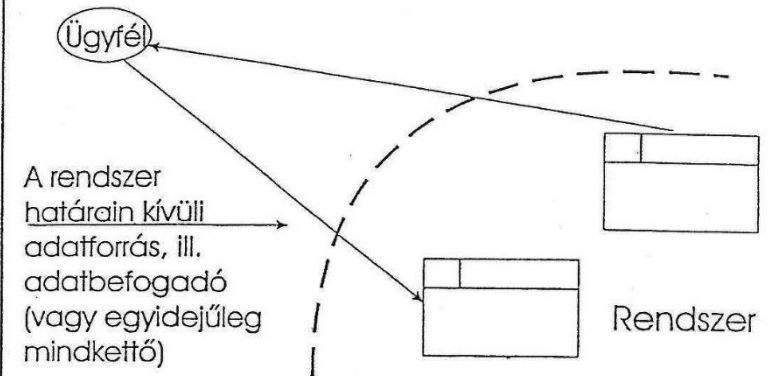


Adatfolyam diagramm

Az adattár



A külső elem



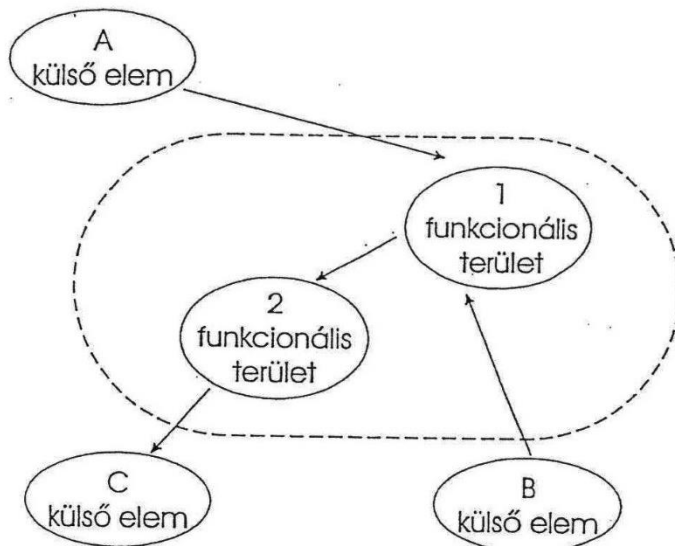
Adatfolyam diagramm

Első szintű AFD kialakítása

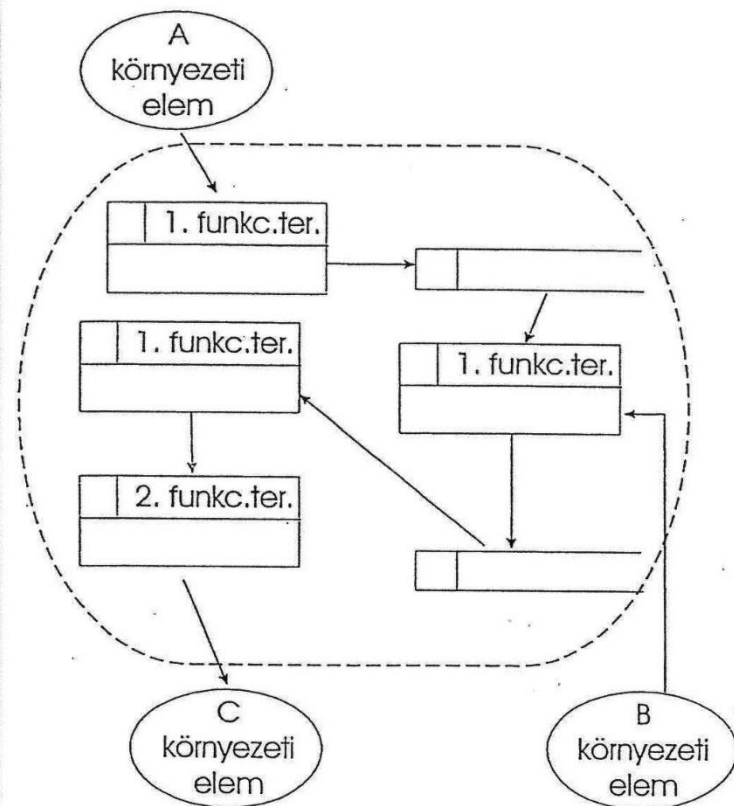
- ↳ A legfontosabb dokumentumok és küldők/címzettek felismerése
- ↳ Dokumentum-áramlási diagram készítése
- ↳ A rendszer határainak kijelölése
- ↳ Eljárások és adattárak
- ↳ További eljárások hozzáadása
- ↳ A teljesség ellenőrzése
- ↳ A részletesség szintjének eldöntése
- ↳ Ellenőrzés a felhasználó bevonásával

Adatfolyam diagramm

A rendszer határainak kijelölése

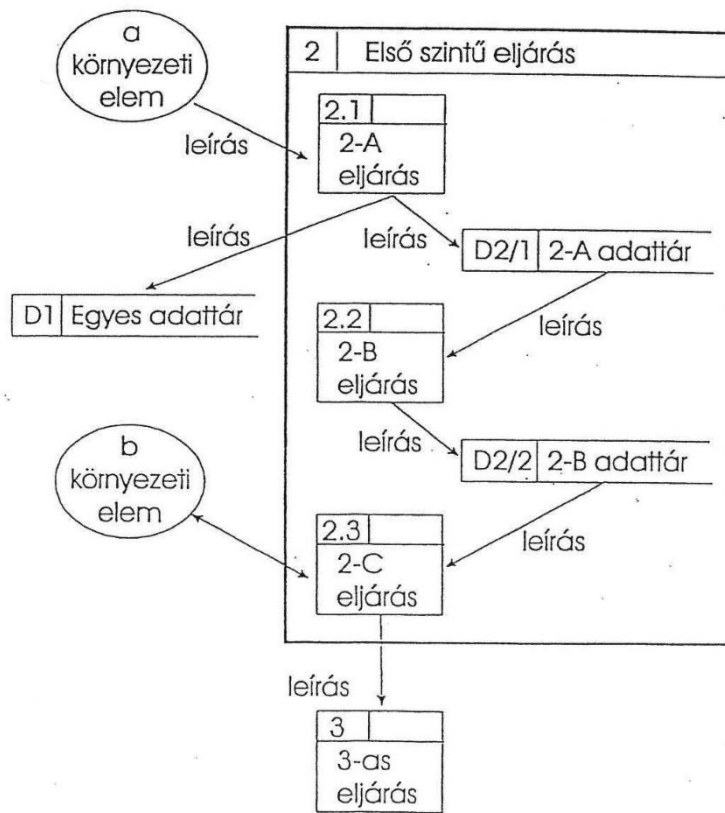


Eljárások és adatfárák felismerése

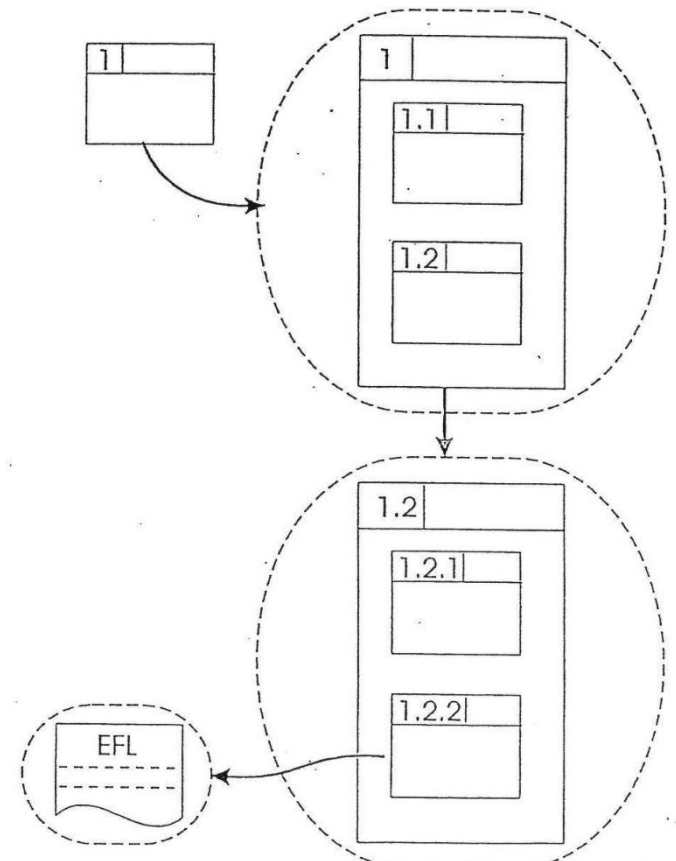


Adatfolyam diagramm

Második szintű AFD-k

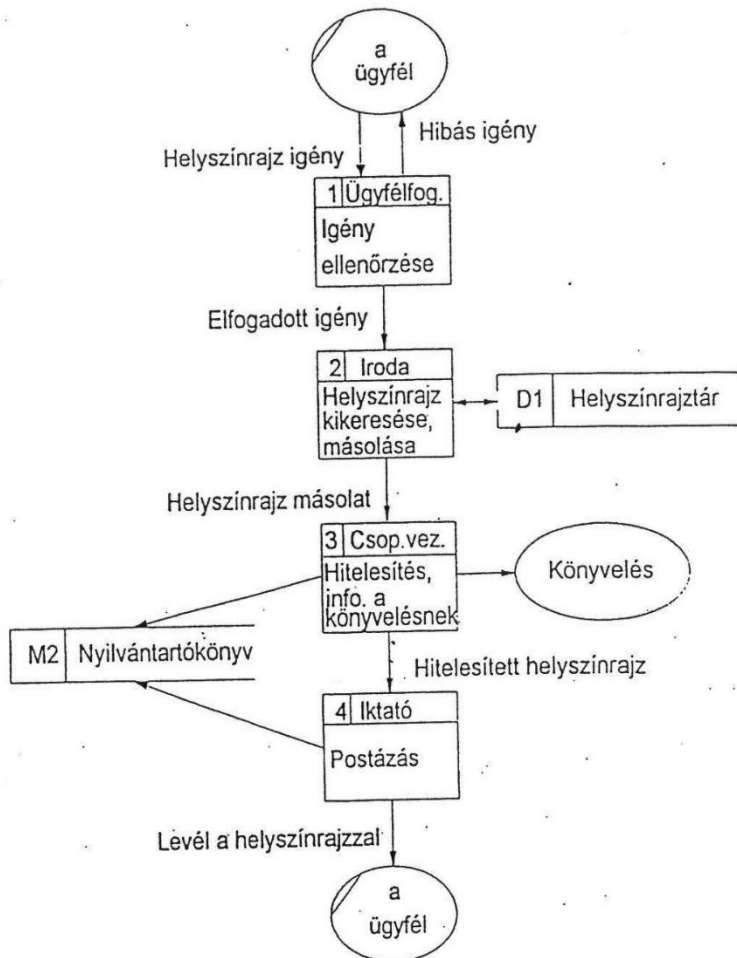


A szintek száma



Adatfolyam diagramm

AFD példa



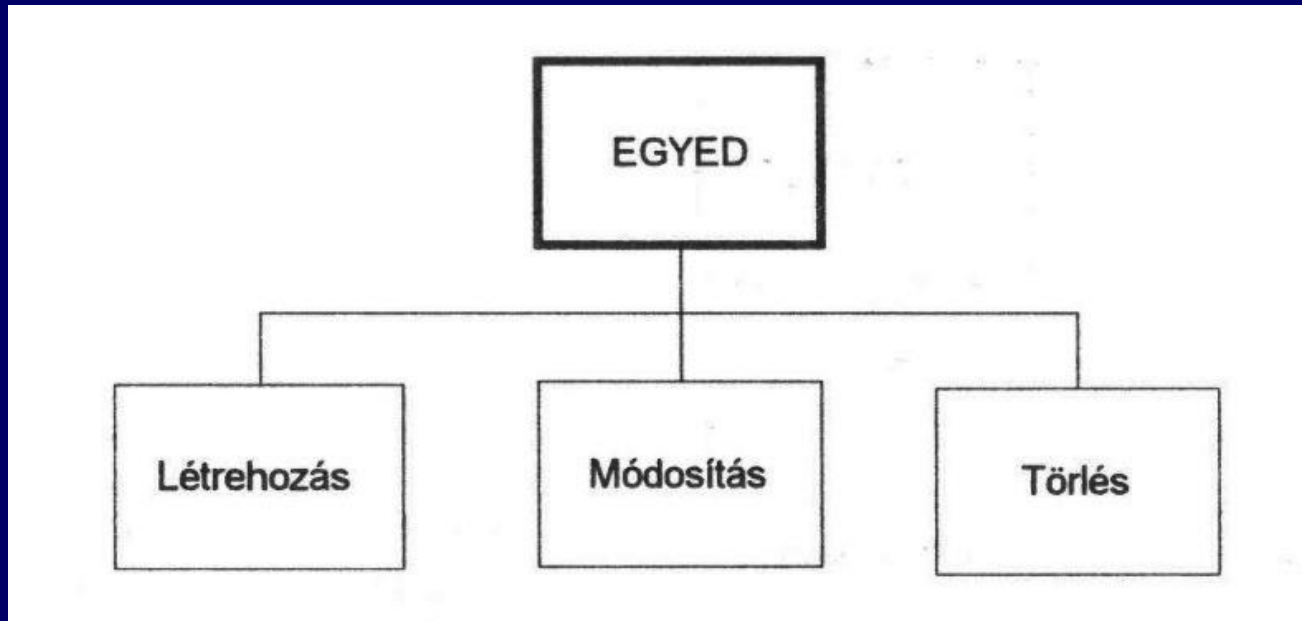
Az adatfolyam diagramok előnyei

- ↳ Gyakorlatias, nem technikai jellegű ábrázolásmód
- ↳ Gyorsan elkészíthető és módosítható
- ↳ Könnyen érthető a felhasználó számára: **kommunikáció**
- ↳ Szintekre bontott, hierarchikus szerkezetben készíthető: **top-down szemlélet**
- ↳ Minimális átfedéssel készíthetők: **minimális redundancia**
- ↳ Segítségükkel a rendszerek akár fizikai, akár logikai szempontból leírhatók és e kétféle típusú diagram egymásba kölcsönösen átalakítható

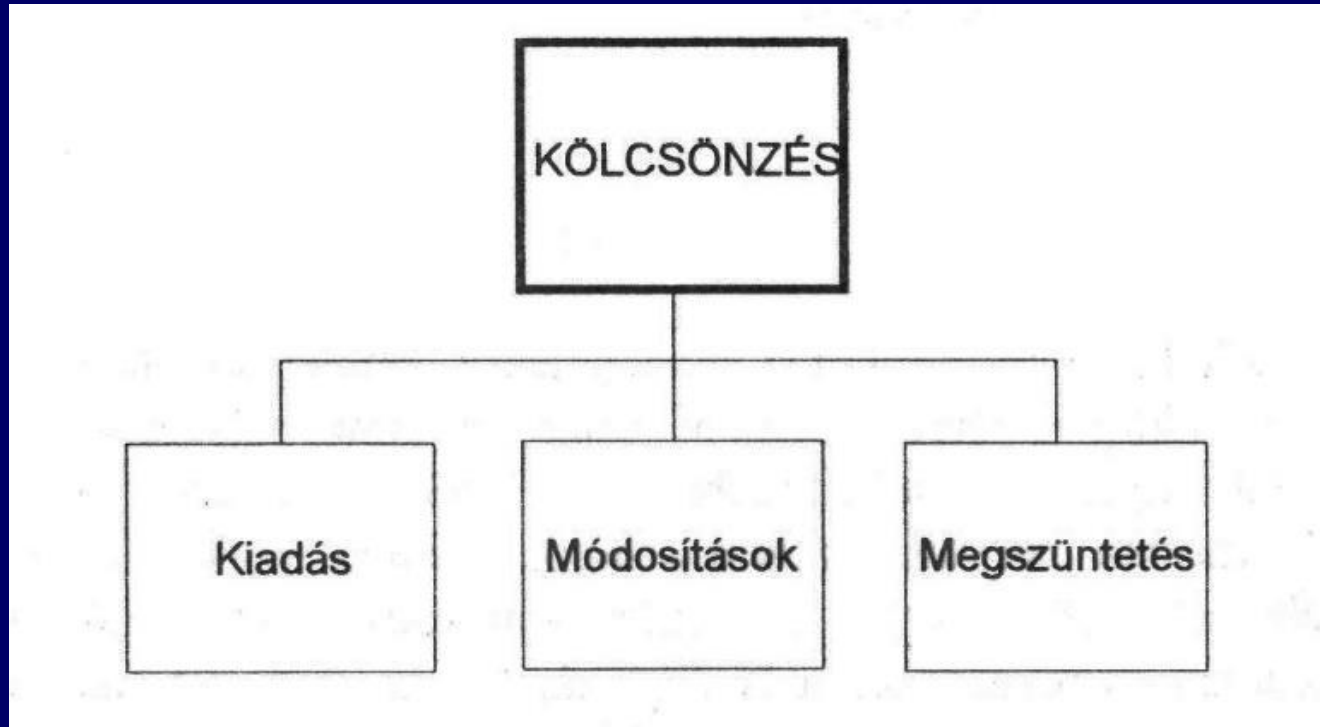
Egyed-élettörténet diagramm - Idő

- Minden egyedtípusra
- Egyedtípus életciklusa
 - Létrehozás
 - Módosítás
 - Megszűnés
- Jackson-struktúrák
 - Szekvencia
 - Szelekció (o)
 - Iteráció (*)

Egyed-élettörténet diagramm



Egyed-élettörténet diagramm

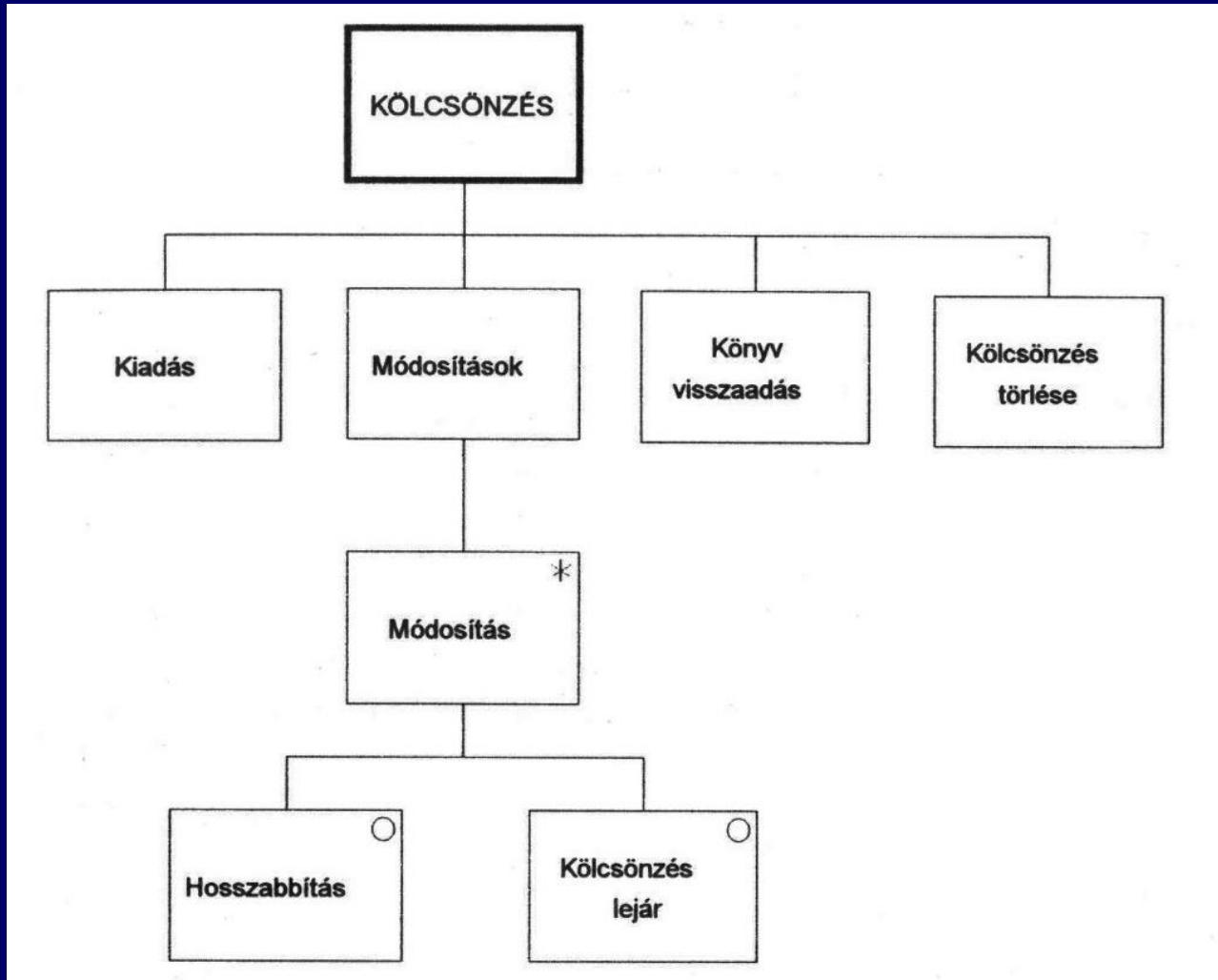


```

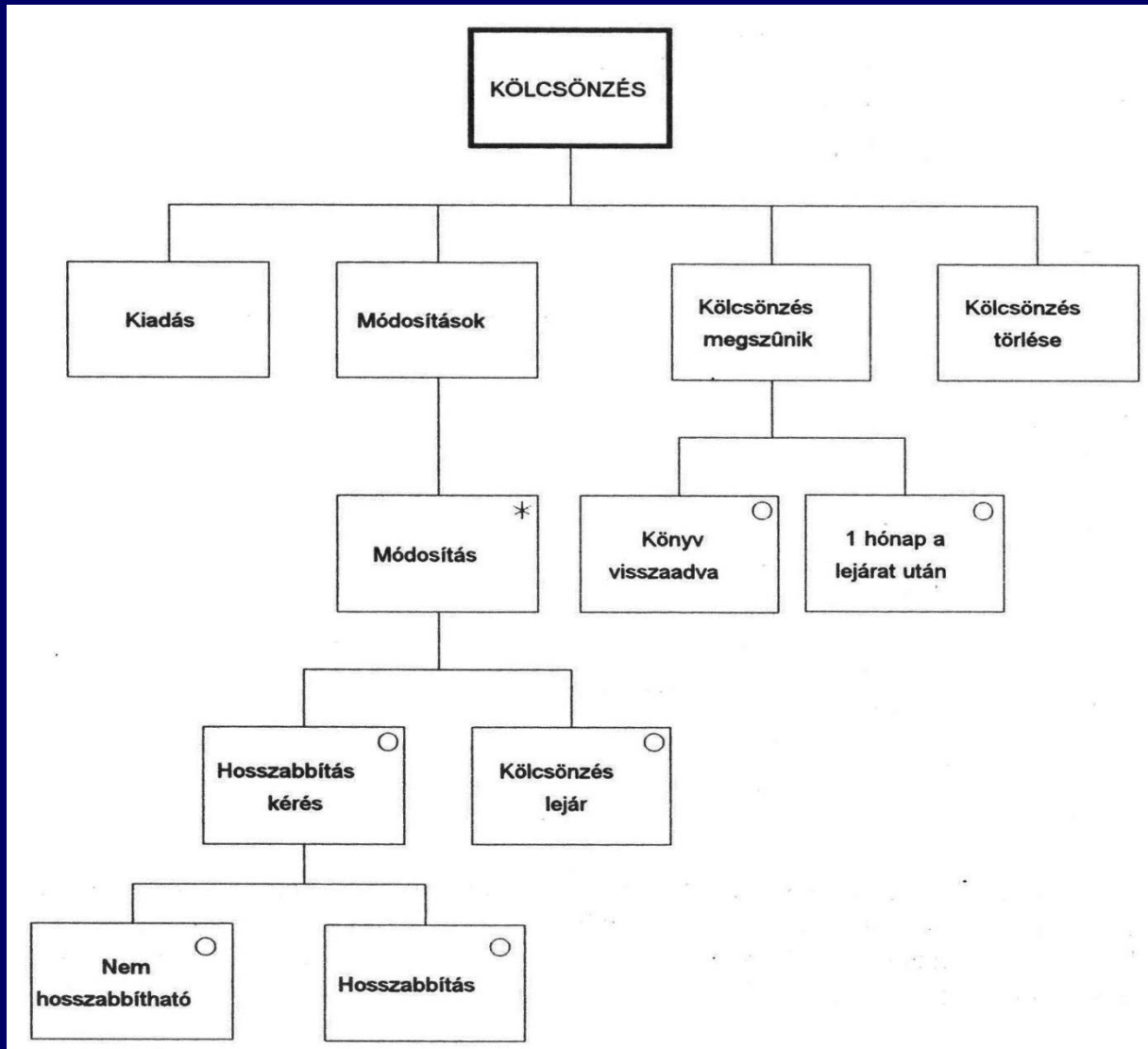
graph TD
    A[KÖLCSÖNZÉS] --> B[Kiadás]
    A --> C[Módosítások]
    A --> D[Könyv visszaadás]
    A --> E[Kölcsönzés törlése]
    C --> F[Módosítás*]

