Architektúra tervek ellenőrzése

Majzik István

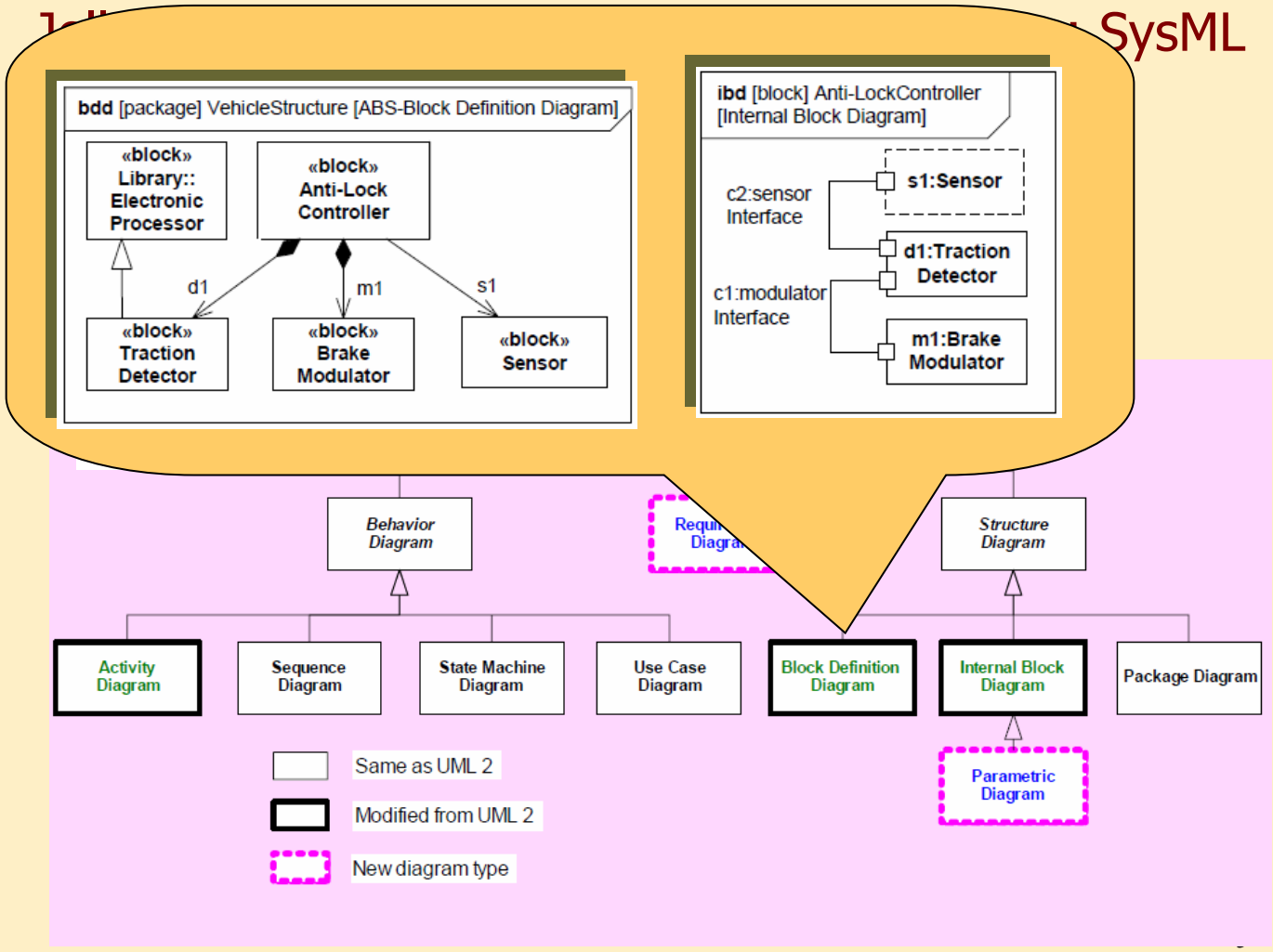
http://www.inf.mit.bme.hu/

Tartalomjegyzék

- Motiváció
  - Mit határoz meg az architektúra?
  - Milyen vizsgálati módszerek vannak?
- Követhetőség
- Szisztematikus vizsgálati módszerek
  - Interfész analízis
  - Hibahatás analízis
- Modell alapú vizsgálatok
  - Megbízhatósági modellezés
  - Teljesítmény modellezés
- ATAM architektúra elemzés

Szoftverarchitektúra

- Architektúra
  - Komponensek (tulajdonságokkal)
  - Közöttük lévő kapcsolatok (telepítés, információáramlás, ...)
- Tervezői döntések
  - Komponensek és kapcsolatok meghatározása
    - Hardver-szoftver együttműködés
    - Hibahatások figyelembe vétele
  - Architektúra minták használata
    - Pl. MVC, N-tier, ...
  - Méretezés
    - Redundancia, teljesítmény, biztonságosság, ...
  - Újrafelhasználás (korábban fejlesztett komponensek)

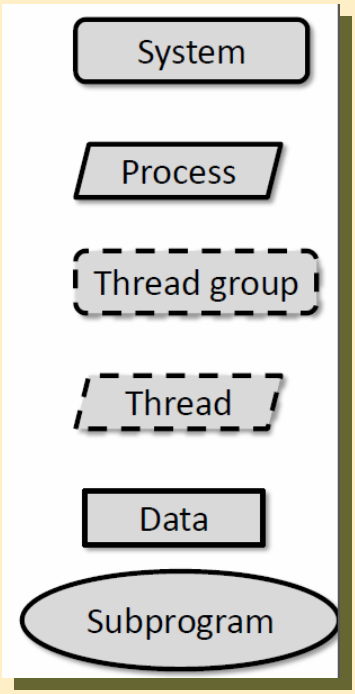


Jellegzetes nyelvek architektúra tervezéshez: AADL

AADL: Architecture Analysis and Design Language (v2: 2009)

- Beágyazott rendszerekhez (SAE)

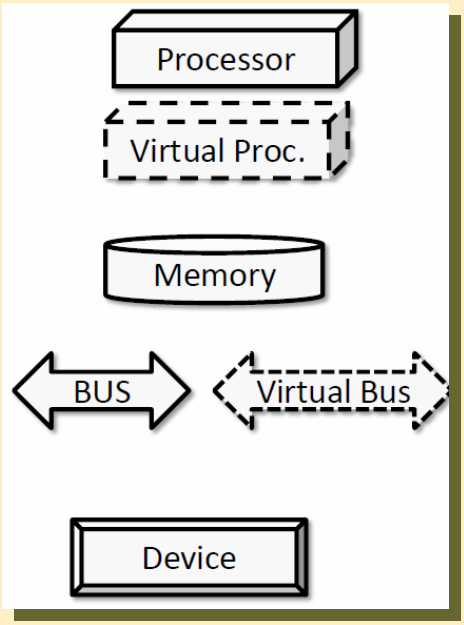
- Szoftver komponensek
  - System: Komponensek hierarchikus elrendezése
  - Process: Védett címtartomány
  - Thread group: Thread-ek logikai csoportja
  - Thread: Ütemezhető konkurens végrehajtási egység
  - Data: Megosztható adat
  - Subprogram: Szekvenciális, hívható kódegység



Jellegzetes nyelvek architektúra tervezéshez: AADL

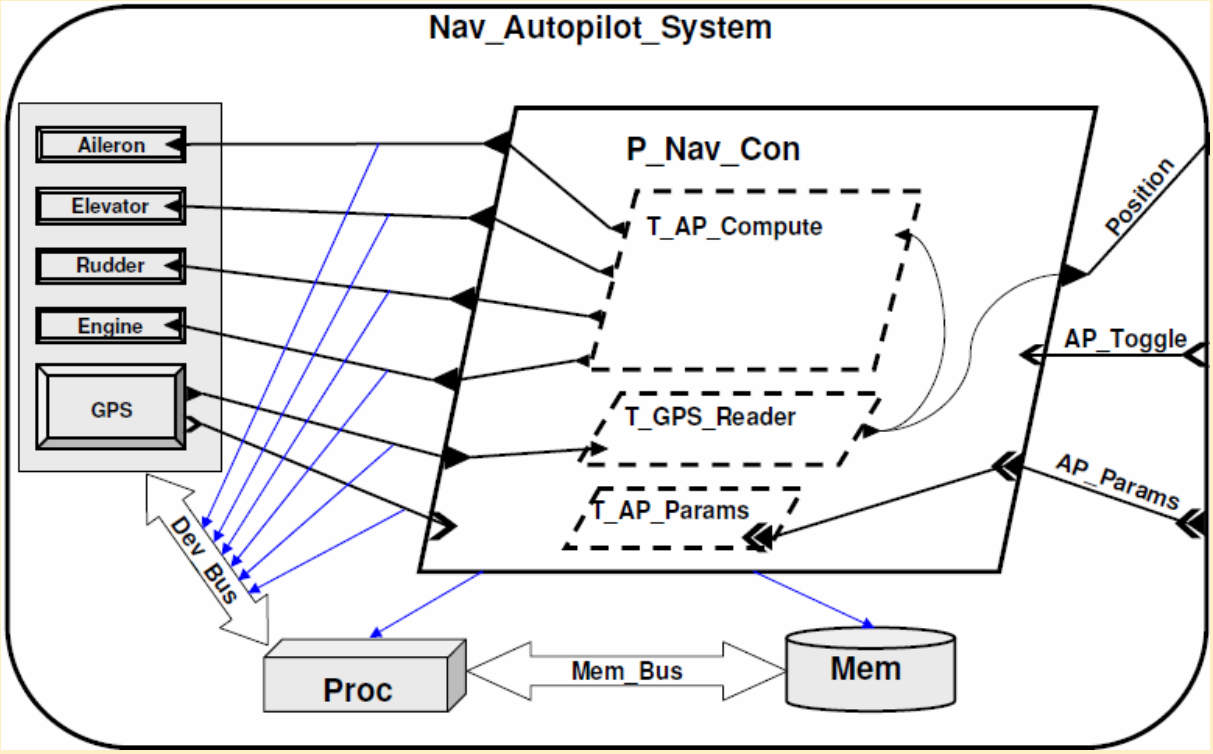
- Hardver komponensek
  - Processor, Virtual Processor: Ütemezés platformja
  - Memory: Adat és futtatható kód tárolója
  - Bus, Virtual Bus: Fizikai vagy logikai kapcsolatok egysége
  - Device: Interfész külső környezethez

- Kötések
  - Szoftver és hardver között
  - Logikai (virtuális) komponensek és fizikai komponensek között



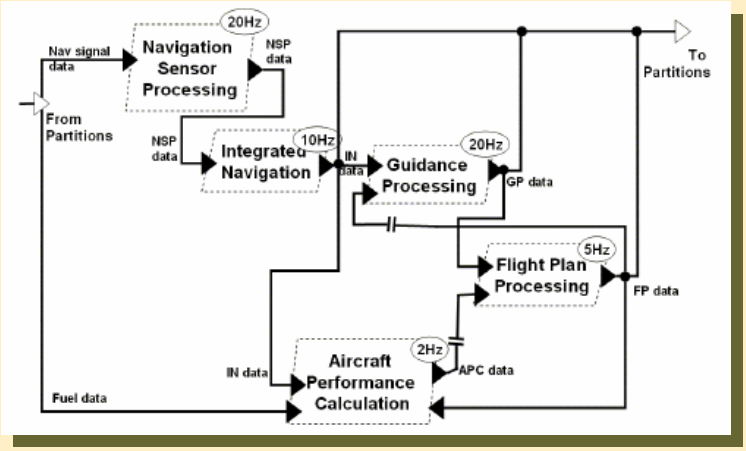
Jellegzetes nyelvek architektúra tervezéshez: AADL

- Kötések szoftver és hardver között



Jellegzetes nyelvek architektúra tervezéshez: AADL

- Kapcsolatok
  - Adatok, események áramlása portokon
- Tulajdonságok megadása analízishez
  - Időzítés
  - Ütemezés
  - Hibaterjedés (annex)
  - ...

