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

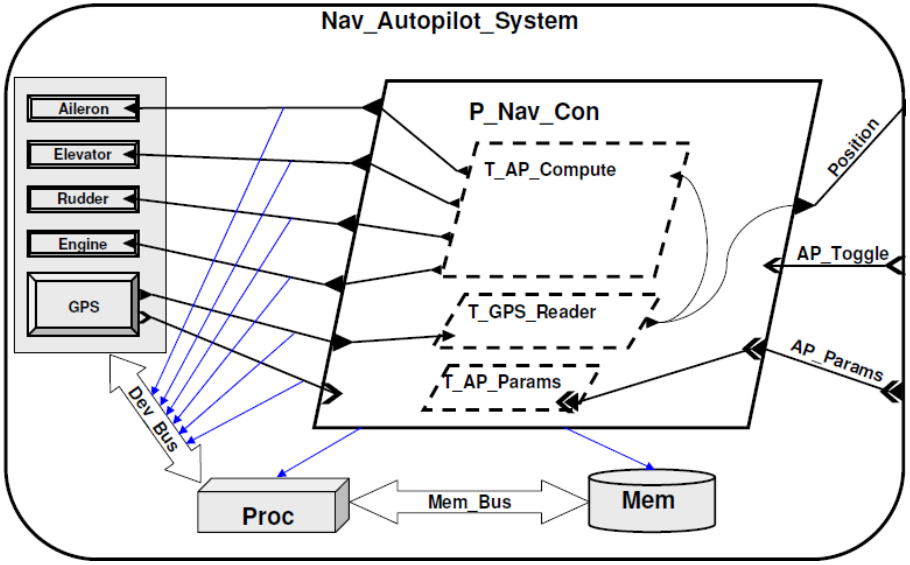
- Architektúra
  - Komponensek (tulajdonságokkal)
  - Közöttük lévő kapcsolatok (telepítés, információáramlás, ...)
- Tervezői döntések
  - Komponensek és kapcsolatok meghatározása
    - Hardver-szoftver együttműködés
    - Hibahatások figyelembe vétele
  - Méretezés (a komponensek és kapcsolatok tulajdonságai)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

Jellegzetes nyelvek architektúra tervezéshez

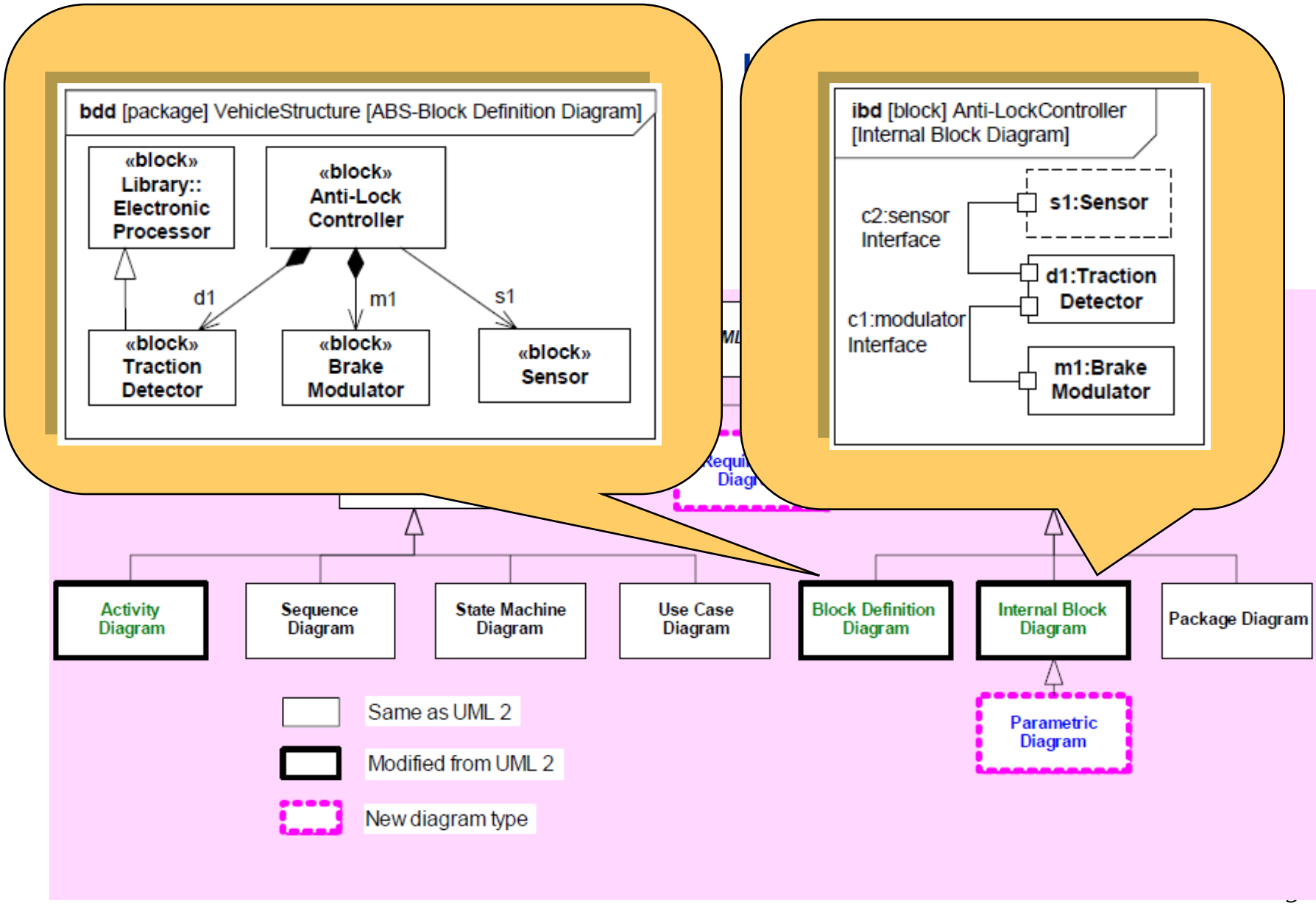
- UML
- SysML (Id. Block diagram)
- AADL: Architecture Analysis and Design Language
  - Komponensek
  - Kapcsolatok: Portokon (adatok, események áramlása)
  - Leképzés hardverre
  - Tulajdonságok analízishez

```
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: SysML

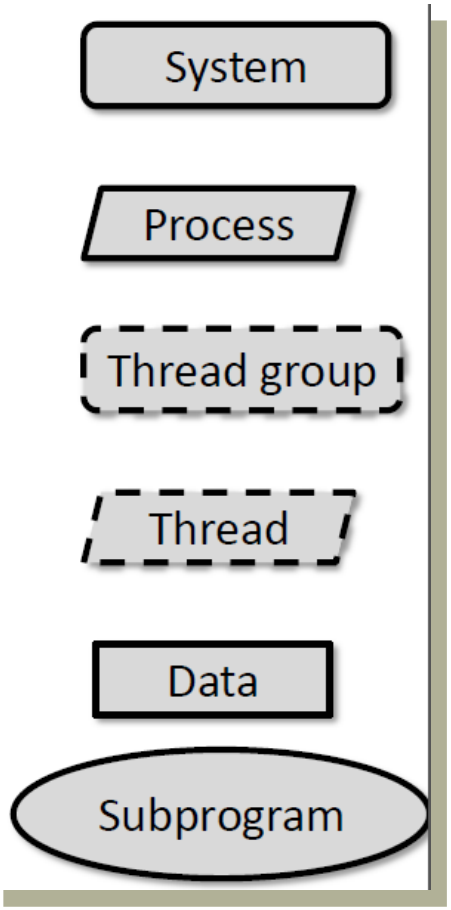


Jellegzetes nyelvek architektúra tervezéshez: AADL

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

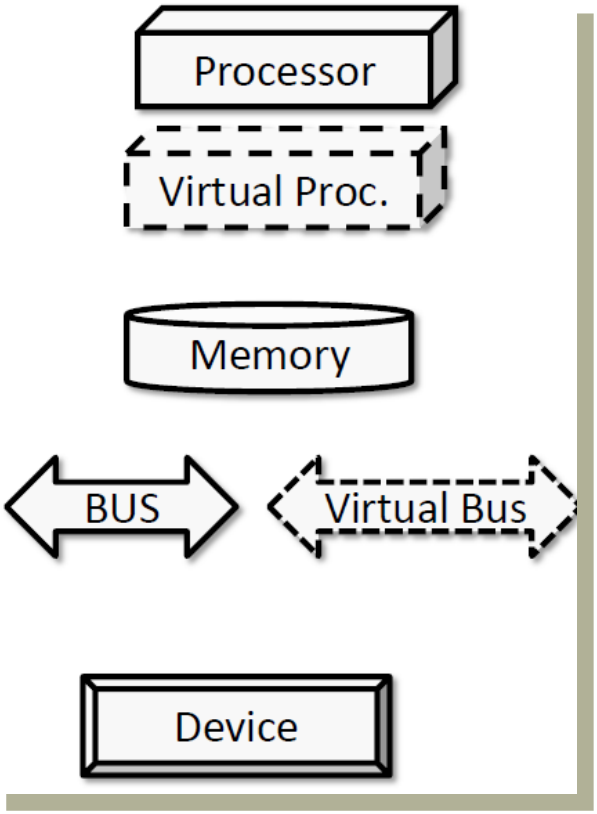
- Beágyazott rendszerekhez (SAE)

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



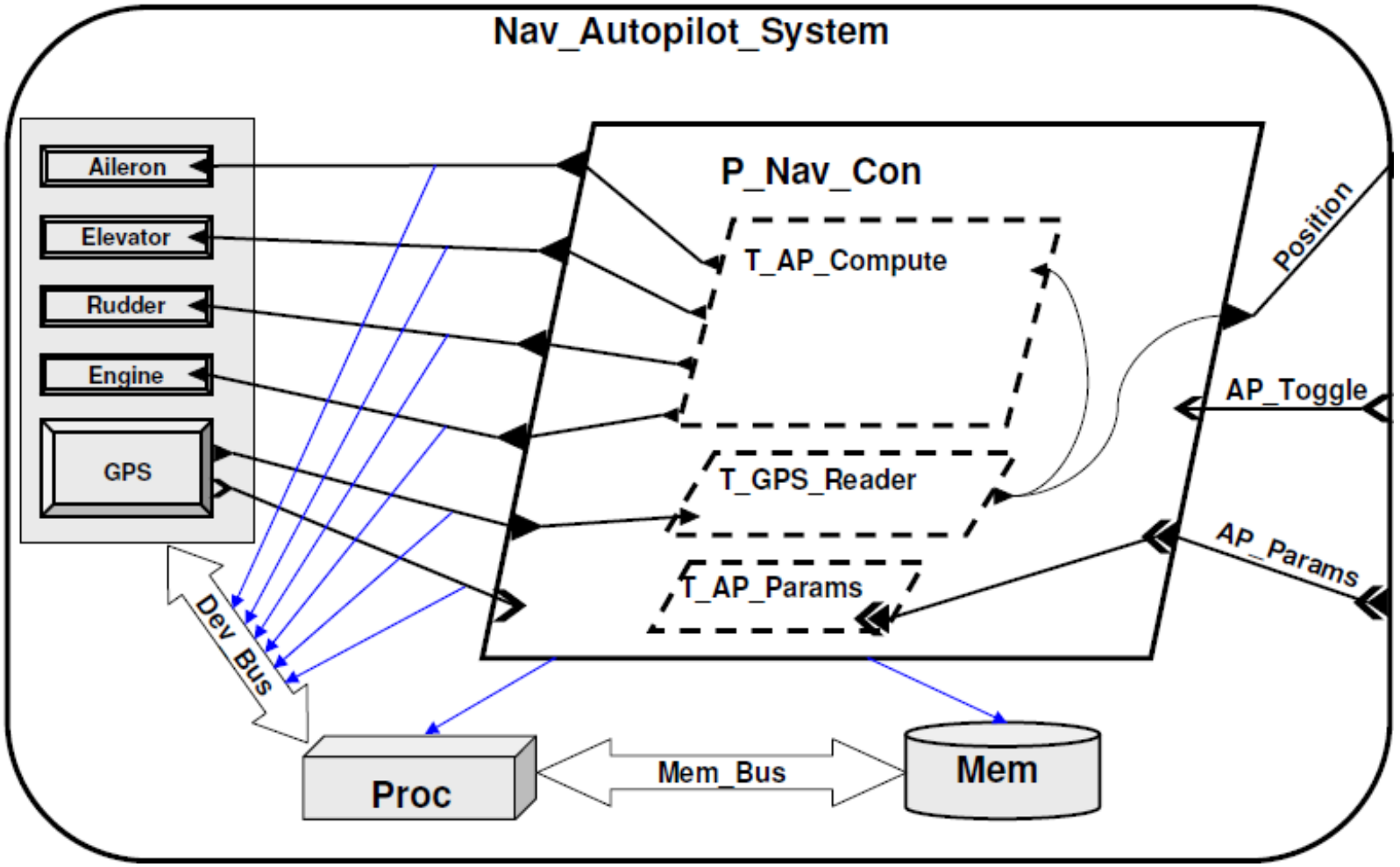
Jellegzetes nyelvek architektúra tervezéshez: AADL

- 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



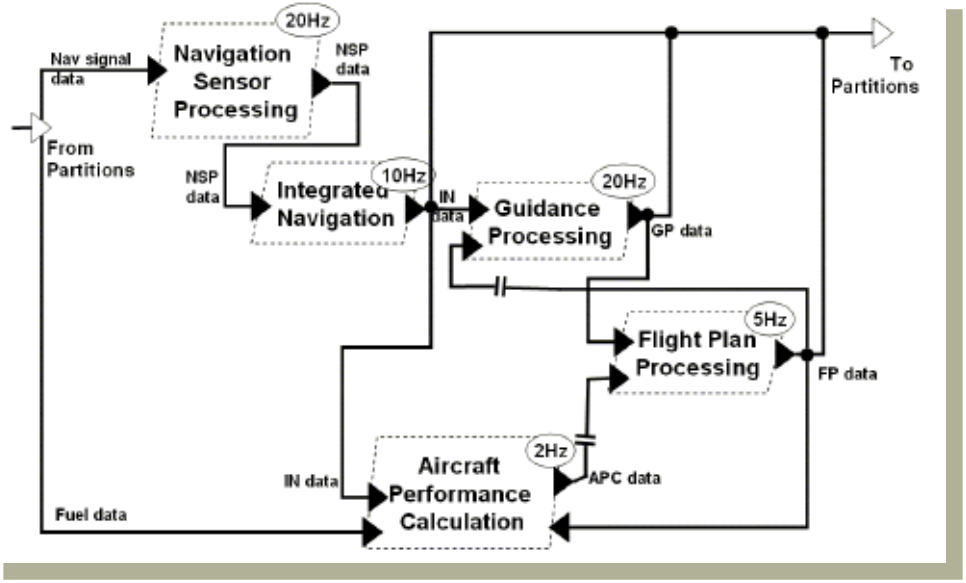
Jellegzetes nyelvek architektúra tervezéshez: AADL

- 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;
```