```

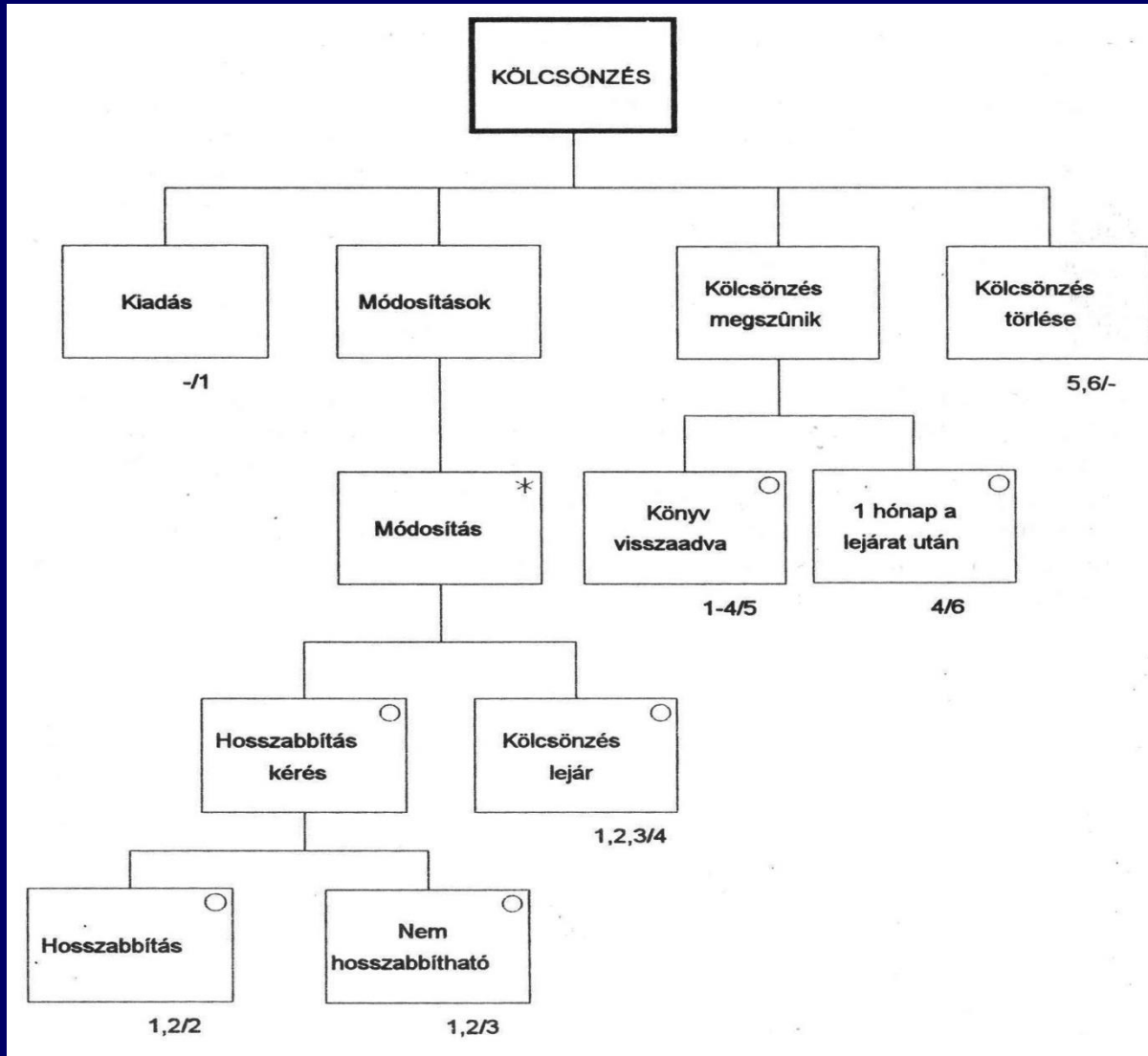
Egyed-élettörténet diagramm



Egyed-éleltörténet diagramm



Egyed-élettörténet diagramm

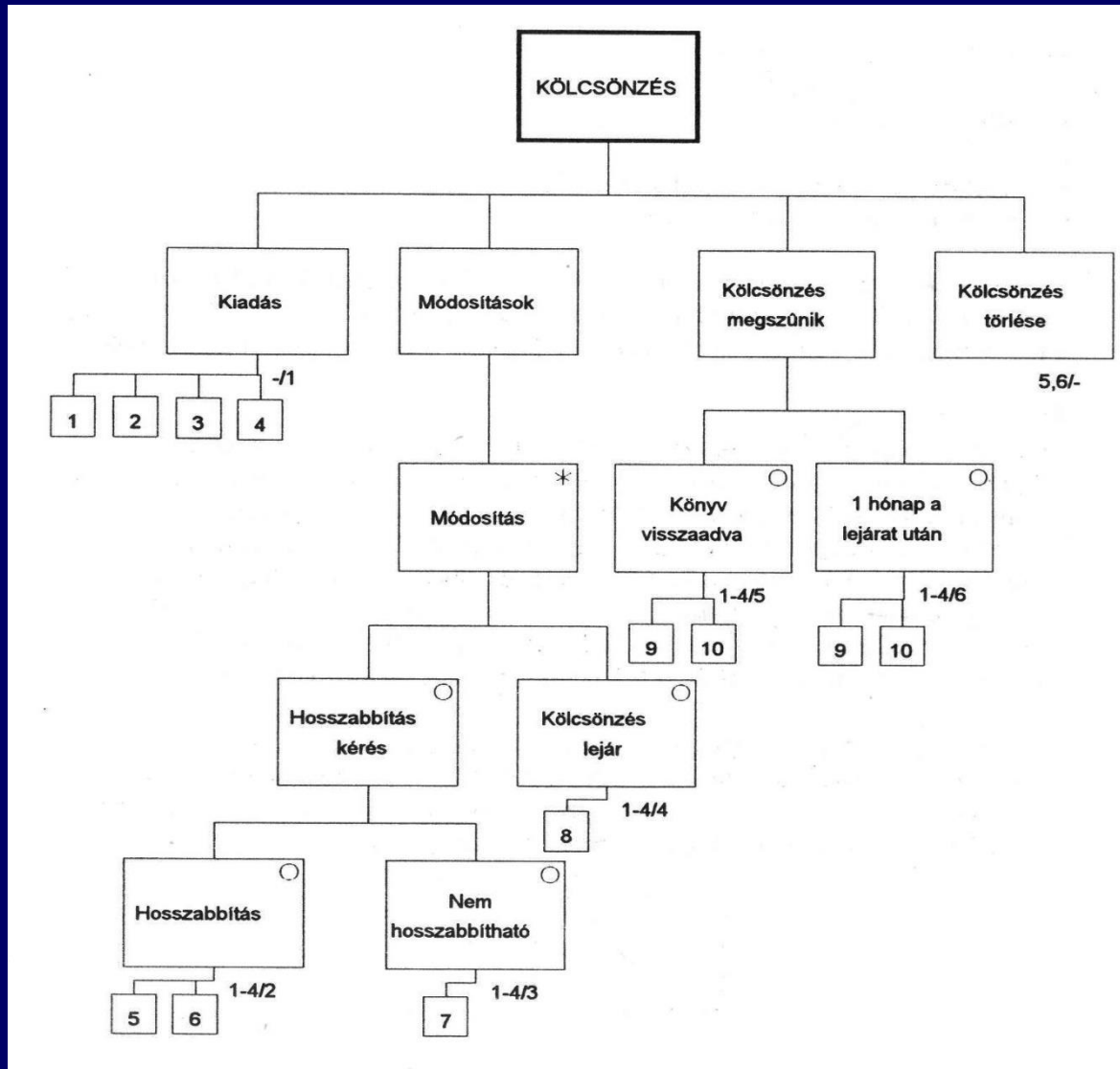


Egyed-éleltörténet diagramm

Az diagrammal kapcsolatos műveletek jegyzéke:

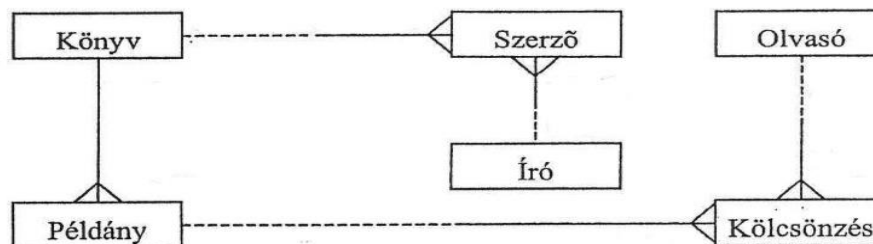
1. Kulcsok beállítása
2. Leíró tulajdonságok értékének beállítása
3. Kapcsolás KÖLCSÖNZŐ-höz
4. Kapcsolás PÉLDÁNY-hoz
5. A lejárat dátum helyettesítése aktuális dátum+14-gyel
6. A hosszabbítás jelzőjének helyettesítése jelző+1-gyel
7. Foglaltság jelző beállítása
8. Lejártság jelző beállítása
9. Leválasztás KÖLCSÖNZŐ-ről
10. Leválasztás PÉLDÁNY-ról

Egyed-élettörténet diagramm

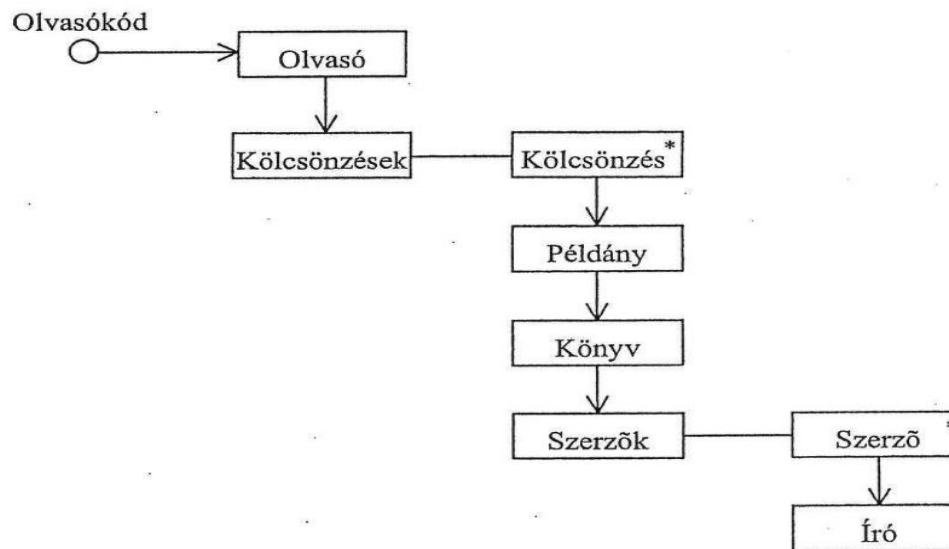


Visszakeresési út modellezése - Folyamat

Egyed-kapcsolat diagram:



Visszakeresési út:



Funkció meghatározás

FUNKCIÓ MEGHATÁROZÁS				SSADM-4	
Projekt/rsz.	Elemző	Dátum	Változat	Allapot	Oldal
BAF	Dr. Bana	93.6.29.	V1	M	1/1

Funkciónév				Funkció azonosító	
Típus					
Felhaszn. szerepek					
Funkcióleírás					
Hibakezelés					
AFD eljárások					
Események				Esemény gyakoriság	
I/O leírások					
I/O szerkezetek					
Követelménykatalógusra hivatkozás					
Tömegszerűség					
Kapcsolódó funkciók					
Lekérdezések				Lekérdezés gyakor.	
Közös feldolgozás					
Dialogusnevek					
Szolgáltatások szintjére vonatkozó körülmények					
Leírás	Célérték	Tűrés	Megjegyzés		

Példa nem
diagrammszerű
technikára

Unified Modeling Language - UML

■ Modell

- A modell egy rendszer (bonyolult probléma vagy szerkezet) absztrakciója, amely a megértést és a kezelhetőséget célozza.
- A modell az adott rendszert egy bizonyos absztrakciós szinten teljes mértékben leírja.

■ Nézet

- A rendszernek lehetnek nézetei. Az egyes nézetek a rendszert más-más szemszögből világítják meg, más-más oldalról mutatják.

■ Modellelem

- A modell alkotóelemei a modellelemek.
- Pl: aktor, használati eset, osztály, objektum, aktivitás, kölcsönhatás, állapot, esemény küldése
- Közöttük kapcsolatok definiálhatók (pl. függőség, társítás)
- Csoportosíthatók (csomag, package).

■ Diagram

- A diagramokat modellelemek és az azok közötti kapcsolatok alkotják.

Unified Modeling Language - UML

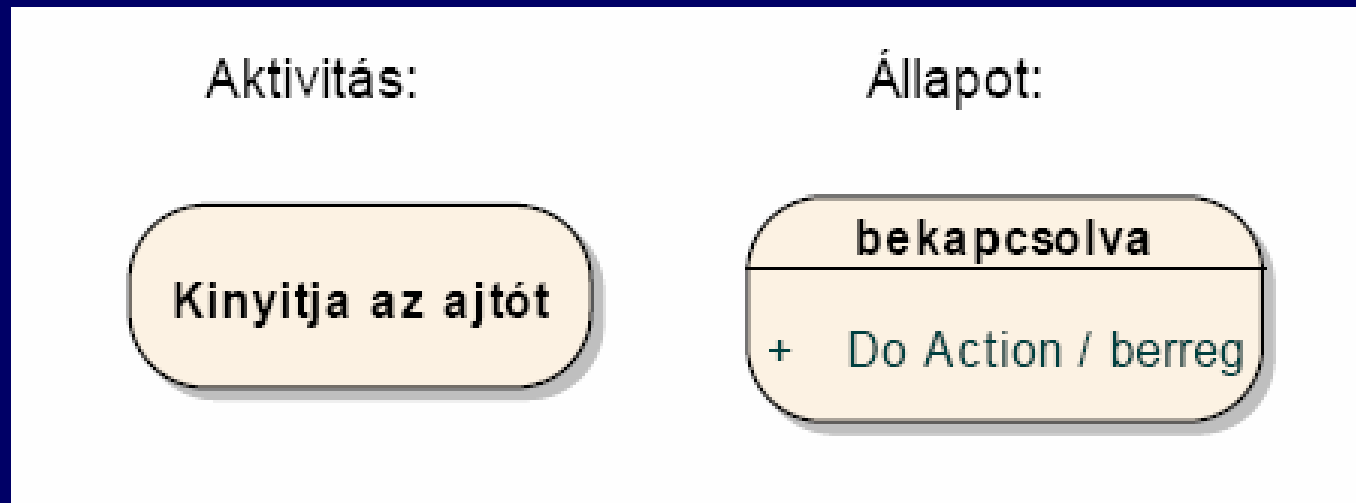
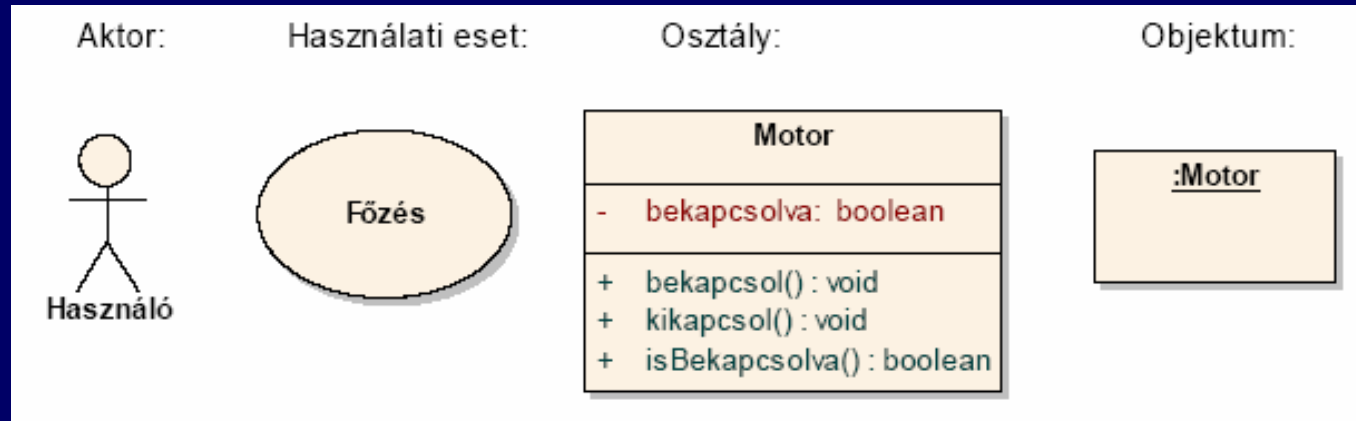
- Az UML egy grafikus modellező nyelv a szoftverrendszer különböző nézeteinek modellezésére.
- Segítségével tervezni és dokumentálni tudjuk a szoftvereket.
- Az UML-ben modellek és diagramok adhatók meg, különböző nézetekben.
- Az UML „csak” egy jelölésrendszer, amely független a szoftverfejlesztési módszertől.
- Az UML nem írja elő , hogy az egyes modelleket, illetve diagramokat mily módon és milyen sorrendben állítjuk elő .

Unified Modeling Language - UML

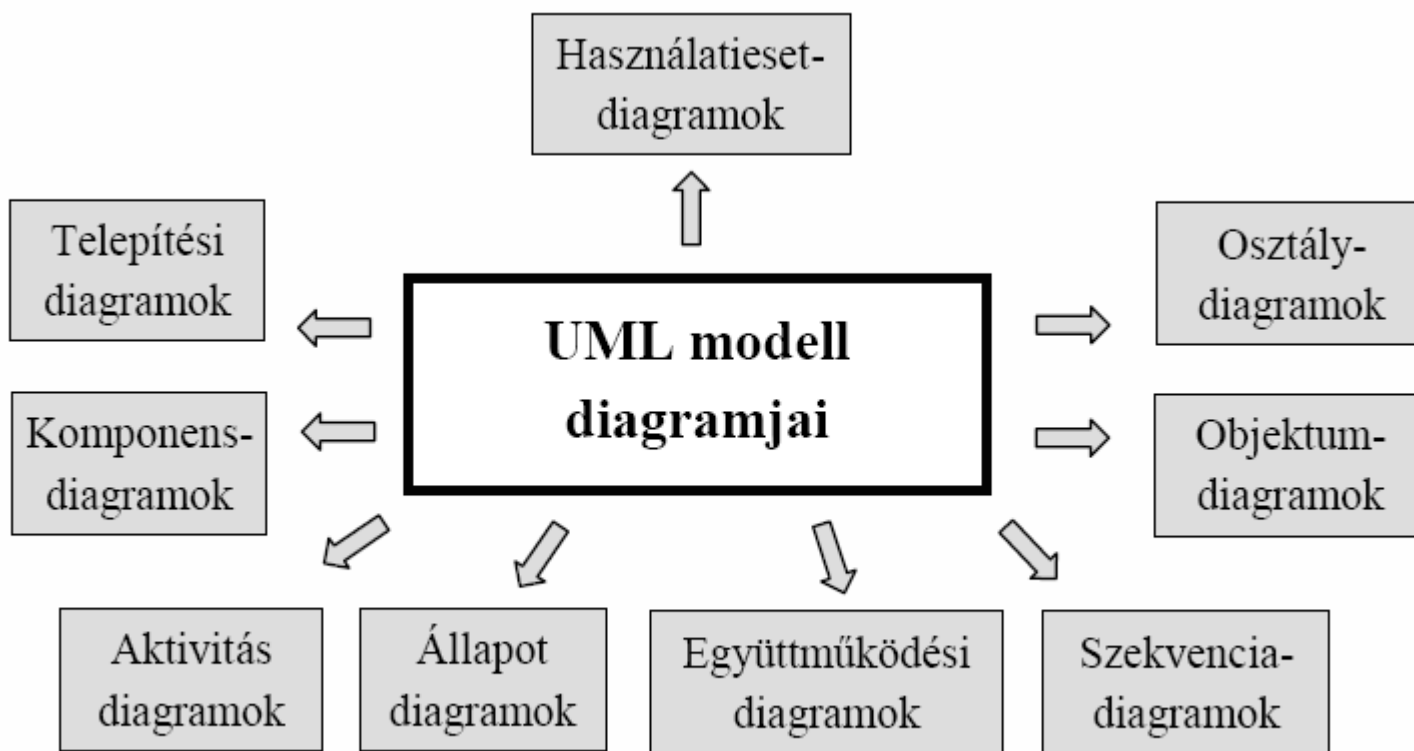
Az OMG (Object Modeling Group) definíciója szerint:

"Az egységesített modellező nyelv (UML) egy grafikus nyelv egy szoftver-intenzív rendszer termékeinek megjelenítésére, specifikálására, felépítésére és dokumentálására. Az UML szabványos lehetőségeket kínál a rendszer felvázolásához, beleértve a fogalmi dolgokat, mint üzleti modellezés és rendszerfunkciók, valamint a konkrét dolgokat, mint programnyelvi utasítások, adatbázis sémák és újrafelhasználható szoftverkomponensek."