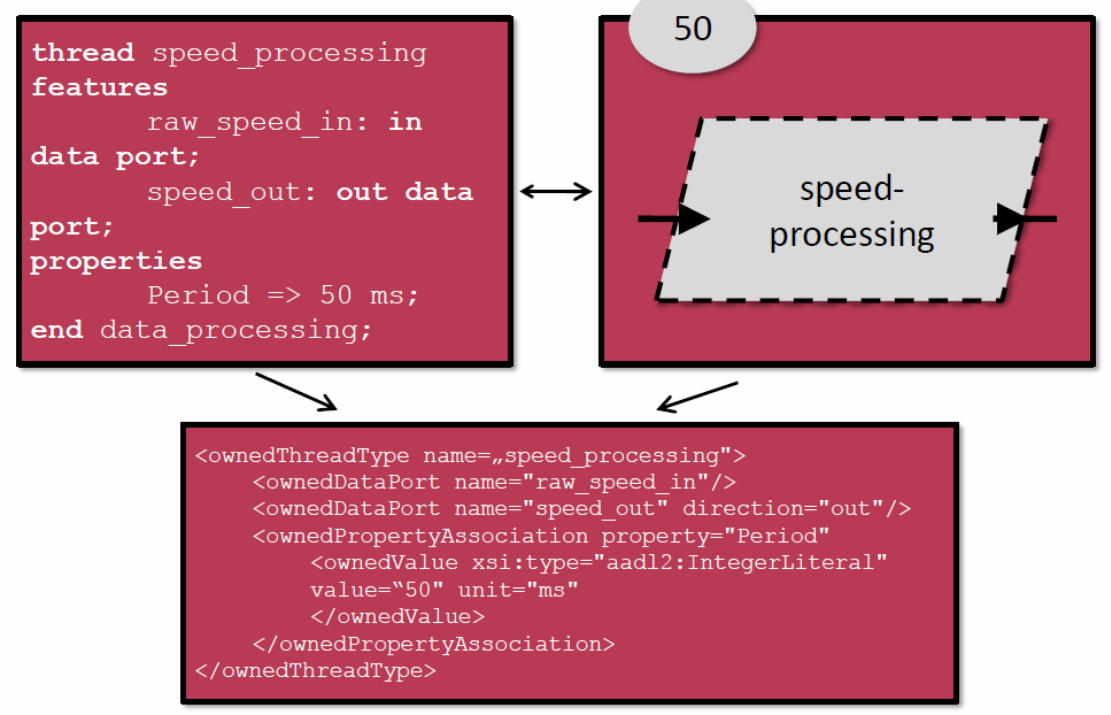



```
thread implementation CoinPublisher.impl
            calls(u: subprogram updateTotal;);
      properties

            Compute_Execution_Time => 30ms .. 40ms;
            Dispatch_Protocol => ( Sporadic );
            annex behavior {**
                  compute(5ms);
                  compute(10ms);
                  compute(15ms);
                  raise(availableContent);
            **};
end CoinPublisher.impl;
```

Jellegzetes nyelvek architektúra tervezéshez: AADL

- Modell megadásának formái:



Mit és hogyan határoz meg az architektúra? 1/2

- Szolgáltatásbiztonság
  - Hibadetektálás: Push/pull, kivételkezelés
  - Helyreállítás: Előrelépés, visszalépés, kompenzáció
  - Hibahatás kezelése: Átkonfigurálás, graceful degradation
- Teljesítmény
  - Erőforrásigény: Hatékony számítások, várakozó kérések korlátozása
  - Erőforrás menedzsment: Párhuzamos feldolgozás, cache, erőforrás ütemezés, dinamikus hozzárendelés
- Adatbiztonság
  - Támadás kivédése: Autentikáció, bizalmas adatkezelés, integritás karbantartása, elérés korlátozása
  - Támadás felderítés: Minták elemzése
  - Helyreállítás

Mit és hogyan határoz meg az architektúra? 2/2

- Módosíthatóság
  - Lokalizálás: Szemantikus koherencia, általános input
  - Dominóhatás elkerülése: Információrejtés, kommunikáció korlátozása, közvetítő használata
  - Kötési idő elhalasztása: Futási idejű regisztráció, konfigurációs leírók, polimorfizmus, közös protokollok
- Tesztelhetőség
  - Vezérelhetőség, megfigyelhetőség biztosítása
  - Rögzítés és visszajátszás biztosítása
  - Interfész és implementáció elválasztása
- Használhatóság
  - Futásidejű technikák: Felhasználói modell, feladatmodell, rendszermodell karbantartása
  - Tervezési idejű technikák: Felhasználói felület elhatárolása

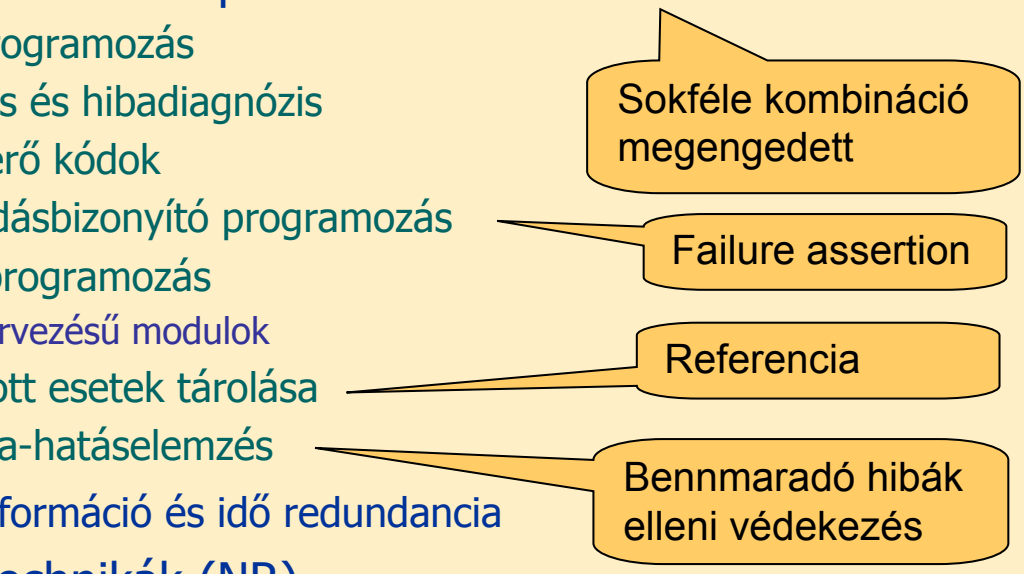
Összefoglalás: Elérendő tulajdonságok és tervezési tér

Elérendő tulajdonság	**Tervezési tér**
Szolgáltatásbiztonság	Hibadetektálás, hibabehatárolás, helyreállítás, hibakezelés
Teljesítmény	Erőforrás hozzárendelés, erőforrás menedzsment
Adatbiztonság	Támadás kivédés, felderítés, helyreállítás
Módosíthatóság	Lokalizálás, dominóhatás elkerülése, kötések elhalasztása
Tesztelhetőség	Vezérelhetőség, megfigyelhetőség, rögzítés és visszajátszás
Használhatóság	Felhasználói és feladatmodell, felület elhatárolása (MVC)

Példa: Előírt módszerek a biztonsághoz (EN 50128)

- SIL 1-től R, SIL 3-tól tipikusan HR technikák
  - Defenzív programozás
  - Hibafelfedés és hibadiagnózis
  - Hibafelismerő kódok
  - Meghibásodásbizonyító programozás
  - Diverziter programozás
    - Eltérő tervezésű modulok
  - Megvalósított esetek tárolása
  - Szoftverhiba-hatáselemzés

  -> Szoftver, információ és idő redundancia

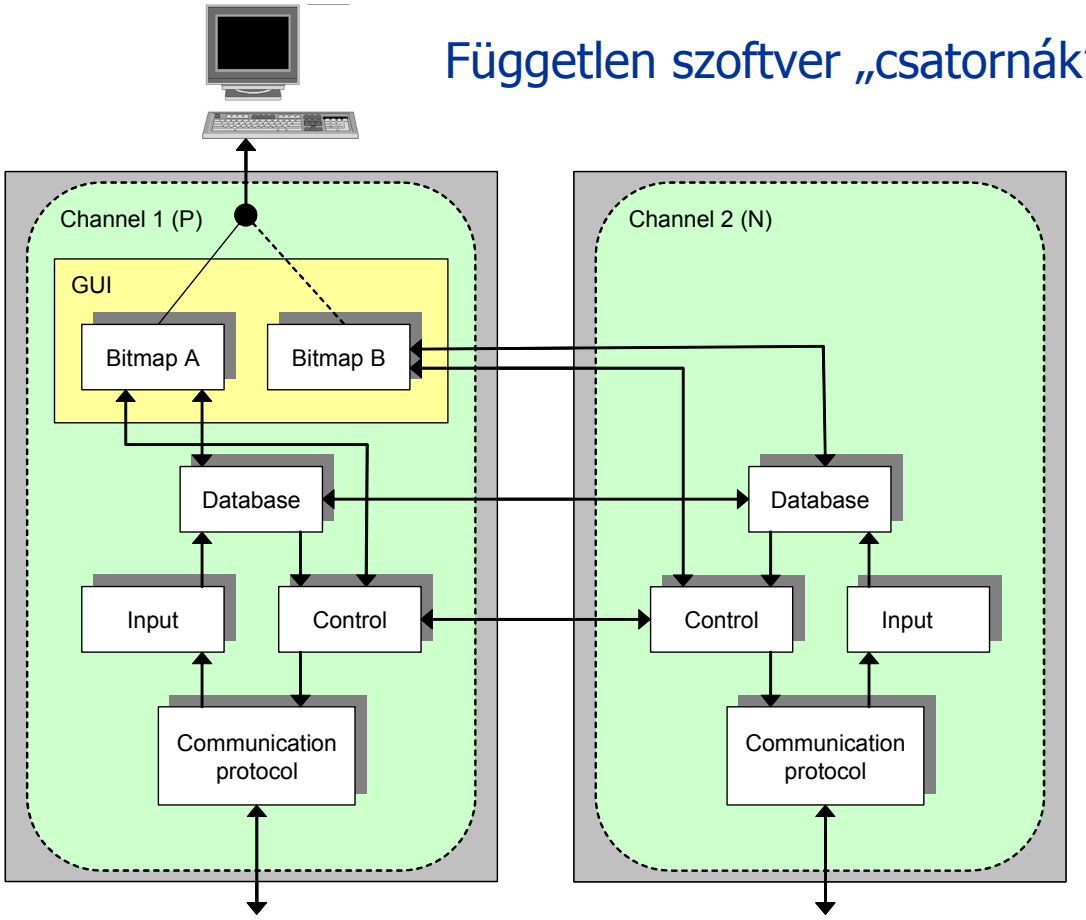


- Ellenjavallt technikák (NR)
  - Előre / visszalépő helyreállítás
  - Mesterséges intelligencia módszerek hibajavításra
  - Dinamikus szoftver rekonfiguráció