Mit és hogyan határoz meg az architektúra? 1/2

- Szolgáltatásbiztonság
  - Hibadetektálás: Push/pull monitorozás, kivételkezelés
  - Helyreállítás: Előrelépés, visszalépés, kompenzáció
  - Hibahatás kezelése: Átkonfigurálás, graceful degradation
- Teljesítmény
  - Erőforrás hozzárendelés: Kritikus számítások kiszolgálása, várakozó kérések korlátozása
  - Erőforrás menedzsment: Erőforrás ütemezés, dinamikus hozzárendelés, párhuzamos feldolgozás
- Adatbiztonság
  - Támadás kivédése: Elérés korlátozása, autentikáció, bizalmas adatok védelme
  - Támadás felderítése: Változások elemzése
  - Helyreállítás: Integritás karbantartása

Mit és hogyan határoz meg az architektúra? 2/2

- Módosíthatóság
  - Lokalizálás: Szemantikus koherencia (egységbezárás)
  - 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
  - Tervezési idejű technikák: Felhasználói felület elhatárolása
  - Futásidejű technikák: Felhasználói modell, feladatmodell, rendszermodell karbantartása

Példa: Előírt módszerek a szoftver biztonsághoz (EN 50128)

- SIL 1-től R, SIL 3-tól tipikusan HR technikák
  - Defenzív programozás
  - Hibafelfedés és hibadiagnózis
  - Hibafelismerő kódok
  - Meghibásodásbizonyító programozás
  - Diverz 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

Sokféle kombináció megengedett

Failure assertion

Referencia

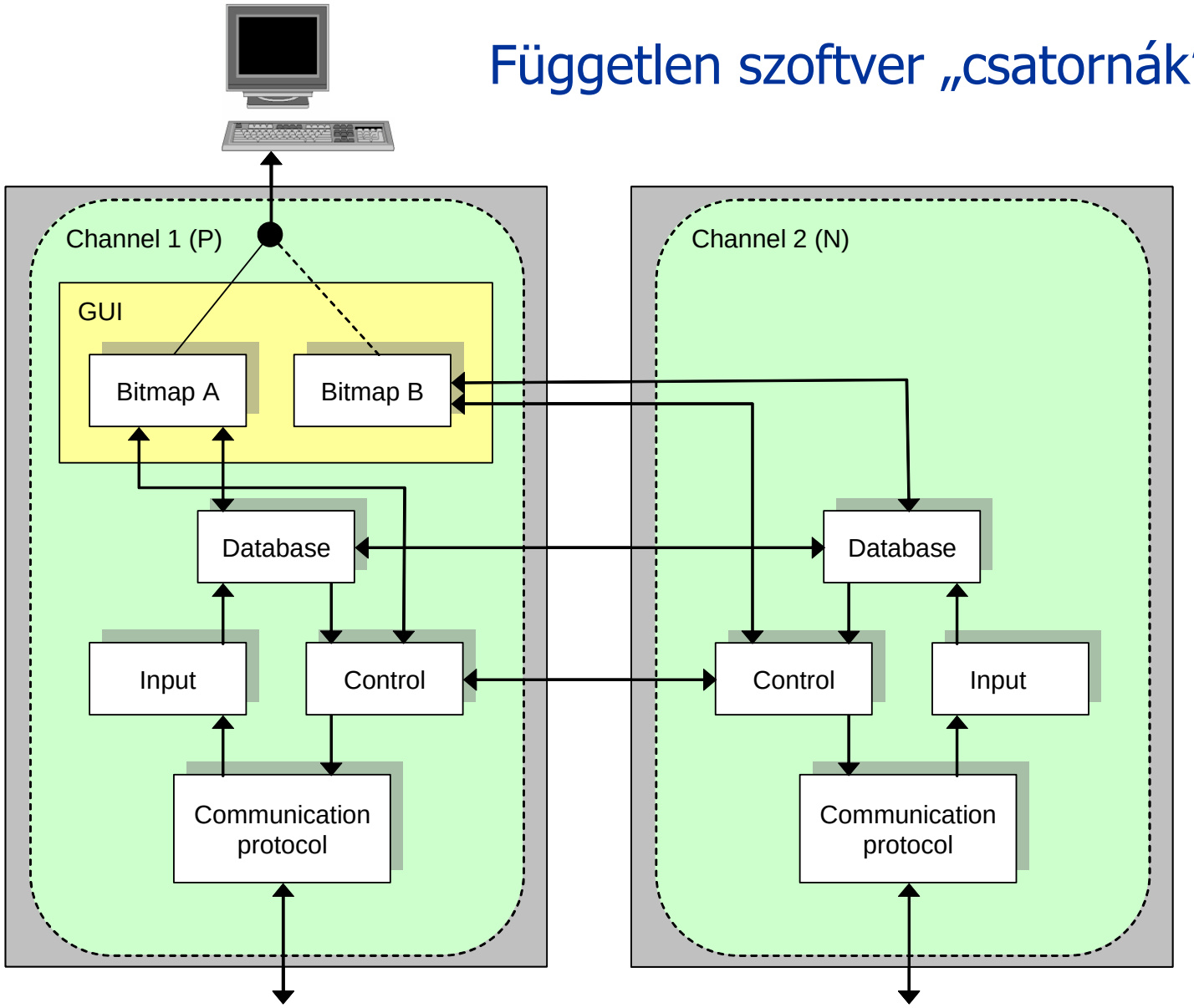
Bennmaradó hibák elleni védekezés

Példa: OTS komponensek használata kritikus rendszerekben

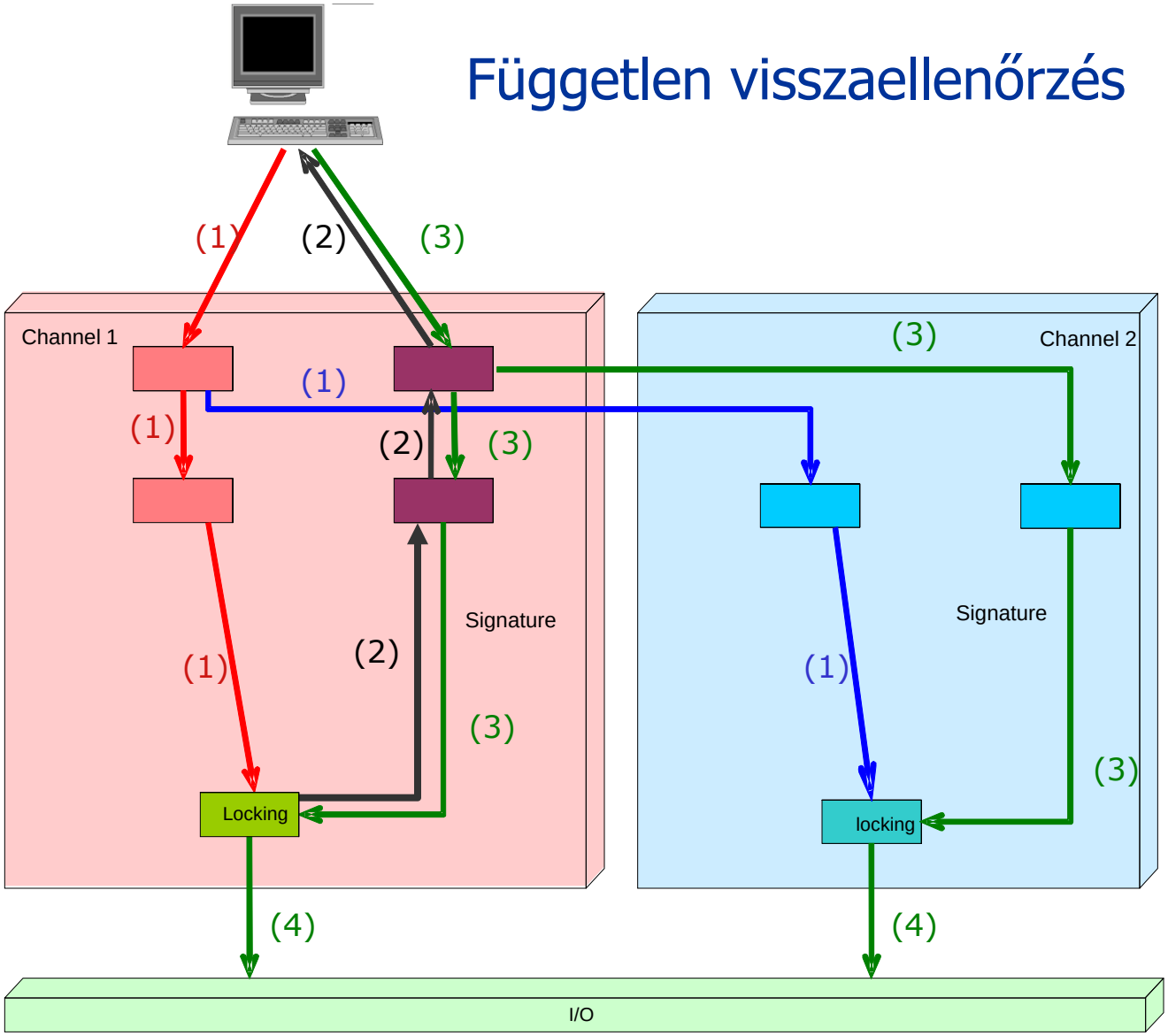
- OTS/COTS komponensek használata adott biztonságintegritási szinten (SIL)
  - SIL 0: Általánosan elfogadott
  - SIL 1 és 2: Be kell vonni a validációba
  - SIL 3 és 4: Szükséges intézkedések:
    - Bevonás a validációba
    - Lehetséges meghibásodások elemzése
    - Védelmi stratégia kialakítása és ennek tesztelése
    - Hibanaplók felvétele és kiértékelése
- Komponensek biztonságintegritási szintje
  - Alapértelmezés: Egyező a rendszerével
  - Csökkentés: Ha megakadályozható a komponens hibájából adódó rendszerszintű hiba
    - Egyszeres hibapont (SPOF) elkerülése