Modellelemek - példák



Diagrammok - példák



Struktúramodellezés

- **Osztálydiagram** (Class Diagram): Megadja a rendszer osztályait, és az azok közötti társítási és öröklési kapcsolatokat. Az adatbázisdiagram (database diagram) egy speciális osztálydiagram, amely megadja a rendszer perzisztens adatainak szerkezetét.
- **Objektumdiagram** (Object Diagram): Megadja a rendszer objektumait, és az azok közötti kapcsolatokat. Az osztálydiagram egy „pillanatfelvétele”.
- **Komponensdiagram** (Component Diagram): Megadja a szoftver fizikai felépítést, vagyis a tervezési elemek szoftver elemekre való leképezését.
- **Telepítési diagram** (Deployment Diagram): Megadja, hogy milyen szoftver elemeket milyen hardverre telepítünk.
- **Összetett szerkezet diagram** (Composite Structure Diagram): az osztály belső szerkezetét mutatja meg.
- **Csomag diagram** (Package Diagram): a rendszer elemeinek logikai csoportokra való osztását és azok összefüggéseit ábrázolja

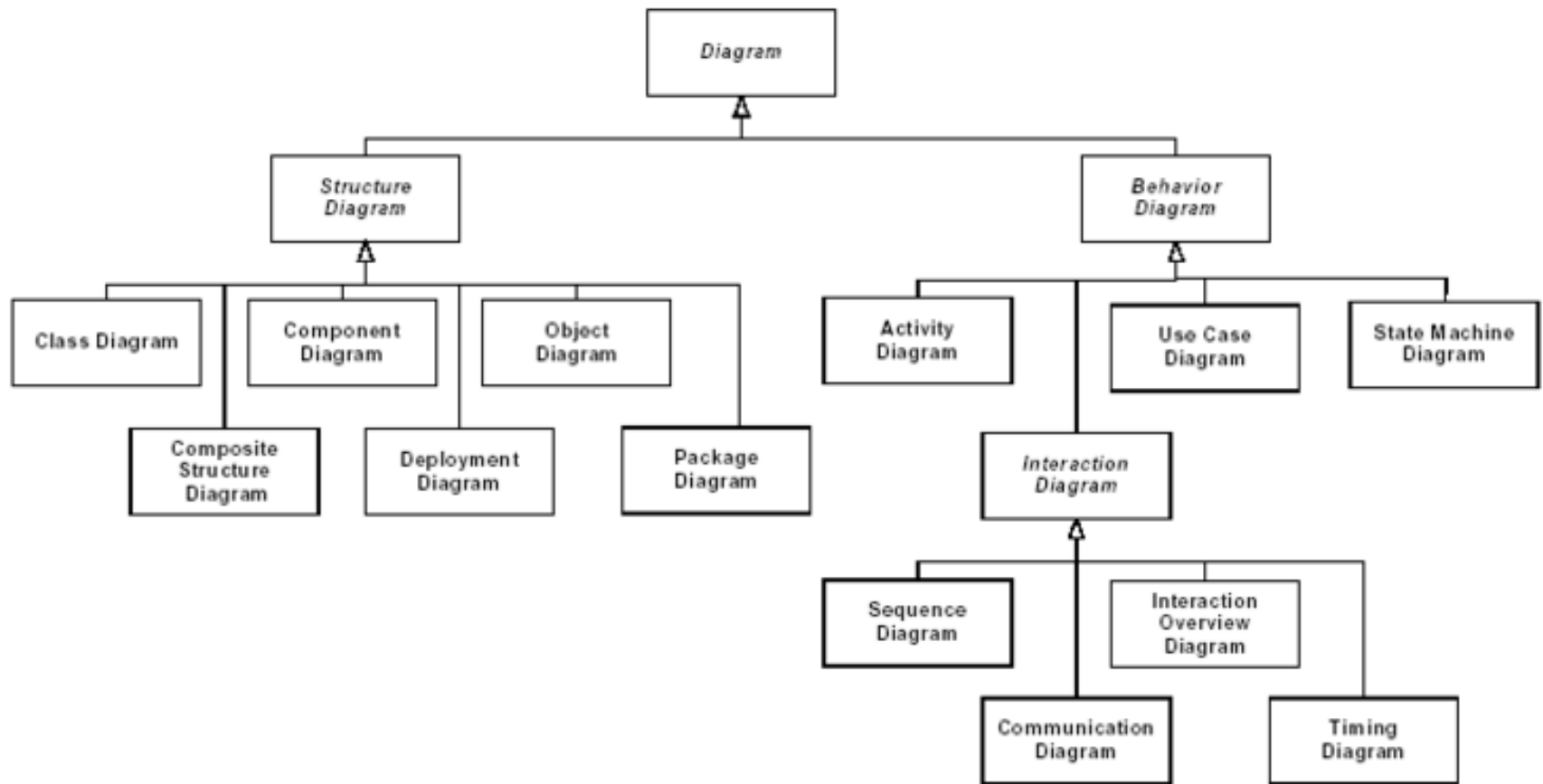
Viselkedésmodellezés

- **Használatieset-diagram** (Use Case Diagram). Megadja, hogy a felhasználó mire tudja használni a rendszert. Aktorokat, használati eseteket és azok kapcsolatait ábrázoló diagram. Funkcionális követelményeket ábrázoló diagram.
- **Állapotdiagram** (State Diagram): Egy adott osztály vagy alrendszer állapotváltozásait írja le.
- **Aktivitásdiagram** (Activity Diagram): Leír egy folyamatot (tevékenységek egymásutánját). Az üzleti folyamat diagram egy speciális aktivitásdiagram, mely leírja a rendszert körülvevő folyamatokat, illetve azt a környezetet, amelybe a rendszert el kell helyezni.

Interakció

- **Szekvenciadiagram** (Sequence Diagram): Aktorokat, objektumokat és az azok közötti kapcsolatokat, kölcsönhatásokat (üzeneteket) ábrázoló diagram. A szekvenciadiagram olyan interakció diagram, mely az idő múlására helyezi a hangsúlyt.
- **Együttműködési diagram** (Collaboration (UML 1.x)/Communication Diagram (UML 2.0)): Megadja a rendszer objektumait, az azok közötti kapcsolatokat és üzeneteket. Az együttműködési diagram a szekvenciadiagram egy más formája – olyan interakció diagram, mely az objektumok közötti kapcsolatra helyezi a hangsúlyt.
- **Interakciós áttekintő diagram** (Interaction Overview UML 2.0) Az aktivitás diagram egy változata, mely egy rendszer vagy üzleti folyamaton belül áttekinti a vezérlő folyamatokat. A diagram minden csomópontja/tevékenysége egy további interakciós diagramot jelenthet.
- **UML Idődiagram** (Timing Diagram UML 2.0) Az elemek állapotainak és feltételeinek időbeli változásait írja le.

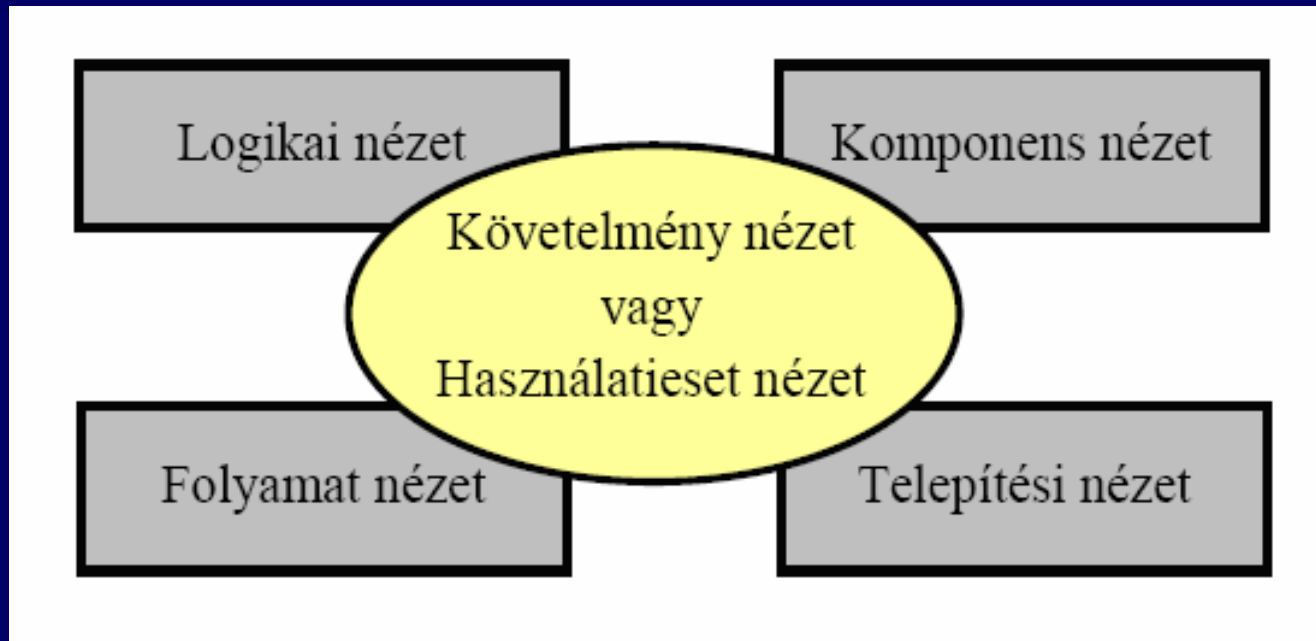
Diagrammok



Az UML modelljei

- **Üzleti modell:** Bemutatja a rendszer üzleti folyamatait.
- **Követelménymodell:** Megadja a rendszerrel szemben támasztott követelményeket. Megadja, hogy a rendszert mire akarják majd használni (funkcionális követelmények), és hogy milyen egyéb feltételeknek kell teljesülniük (nemfunkcionális követelmények; pl. képernyőn való megjelenés, nyomtatás, teljesítmény, tesztelés).
- **Tervezési modell:** Megoldási útmutató .
- **Implementációs modell:** Kivitelezési útmutató .
- **Teszt modell:** A rendszer működésének kipróbálására vonatkozó útmutató .

Az UML nézetei



■ Követelmény nézet (Requirements View) vagy Használatieset-nézet (Use Case View)

- A rendszer modellezése a felhasználó , megrendelő szemszögéből.
- A rendszerrel szemben támasztott felhasználói követelmények; kik, és mire akarják használni a rendszert.
- Itt írjuk le a projekt funkcionális és nemfunkcionális követelményeit.
- Ez a nézet a fejlesztés kezdeti fázisaiban lényegében kialakul, és végigkíséri a teljes fejlesztést.

■ Dinamikus nézet (Dynamic View) vagy Folyamat nézet (Process View)

- A rendszer működését, folyamatát, dinamikáját ábrázolja.
- Elsősorban interakció , aktivitás és állapotdiagramokat tartalmaz.

■ Logikai nézet (Logical View) vagy Tervezési nézet (Design View)

- A rendszerben szereplő osztályokat, interfészeket, illetve ezek kapcsolatát mutatja
- Ez a programterv lényegi része, ez alapján történik a kódolás, illetve a kódgenerálás.

■ Komponens nézet (Component View) vagy Implementációs nézet (Implementation View)

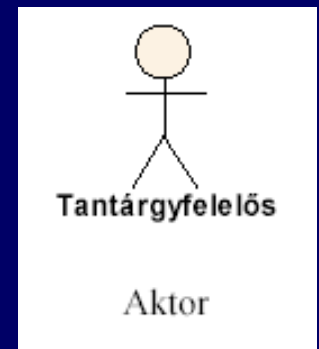
- A rendszert alkotó fizikai komponensek, szoftverelemek modellezése.
- A logikai nézet osztályainak forráskomponensekhez való hozzárendelése, valamint a forráskódok hozzárendelése futtatható komponensekhez.
- A rendszer állományait modellezi.
- Jellemző diagramja a komponensdiagram.

■ Telepítési nézet (Deployment View)

- A rendszer topológiájának modellezése.
- Leírja a rendszerszoftver elemeinek a hardver elemekhez való rendelését.
- Itt adható meg, hogy milyen számítógépen milyen szoftver fut.
- A nézet jellemző diagramja a telepítési diagram.

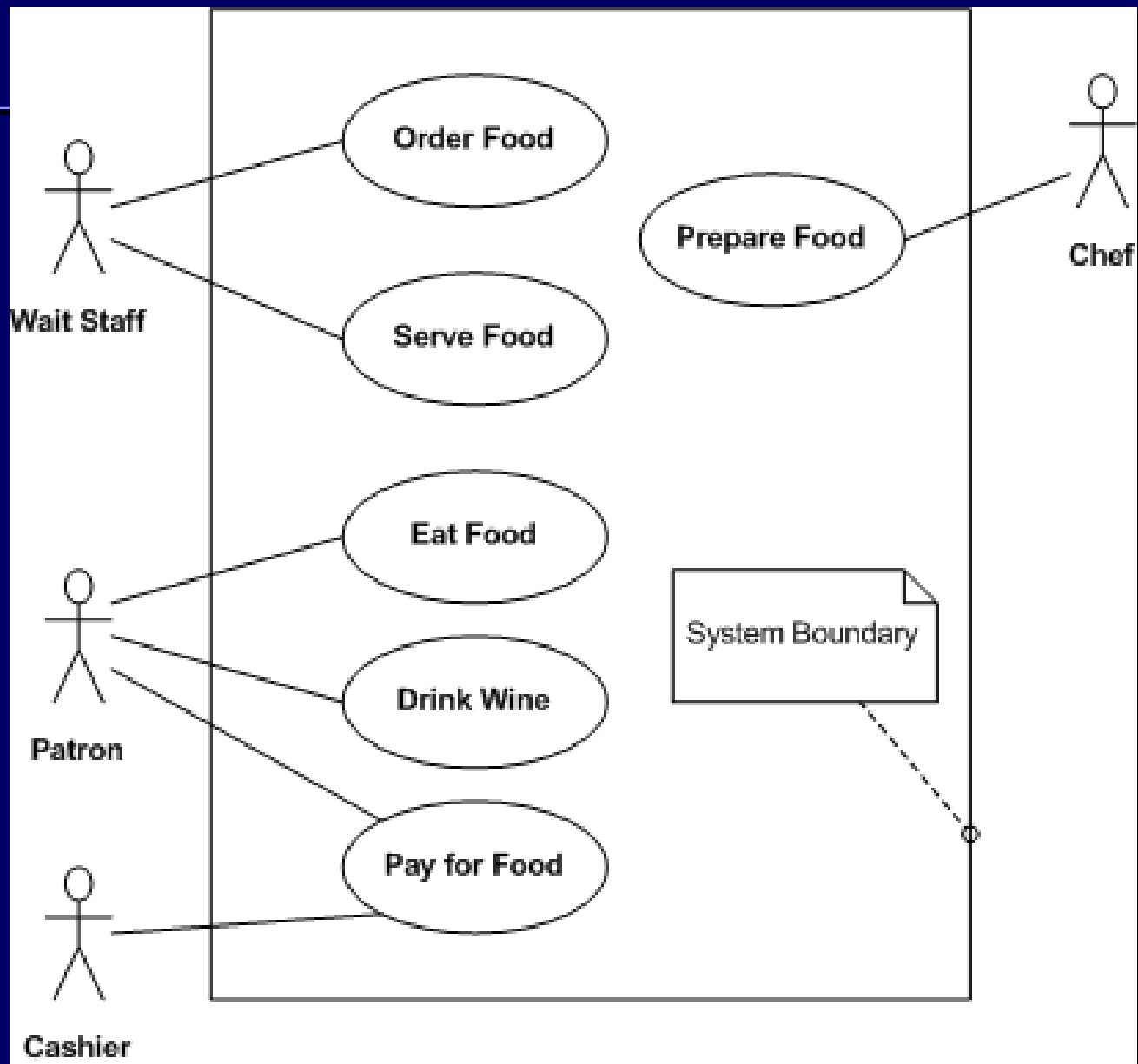
Használatieset-diagram

- A használatieset-diagram (HE) a rendszer viselkedésének egy kiragadott részét írja le külső aktorok szemszögéből.
- A HE diagram elemei:
 - aktorok,
 - használati esetek,
 - valamint az ezek közötti kapcsolatok.
- Egy aktor üzeneteken keresztül kommunikál a rendszerrel. Az aktor üzenetet küld a rendszernek, a rendszer pedig a használati eset végrehajtása közben üzeneteket küldhet vissza az aktor számára.



Használatieset-diagram

- Használati eset: a használatnak egy értelmes, kerek egysége (pl. egy szövegszerkesztő programmal kinyomtatunk egy dokumentumot)
- Aktor: aki vagy ami a rendszert használja (ember vagy a rendszerhez csatlakozó külső egység, pl. mérőeszköz)



Használatieset-diagram

- A használati eset a szoftver használatának egy értelmes egysége, az aktor kommunikációja, párbeszéde a szoftverrel.
- A használati eset a rendszer viselkedését írja le a rendszeren kívülről.
- A használati esetek vezérlik a fejlesztést.
- Csak a MIT írja le, a HOGYAN-t nem.
- Mindig az aktor indítványozza.
- A használati eset az UML osztályszerű eleme, melynek egy példánya a forgatókönyv.

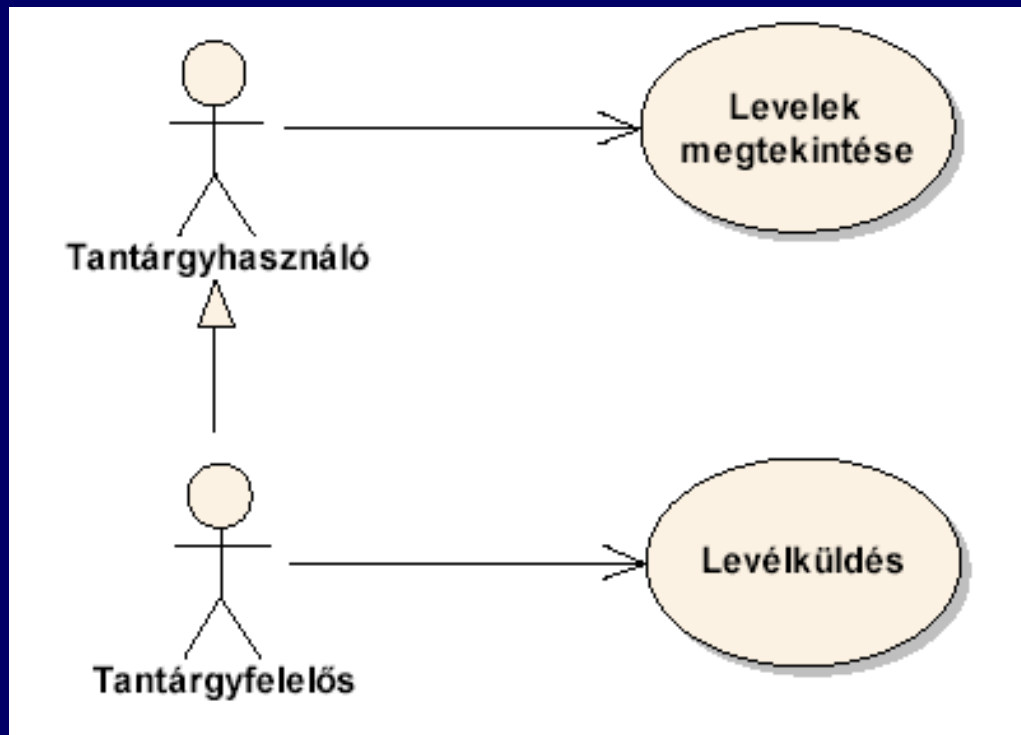
Használatieset-diagram

- A forgató könyv a használatnak egy konkrét esete. A forgatókönyvet interakció diagramokkal lehet modellezni, kifejteni.
- Egy használati esetnek több forgatókönyve lehetséges (pl. bejelentkezés)
- A használati eseteket szövegesen le kell írni, dokumentálni kell.
- Jele: ellipszis