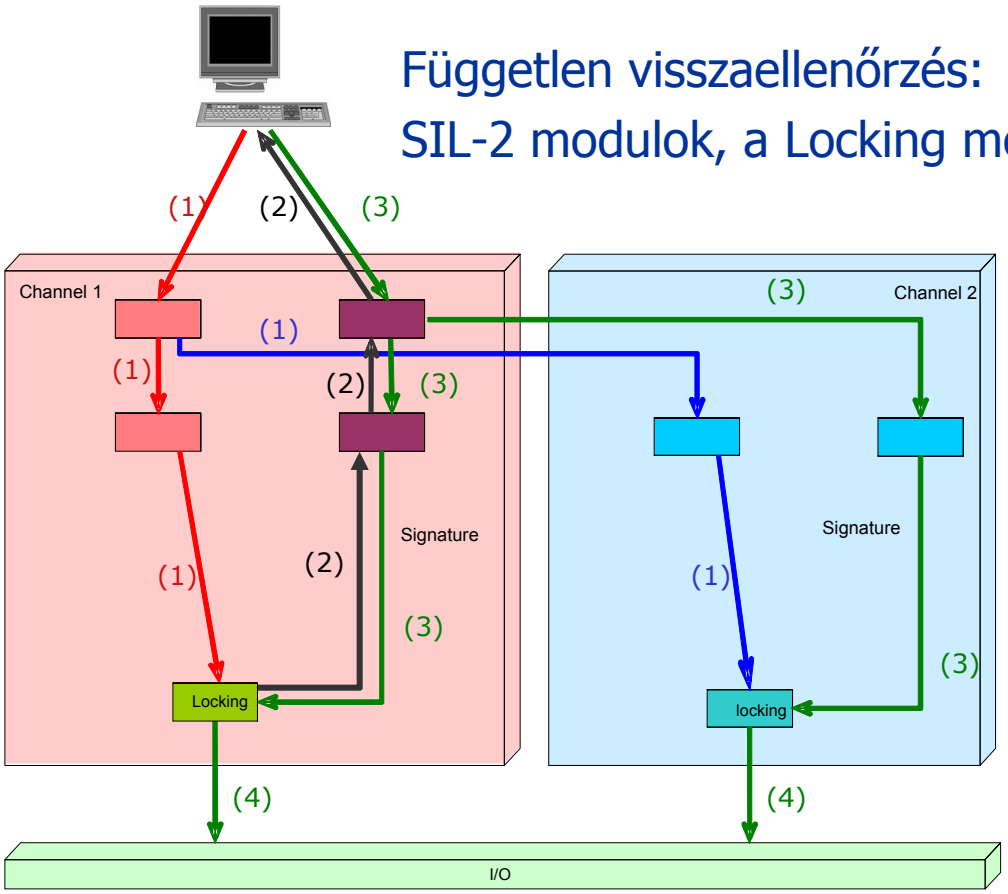
Példa: Architekturális megoldás SIL csökkentésre

- Komponensek SIL szintje
  - Alapértelmezés: Egyező a rendszerével
  - Csökkentés: Megakadályozható a szoftver komponens hibájából adódó rendszerszintű hiba
    - SPOF elkerülése
    - Függetlenséget igazolni kell
- (C)OTS komponensek használata
  - SIL 0: Általánosan elfogadott
  - SIL 1 és 2: Validációnak tartalmaznia kell
  - SIL 3 és 4:
    - Validációnak tartalmaznia kell +
    - Lehetséges meghibásodások elemzése kell +
    - Védelmi stratégia és ennek tesztelése kell +
    - Hibanaplók felvétele és kiértékelése szükséges

Példa: Architekturális megoldás SIL csökkentésre



Példa: Architekturális megoldás SIL csökkentésre



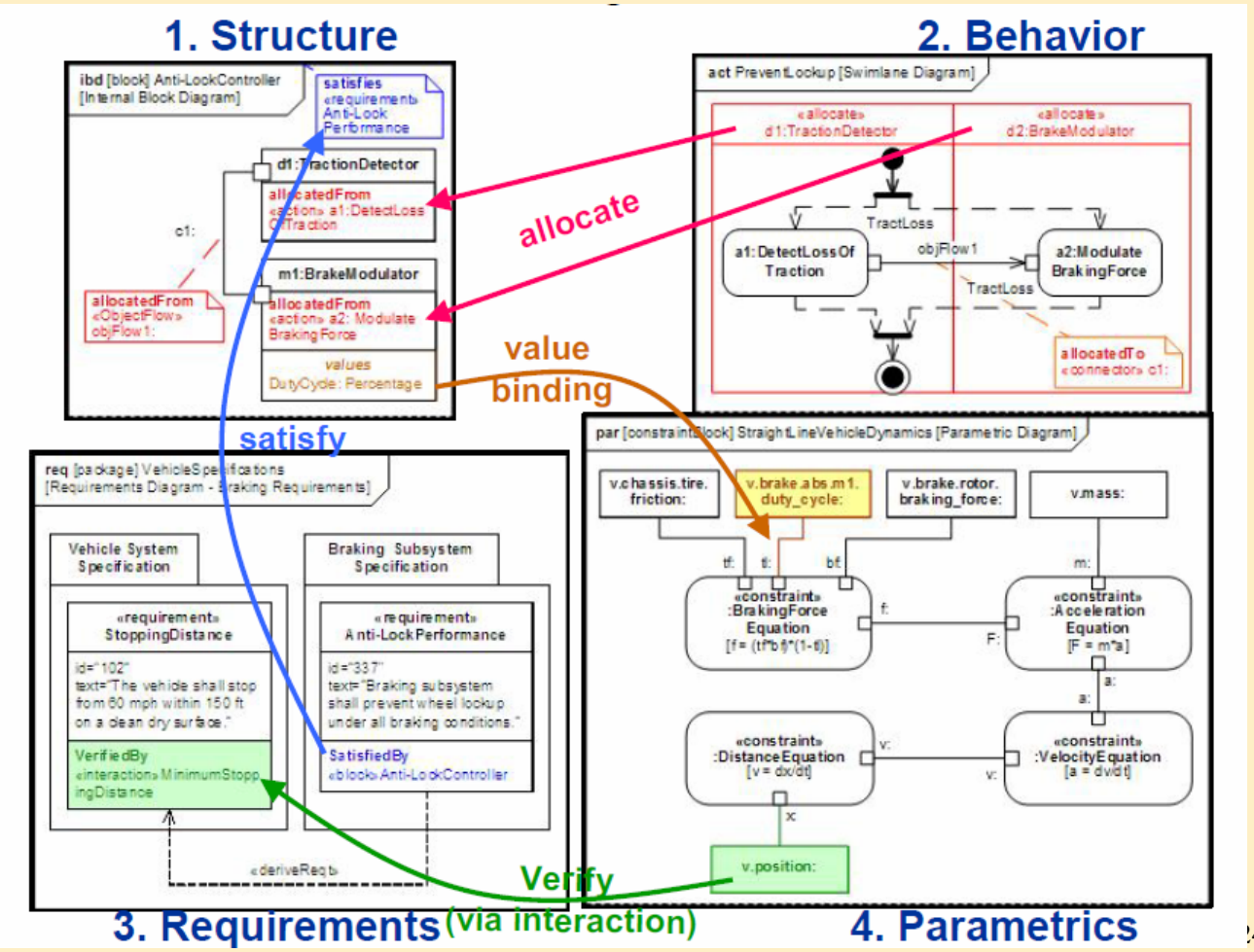
Áttekintés: Milyen vizsgálati technikák vannak?

- (Funkcionális) követelményekkel való összevetés
  - Architektúra megfelelősége: Ld. követhetőség
- Szisztematikus elemzés: Az architektúra „bejárása”
  - Interfész analízis
    - Elvárt és nyújtott interfészek megfelelősége
  - Hibahatás analízis kombinatorikus módszerekkel
    - Komponens szintű hibák ↔ Rendszerszintű hatások
- Modell alapú vizsgálatok
  - Extra-funkcionális jellemzők meghatározása
    - Tipikusan sztochasztikus módszerek
    - Lokális (komponens illetve kapcsolati szintű) paraméterek alapján rendszerszintű jellemzők számítása
    - Az analízis modell konstruálása: Az architektúra alapján
- Trade-off analízis

Tartalomjegyzék

- Motiváció
  - Mit határoz meg az architektúra?
  - Milyen vizsgálati módszerek vannak?
- Követhetőség
- Szisztematikus vizsgálati módszerek
  - Interfész analízis
  - Hibahatás analízis
- Modell alapú vizsgálatok
  - Megbízhatósági modellezés
  - Teljesítmény modellezés
- ATAM architektúra elemzés

Példa: Követhetőség SysML relációk alapján



Tartalomjegyzék

- Motiváció
  - Mit határoz meg az architektúra?
  - Milyen vizsgálati módszerek vannak?
- Követhetőség
- Szisztematikus vizsgálati módszerek
  - Interfész analízis
  - Hibahatás analízis
- Modell alapú vizsgálatok
  - Megbízhatósági modellezés
  - Teljesítmény modellezés
- ATAM architektúra elemzés

Szisztematikus vizsgálat: Interfész analízis

- Célkitűzés
  - Komponens interfészek megfelelősége
  - Teljesség: A kapcsolatok szisztematikus „bejárása” biztosítja
- Szintaktikus analízis
  - Hívási paraméterek száma, típusa (signature)
- Szemantikus analízis
  - Komponensekhez rendelt funkcionalitás leírása alapján
  - Szerződés (contract) alapú analízis
- Viselkedési analízis
  - Komponens viselkedés specifikáció alapján
  - Összetett protokollok esetén szokásos (konformancia)
  - Viselkedési ekvivalencia relációk definiálhatók
    - Trace ekvivalencia
    - Biszimuláció
  - Időzítések is vizsgálhatók

Példa: Interfész analízis

- „Contract based” komponens specifikáció példa: JML

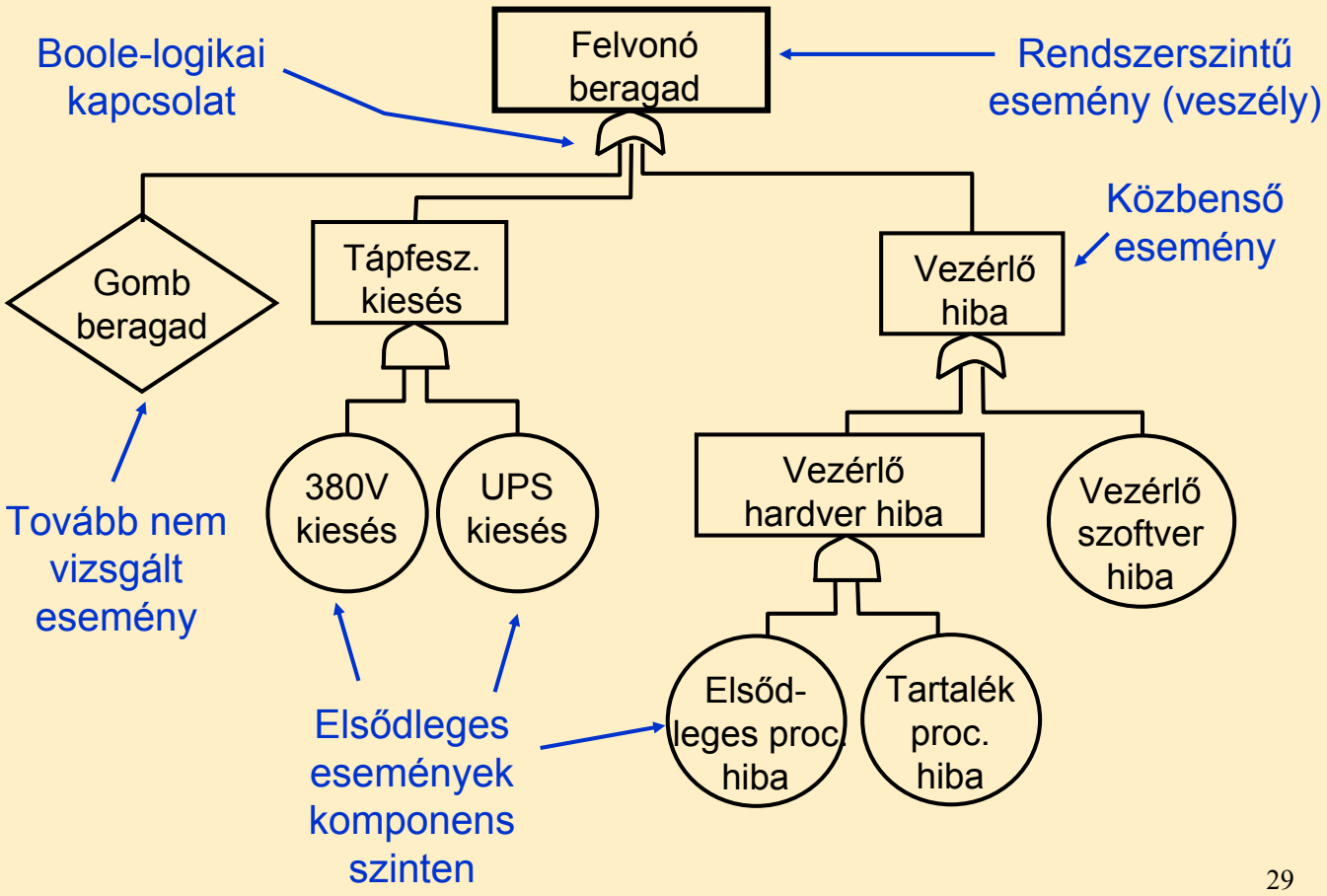
```
public class Purse {
   final int MAX_BALANCE;
   int balance;
     /*@ invariant pin != null && pin.length == 4  @*/
   byte[] pin;
     /*@ requires amount >= 0;
        @ assignable balance;
        @ ensures balance == \old(balance) – amount
                && \result == balance;
        @ signals (PurseException) balance == \old(balance);
        @*/
   int debit(int amount) throws PurseException {
     if (amount <= balance) {
        balance -= amount;
        System.out.println("Debit placed"); return balance; }
     else {
        throw new PurseException("overdrawn by " + amount); }}
```