Példa: Architekturális megoldás SIL csökkentésre

Független szoftver „csatornák”



Példa: Architekturális megoldás SIL csökkentésre



Összefoglalás: Elérendő tulajdonságok és tervezési tér

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

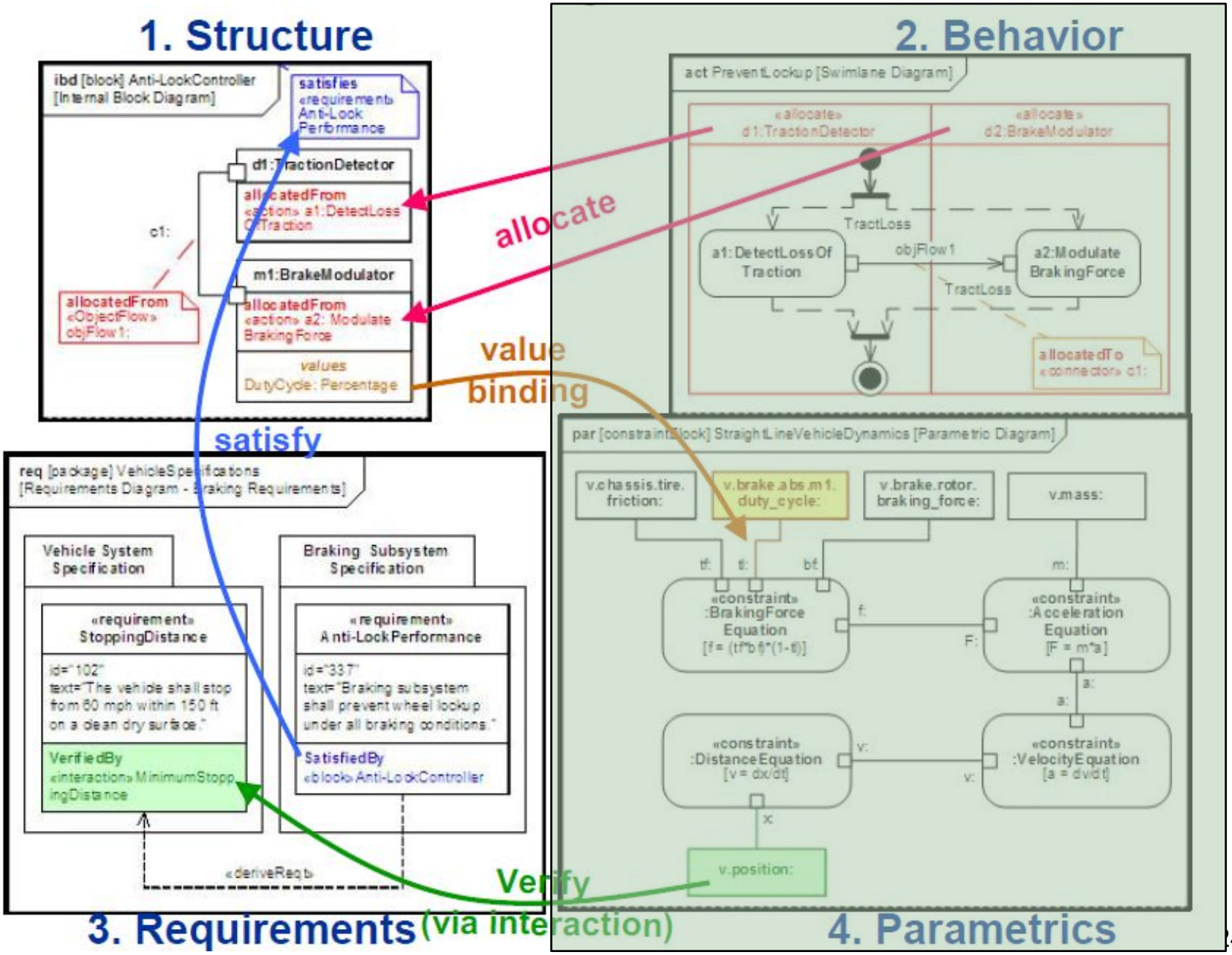
Áttekintés: Milyen vizsgálati technikák vannak?

- Követhetőség a követelmények felől
  - Követelmények „lefedése” architektúra elemekkel
- Szisztematikus elemzések: az architektúra bejárása
  - Interfész analízis
    - Elvárt és nyújtott interfészek megfelelősége
  - Hibahatás analízis kombinatorikus módszerekkel
    - Komponens szintű hibák ↔ Rendszerszintű hatások
- Modell alapú elemzések: analízis modell készítés
  - Sztochasztikus analízis: megbízhatósági illetve teljesítmény jellemzők meghatározása
    - Lokális (komponens illetve kapcsolati szintű) paraméterek alapján rendszerszintű jellemzők számítása
    - Az analízis modell konstruálása: Az architektúra alapján
- Architektúra trade-off analízis (ATAM)