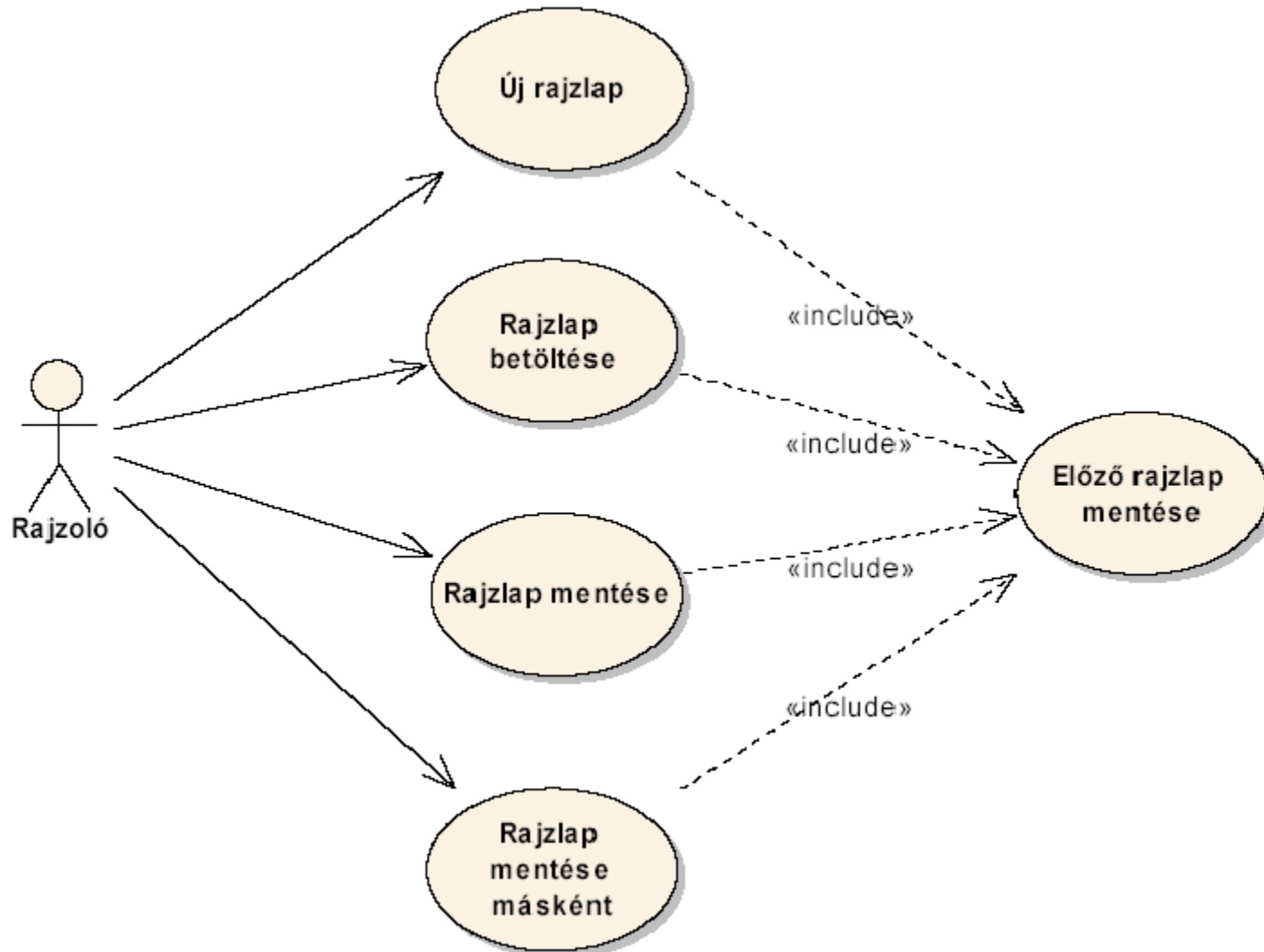


- A diagram elemei között lehetséges kapcsolatok:
 - Két aktor között: öröklési és kommunikációs.
 - Aktor és HE között: társítási (kommunikációs) kapcsolat. Van kezdeményező és résztvevő aktor.
 - Két HE között: tartalmazás (include), kiterjesztés (extends), öröklés

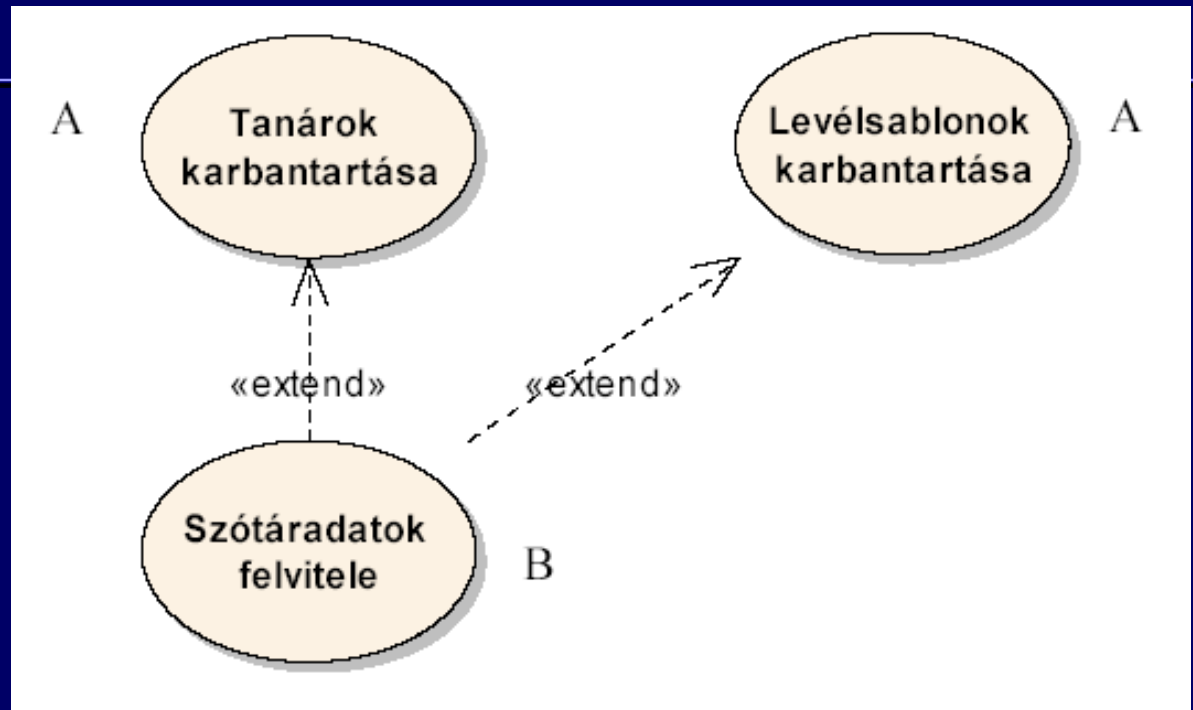
Két aktor között lehetséges öröklési kapcsolat.



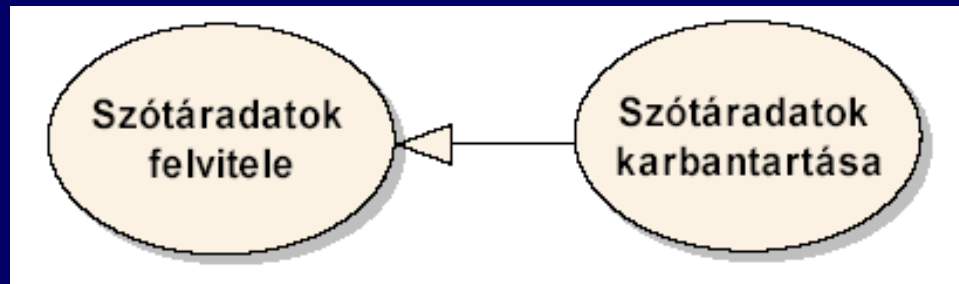
Két használati eset tartalmazási kapcsolata

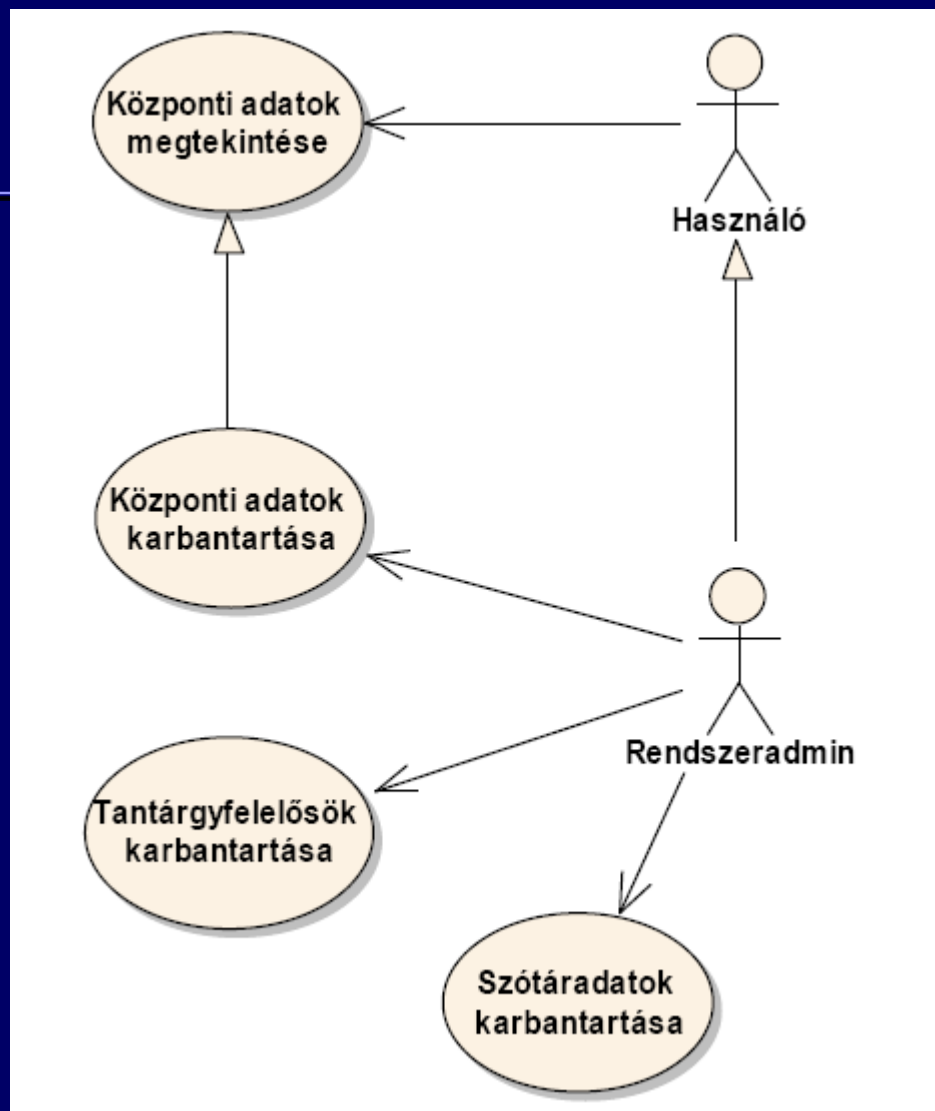


Bővítési kapcsolat



Öröklési kapcsolat





Tanári levelező program használatieset-diagramja (részlet)

- A HE diagramok és leírások alkotják a használatieset-modellt.
- A fejlesztés HE centrikus; a használati esetek a teljes fejlesztés során központi szerepet játszanak (ld. RUP).
- Használati esetek vezérlik a következő dolgokat:
 - felhasználói igények (követelmények) meghatározása és ellenőrzése;
 - fejlesztés ütemezése;
 - analízis és tervezés;
 - tesztelés;
 - felhasználói dokumentáció struktúrája;
 - átadás.
- A rendszer használati esetei nem mindig egyértelműek. Ugyanahhoz a feladathoz többféle HE diagram is felrajzolható. Fontos, hogy a használati esetek lefedjék a rendszer funkcióit.

A használati eset dokumentációja

- **Leírás:** Érthető szöveg, melyben röviden felvázoljuk a HE célját, jellemzőit.
- **Aktor:** A HE használója (használói).
- **Megszorítások (korlátozások):** Szabályok, amelyek megtehetőek vagy meg kell tenni.
 - Előfeltételek, amelyeknek teljesülniük kell a HE végrehajtása előtt.
 - Utófeltételek, melyek majd teljesülnek a HE végrehajtása után.
 - Alapfeltételek: Olyan dolgok (állítások, feltételek), melyek az egész HE során igazak.

A használati eset dokumentációja

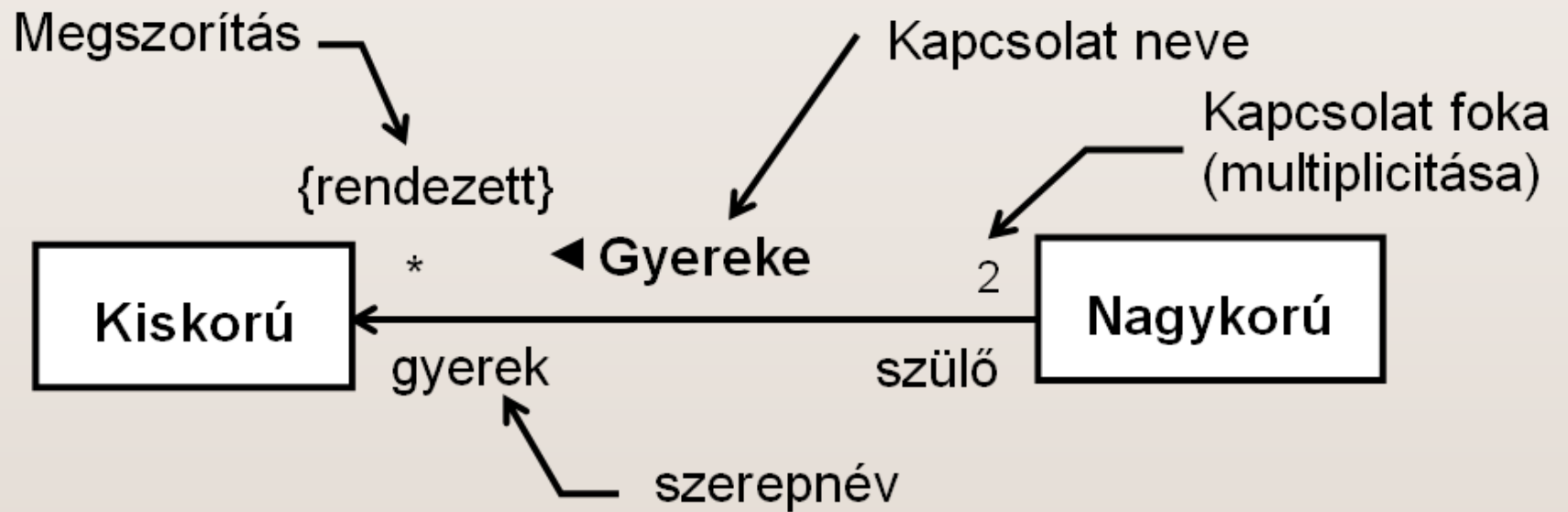
- **Forgatókönyvek** (eseményfolyamok): Egy forgatókönyv a HE egy konkrét végrehajtása.
 - Cél, hogy a HE minél több jellemző ágát bejárjuk.
 - Minden egyes, az előzőtől különböző forgató könyv megadása közelebb visz a megoldáshoz.
 - A forgatókönyvek segítségével újabb osztályok, szabályok fedezhetők fel.
 - Egy forgató könyv alátámasztható interakció diagrammal (szekvencia vagy együttműködési).

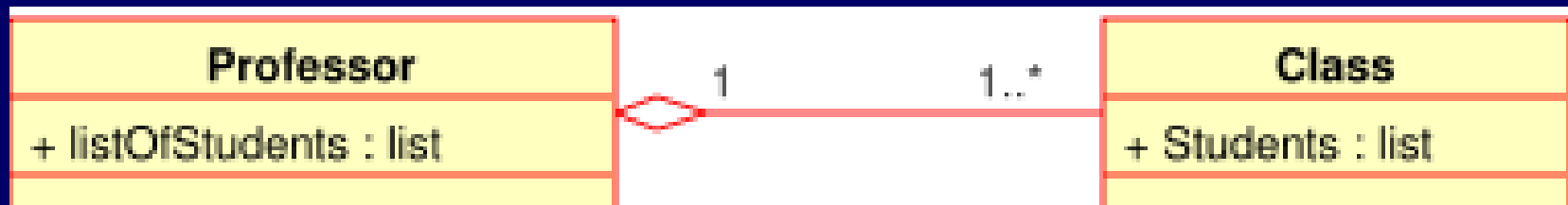
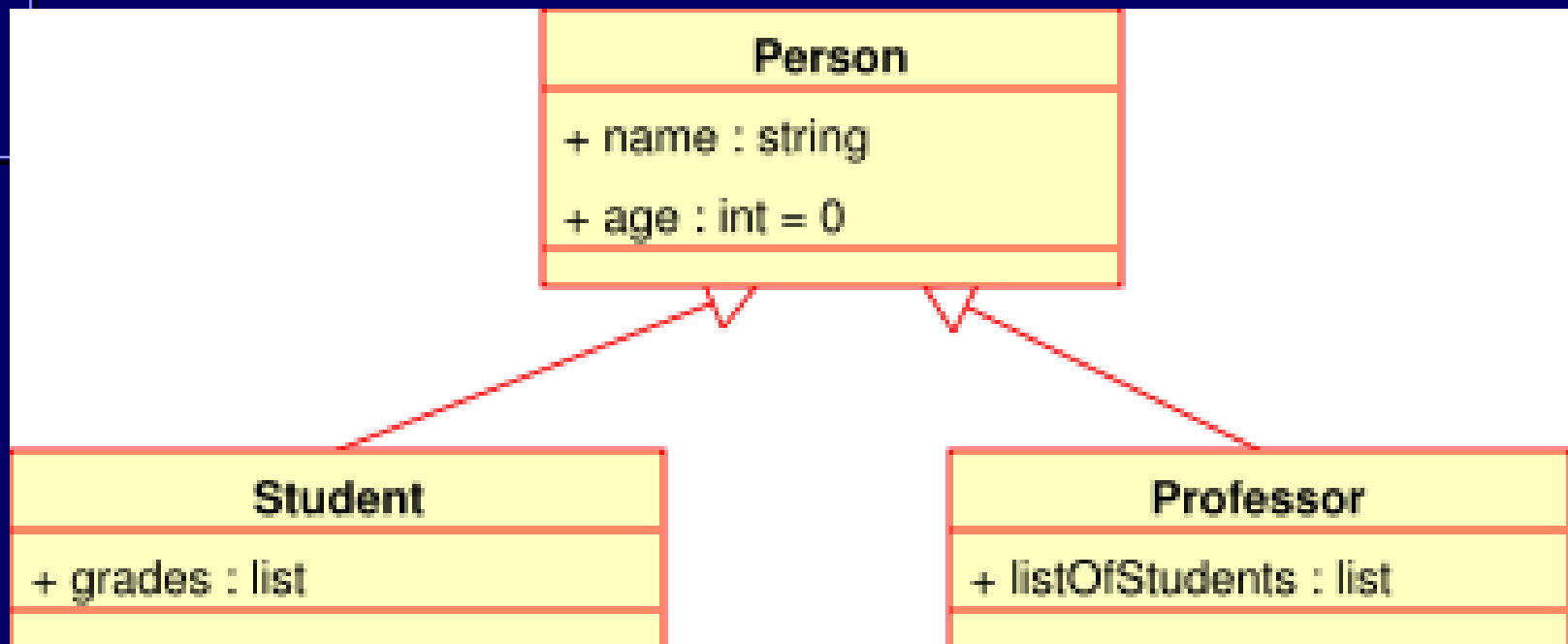
Pl. Főzés – használati eset

- **Leírás** A Használó beteszi az ételt a mikróba, és azt egy adott ideig főzi. A főzés használati eset összetett, annak számtalan forgatókönyve lehetséges.
- **Előfeltétel** Nincs.
- **Utófeltétel** Az étel meg van főzve.
- **Egy perc főzés – forgatókönyv**
A Használó kinyitja az ajtót, beteszi az ételt, becsukja az ajtót. Megnyomja a Perc plusz gombot. Elindul a főzés, a lámpa ég. Egy perc lejártával befejeződik a főzés, a lámpa kialszik. Használó kinyitja az ajtó t, a lámpa kigyullad. Kiveszi az ételt. Becsukja az ajtót, a lámpa elalszik.
- **Három perc főzés, közben 1:33-nál ajtónyitás – forgatókönyv**
A Használó háromszor lenyomja a Perc plusz gombot. Az elsőre elindul a melegítés. 1:33-nál kinyitja az ajtót, mire a főzés félbeszakad. Később becsukja az ajtót, a főzés újra indul onnan, ahol abbamaradt. A három perc lejártával a sütő leáll, a lámpa kialszik. A használó kinyitja az ajtót.

Osztálydiagram

- Osztálydiagram (class diagram): Olyan diagram, amely az osztályokat és a közöttük lévő társítási és öröklési (általánosítási) kapcsolatokat ábrázolja.
- Az objektumdiagram az osztálydiagram előfordulása, példánya.
- Az osztálydiagram rögzíti az objektumok közötti kapcsolatok szabályait.
- Két osztály közötti társítási kapcsolat főbb jellemzői:
 - ismeretségi vagy tartalmazási kapcsolat;
 - név;
 - multiplicitás (egy–egy, egy–sok vagy sok–sok jellegű; kötelező vagy opcionális);
 - szerepnév;
 - megszorítás.