- Tételbizonyítás (EscJava2), futásidejű verifikáció (jmlc, jml)

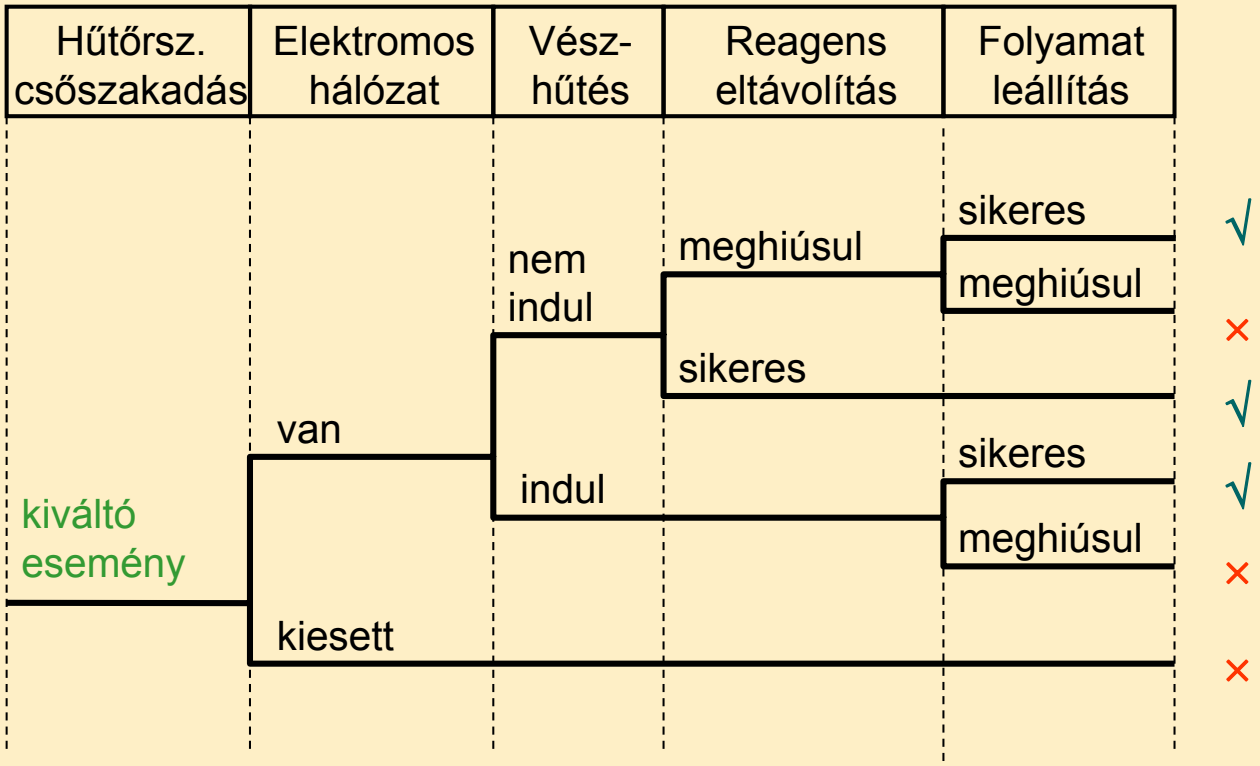
Szisztematikus vizsgálat: Hibahatás analízis

- Célkitűzés
  - Komponens jellemzők és rendszerszintű jellemzők közötti kapcsolatok megállapítása
  - Teljesség: Minden komponens hibamódjai szerepelnek
- Analízis megközelítése
  - Az architektúra szempontjából
    - Alulról felfelé: A komponensek (alrendszerek) felől
    - Felülről lefelé: Rendszerszintről lebontva
  - Ok-okozati szempontból:
    - Előrelépő (induktív): Esemény hatásainak vizsgálata
    - Visszalépő (deduktív): Veszély okainak felderítése
- Tipikus módszerek a vizsgálatra
  - Hibafa
  - Eseményfa
  - Ok-következmény analízis
  - Hibamód és -hatás analízis (FMEA)

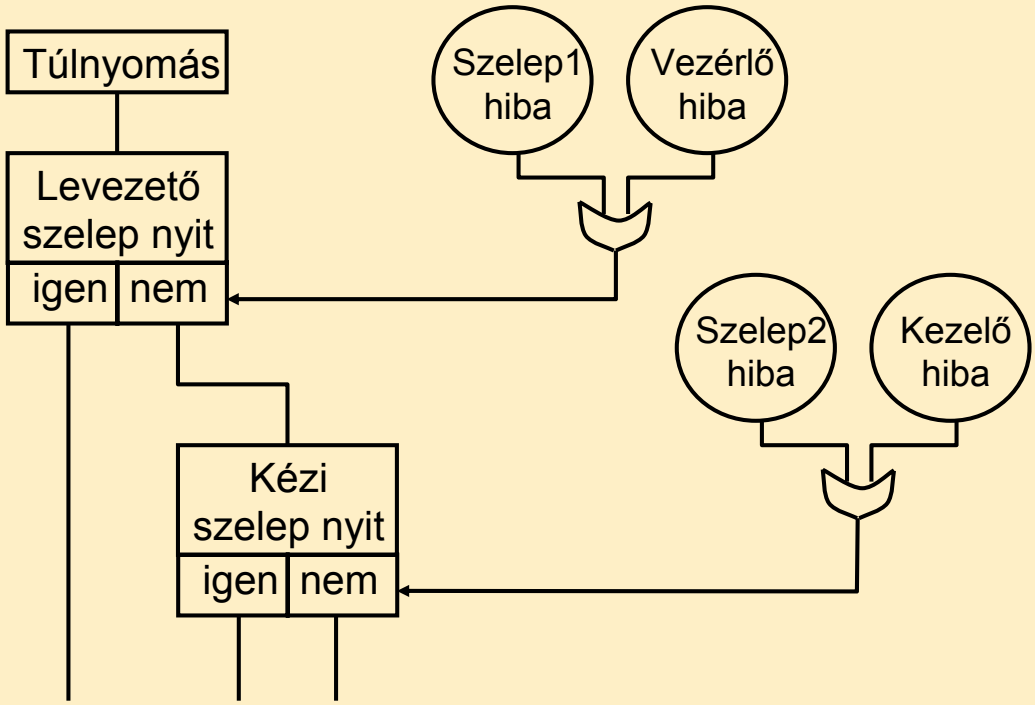
Hibafa példa: Felvonó



Eseményfa példa: Reaktorhűtés



Ok-következmény analízis példa: Túlnyomás védelem



Hibamód és -hatás analízis (FMEA) példa

- Hibák és hatásaik felmérése
- A táblázat teljessége
  - Minden komponens
  - Minden hibamód
- Hatás analízis nem egyszerű

Komponens	Hibamód	Valószínűség	Hatás
L határérték-túllépés vizsgálat	> L átmegy ≤ L nem megy át	65% 35%	- túlnyomás - technológiai hiba
...	...	...	...

Tartalomjegyzék

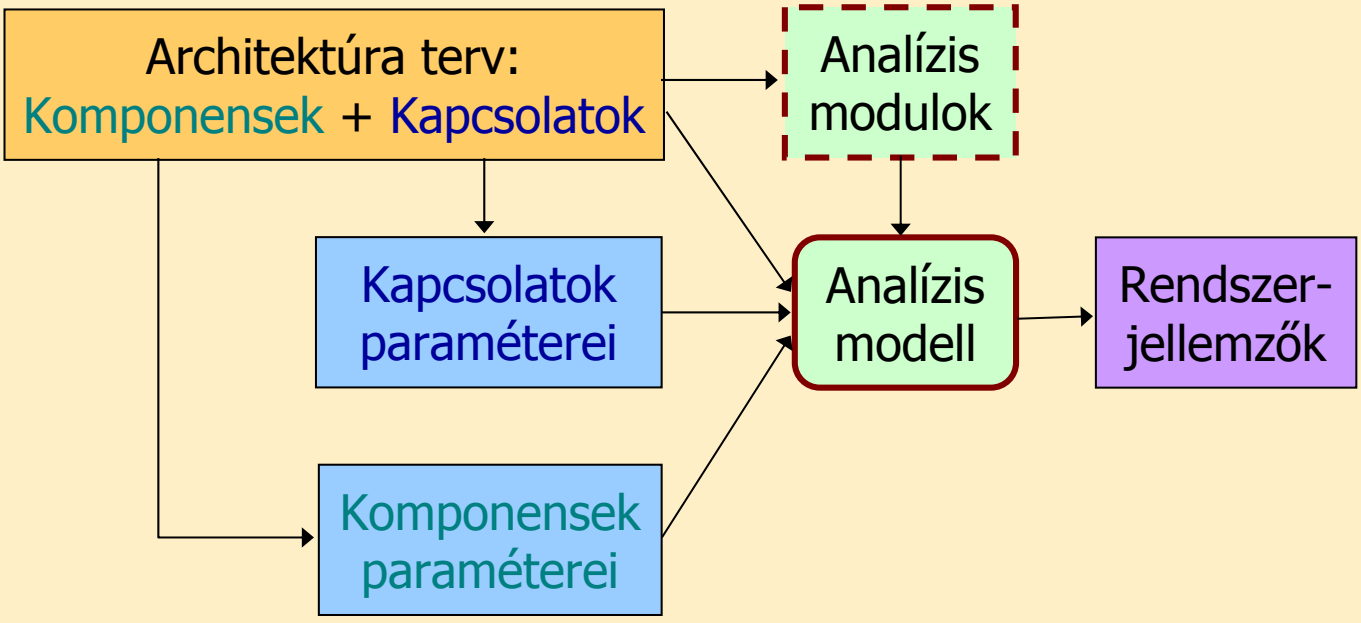
- Motiváció
  - Mit határoz meg az architektúra?
  - Milyen vizsgálati módszerek vannak?
- Követhetőség
- Szisztematikus vizsgálati módszerek
  - Interfész analízis
  - Hibahatás analízis
- Modell alapú vizsgálatok
  - Megbízhatósági modellezés
  - Teljesítmény modellezés
- ATAM architektúra elemzés

Modell alapú architektúra elemzés

Cél: Architektúra változatok kiértékelése

- Analízis modellek készítése és paraméterezése az architektúra modellje alapján
  - Teljesítmény modell
  - Megbízhatósági modell
  - Biztonsági modell
  - Adatbiztonsági modell
- Moduláris modellalkotás – az automatizálás alapja
  - Architektúra: Komponensek és kapcsolatok
  - Analízis modell: Ezekhez analízis modellkönyvtár elemei
- Rendszerszintű analízis modellek megoldása
  - Lokális (komponens illetve kapcsolati szintű) paraméterek alapján rendszerszintű jellemzők számítása