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

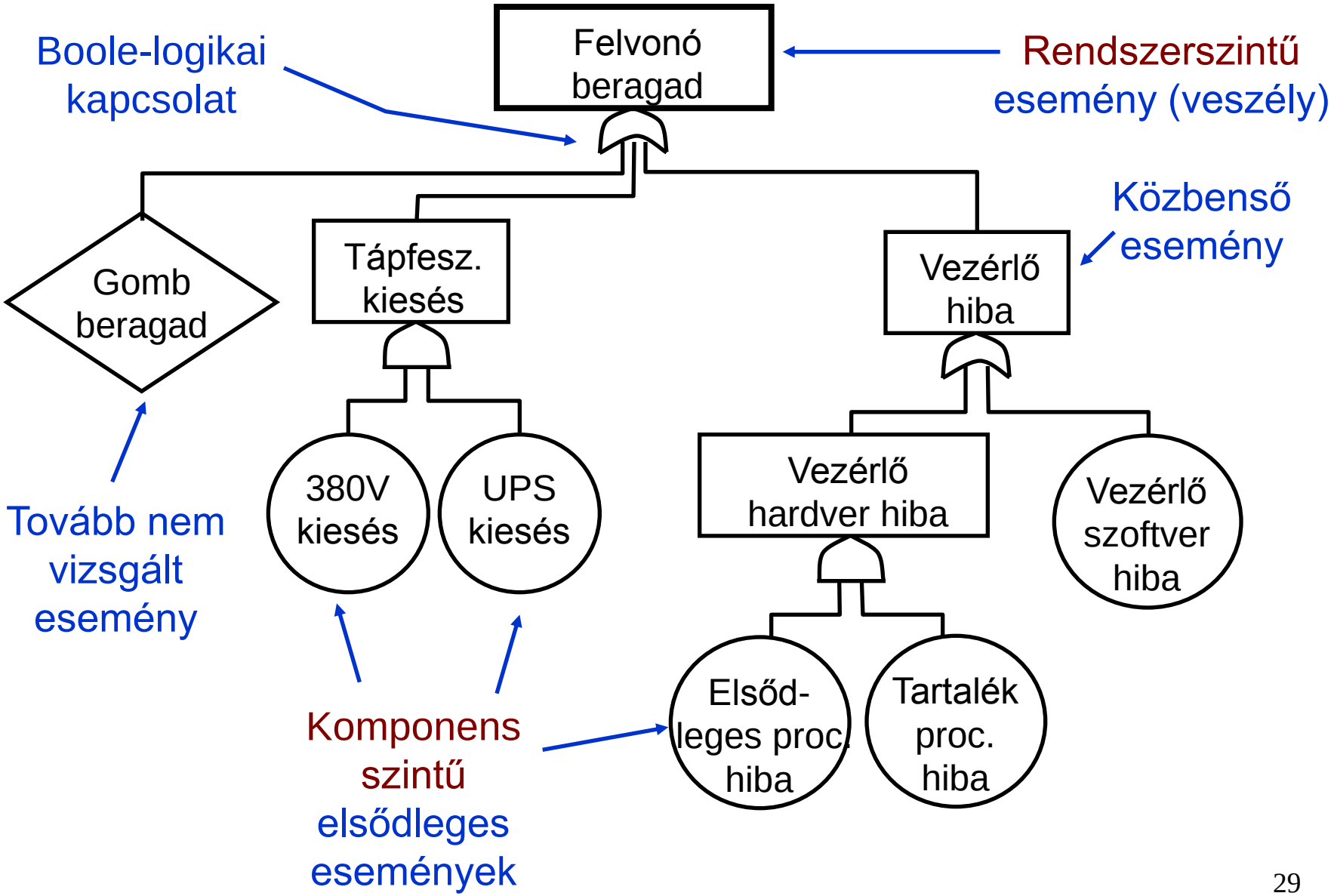
```
public class Purse {
    final int MAX_BALANCE;
    int balance;
      /*@ invariant pin != null && pin.length == 4  @*/
    byte[] pin;
      /*@ requires amount >= 0;
          @ assignable balance;
          @ ensures balance == \old(balance) – amount
                  && \result == balance;
          @ 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 releváns jellemzői szerepelnek
- Analízis megközelítése
  - Az architektúra szempontjából
    - Alulról felfelé: A komponensek (alrendszerek) felől
    - Felülről lefelé: Rendszerszintről lebontva
  - Ok-okozati szempontból:
    - Előrelépő (induktív): Ok hatásainak vizsgálata
    - Visszalépő (deduktív): Okozat okainak felderítése
- Jellegzetes példa: Hibahatás analízis
  - Hibafa
  - Eseményfa
  - Ok-következmény analízis
  - Hibamód és -hatás analízis (FMEA)

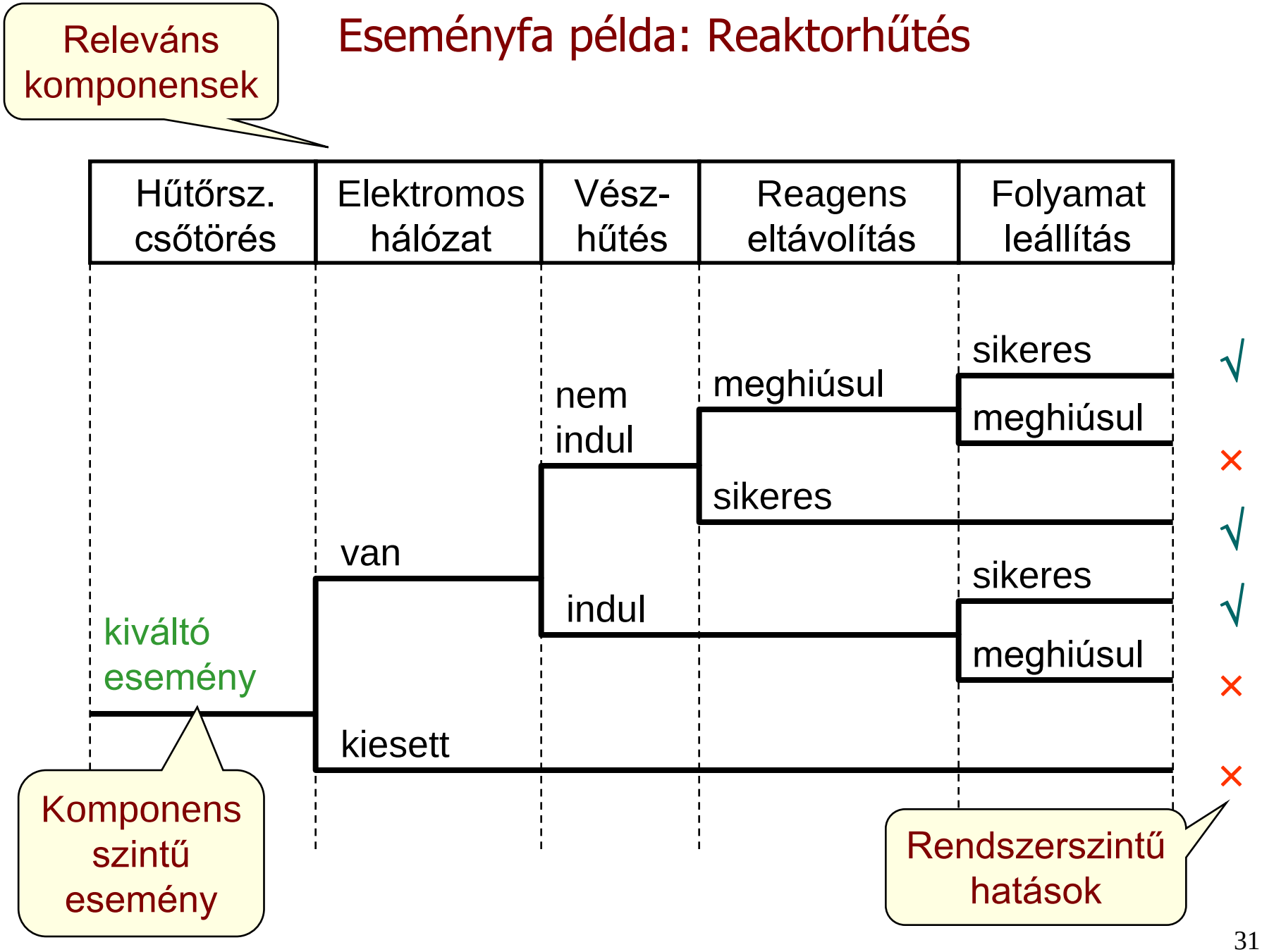
Hibafa példa: Felvonó



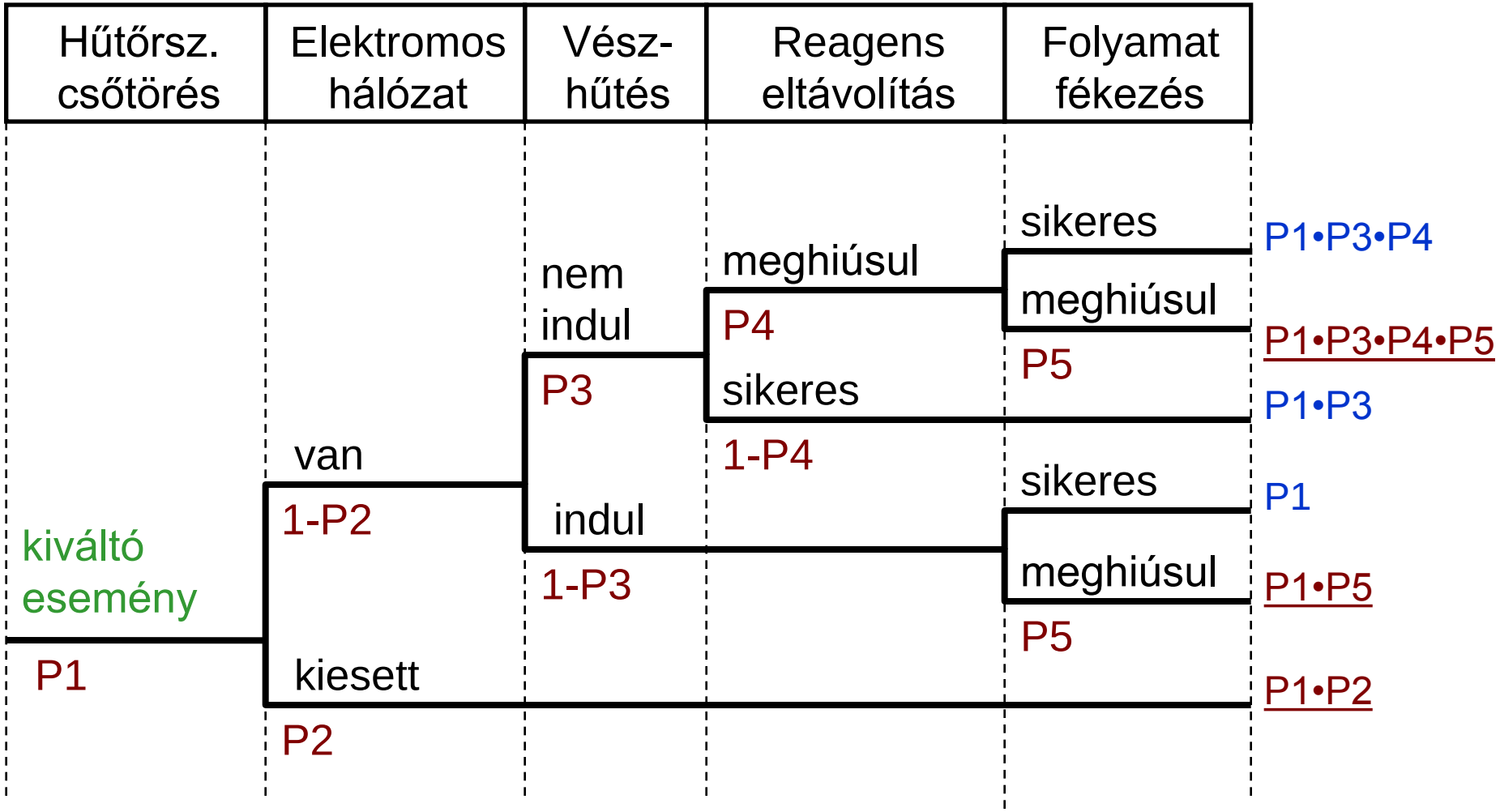
Hibafa analízis

- Minőségi (kvalitatív) analízis:
  - Hibafa redukció: Közbenső események feloldása
    $\rightarrow$ diszjunktív normál forma (OR a legtetején)
  - Azonosítható
    - Egyszeres hibapont (SPOF)
    - Kritikus esemény (több ágban is szerepel)
- Mennyiségi (kvantitatív) analízis:
  - Alapszintű eseményekhez rendelt valószínűségek
    - Komponens-adat, tapasztalat, becslés
  - Rendszerszintű veszély valószínűség számítása
    - AND kapu: szorzat (független eseményekre)
    - OR kapu: összegzés (felső becslés)
  - Problémák:
    - Korreláló hibák, időbeli (hiba)szekvenciák kezelése

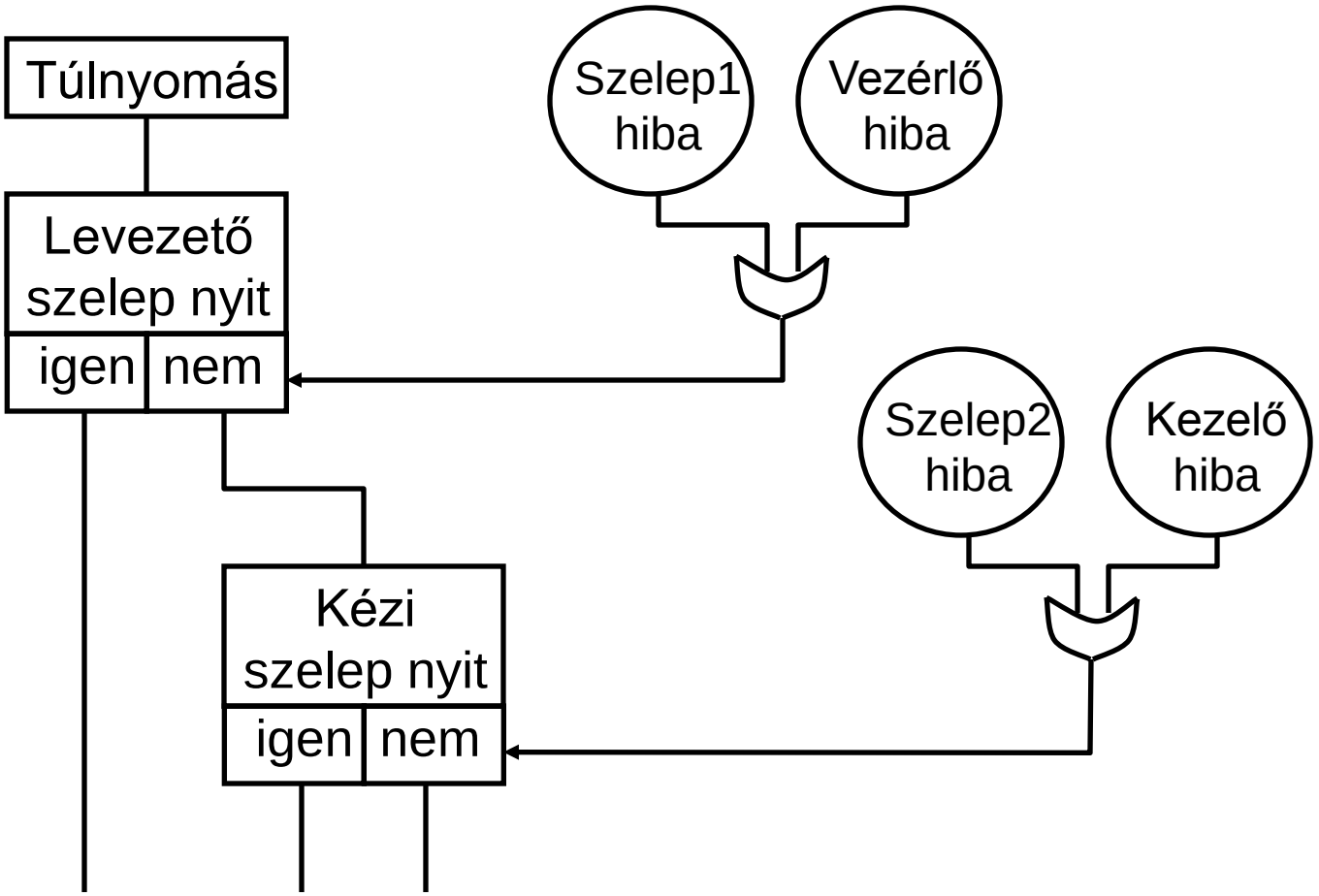
Eseményfa példa: Reaktorhűtés



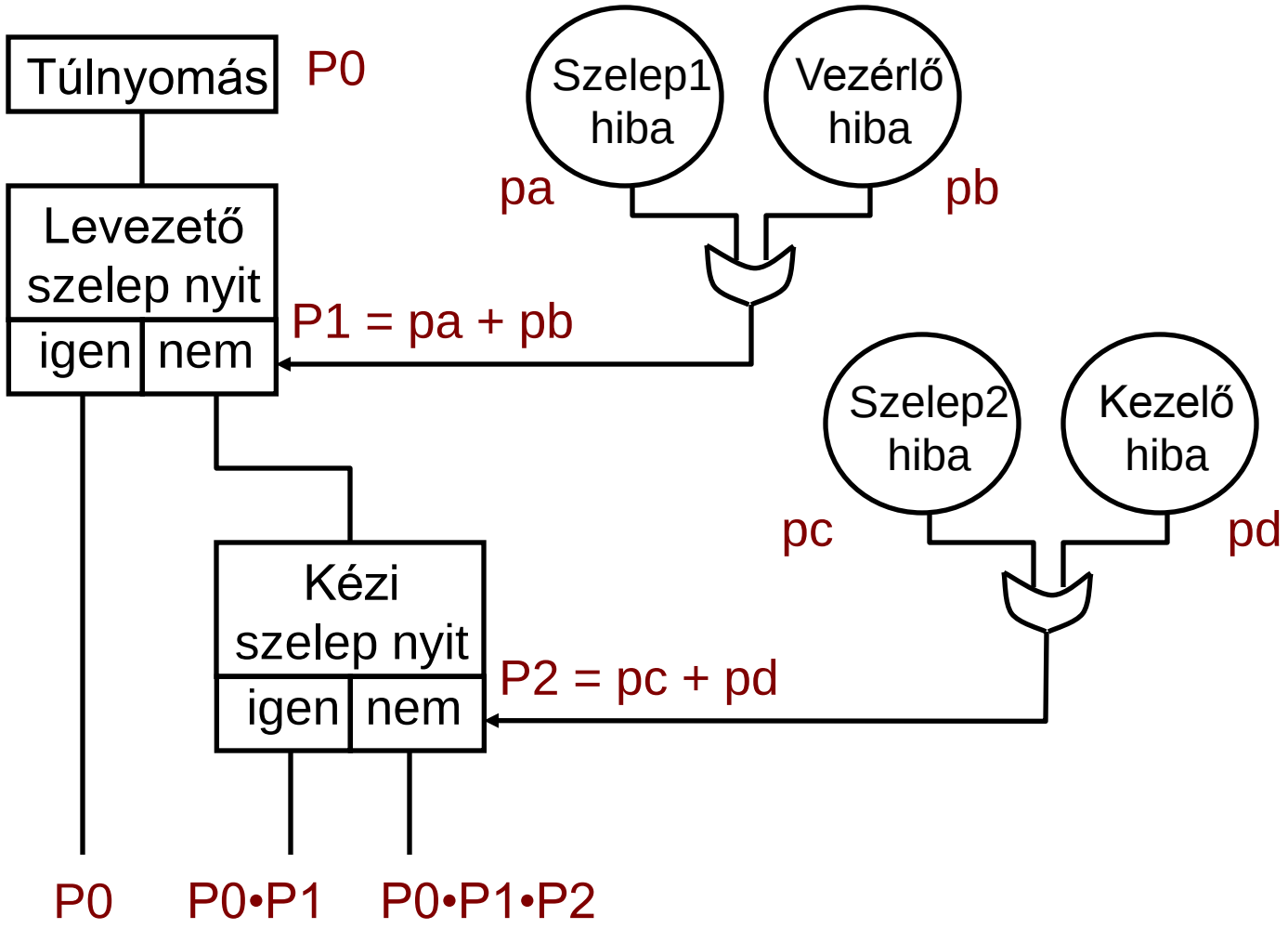
Eseményfa példa: Forgatókönyv valószínűségek



Ok-következmény analízis példa: Túlnyomás védelem



Ok-következmény analízis példa: Valószínűségek



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

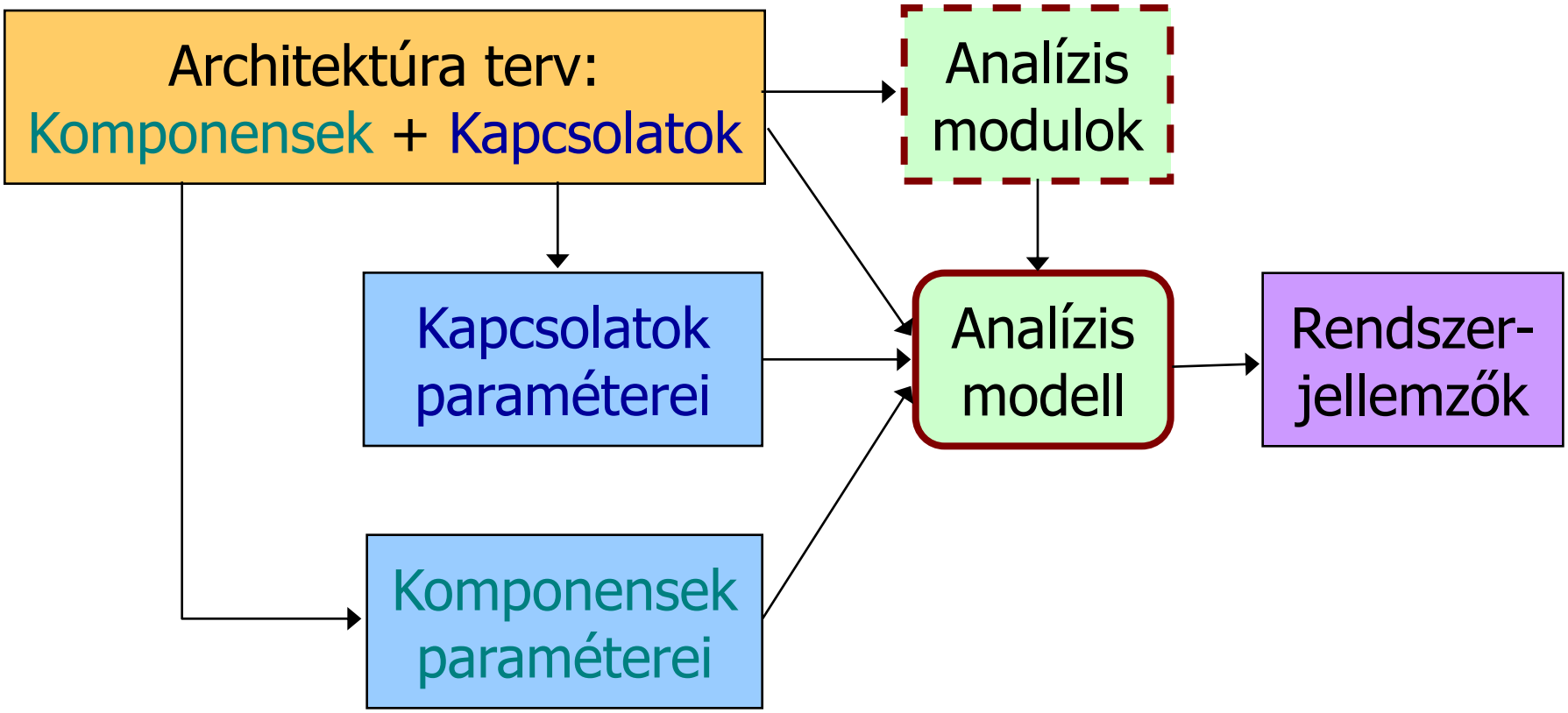
- Motiváció
  - Mit határoz meg az architektúra?
  - Milyen vizsgálati módszerek vannak?
- Követhetőség
- Szisztematikus vizsgálati módszerek
  - Interfész analízis