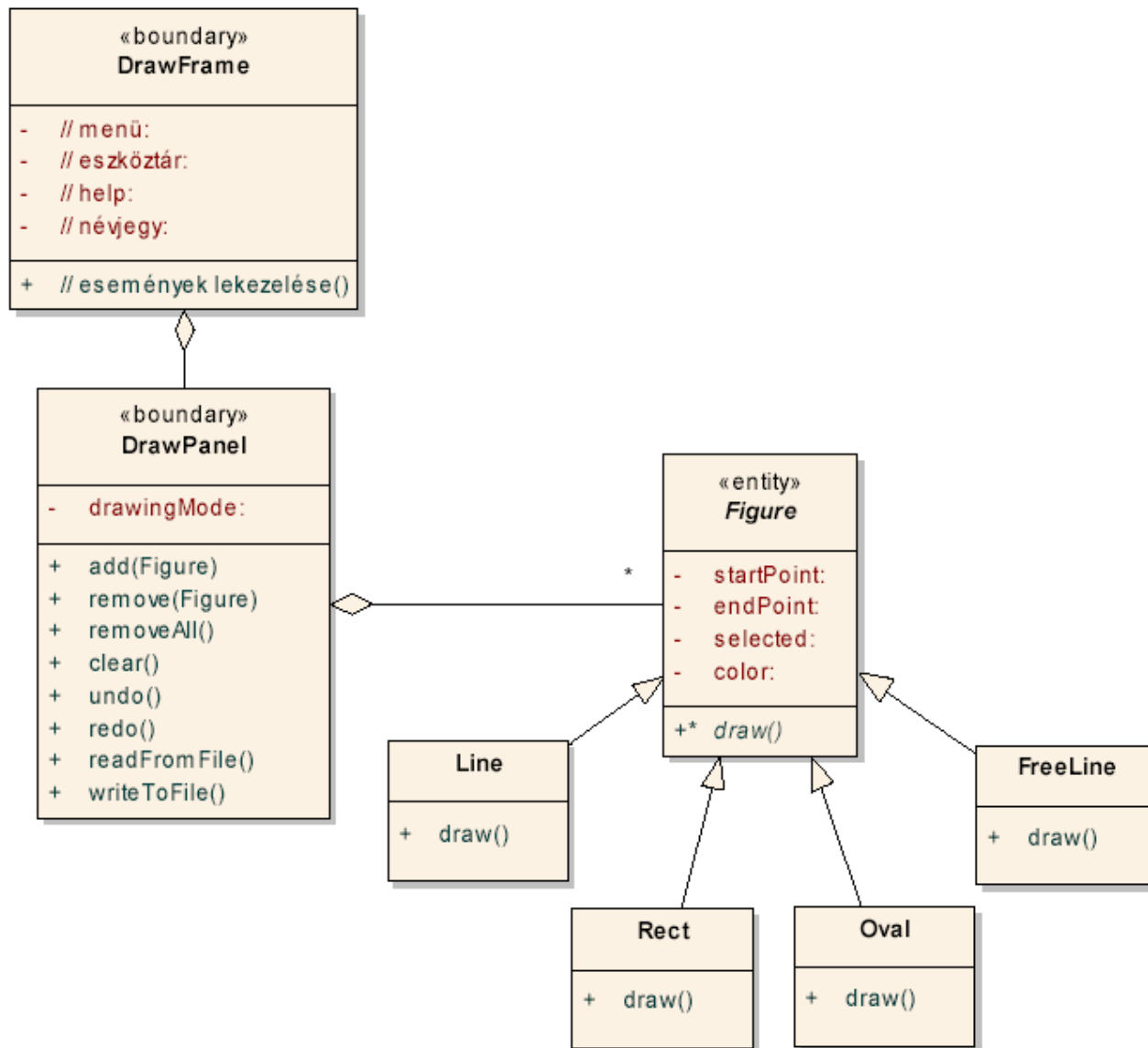


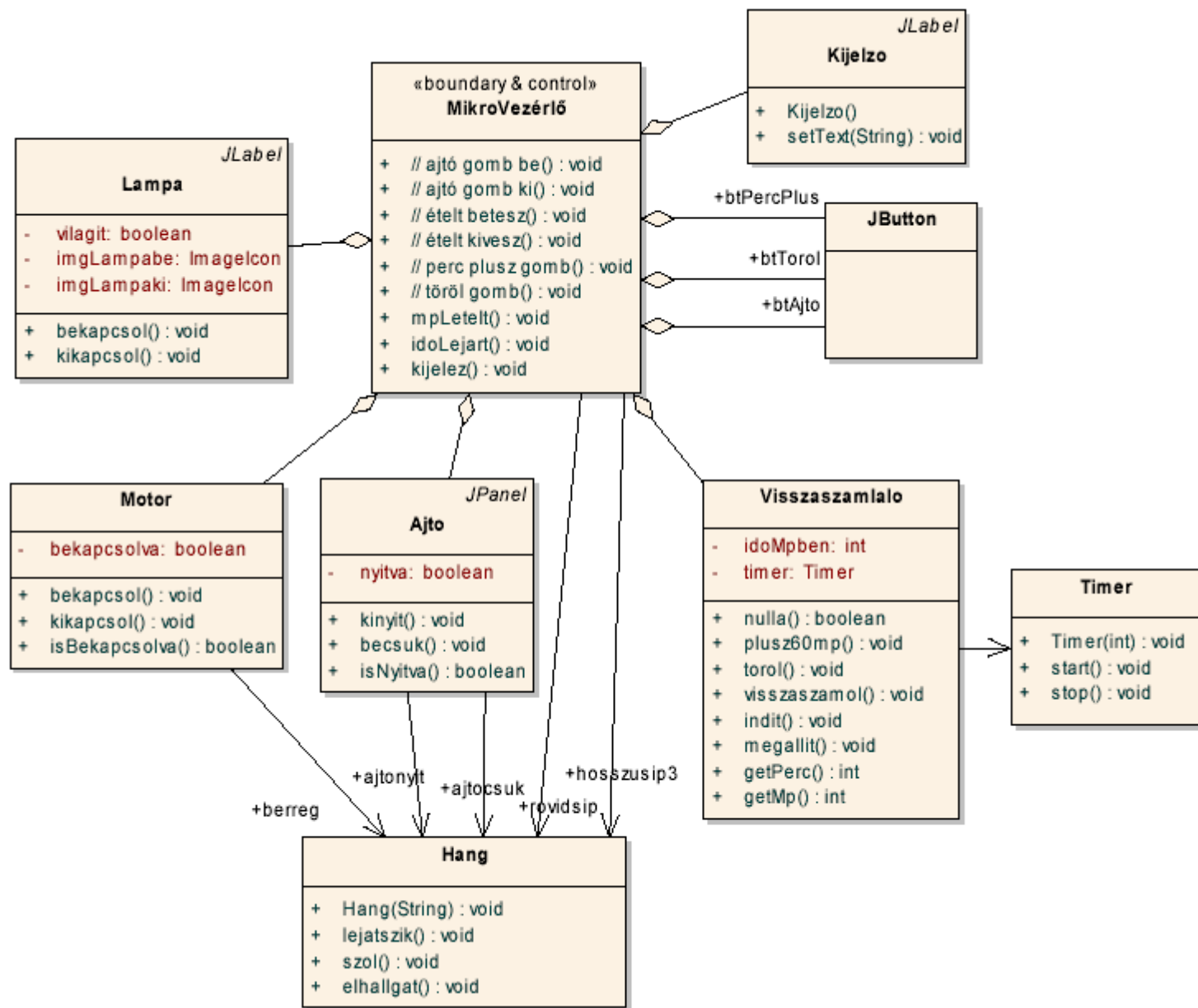


■ A rendszer modellezésének végső célja

- a rendszerben szereplő osztályok és azok kapcsolatainak meghatározása,
- az osztályok forráskódhoz rendelése és kódolása,
- a szoftver komponensek konkrét hardver eszközökhöz való hozzá rendelése.

■ A modellezés minden változata e cél szolgálatában áll.





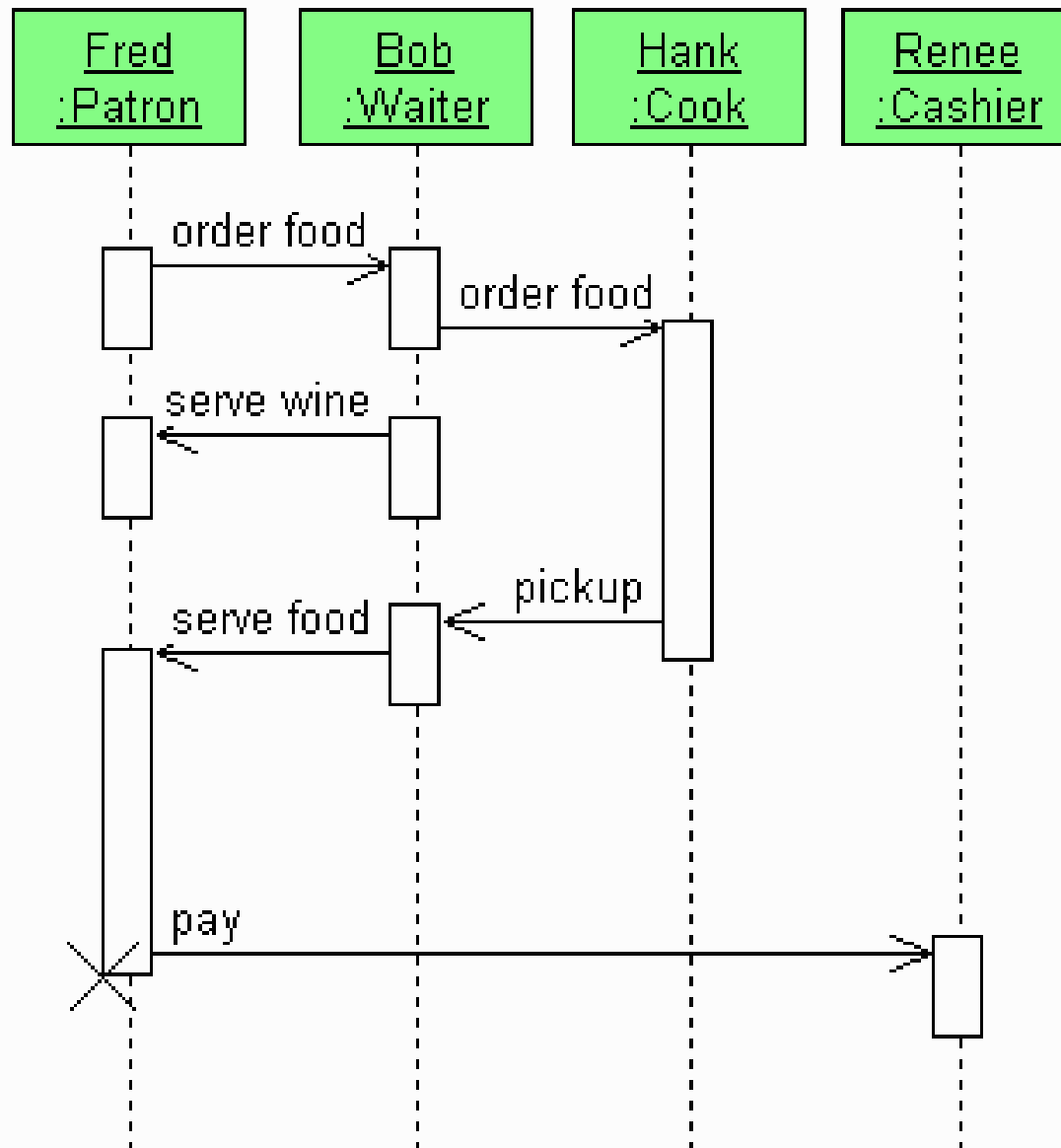
Interakciódiagramok

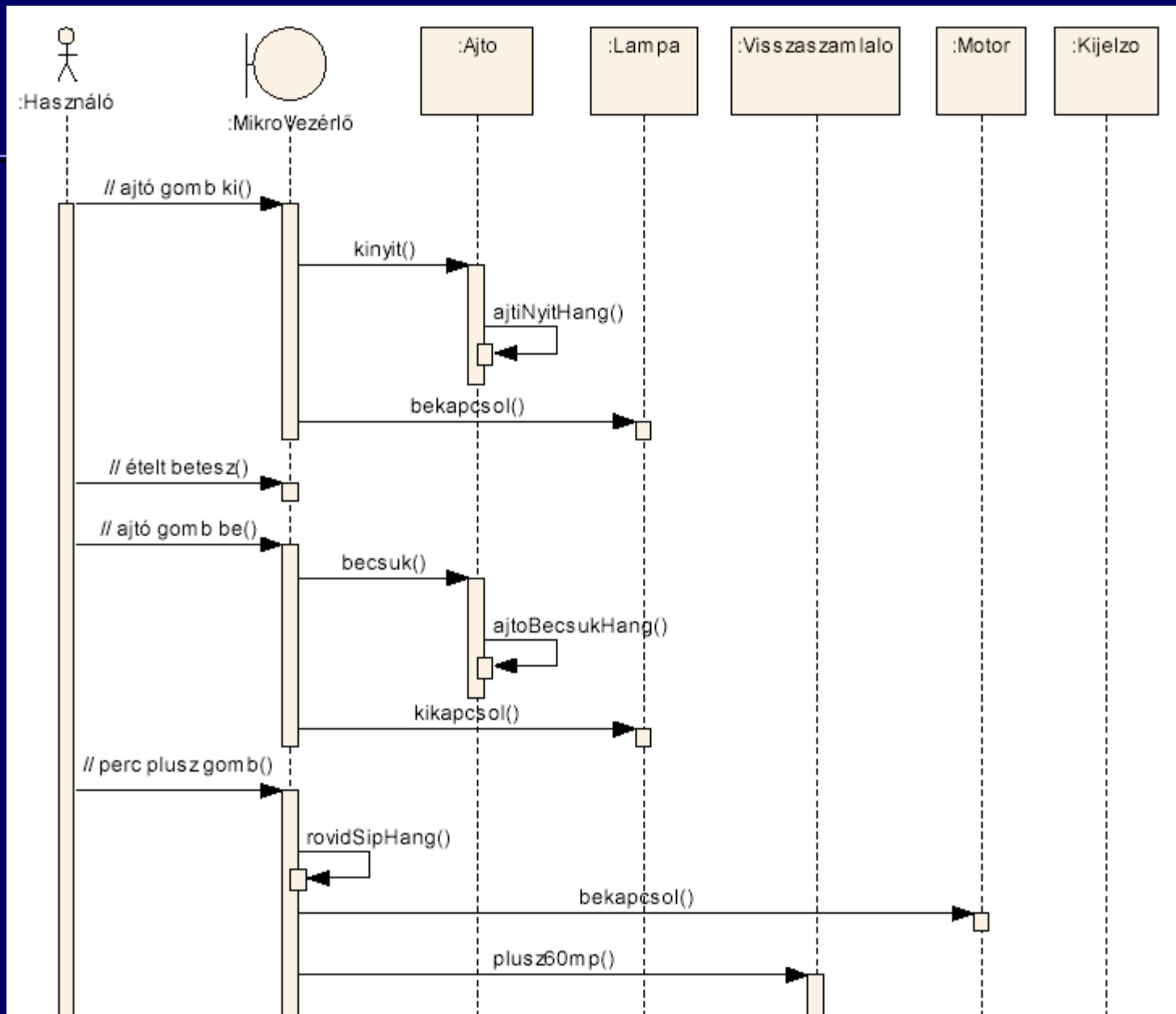
- Az interakció diagram objektumokat és az azok közötti üzeneteket ábrázolja.
- Segítségével az objektumok együttműködését, a végrehajtandó üzenetsorokat szemléltethetjük.
- Egy interakció diagram tipikusan egy használati eset egy forgatókönyvét írja le.
- A diagram használatával az objektumok/osztályok közötti felelősségek könnyebben feltárhatók, illetve szétoszthatók.
- A két legfontosabb interakció diagram : **szekvenciadiagram** és **együttműködési diagram**.
- A két diagram ugyanazt az információt tárolja, a különbség megjelenésükben van:
 - míg a szekvenciadiagramon az idő a fontos,
 - az együttműködési diagramon az objektumok kapcsolata az elsődleges információ.
- A két diagram átalakítható egymásba.

- Egy használati eset forgatókönyveit interakció diagrammal szemléltetjük.
- A szoftver tervezésekor minden egyes aktor–HE párhoz nulla, egy vagy több interakció diagramot szokás készíteni, a HE bonyolultságától függően.
- Az objektumoknak küldött üzenet a megfelelő osztályban megjelenik.
- Az interakció diagram elsődlegesen a használati esetek analizálásának eszköze.
- Nem a lépéssorozat struktúrája (elágazások, iterációk) a lényeg, hanem az, hogy mely objektumoknak milyen üzenetet küldünk. A küldött üzenet ugyanis az objektum osztályában szerepelni fog.
- A küldés vonalán kirajzolódnak az osztályok, az osztályok közötti kapcsolatok, valamint az osztályok metódusai.
- Az interakció diagram alapján az osztálydiagram egy része vagy egésze felépíthető

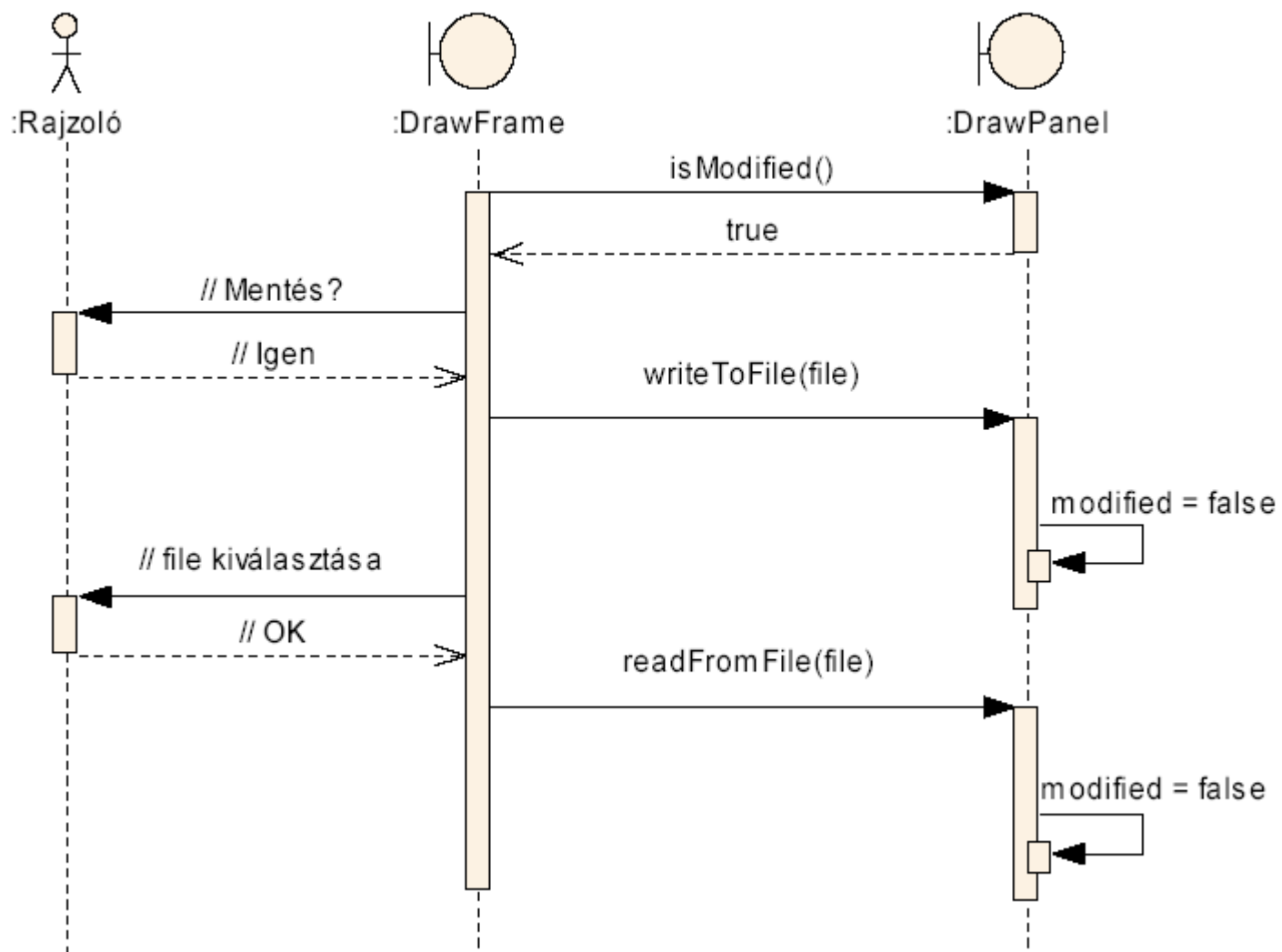
Szekvenciadiagram

- A szekvenciadiagramon az idő a fontos.
- Az X tengelyen vannak az objektumok, az Y tengely az életvonal (object lifeline).
- Az idő fentről lefelé telik.
- Két objektum között itt nem mutatható ki közvetlenül a kapcsolat, a kapcsolat az üzenetek által érzékelhető
- Az üzenetek sorszámozhatók.
- Feltüntethetünk elágazást, iterációt
- A szekvenciadiagram segítségével elsősorban azt akarjuk feltérképezni, hogy „ki üzen kinek”.
- Az egyes metódusok pontos struktúráját főleg az állapot- és aktivitási diagramokkal fogjuk megállapítani.