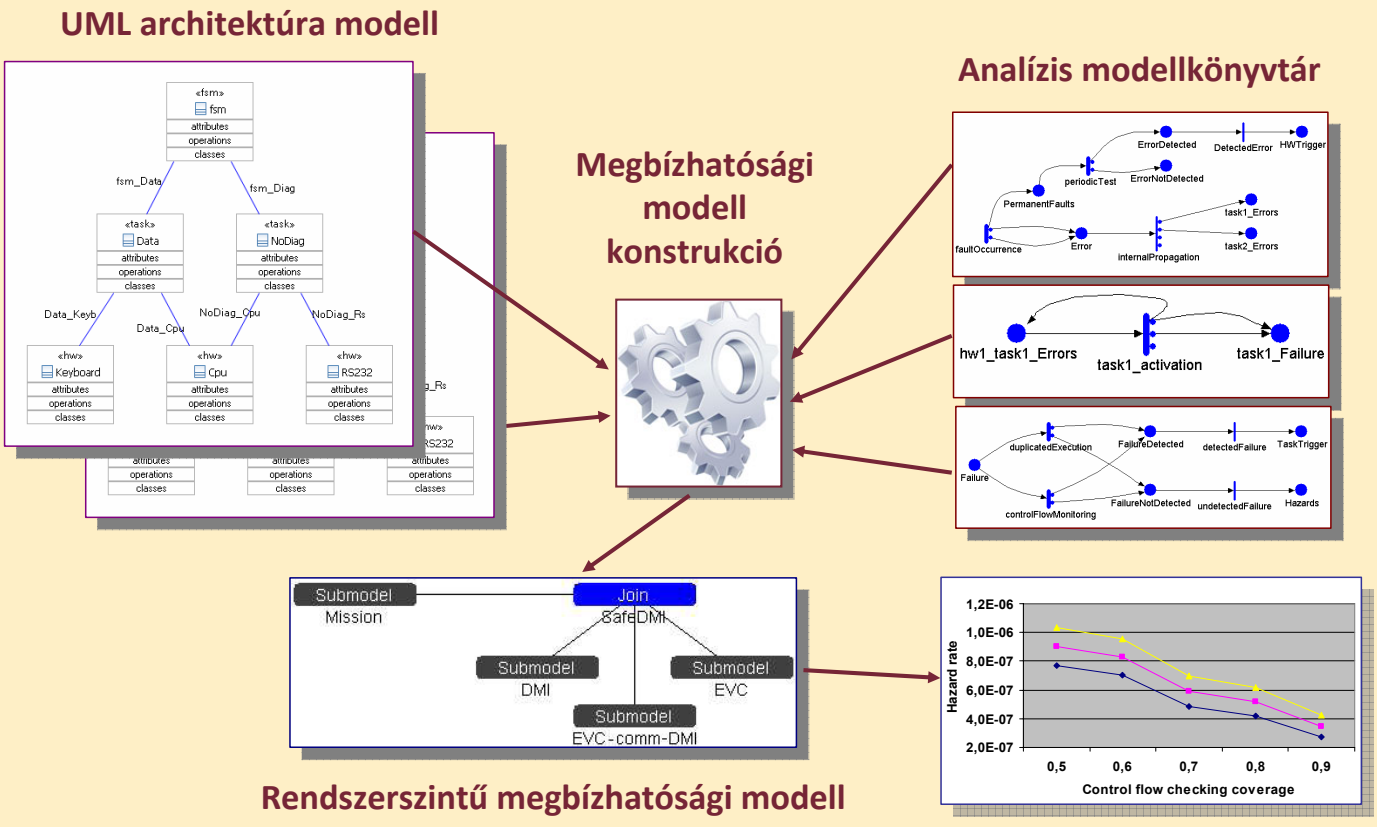
Modell alapú vizsgálatok



Rendszerszintű analízis modellek

	Megbízhatósági modell	**Teljesítmény modell**	**Biztonsági modell**
Komponens paraméterek	Meghibásodási tényező, lappangási idő, javítási tényező, hibafedés, …	Funkció lokális végrehajtási idő, taszk prioritás, processzor ütemezés	Veszély gyakoriság
Kapcsolat paraméterek	Hibaterjedési valószínűség, hibaterjedés feltételei, javítási stratégia	Hívás továbbítási gyakoriság, hívás szinkronitás	Veszély forgatókönyv, veszély kombinációk
Modell	Markov-lánc, Petri-háló	Sorbanállási háló	Markov-lánc, Petri-háló
Rendszer jellemzők (számított)	Megbízhatóság, rendelkezésre állás, készenlét, MTTF, MTTR, MTBF	Kiszolgálási idő, taszk áteresztő-képesség, processzor kihasználtság	Rendszerszintű veszély gyakoriság

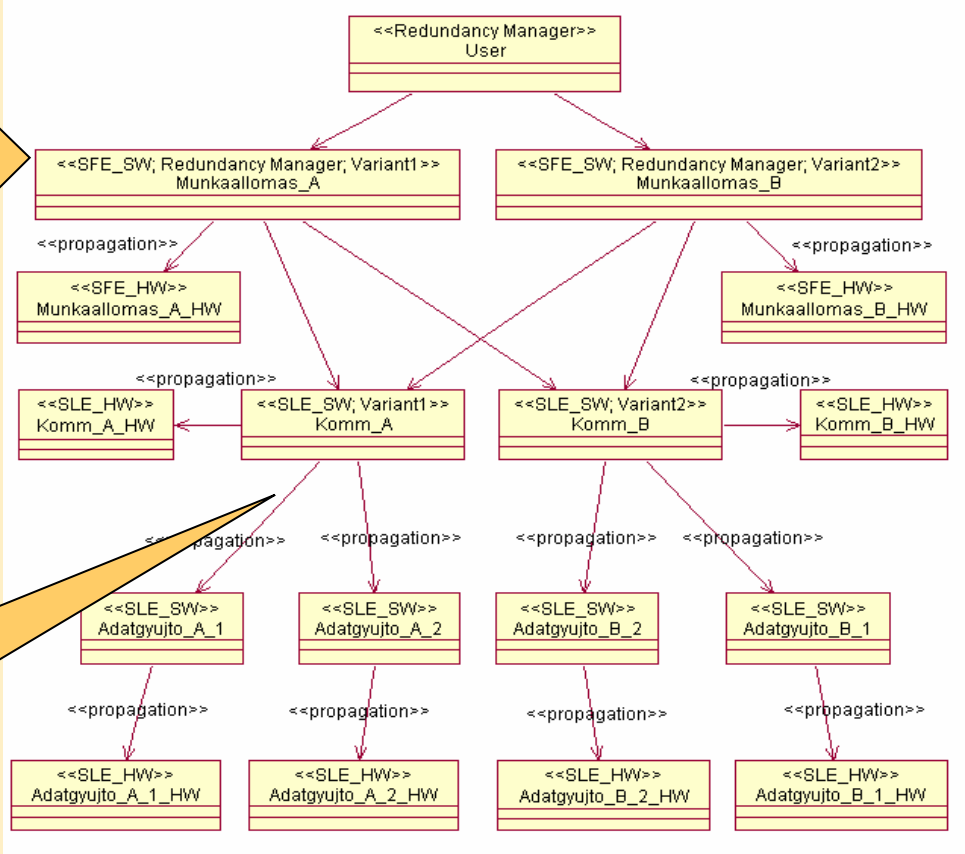
Példa: UML alapú megbízhatósági modellezés



Példa: Megbízhatósági modellezés – Architektúra modell

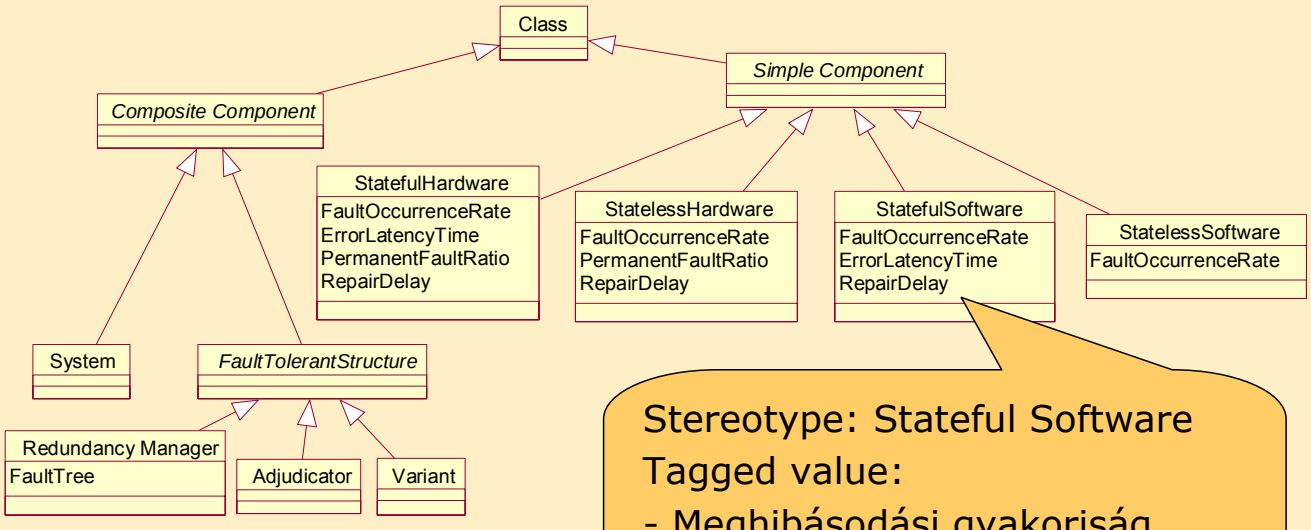
Komponens:
- Típus (HW, SW)
- Szerep (variáns, menedzser)
- Meghibásodási jellemzők:
  * meghibásodási gyakoriság,
  * lappangási idő,
  * javítási idő

Kapcsolat:
- Hibaterjesztési jellemzők:
  * hibaterjesztési valószínűség



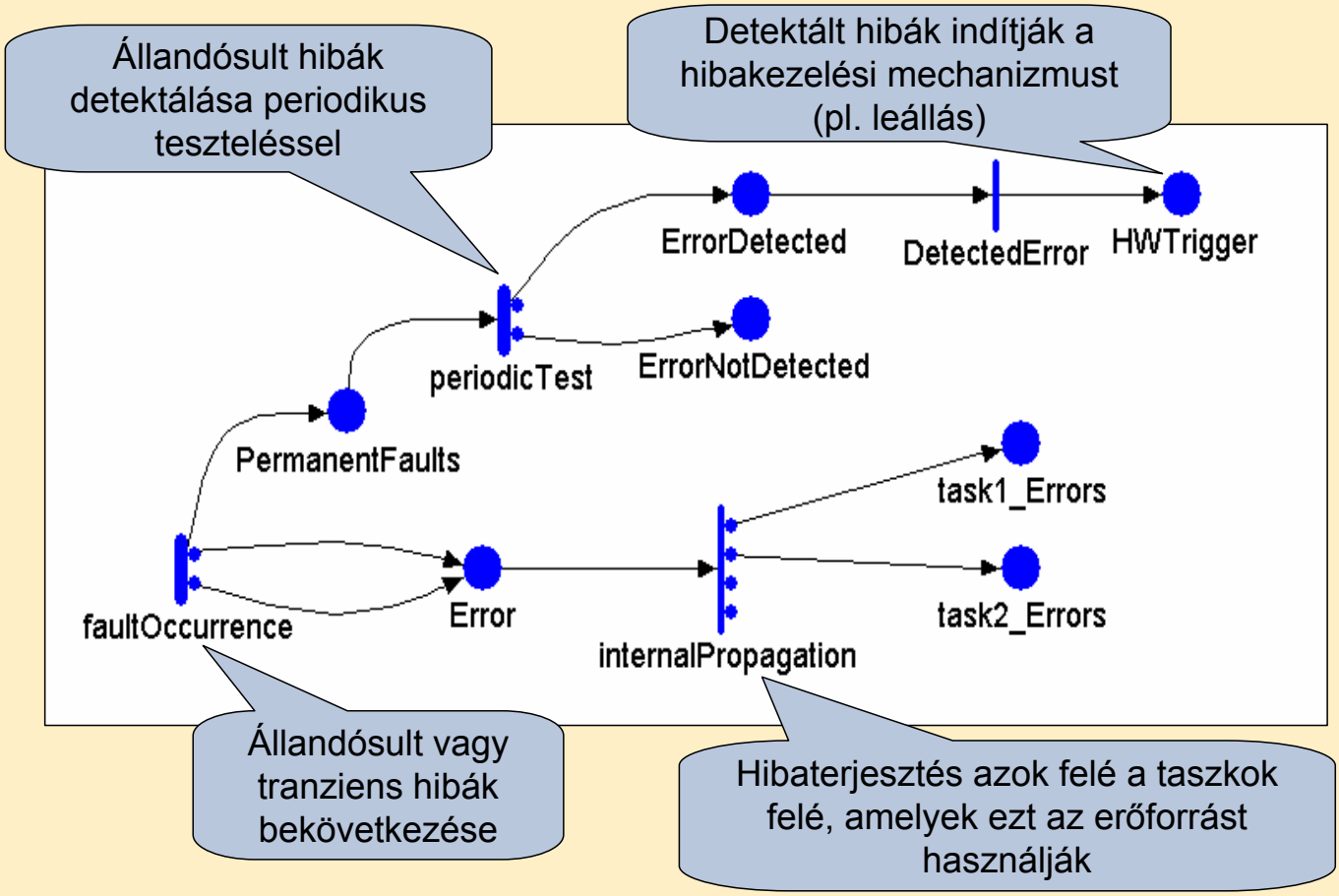
Példa: Komponensek lokális tulajdonságai