  - Hibahatás analízis
- Modell alapú vizsgálatok
  - Teljesítmény modellezés
  - Megbízhatósági modellezés
- ATAM architektúra elemzés

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



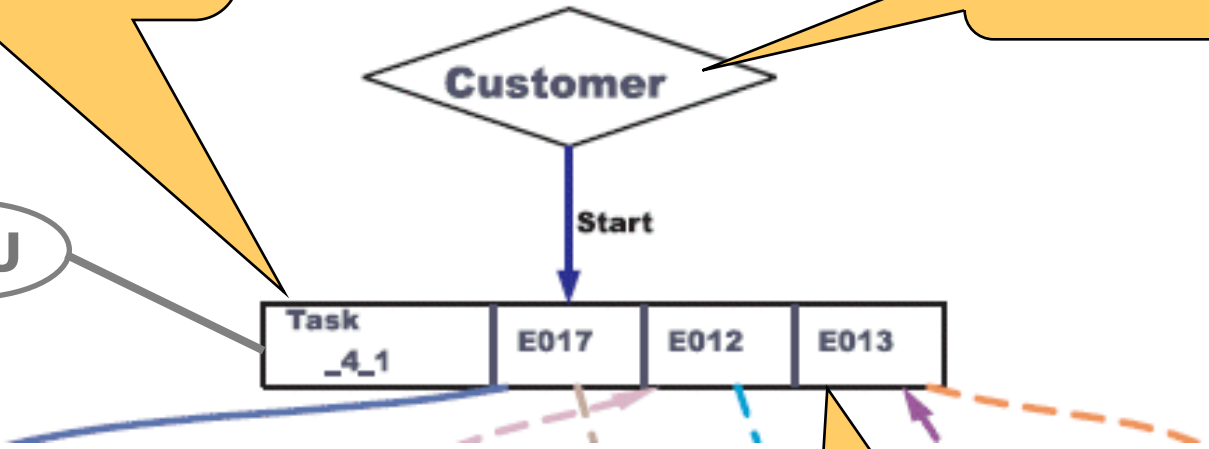
Rendszerszintű analízis modellek

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

Példa: Teljesítmény modellezés (LQN)

Taszk (szerver):
- Funkciók (szolgáltatás hívási pontok)
- Prioritások

Kliens (kérés):
- Hívási gyakoriság



Processzor:
- Telepítés
- Ütemezés

Funkció (szolgáltatás):
- Lokális végrehajtási idő
- Továbbhívás gyakoriság

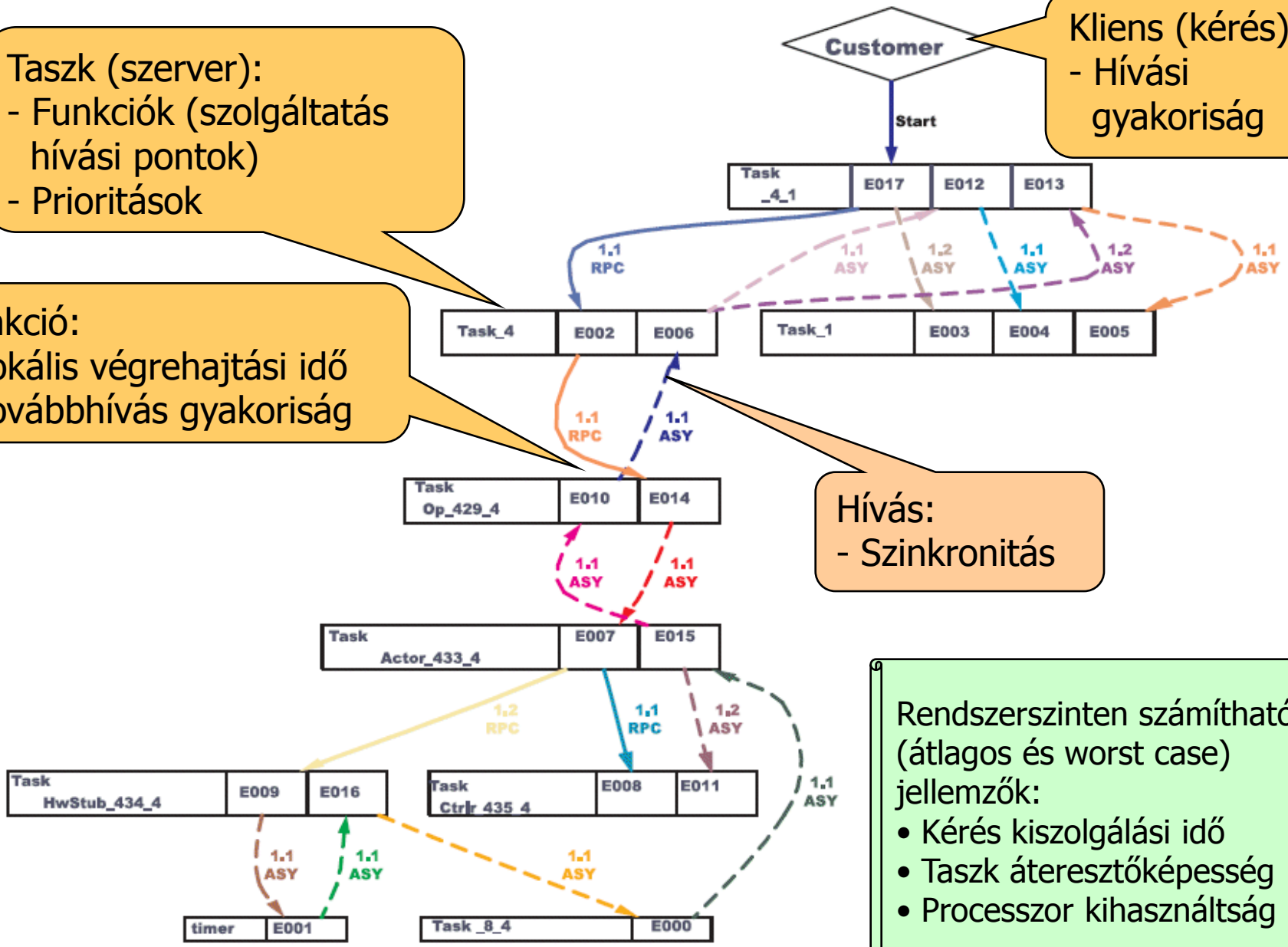
Példa: Teljesítmény modellezés (LQN): Rétegek

Taszk (szerver):
- Funkciók (szolgáltatás hívási pontok)
- Prioritások

Kliens (kérés):
- Hívási gyakoriság

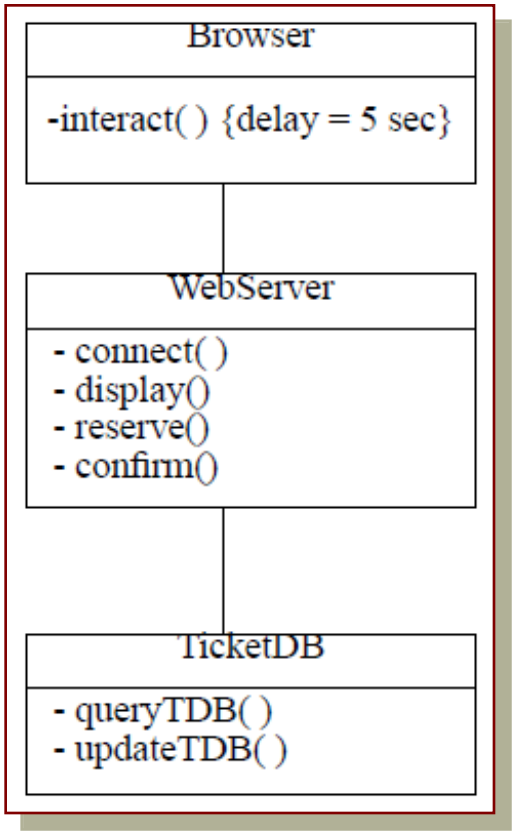
Funkció:
- Lokális végrehajtási idő
- Továbbhívás gyakoriság

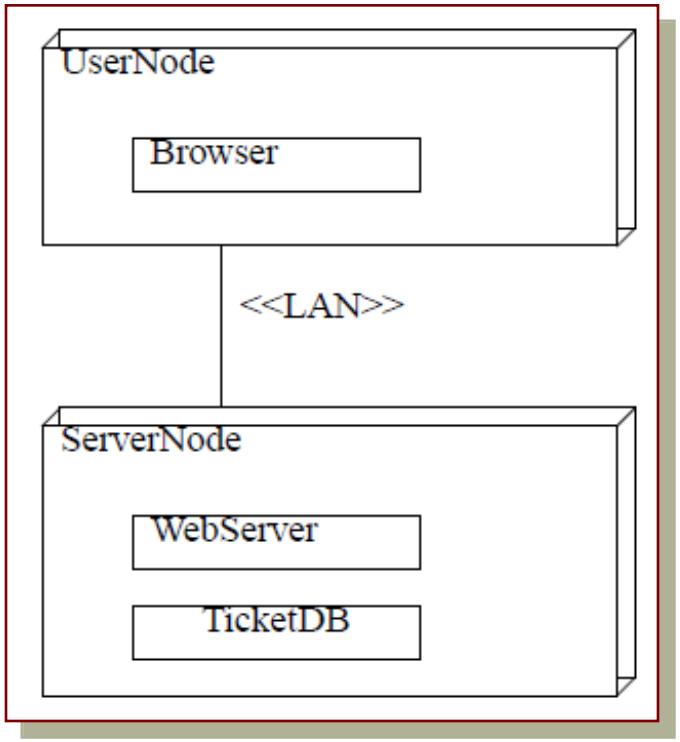
Hívás:
- Szinkronitás

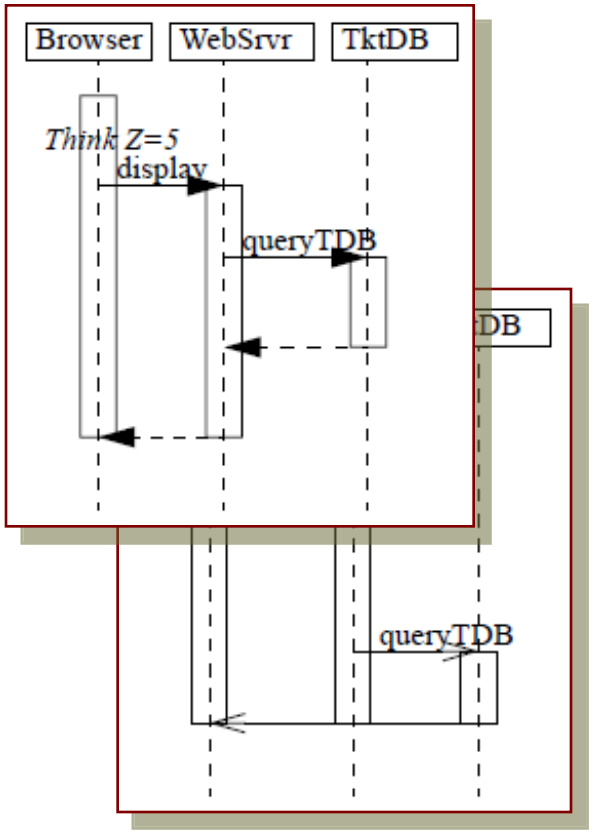


Rendszerszinten számítható (átlagos és worst case) jellemzők:
- Kérés kiszolgálási idő
- Taszk áteresztőképesség