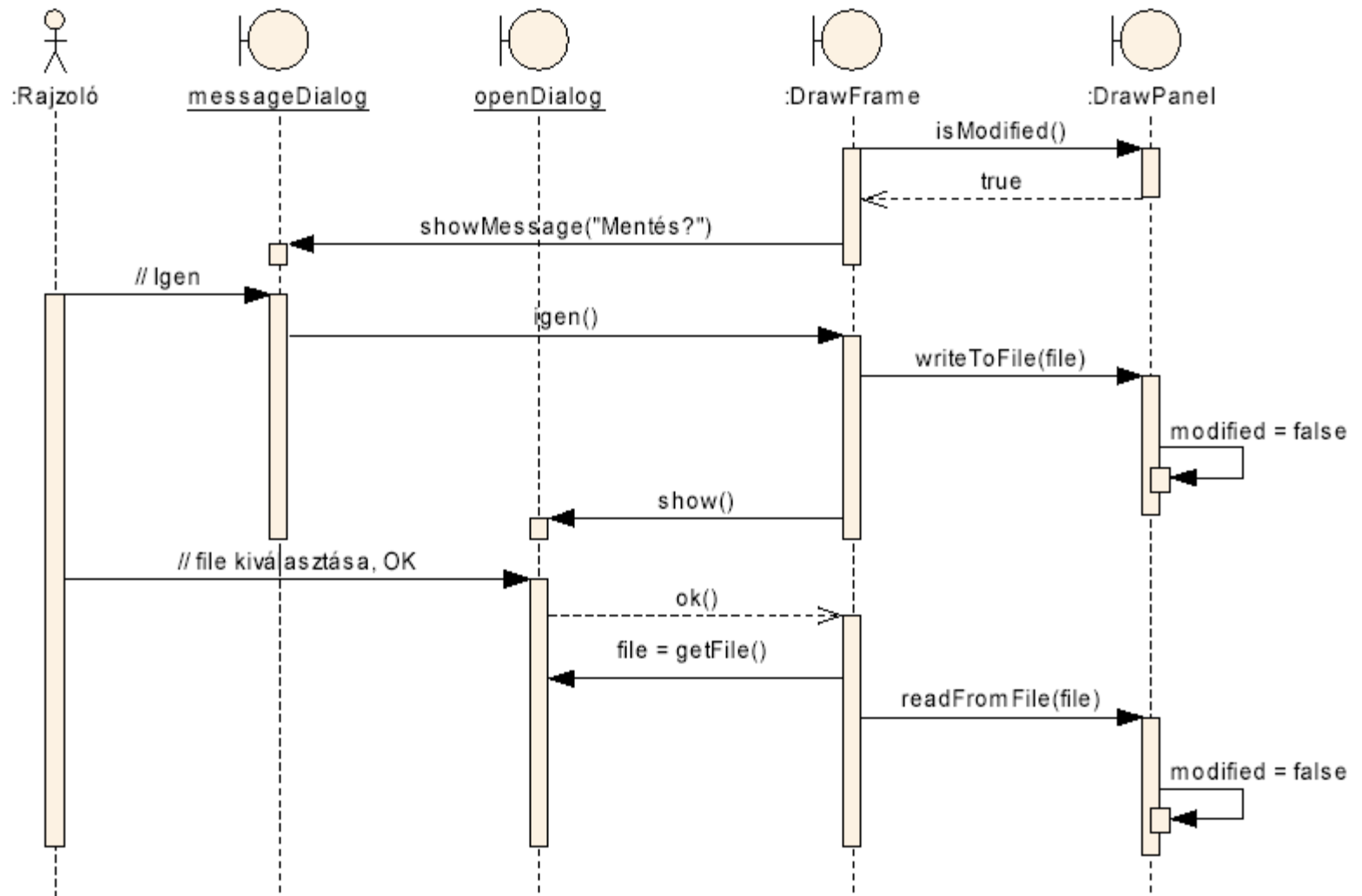




Rajzlap betöltése

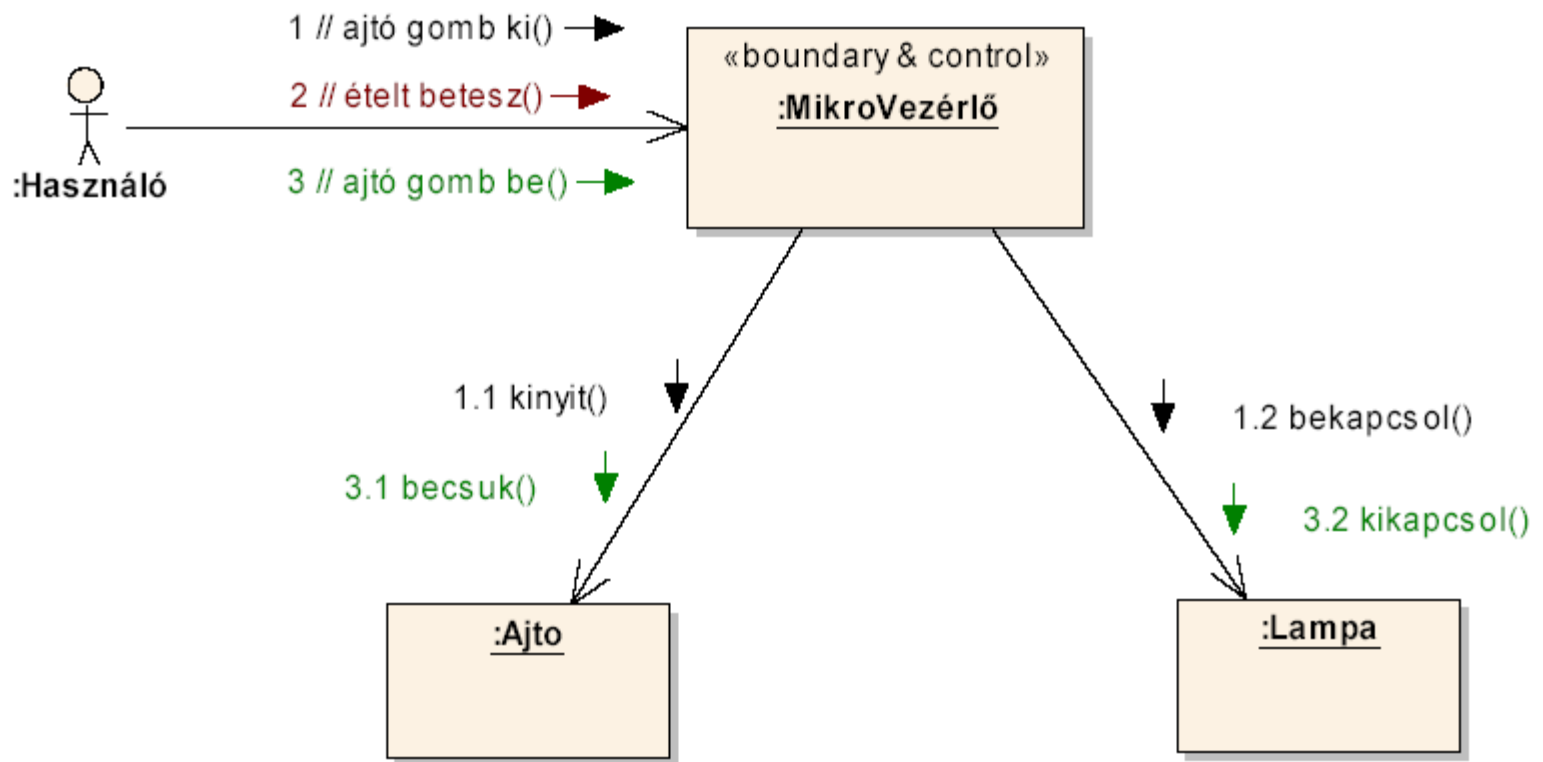


Rajzlap betöltése, dialógusokkal



Együttműködési diagram

- Az **együttműködési diagram** (collaboration diagram) egy gráf, ahol
 - a csúcsok az objektumok,
 - az élek a kapcsolatok.
- Itt nem az idő, hanem az objektumok közötti kapcsolatok a fontosak.
- Az üzenetek többszintesen sorszámozhatók (ezek a szekvencia számok), így módon az idő itt is kifejezhető



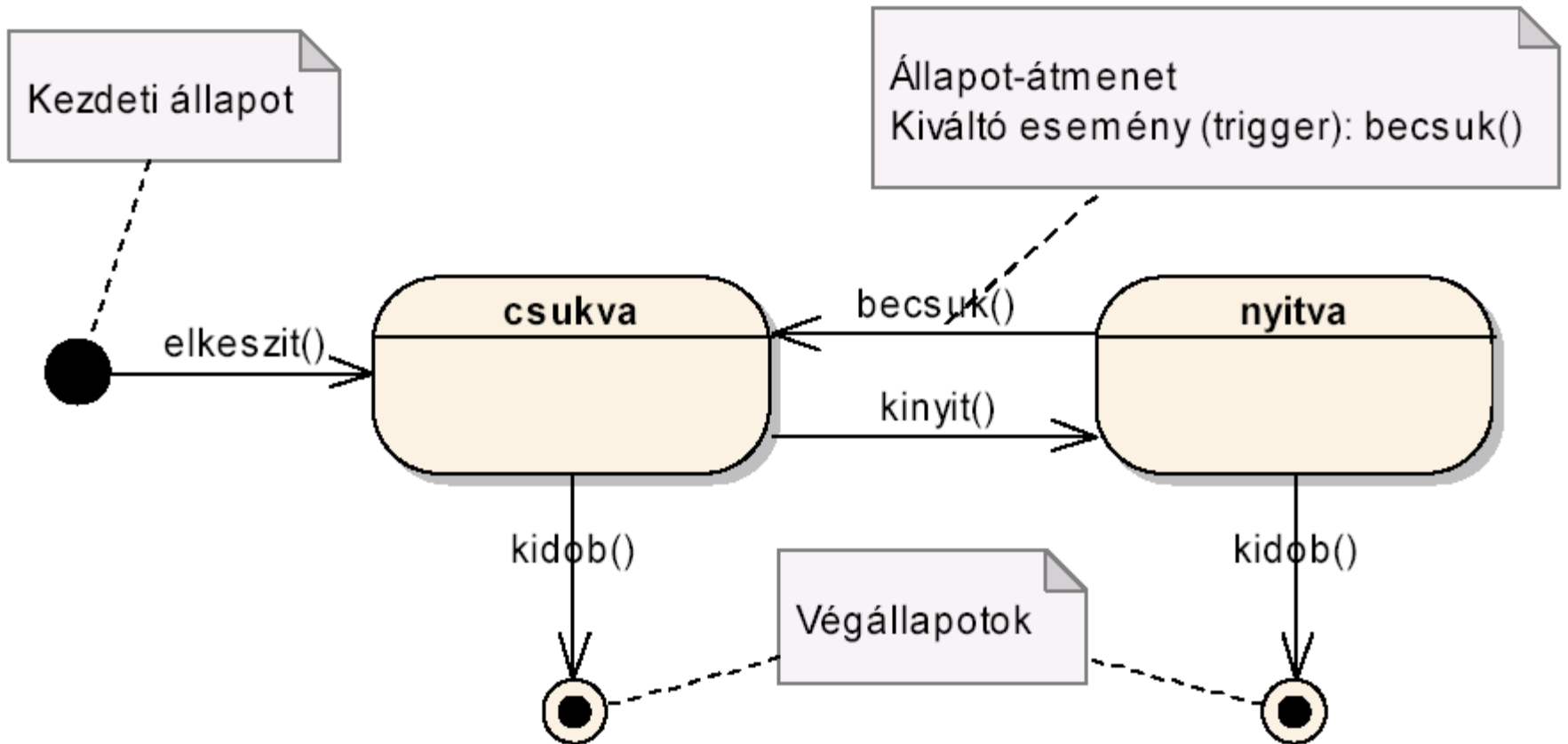
Állapot diagram

- Az állapot diagram (state diagram) egyetlen osztály (annak egy elő fordulásának) dinamikus viselkedését, a külvilággal való kapcsolatát ábrázolja.
- Az állapot diagram egy gráf, melynek csomó pontjai állapotok, élei pedig átmenetek.
- Megadja, hogy az objektum mely események hatására milyen állapotból milyen állapotba kerül.
- Ha egy objektumnak intenzív a dinamikus működése, akkor az objektum viselkedésének feltárásában az állapot-átmeneti diagram igen hatékony eszköz.
- Az állapot jele az UML-ben egy lekerekített téglalap, melynek részei (bármelyik rész elhagyható):
 - Név: a téglalap felső része, az állapot neve
 - Belső átmenetek: Olyan akciókat, illetve aktivitásokat tartalmazhat, melyek nem okoznak állapotváltozást.

■ Az objektumnak van

- pontosan egy kezdeti állapot (initial state), melybe az objektum születésekor kerül. Jele egy tömör kör.
- valahány végállapot (final state), ahonnan további állapotváltozás nem lehetséges.

Ajto	
-	nyitva: boolean
+	elkészít() : void
+	kidob() : void
+	isNyitva() : void
+	kinyit() : void
+	becsuk() : void
+	becsap() : void



Aktivitásdiagram

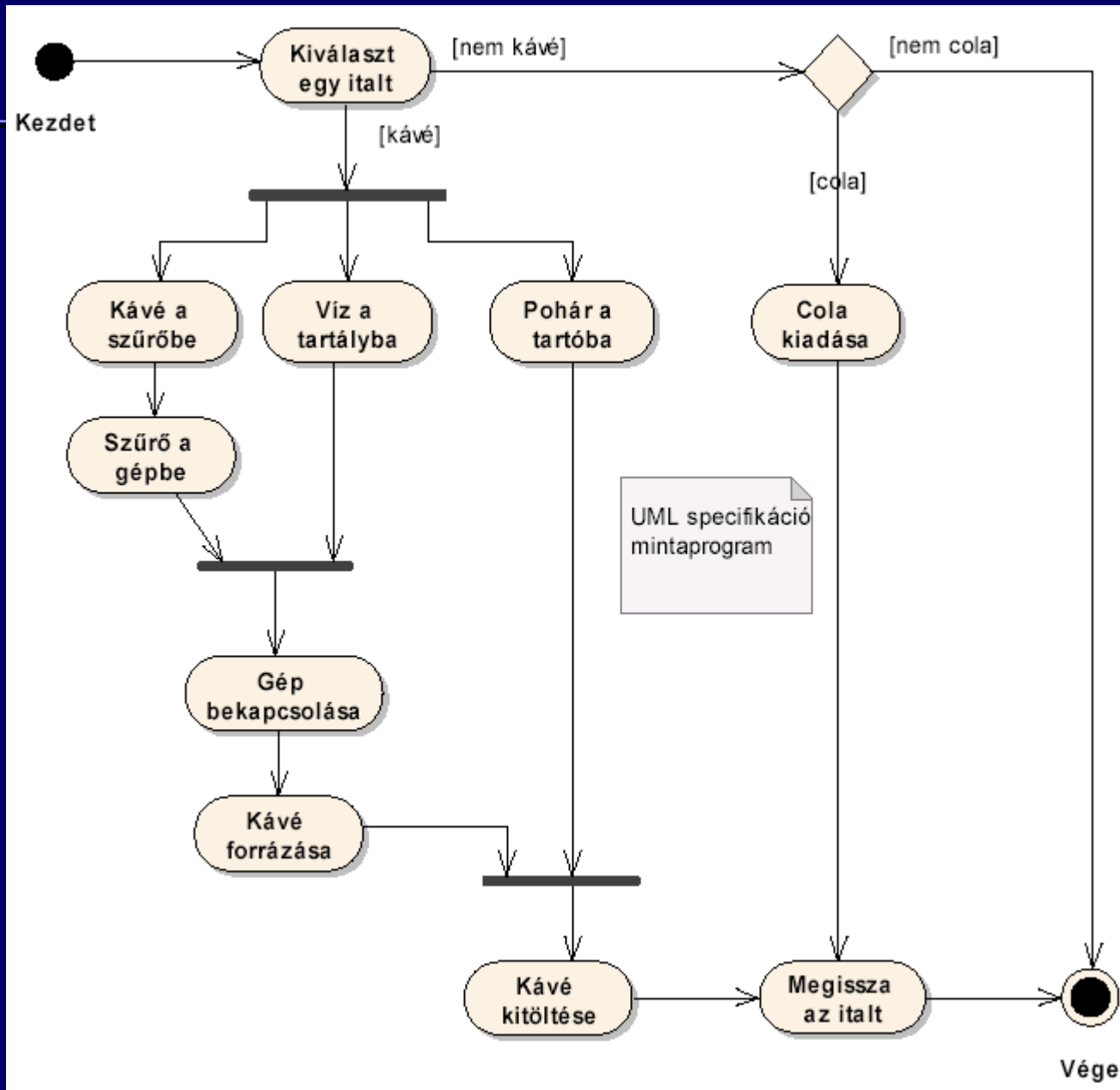
- Az aktivitásdiagram lényegében egy folyamatábra, segítségével a program dinamikáját ábrázolhatjuk.
- A diagram megmutatja, hogy az egyes tevékenységek egymáshoz képest mikor és milyen feltételekkel hajtódnak végre.
- Az aktivitásdiagram tevékenységei több osztály felelősségi köréhez is tartozhatnak.

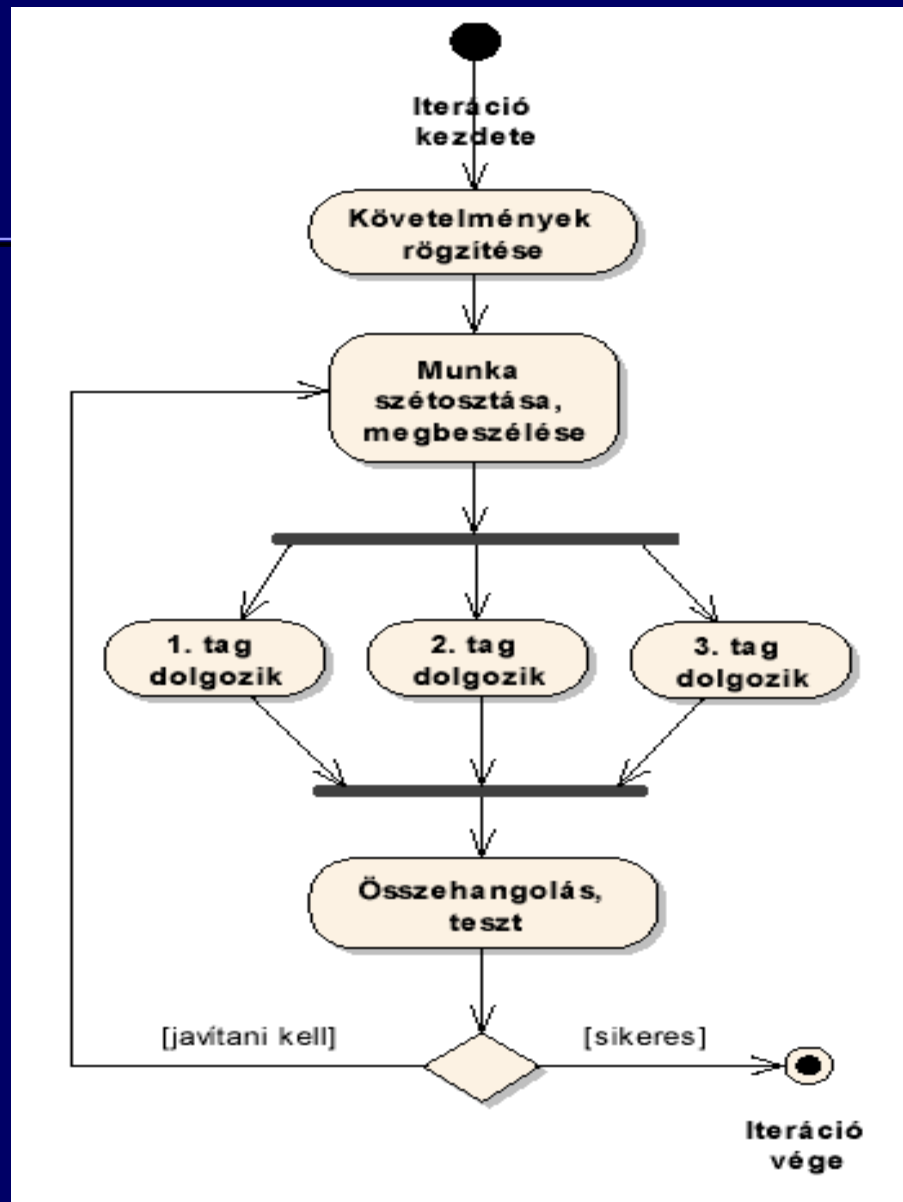
Aktivitásdiagram

- Az aktivitásdiagram használható:
 - üzleti modellezésre: ekkor a tevékenység lehet az ember vagy a számítógép által elvégzendő tetszőleges tevékenység;
 - programspecifikálásra: ekkor a tevékenység általában egy operáció, illetve metódus.

Aktivitásdiagram

- Az aktivitásdiagram az állapotdiagram egy speciális esete, ahol az állapot tevékenység (tevékenység-állapot), a kiváltó esemény (trigger) pedig a tevékenység befejezése.
- Az aktivitásdiagram általában a bonyolultabb használati esetek vagy metódusok implementációjához kötődik.
- Egy használati eset viselkedésének modellezéséhez mindig olyan diagramot (szekvencia, együttműködési, aktivitás vagy állapot) használjunk, amely arra a legalkalmasabb!

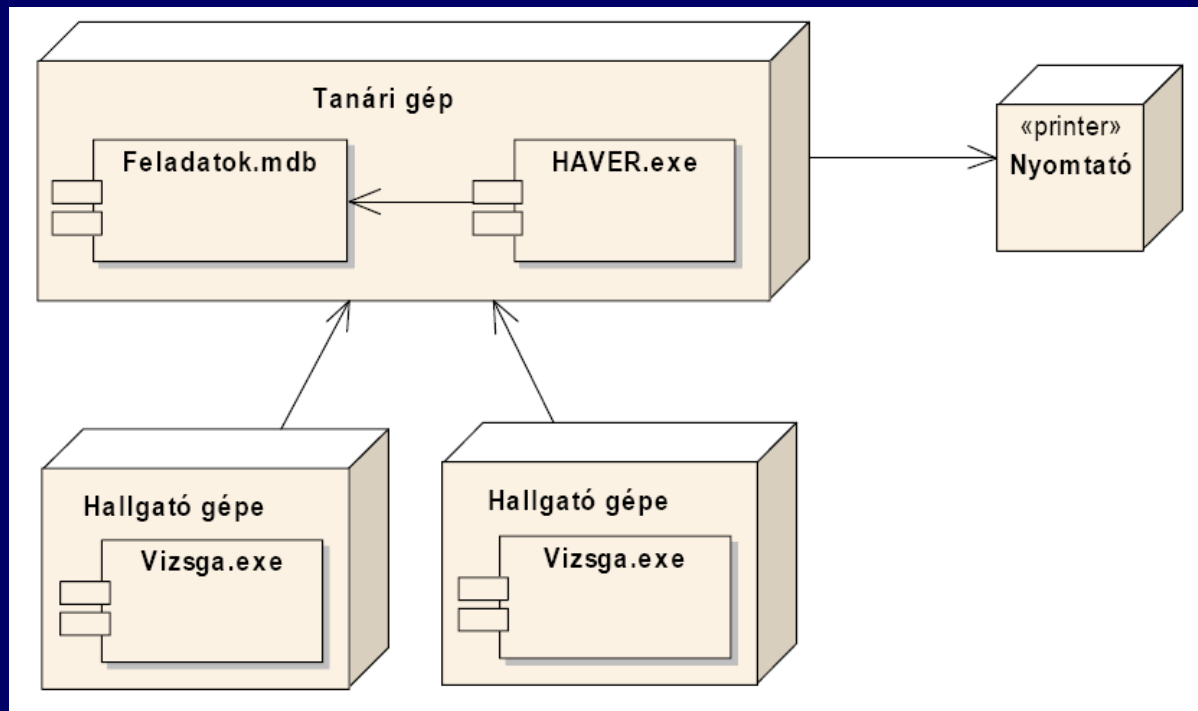




A "Szoftverfejlesztés csapatmunkában" egy iterációja

Telepítési diagram

- A telepítési diagram segítségével a rendszer szoftver elemeit a hardver elemekhez rendeljük.
- Megadjuk, hogy mely komponensek (szoftver elemek) milyen számítógépen futnak. A telepítési diagram az implementációs nézet jellemző diagramja.

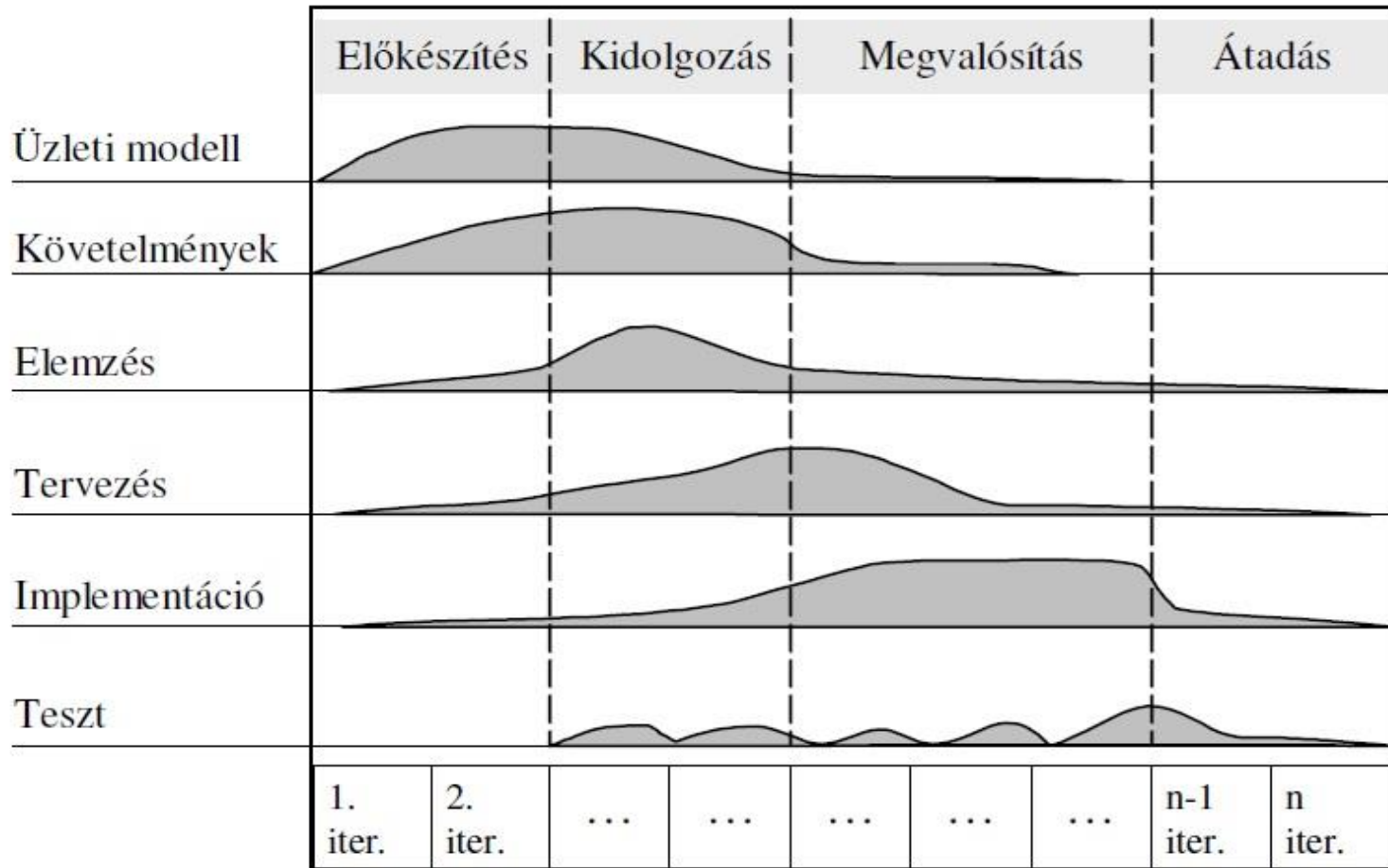


Rational Unified Process

A RUP a rendszerfejlesztés folyamatát két dimenzióval írja le :

- A fejlesztés időbelisége, dinamikája
- Statikus, az eljárás elemeit határozza meg, az elkészítendő dokumentumokat, diagramokat és forráskódokat

Rational Unified Process



Fázisok és munkafolyamatok

RUP fázisok - Előkészítés

Előkészítés (Inception)

A rendszer eredeti ötletének átdolgozása egy olyan részletes megfogalmazássá, amely felhasználásával a fejlesztés tervezhető lesz, a költségei pedig megbecsülhetők. Ebben a fázisban kerül leírásra az, hogy a felhasználók milyen módon használják majd a rendszert és hogy annak milyen alapvető belső szerkezete, architektúrája kerül kialakításra.