- UML profil az extra-funkcionális tulajdonságokhoz

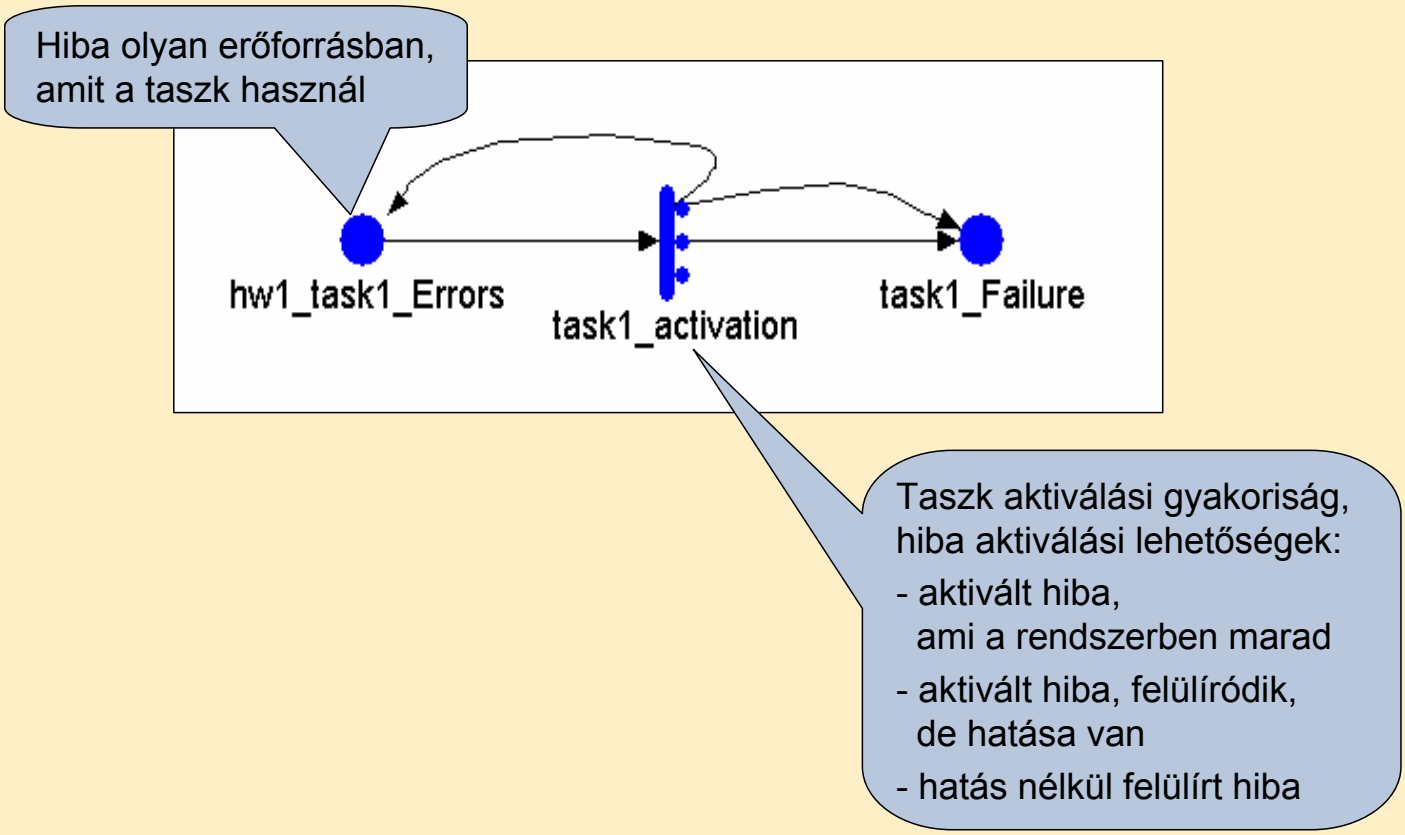


Stereotype: Stateful Software
Tagged value:
- Meghibásodási gyakoriság
- Hiba lappangási idő
- Javítási tényező

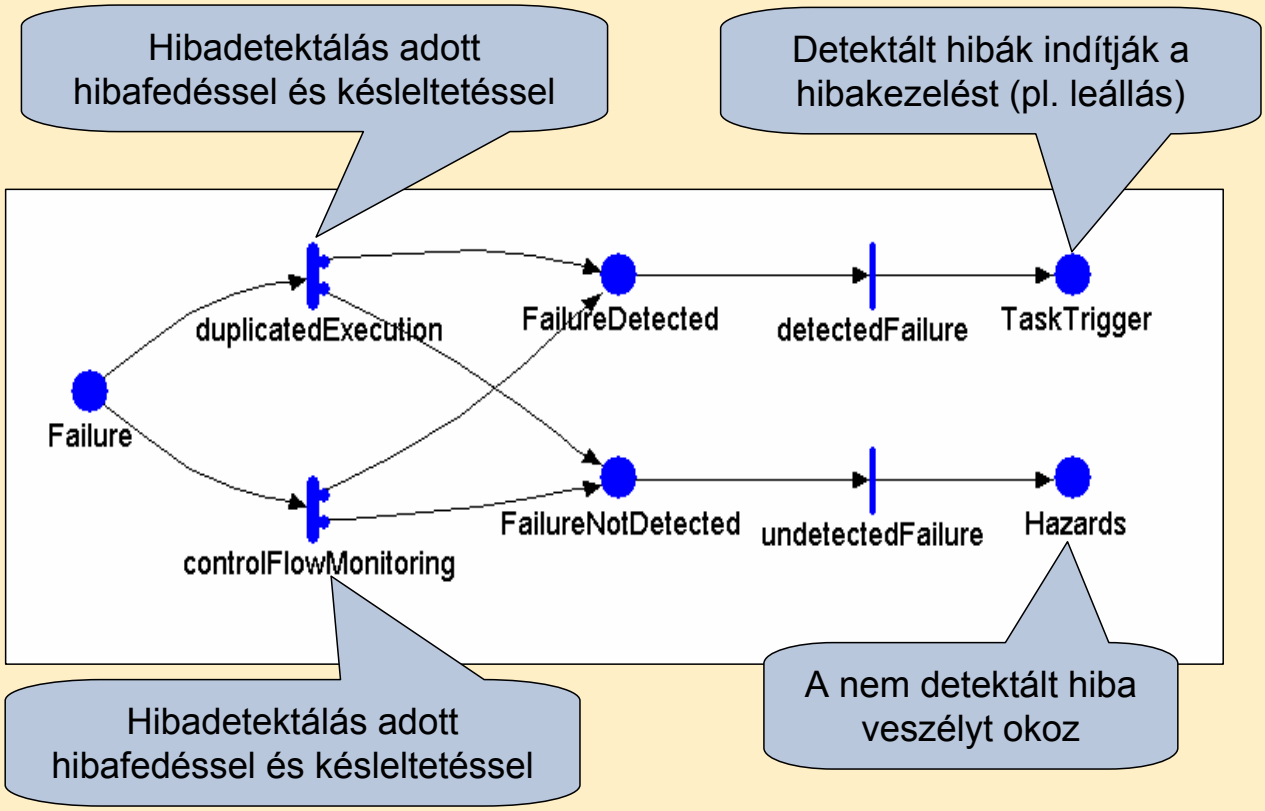
Példa: Komponens analízis alháló: Egy HW erőforrás (memória)



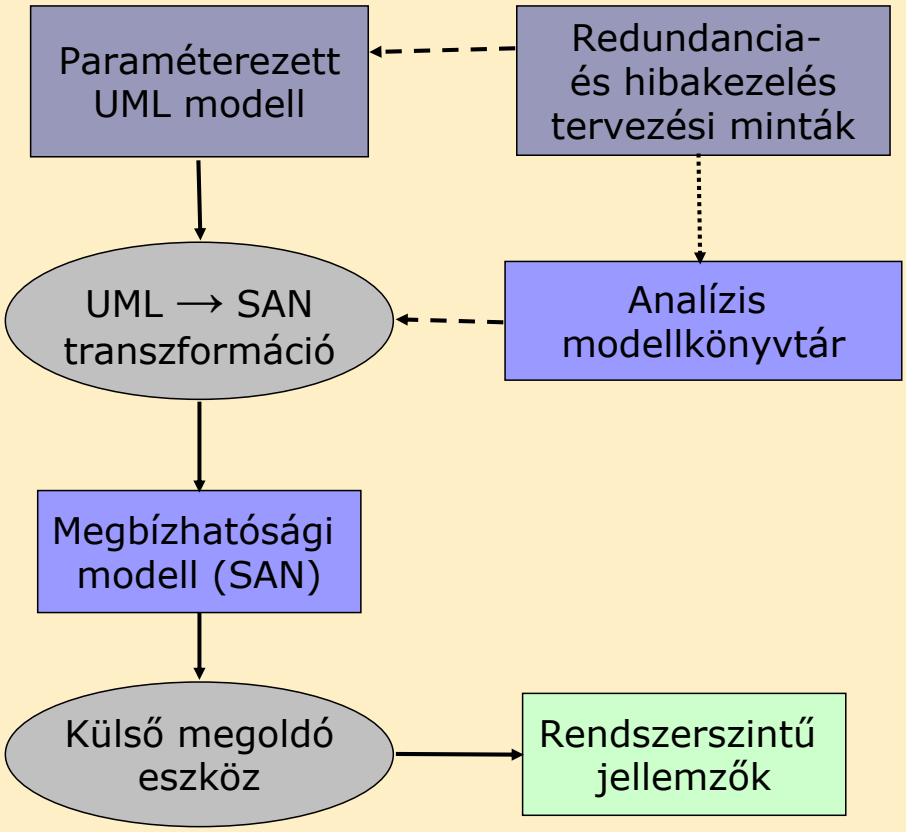
Példa: Kapcsolat analízis alháló: Hibaterjesztés