- Processzor kihasználtság

Példa: Az architektúra leképezése a teljesítmény modellre





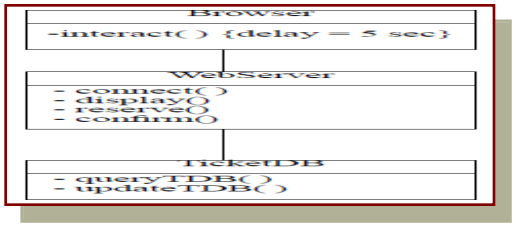


Osztályok és lokális jellemzők

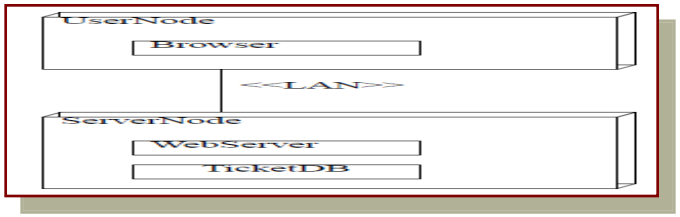
Processzorok és telepítés

Interakciók (hívások)

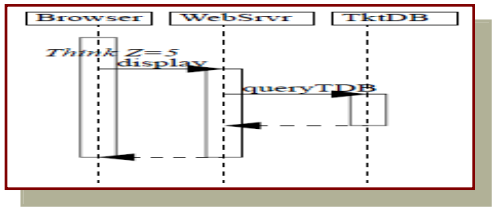
Példa: Az architektúra leképezése a teljesítmény modellre



Osztályok

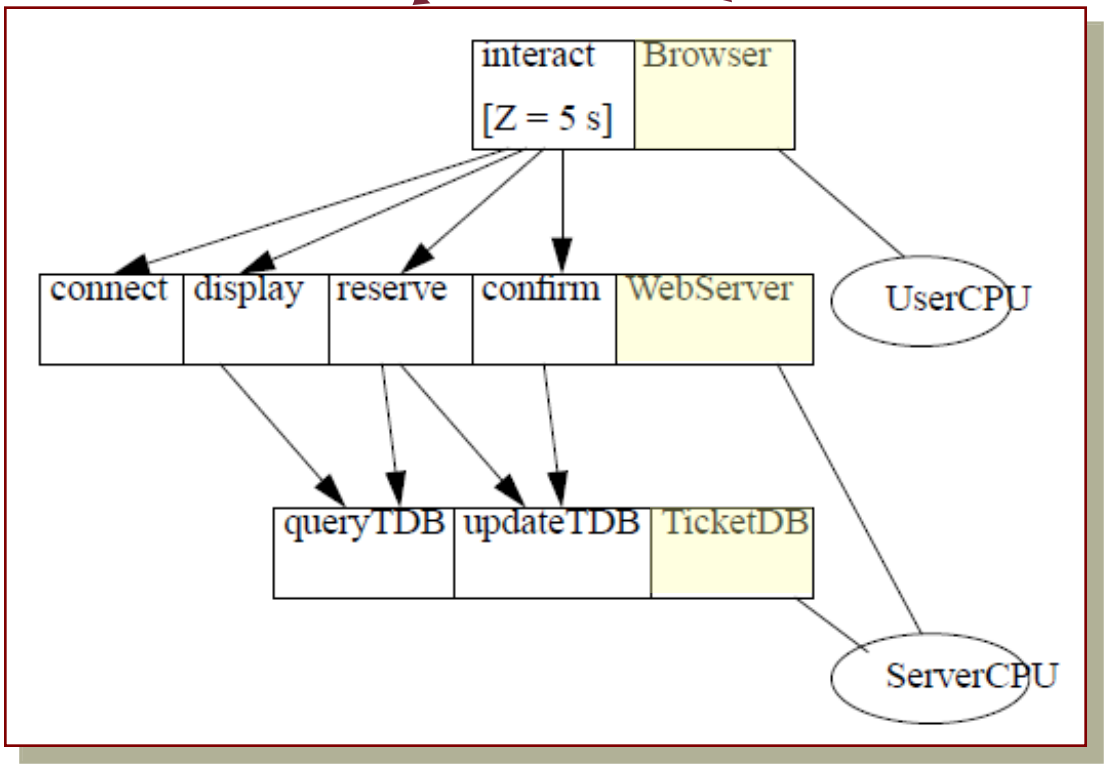


Telepítés



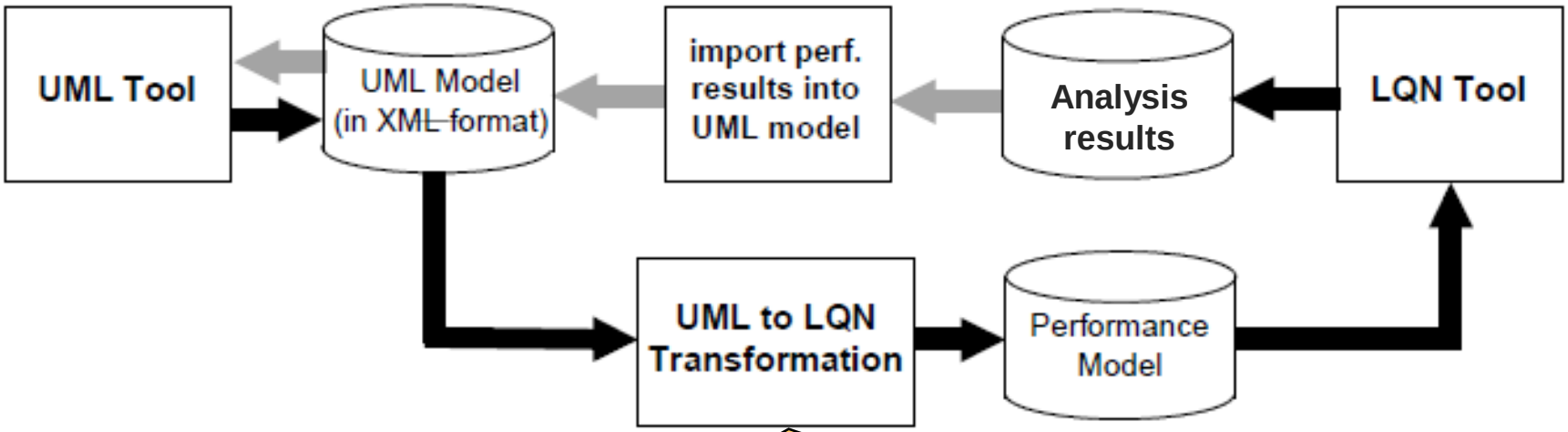
Interakciók

Modell-
transzformáció



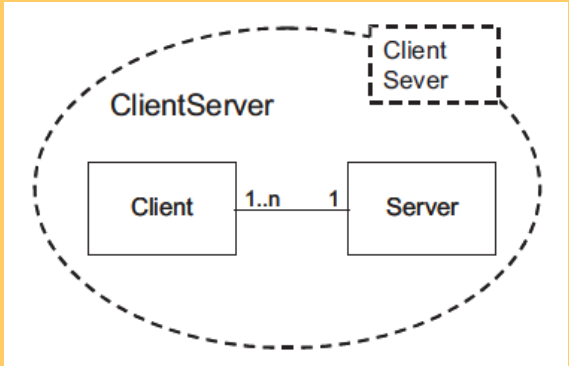
LQN
teljesítmény-
modell

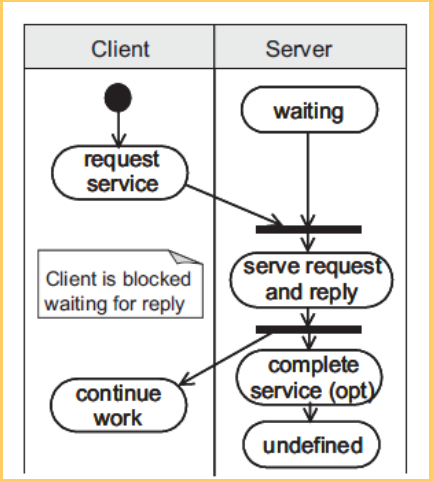
Példa: Az architektúra leképezése a teljesítmény modellre



- Architektúra minták használata

Szinkron üzenetküldés:





Tartalomjegyzék

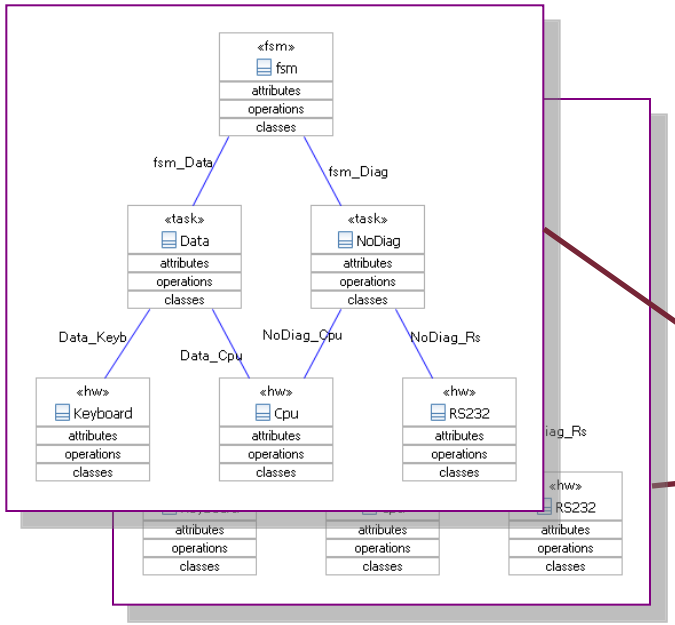
- Motiváció
  - Mit határoz meg az architektúra?
  - Milyen vizsgálati módszerek vannak?
- Követhetőség
- Szisztematikus vizsgálati módszerek
  - Interfész analízis
  - Hibahatás analízis
- Modell alapú vizsgálatok
  - Teljesítmény modellezés
  - Megbízhatósági modellezés