RUP fázisok - Előkészítés

Az *előkészítés (inception)* elsősorban üzleti szempontból írja le a fejlesztést és meghatározza az alkalmazás határait. Az üzleti szempont röviden a következőket jelenti:

- sikertényezők meghatározása
- kockázati tényezők felmérése
- erőforrás-bebecslés
- projekterv: a mérföldkövek dátumainak meghatározása

A fázis végén meghatározzuk az egyes iterációs ciklusok célját és döntünk a folytatásról.

RUP fázisok - Kidolgozás

Kidolgozás (Elaboration)

A *használati módok*, a „*használati* esetek” részleteiben is kidolgozásra kerülnek, valamint összeáll egy stabil alaparchitektúra (*architecture baseline*).

A *Unified Process* készítőinek elképzelése alapján a teljes rendszer egy testnek tekinthető, csontváznak, bőrnek és izmoknak. Az alaparchitektúra ebből a bőrrel borított csontváz, mely mindössze a minimális összekötő izomzatot tartalmazza, annyit, amennyi a legalapvetőbb mozdulatokhoz elegendő. Az alaparchitektúra segítségével a teljes fejlesztés folyamata ütemezhető és a költségei is tisztázhatók.

RUP fázisok - Kidolgozás

A *kidolgozás (elaboration)* során a problémát elsősorban szakterületi szempontból elemezzük. Ehhez a következő feladatokat kell végrehajtanunk:

- megbízható architektúrát alakítunk ki
- meghatározzuk a projekt-tervet
- megszüntetjük a legkritikusabb kockázati tényezőket

A fázis végén kiválasztottuk az architektúrát és megszüntettük a főbb kockázati tényezőket. A mérföldkőnél itt is elemezzük az elért eredményeket és döntünk a folytatásról.

RUP fázisok - Megvalósítás

Megvalósítás (Construction)

Ezen fázis során kerül kifejlesztésre a teljes rendszer. Az összes „izomzat” beépítésre kerül.

A megvalósítás (construction) során a teljes rendszert iteratív és inkrementális módon kifejlesztjük. Ehhez:

- specifikáljuk az elmaradt használati eseteket
- a tervezésre helyezzük a hangsúlyt
- kiegészítjük az implementációt
- teszteljük az elkészített alkalmazást

A fázis végén már működőképes szoftvert állítunk el, melyről döntenünk kell, hogy az kibocsátható-e a felhasználóknak.

RUP fázisok - Átadás

Átadás (Transition)

A rendszer bétaváltozatának kipróbálása. Tapasztalt felhasználók tesztelik a rendszert és jelentést készítenek annak helyességéről, hibáiról és hiányosságairól.

A rendszer javítása a rendszer módosítását, majd ezt követően újabb tesztelését jelenti.

RUP fázisok - Átadás

Az átadás (transition) lezáró fázisa során az alkalmazást átadjuk a felhasználóknak.

- Ez a fázis rendszerint az alkalmazás béta-tesztjével kezdődik.
- A rendszer hangolása miatt szükség lehet kiegészítő fejlesztésekre.
- A megjelenő hibákat ki kell küszöbölni.
- A még hiányzó részeket ki kell fejleszteni.

A fázis végén dönteni kell, hogy elértük-e a fejlesztés során kitűzött célokat, illetve, hogy indítanunk kell-e újabb fejlesztési ciklust. Ugyancsak célszer, hogy a projekt tapasztalatait elemezve vizsgáljuk meg azt, hogy miként módosíthatunk a fejlesztési módszerünkön.

RUP – Mérföldkövek, iterációk

Minden fázis vége a fejlesztés egy-egy jól meghatározott *mérföldkövét* (*milestone*) jelenti, azaz olyan pontot, ahol egy célt kell elérnünk, illetve ahol kritikus döntéseket kell meghozni.

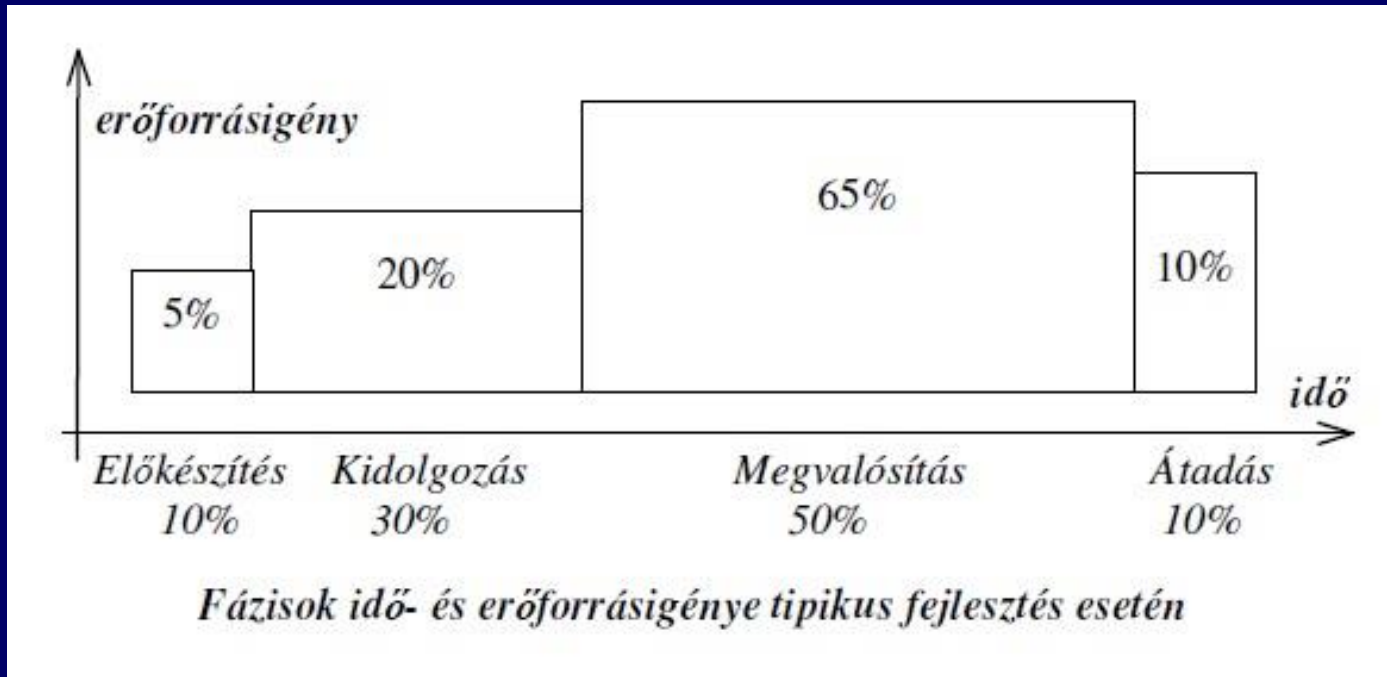
A fázisok további kisebb egységekre, *iterációkra* (*iteration*) bonthatók. Minden iteráció egy teljes, illetve részben önálló fejlesztési ciklust jelent, mivel az iteráció végén egy működő és végrehajtható alkalmazásnak kell előállnia. Minden iteráció végén így a végső, teljes rendszer egyre bővülő részét kapjuk eredményül, melyeket a rendszer egymás utáni *kibocsátásai* (*release*), vagy belső változatai. A belső változatokkal a fejlesztők kipróbálhatják és a tapasztalatok alapján esetleg módosíthatják a fejlesztés ütemezését.

RUP – Mérföldkövek, iterációk

Előkészítés		Kidolgozás		Megvalósítás			Átadás	
1. iter.	2. iter.	...	...	...	...	...	n-1 iter.	n iter.

Mérföldkövek

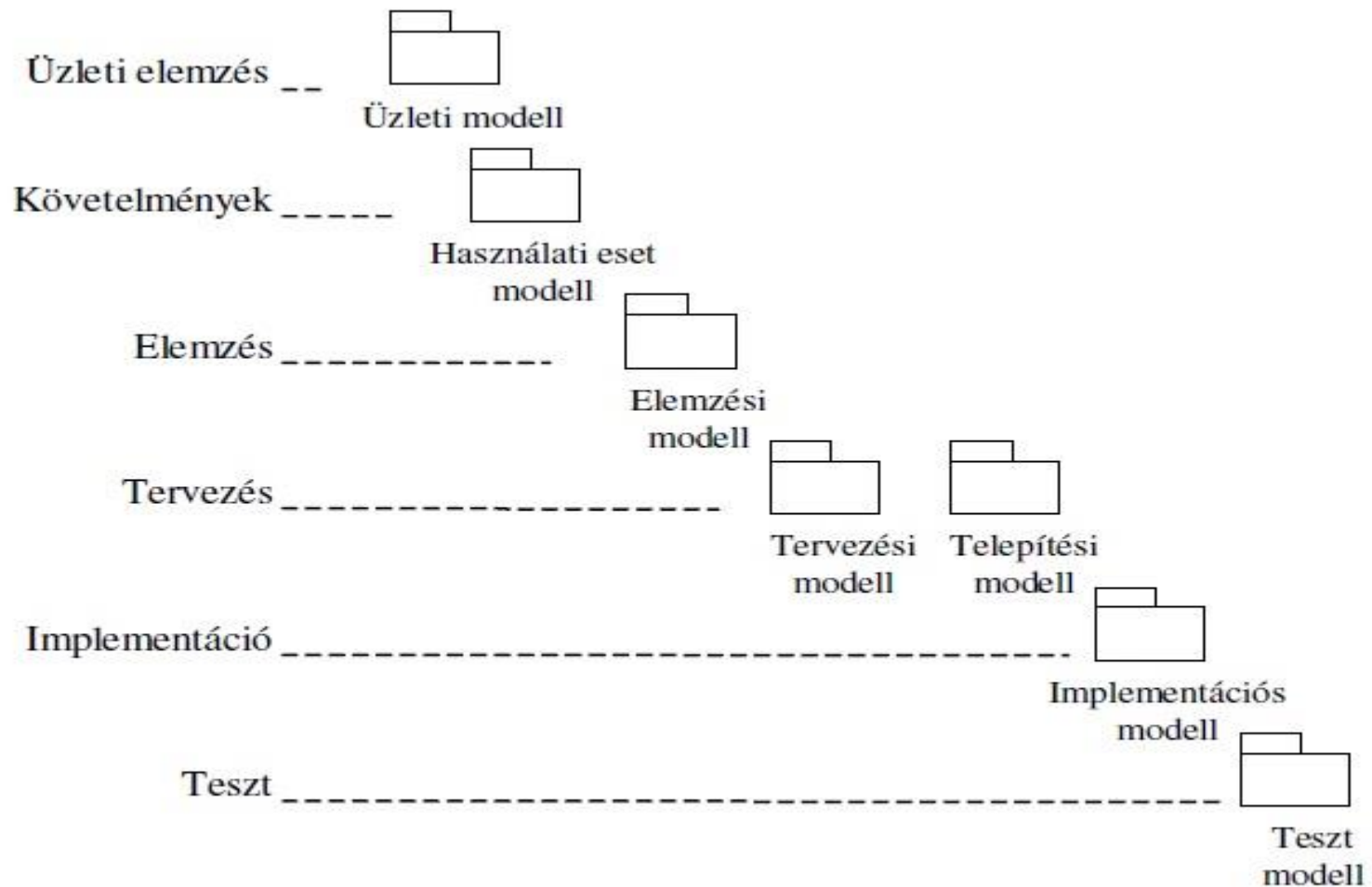
RUP – Idő- és erőforrásigény



RUP – Munkafolyamatok

A RUP másik (statikus) dimenziója az eljárás elemeit határozza meg, hogy a fejlesztés során milyen dokumentumokat, diagramokat, forráskódokat , azaz *produktumokat (artifact)* kell elkészíteni. Az elkészítendő produktumok természetesen a megfelelő tevékenységekkel állíthatók össze, mely tevékenységeket pedig adott szakismerettel rendelkező személyek („dolgozók”), adott sorrendben hajthatnak végre. A tevékenységek, azok időbeli sorrendje és az azt végrehajtó dolgozók együttesen egy munkafolyamattal (*workflow*) írhatók le.

RUP – Modellek



Modellek

RUP – Munkafolyamatok

Először a fejlesztéshez egy megfelelő kiindulópontot keresünk. Az első tevékenységcsoport így az *üzleti modellezés (business model)*, mely során a készítendő rendszer üzleti vagy más néven szakterületi környezete kerül felderítésre, mely alapvetően az üzleti fogalmakat és folyamatokat, illetve az azokra hatást gyakorló üzleti munkatársakat írja le.

RUP – Munkafolyamatok

A következő tevékenység a *követelmények meghatározása* (*requirements capture*). Ezen munkafolyamat során a rendszer működésével szemben támasztott kezdeti elképzelések kerülnek összegyűjtésre és felsorolásra. Megfogalmazásra kerül a rendszer működési környezete, valamint megadjuk a funkcionális (működéssel kapcsolatos) és nem-funkcionális (pl. válaszidők, bővíthetőség, alkalmazott technológiák, stb.) követelményeket.

RUP – Munkafolyamatok

A követelmények meghatározása során alapvetően a felhasználók szempontjából írjuk le a rendszert, így annak egy külső képét rögzítjük.

A következő munkafolyamat, az *elemzés (analysis)* folyamán a követelményeket a fejlesztők szempontjának megfelelően rendezzük át, így azok együttesen a rendszer egy belső képét határozzák meg, mely a további fejlesztés kiindulópontja lesz.

Az elemzés során rendszerezünk és részletezzük az összegyűjtött használati eseteket, valamint azok alapján meghatározzuk a rendszer alapstruktúráját.

RUP – Munkafolyamatok

Az elemzés célja a szerkezeti váz kialakítása, mely vázat a következő munkafolyamat, a *tervezés (design)* formálja teljes alakká és tölti fel konkrét tartalommal, mely az összes - funkcionális és nem-funkcionális - követelménynek is eleget tesz. A tervezésnek az implementációval kapcsolatos összes kérdést meg kell válaszolnia, így részletesen le kell írni az összes felhasznált technológiát, a rendszert független fejlesztői csoportok által kezelhető részekre kell bontani, meg kell határozni az alrendszereket és közöttük a kapcsolódási módokat, protokollokat. A tervezésnek a rendszert olyan részletezettségi szinten kell vázolnia, melyből az közvetlenül, egyetlen kérdés és probléma felvetése nélkül implementálható. A RUP szóhasználata szerint el kell állítania az implementáció tervrajzát (*blueprint*).

RUP – Munkafolyamatok

Az *implementáció* (*implementation*) során a rendszert az UML terminológiája szerinti komponensekként állítjuk el, melyek forráskódokat, bináris és futtatható állományokat, szövegeket (pl. súgó), képeket, stb. jelentenek. Az állományok elállítása egyben azok függetlenül végrehajtható, önálló tesztjeit is jelentik. Az implementáció feladata még az architektúra, illetve a rendszer, mint egészzel kapcsolatos kérdések megválaszolása, így az iteráció esetén szükséges rendszerintegráció tervezése, az osztottság (*distribution*) tervezése.

RUP – Munkafolyamatok

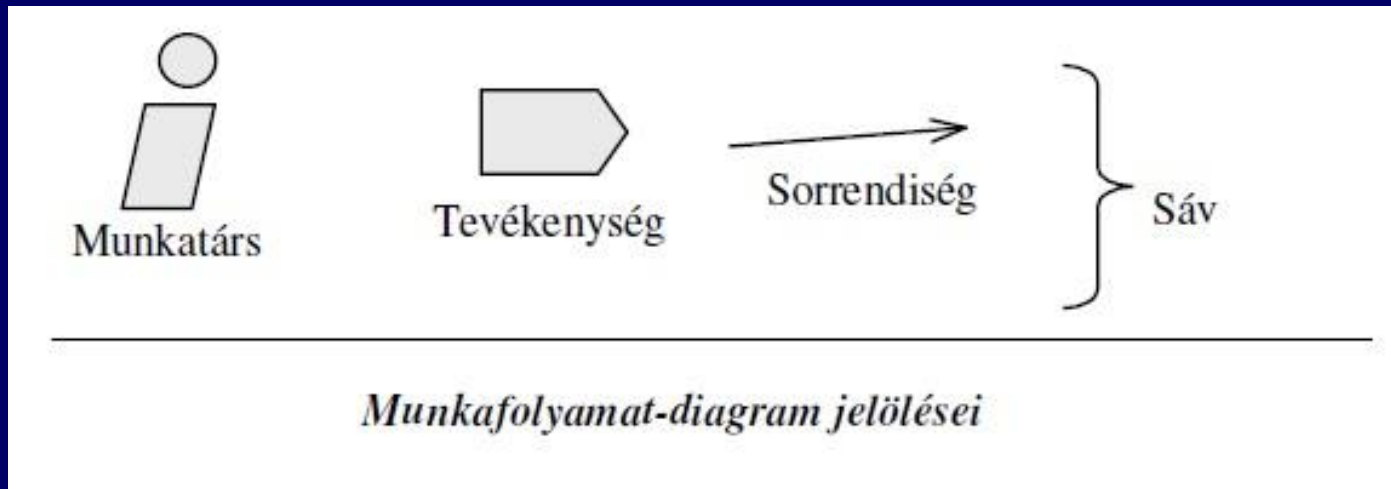
A *teszt (test)* során (utolsó munkafolyamat) összeállítjuk az iterációkon belüli integrációs tesztek és az iterációk végén végrehajtandó rendszertesztek ütemtervét. Megtervezzük és implementáljuk a tesztek, azaz teszt-esetekként megadjuk, hogy mit kell tesztelnünk, teszt-eljárásokként megadjuk azok végrehajtási módját, és programokat készítünk, ha lehetséges a tesztek automatizálása. A tesztek végrehajtásával párhuzamosan azok eredményeit szisztematikusan feldolgozzuk, majd hibák vagy hiányosságok esetén újabb tervezési vagy implementációs tevékenységeket hajtunk végre.

RUP – Munkafolyamatok

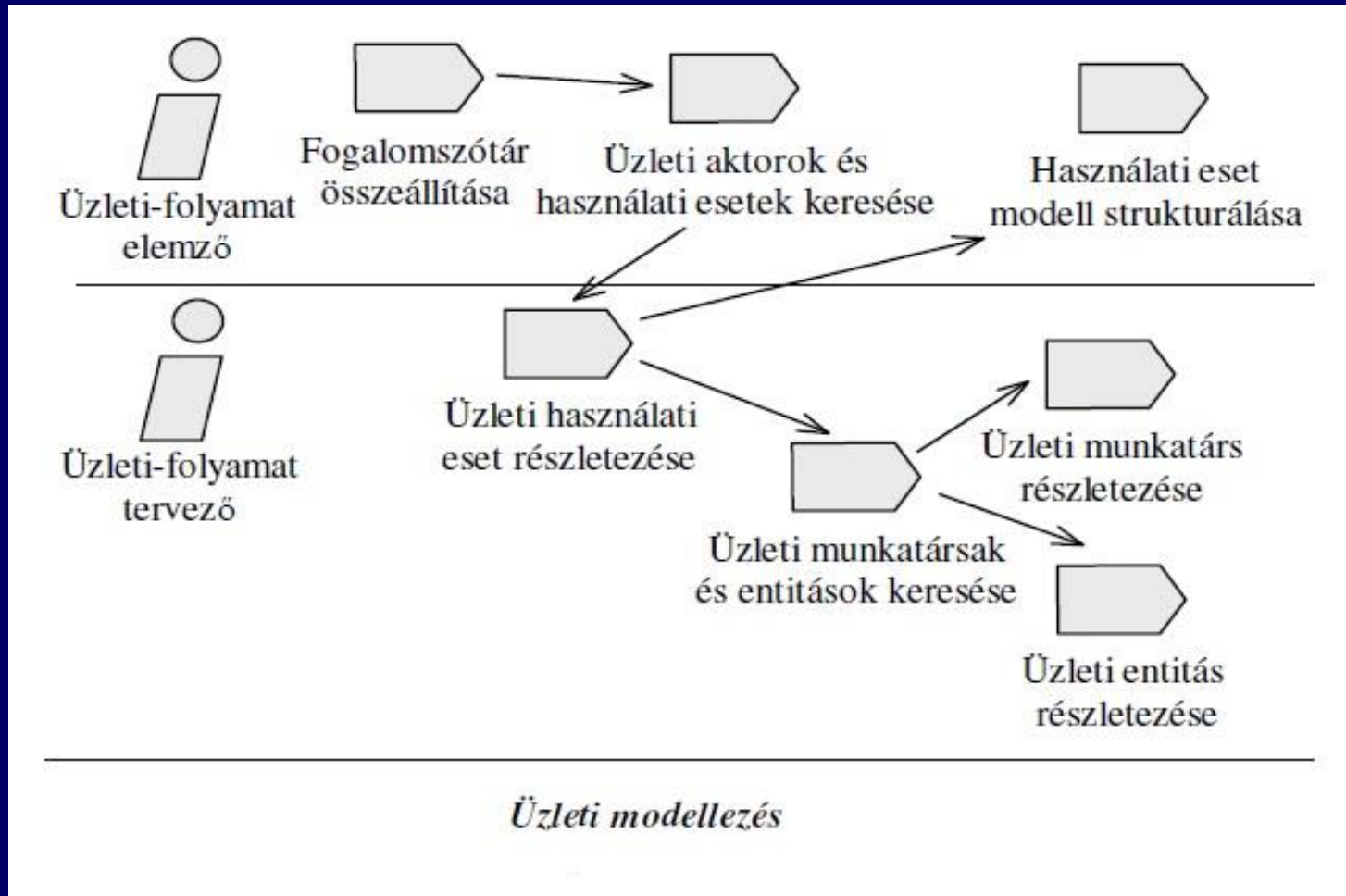
A RUP munkafolyamat (*workflow*) diagramjai

A diagram alapelemei a tevékenységek, melyek mindegyike egy vagy több produktumot (dokumentumot, kódot, stb.) hoz létre, vagy azokon módosít. A diagram a tevékenységeket végrehajtó személyek, ún. munkatársak alapján vonalakkal elválasztott sávokra (*swimlanes*) van bontva. Minden tevékenység az azért felelős munkatárs mezőjében helyezkedik el, a tevékenységek ajánlott időbeli egymásutánisága, sorrendisége pedig nyilakkal van jelölve.