Példa: Komponens analízis alháló: Egy szoftver taszk



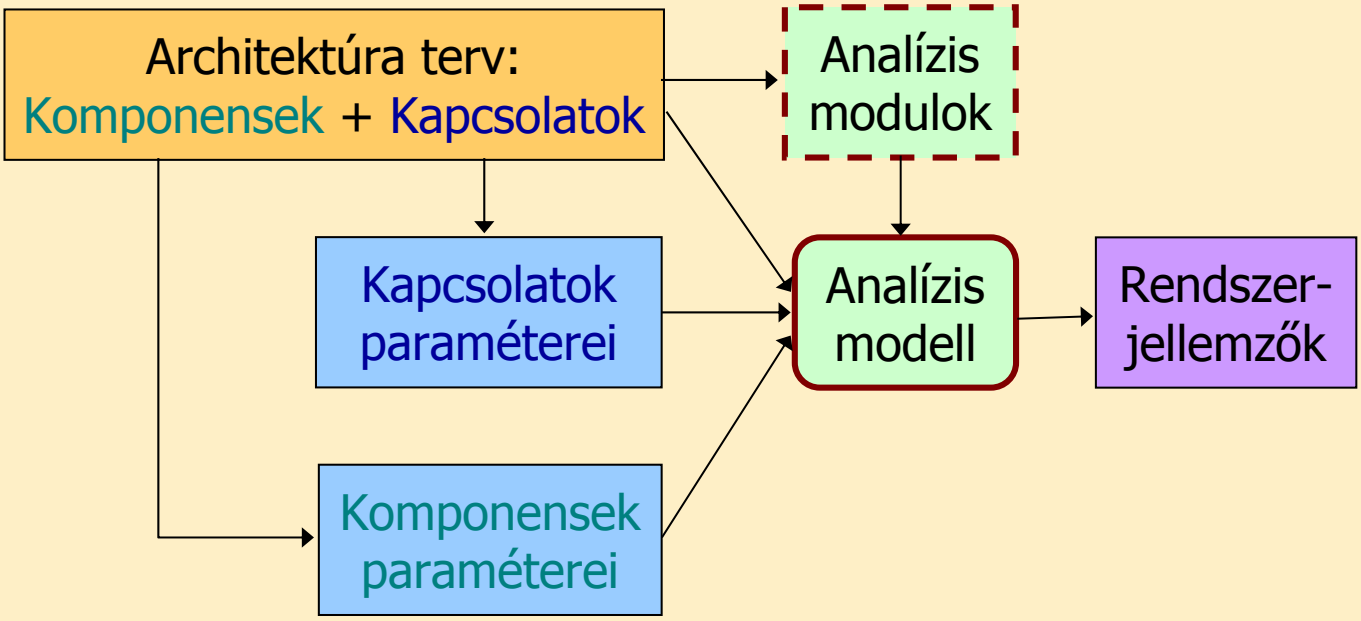
Példa: Automatikus analízis eszköz



Tartalomjegyzék

- Motiváció
  - Mit határoz meg az architektúra?
  - Milyen vizsgálati módszerek vannak?
- Követhetőség
- Szisztematikus vizsgálati módszerek
  - Interfész analízis
  - Hibahatás analízis
- Modell alapú vizsgálatok
  - Megbízhatósági modellezés
  - Teljesítmény modellezés
- ATAM architektúra elemzés

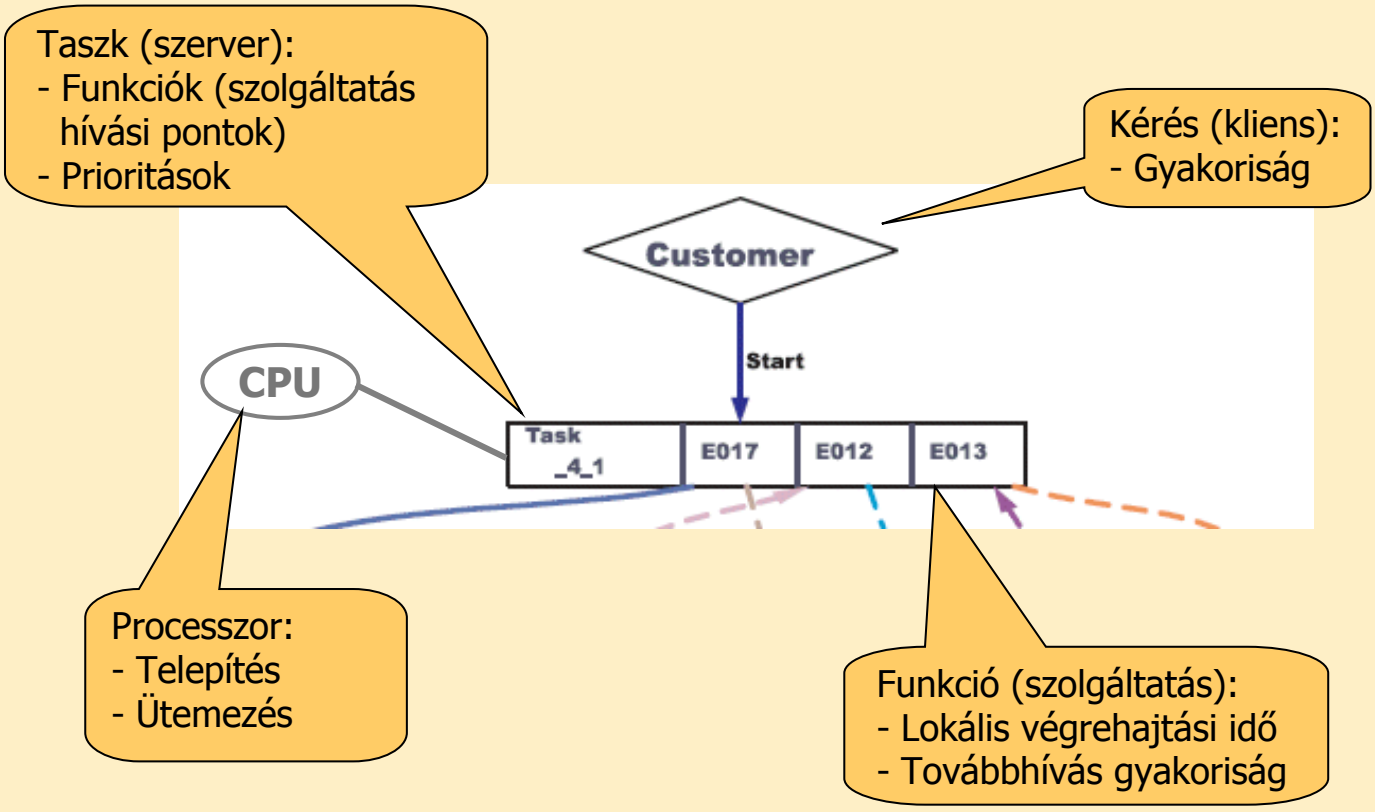
Modell alapú vizsgálatok



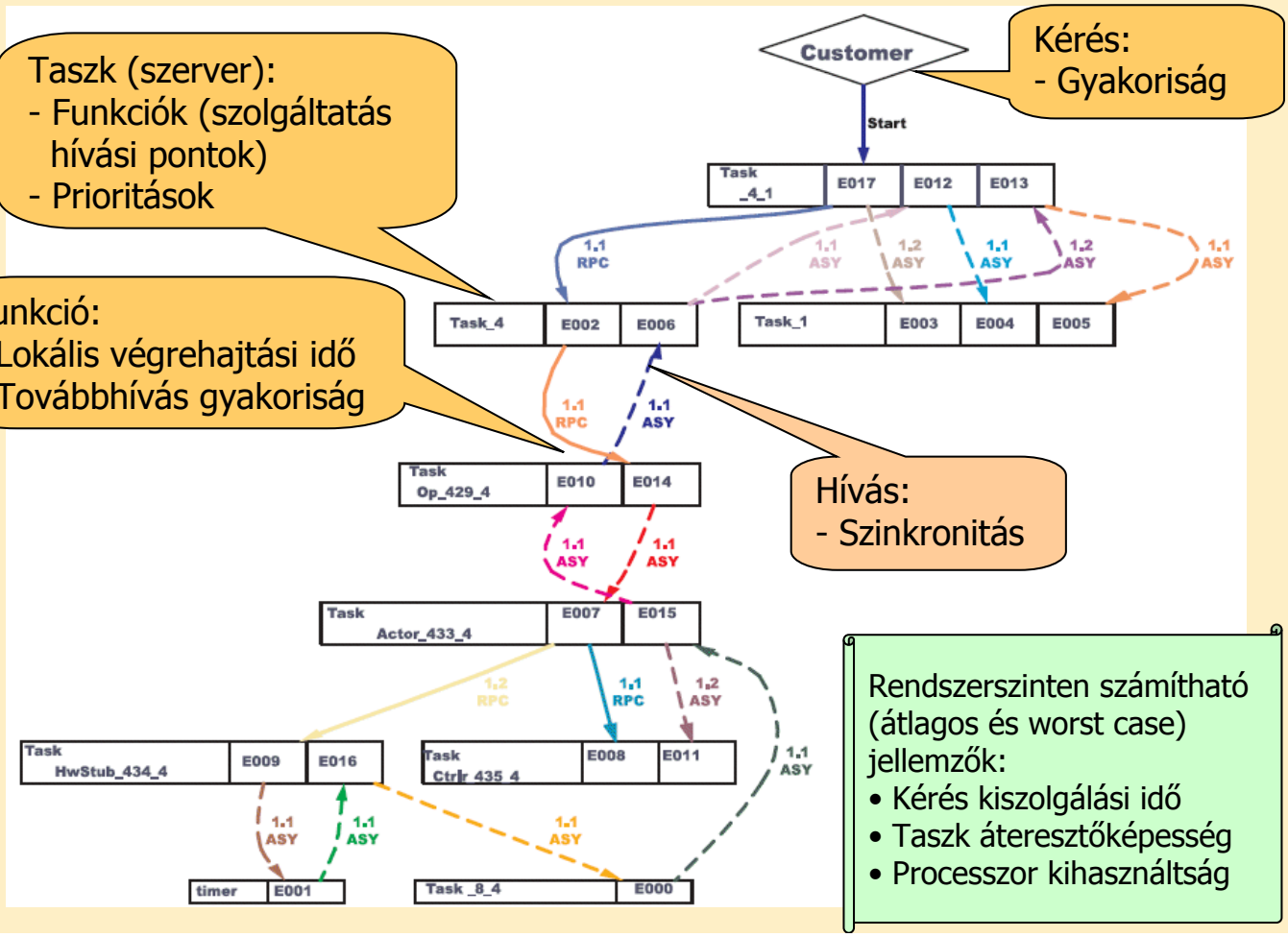
Modell alapú architektúra vizsgálatok

	Megbízhatósági modell	**Teljesítmény modell**	**Biztonsági modell**
Komponens paraméterek	Meghibásodási tényező, lappangási idő, javítási tényező	Funkció lokális végrehajtási idő, taszk prioritás, ütemezés	Veszély gyakoriság
Kapcsolat paraméterek	Hibaterjedési valószínűség, hibaterjedés feltételei, javítási stratégia	Hívás továbbítási gyakoriság, hívás szinkronitás	Veszély forgatókönyv, veszély kombinációk
Modell	Markov-lánc, Petri-háló	Sorbanállási háló	Markov-lánc, Petri-háló
Rendszer jellemzők (számított)	Megbízhatóság, rendelkezésre állás, készenlét, MTTF, MTTR, MTBF	Kiszolgálási idő, taszk áteresztő-képesség, processzor kihasználtság	Rendszerszintű veszély gyakoriság

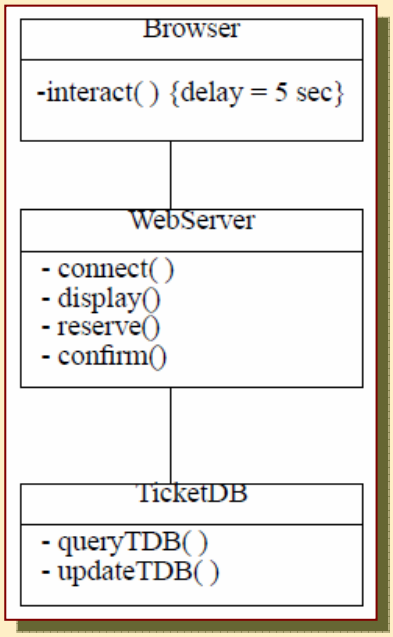
Példa: Teljesítmény modellezés (LQN)