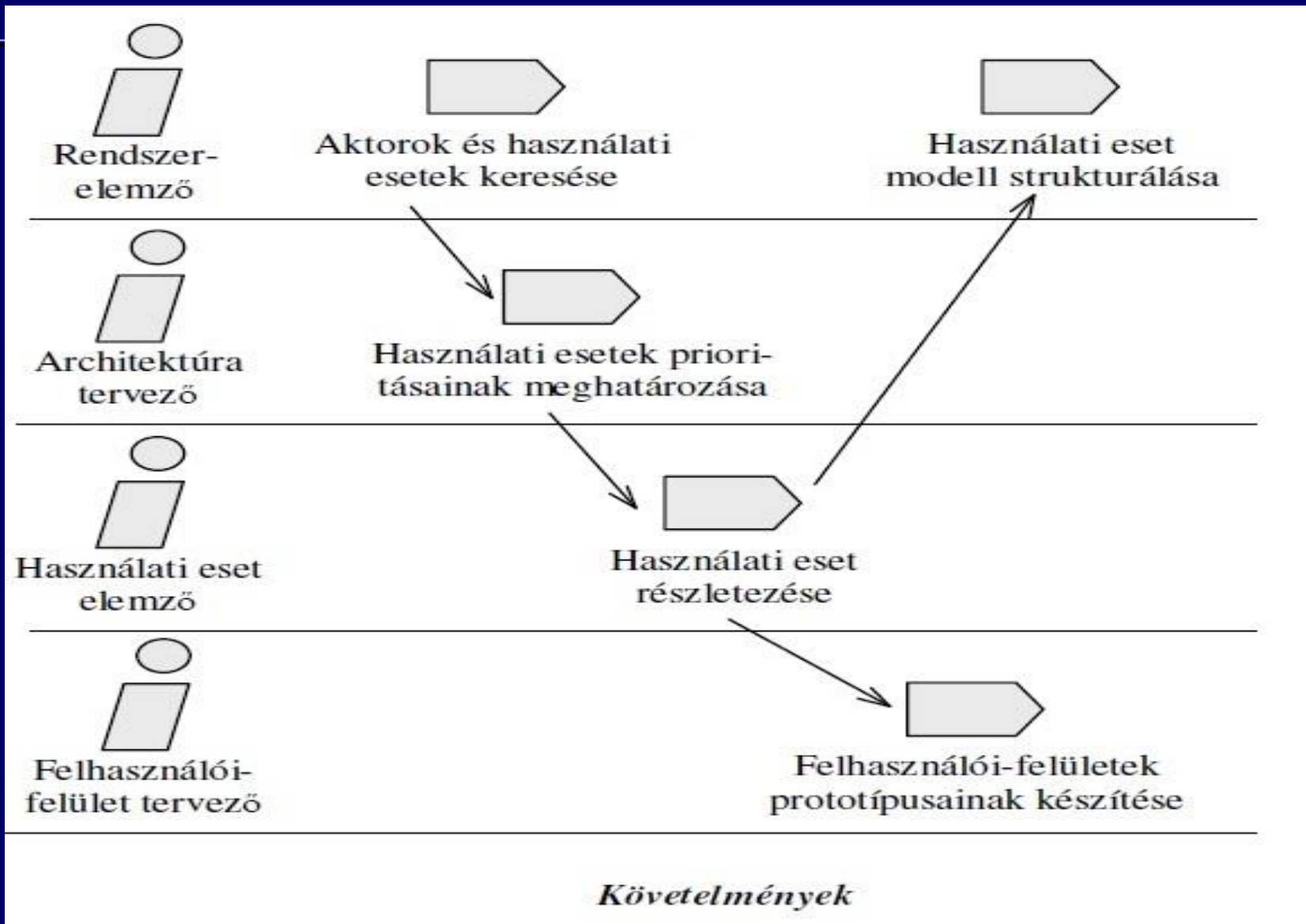
RUP – Munkafolyamatok



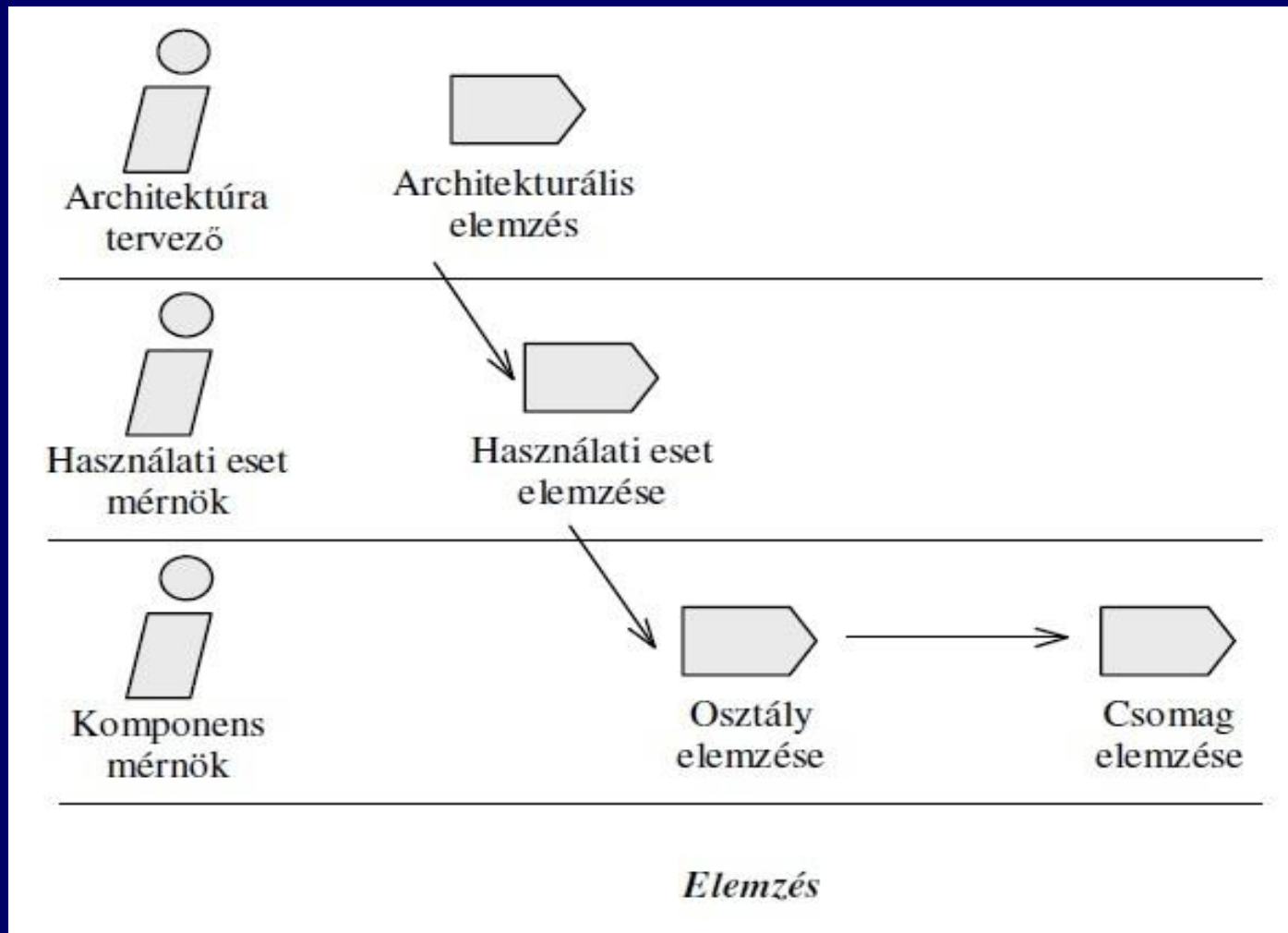
RUP – Munkafolyamatok



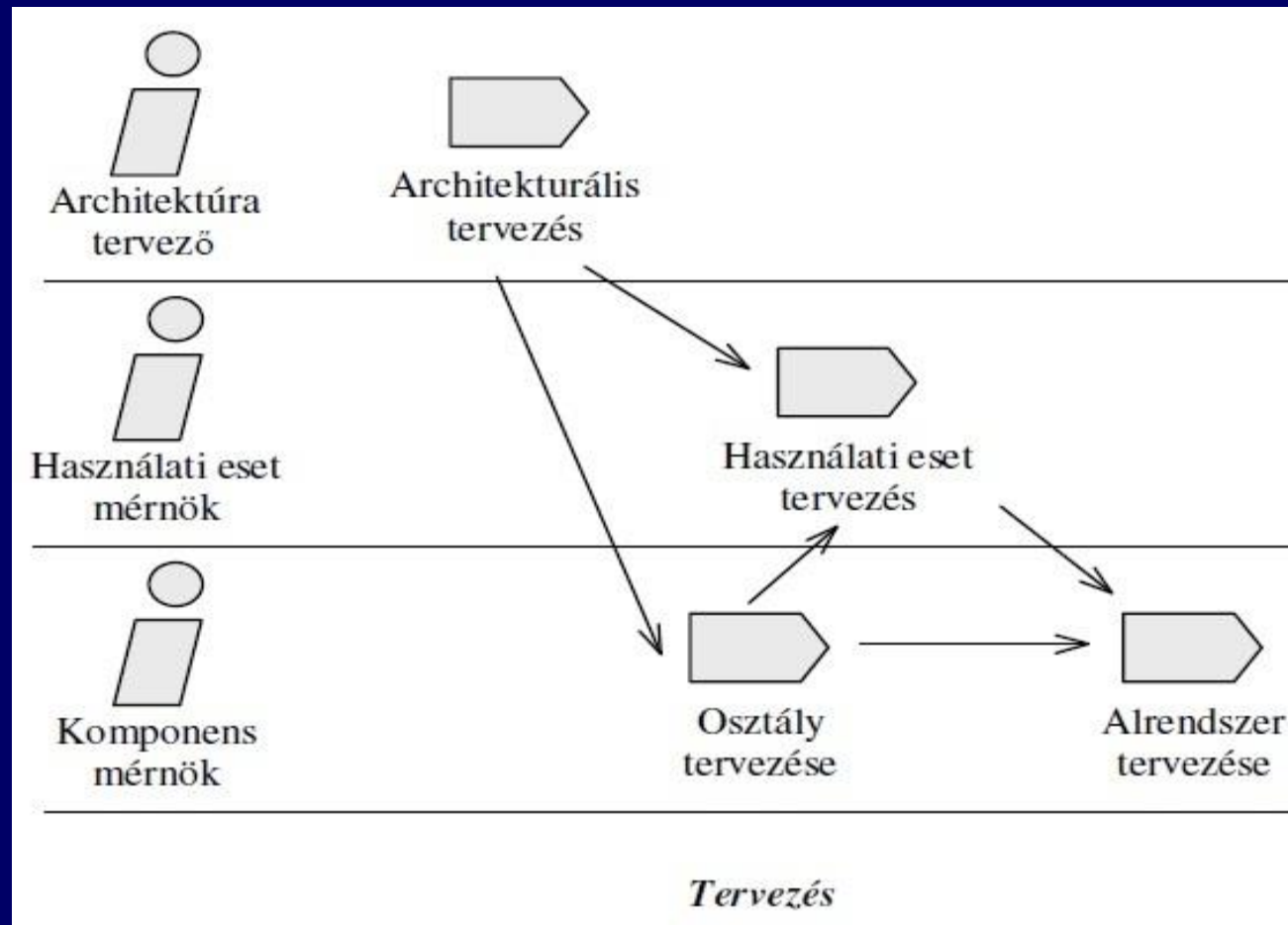
RUP – Munkafolyamatok



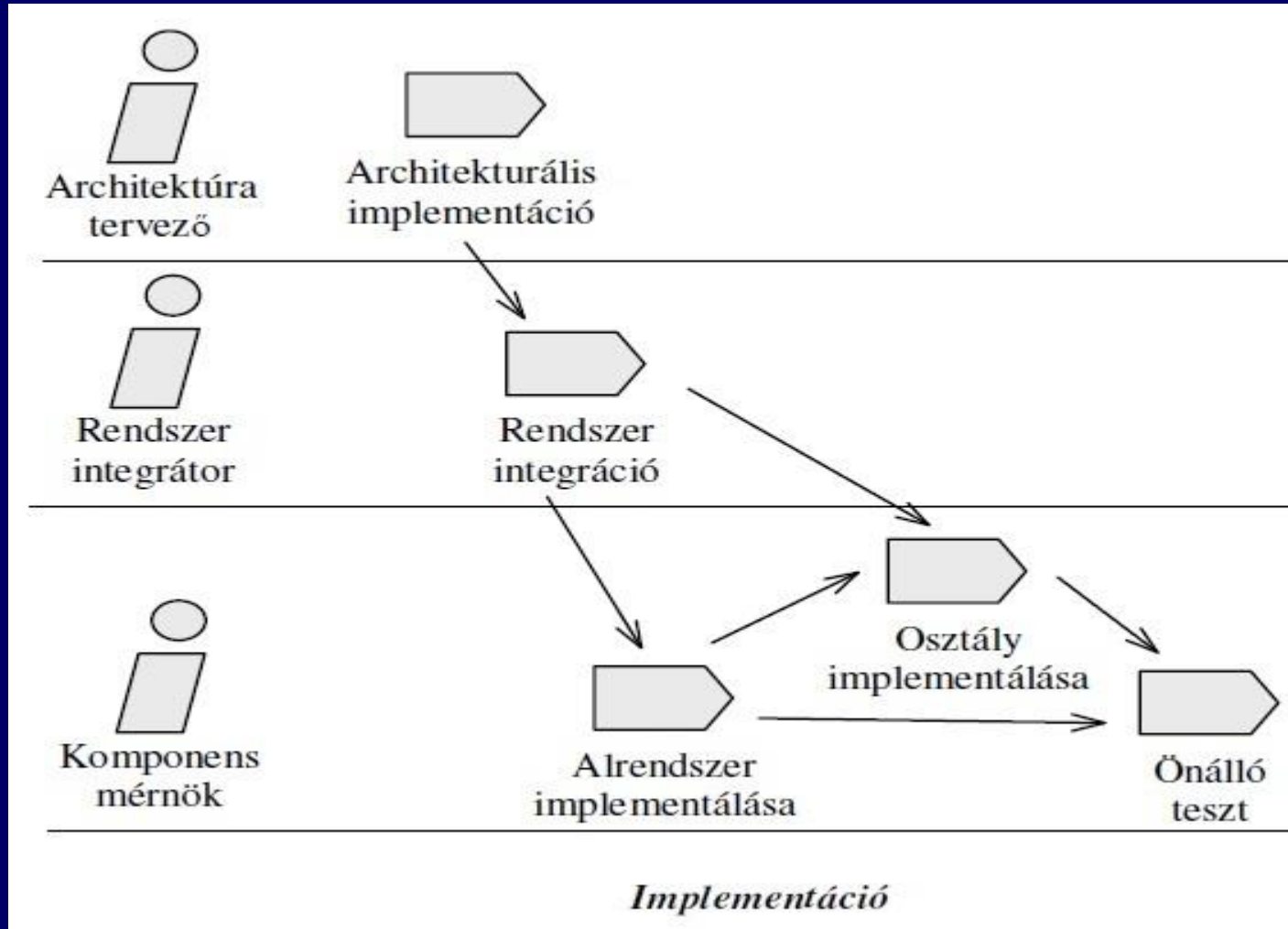
RUP – Munkafolyamatok



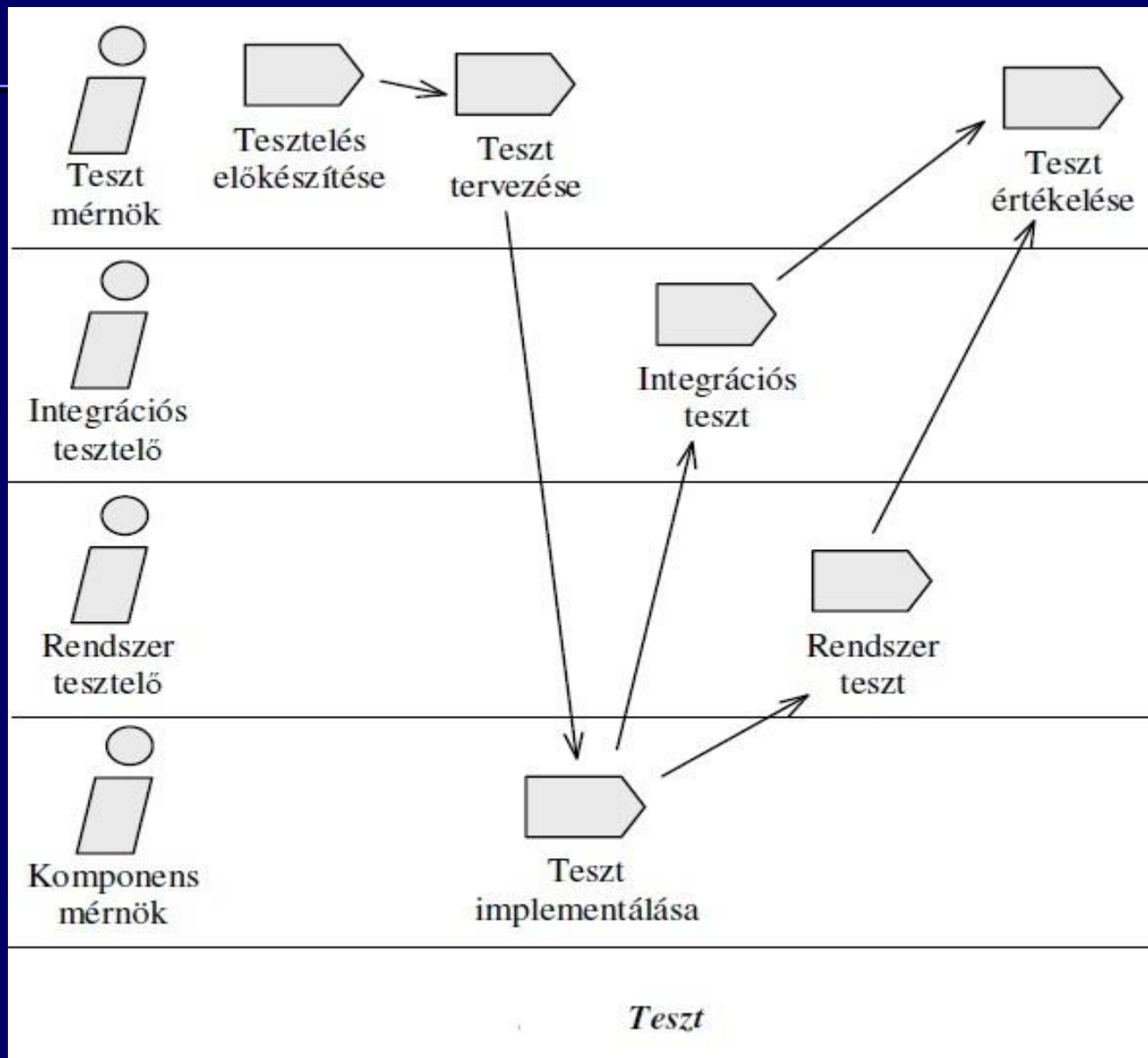
RUP – Munkafolyamatok



RUP – Munkafolyamatok



RUP – Munkafolyamatok



A RUP jellemzői

A RUP egy ún. *használati eset vezérelt (usecase-driven)* eljárás, amely azt jelenti, hogy a fejlesztés teljes folyamata során a használati eseteket eszközszerét, illetve az azzal kialakított modellt alkalmazzuk a fejlesztés ütemezésére.

A használati esetek önmagukban még nem adnak a teljes fejlesztési folyamathoz elegendő információt. A szükséges többletet a RUP a rendszer architektúrájaként nevezi meg, mely a rendszernek a fejlesztésben résztvevő összes munkatárs által közösen kialakított vázát, leglényegesebb elemeinek gyűjteményét jelenti.

A RUP jellemzői

A RUP *architektúra központú (architecture-centric)* eljárás, mivel a munkafolyamatokban nagy hangsúlyt kap a megfelelő architektúra kialakítása.

A rendszer architektúrája annak leglényegesebb elemeit tartalmazza, melyek együttesen a rendszer alapszerkezetét adják meg.

A RUP külön definiálja az *alaparchitektúra (architecture baseline)* fogalmát, mely a teljes rendszer egy önállóan is működő, erősen egyszerűsített változata. A későbbi iterációk ezen alaparchitektúra alapján, illetve az köré építkeznek.

A RUP jellemzői

A RUP a fejlesztés teljes folyamatát kisebb részekre bontja, melyek önállóan is egy teljes fejlesztési folyamatot jelentenek. Az egyes részeket iterációknak nevezzük, mivel a fejlesztést ismétlődő, iteratív ciklusokban hajtjuk végre. Egy iteráció nem a rendszer egy, a korábbitól független részét állítja elő, hanem a rendszert újabb funkcionálitással bővíti, vagy a korábbi funkcionálitást teszi változatosabbá, gazdagítja. Az iteráció bővítését és módosítását inkrementumnak nevezzük. A RUP így egy *iteratív és inkrementális (iterative and incremental)* fejlesztési módszer.