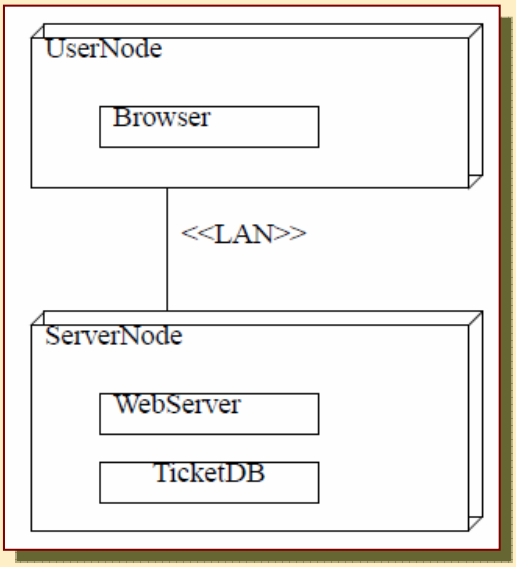


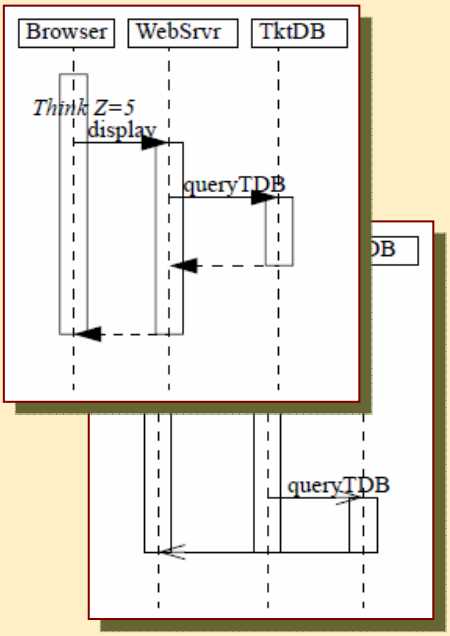
Példa: Teljesítmény modellezés (LQN): Rétegek



Példa: Az architektúra leképezése a teljesítmény modellre





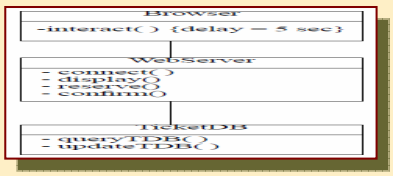


Osztályok és lokális jellemzők

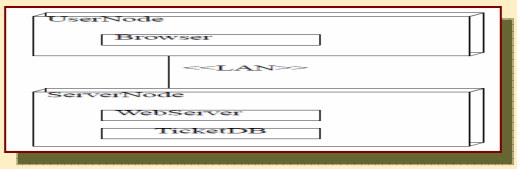
Processzorok és telepítés

Interakciók (hívások)

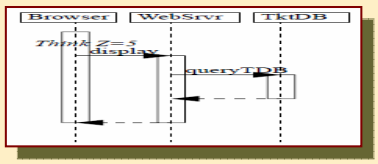
Példa: Az architektúra leképezése a teljesítmény modellre



Osztályok

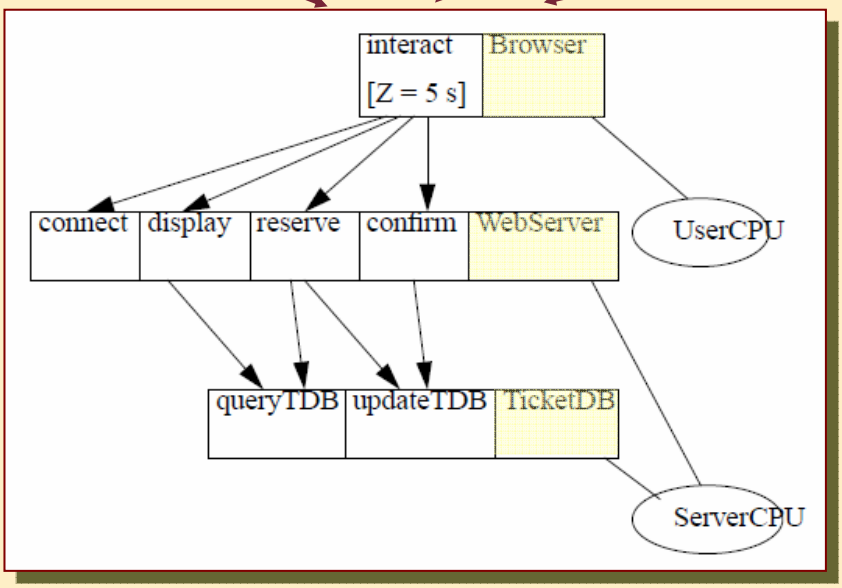


Telepítés



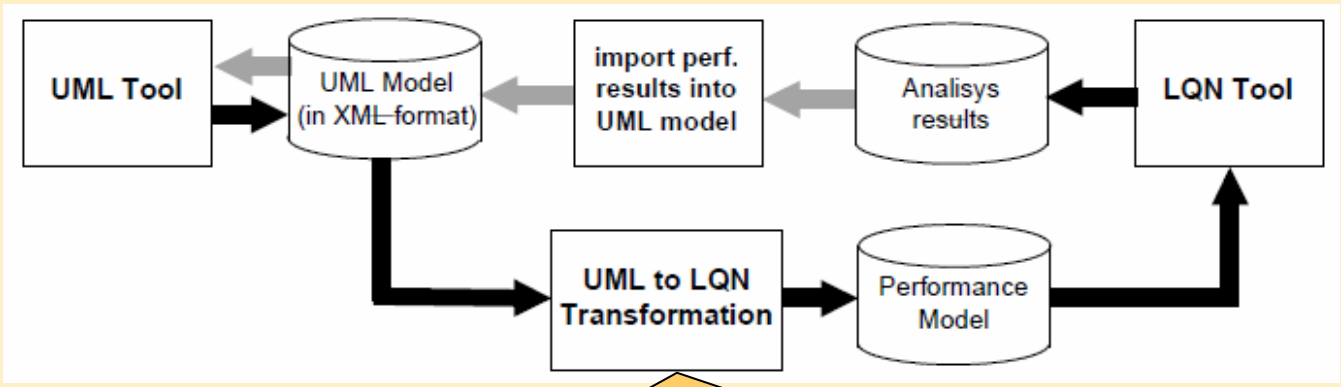
Interakciók

Modell-transzformáció



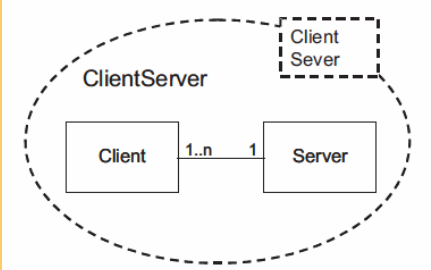
LQN teljesítmény-modell

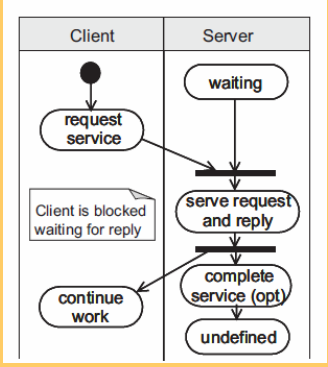
Példa: Az architektúra leképezése a teljesítmény modellre



- Architektúra minták használata

Szinkron üzenetküldés:





Tartalomjegyzék

- Motiváció
  - Mit határoz meg az architektúra?
  - Milyen vizsgálati módszerek vannak?
- Követhetőség
- Szisztematikus vizsgálati módszerek
  - Interfész analízis
  - Hibahatás analízis
- Modell alapú vizsgálatok
  - Megbízhatósági modellezés
  - Teljesítmény modellezés
- ATAM architektúra elemzés

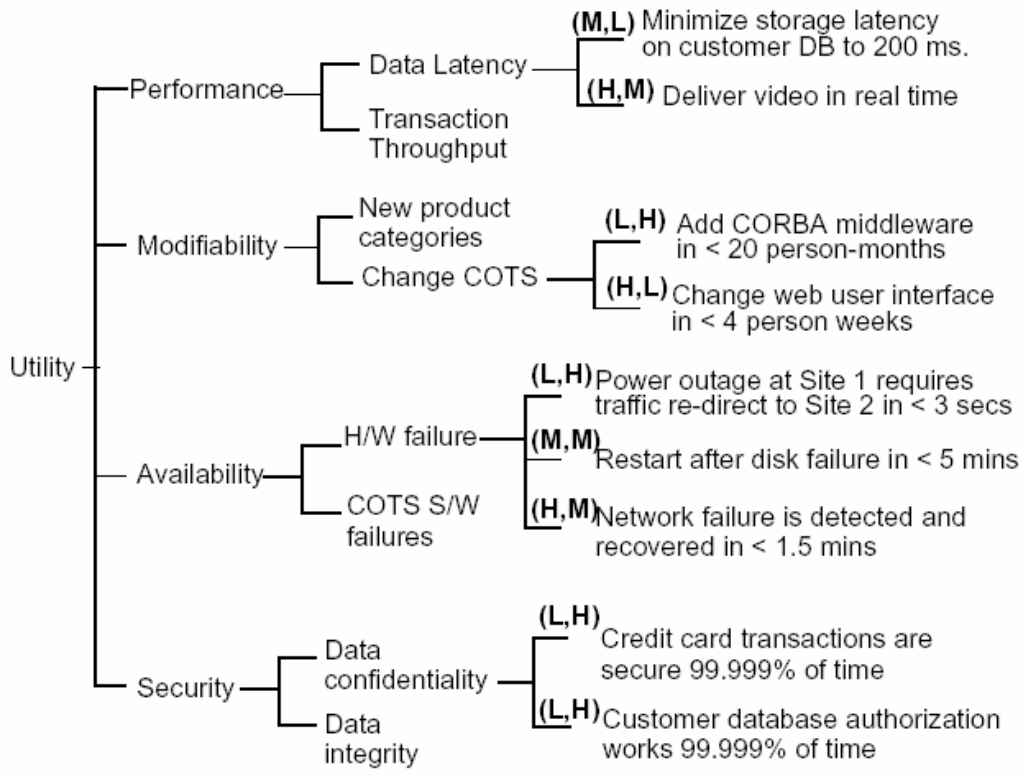
Architektúrák elemzése: ATAM

- Architecture Tradeoff Analysis Method (ATAM)
  - Hogyan elégíti ki az architektúra a minőségi célokat?
  - Hogyan viszonyulnak egymáshoz a célok?
- Fázisok
  1. Előkészítés: Ütemezés, közreműködők meghatározása
  2. Kiértékelés a tervezőkkel
  3. Kiértékelés a további érdekeltekkel
  4. Eredmények rögzítése

Kiértékelés a tervezőkkel

- Üzleti célok bemutatása (fejlesztés vezetője)
  - Funkciók, üzleti célok, elérendő minőségi jellemzők, érdekeltek
  - Megkötések: technikai, gazdasági, menedzsment
- Az architektúra bemutatása (tervezők)
  - Statikus és dinamikus nézetek
- Az architekturális tényezők azonosítása (tervezők, elemzők)
  - Minták felismerése, elnevezése
- Hasznossági fa elkészítése (tervezők és elemzők)
  - Gyökér: Általános megfelelőség
  - Második szint: Jellemzők (rendelkezésre állás, teljesítmény, ...)
  - További szintek: Finomított jellemzők
  - Levelekhez forgatókönyvek rendelése: Jellemző szerepének bemutatása
    - Bemenetek, hatások
    - Környezet
    - Elvárt reakció
  - Prioritások felvétele a forgatókönyvek (azaz jellemzők) között
- Az architektúra analízise (elemzők)
  - Hogyan támogatja az architektúra a fontos forgatókönyveket?
  - Mik a kockázatok, erős és gyenge pontok?

Hasznossági fa



Kiértékelés a további érdekeltekkel

- Új forgatókönyvek felvétele
  - Tesztelők, felhasználók, ... közreműködése
    - Karbantartás, tesztelés, ergonómia stb. szempontjai
  - Brainstorming
  - Prioritások meghatározása
- Az architektúra analízise
  - Elég nagy prioritású új forgatókönyvek esetén
- Összefoglaló készítése

Mintapélda: AbiWord architektúra elemzése

- Célok
- Architektúra
- Hasznossági fa, forgatókönyvek
- Analízis