- ATAM architektúra elemzés

Rendszerszintű analízis modellek

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

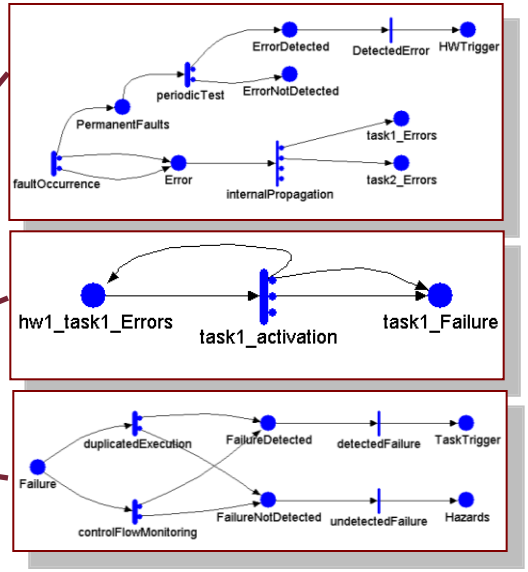
Példa: UML alapú megbízhatósági modellezés

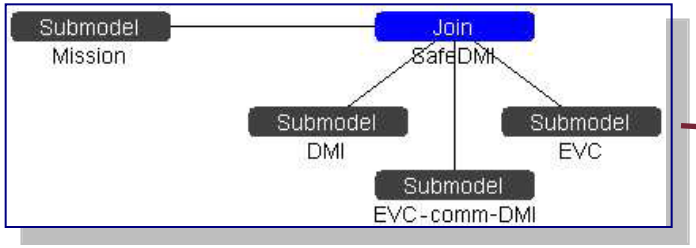
UML architektúra modell



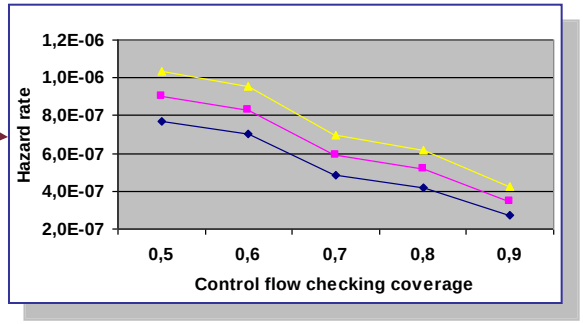
Megbízhatósági modell konstrukció

Analízis modellkönyvtár





Rendszerszintű megbízhatósági modell

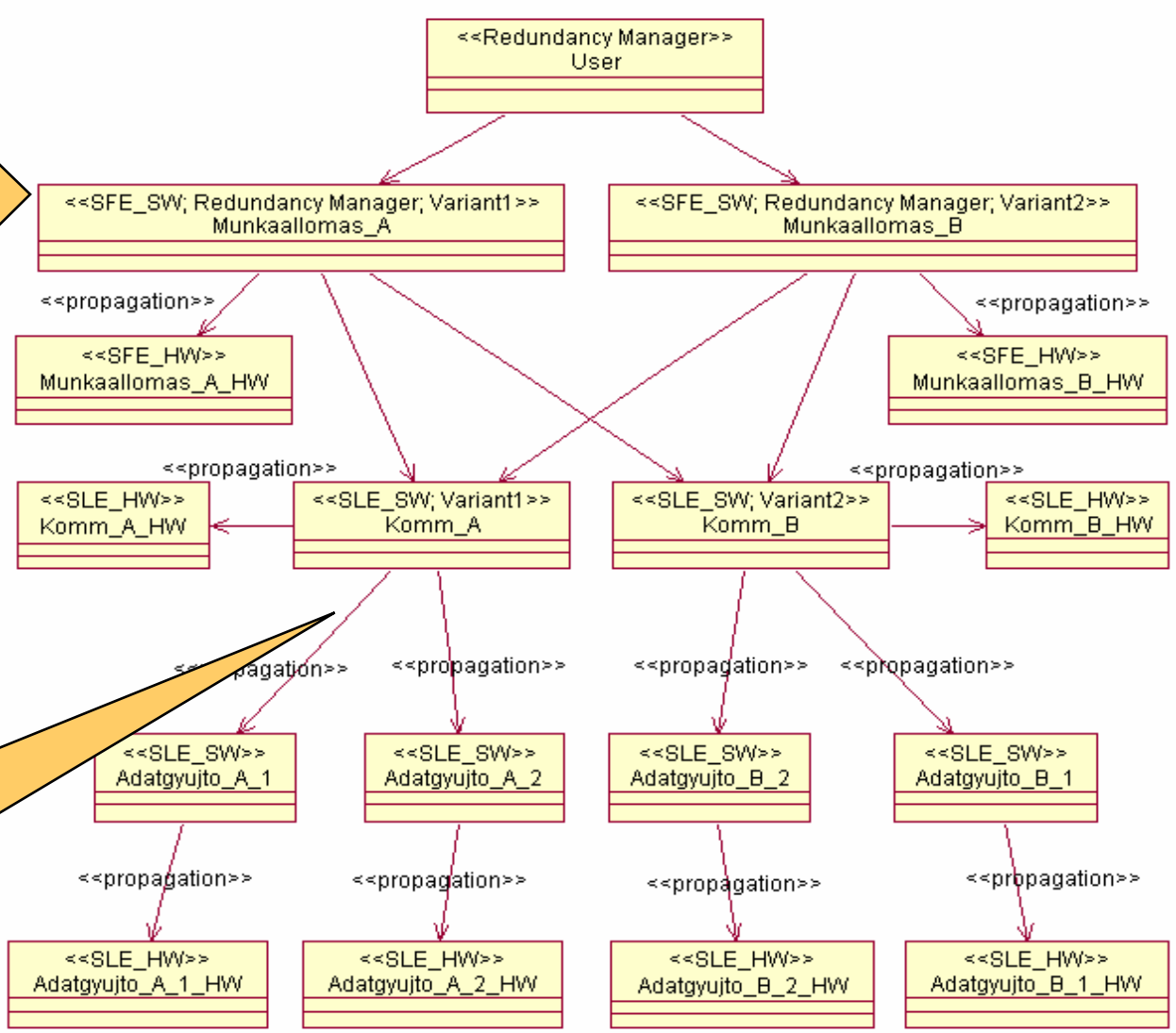


Analízis eredmények

Példa: Megbízhatósági modellezés – Architektúra modell

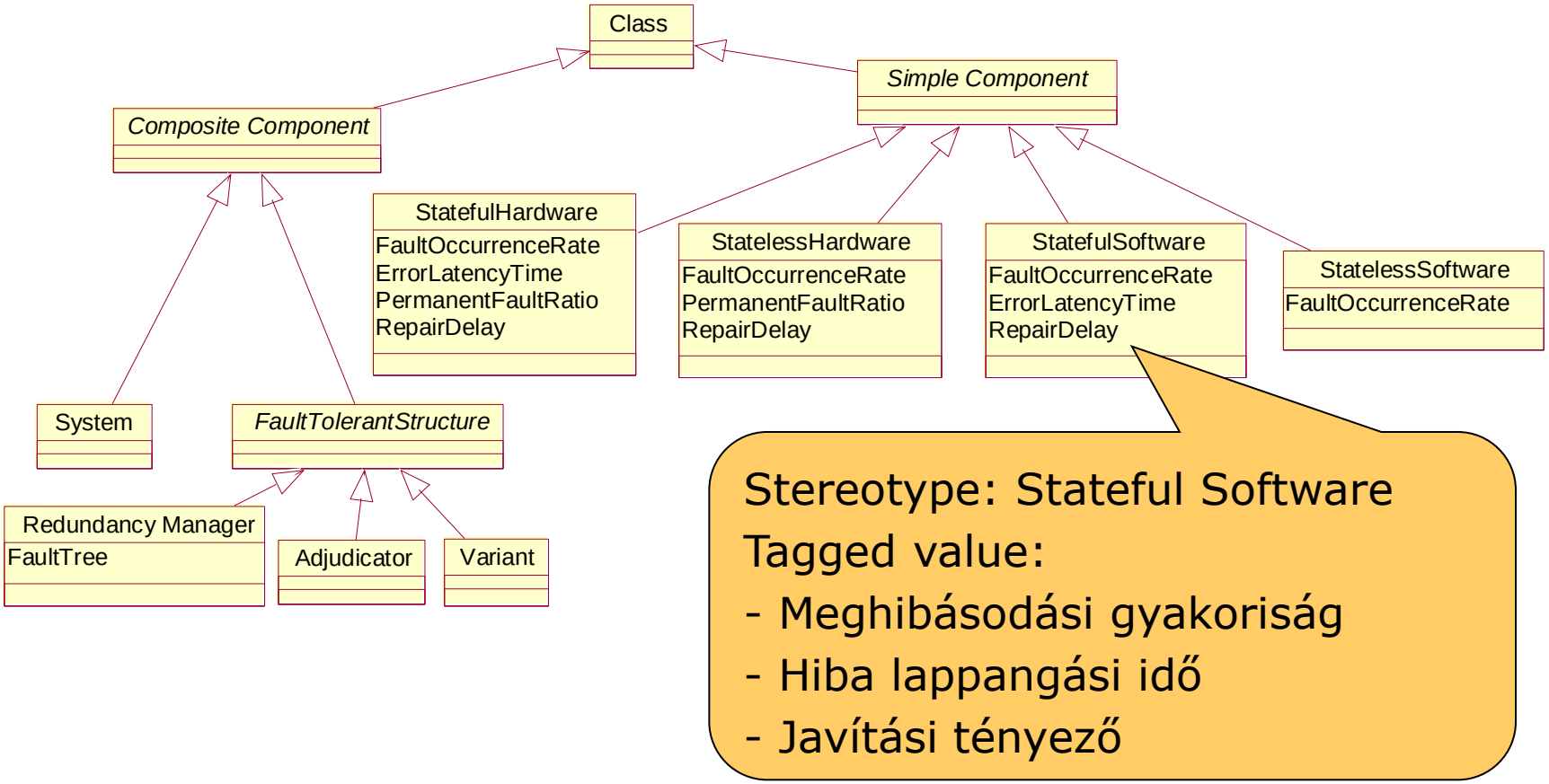
Komponens:
- Típus (HW, SW)
- Szerep (variáns, menedzser)
- Meghibásodási jellemzők:
  * meghibásodási gyakoriság,
  * lappangási idő,
  * javítási idő

Kapcsolat:
- Hibaterjesztési jellemzők:
  * hibaterjesztési valószínűség

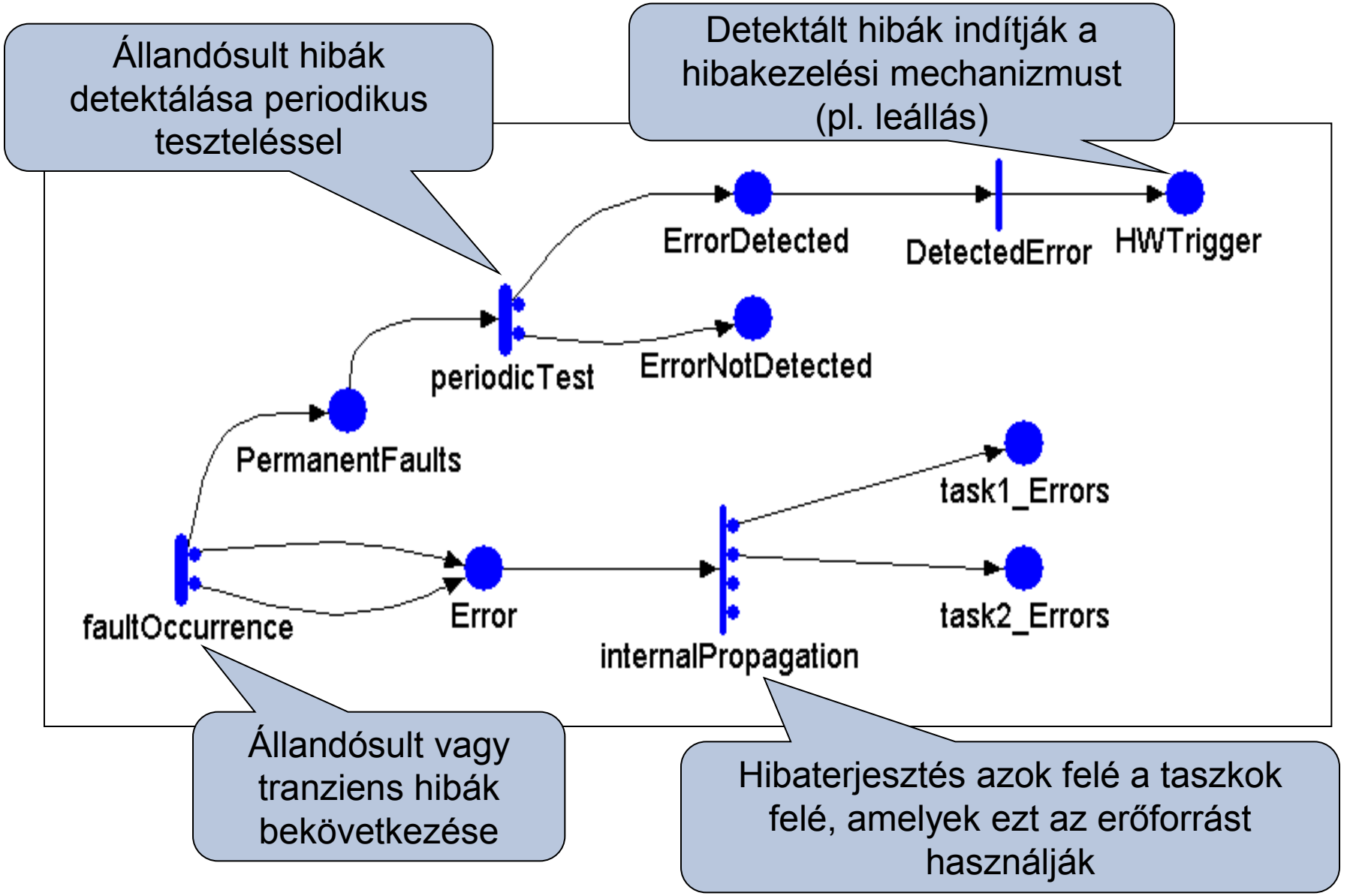


Példa: Komponensek lokális tulajdonságai

- UML profil az extra-funkcionális tulajdonságokhoz

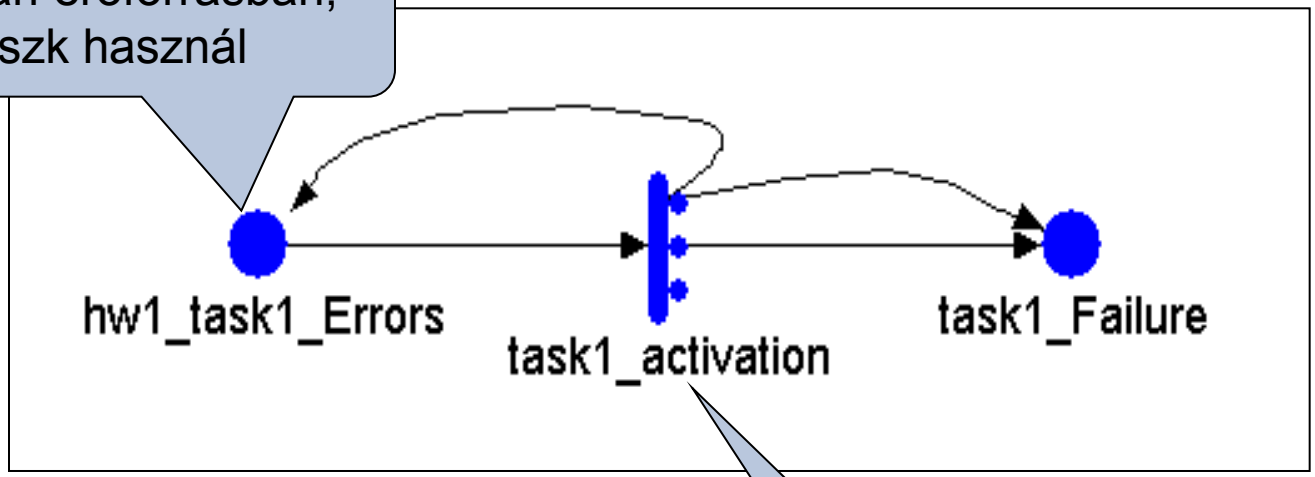


Példa: Komponens analízis alháló: Egy HW erőforrás (memória)



Példa: Kapcsolat analízis alháló: Hibaterjesztés

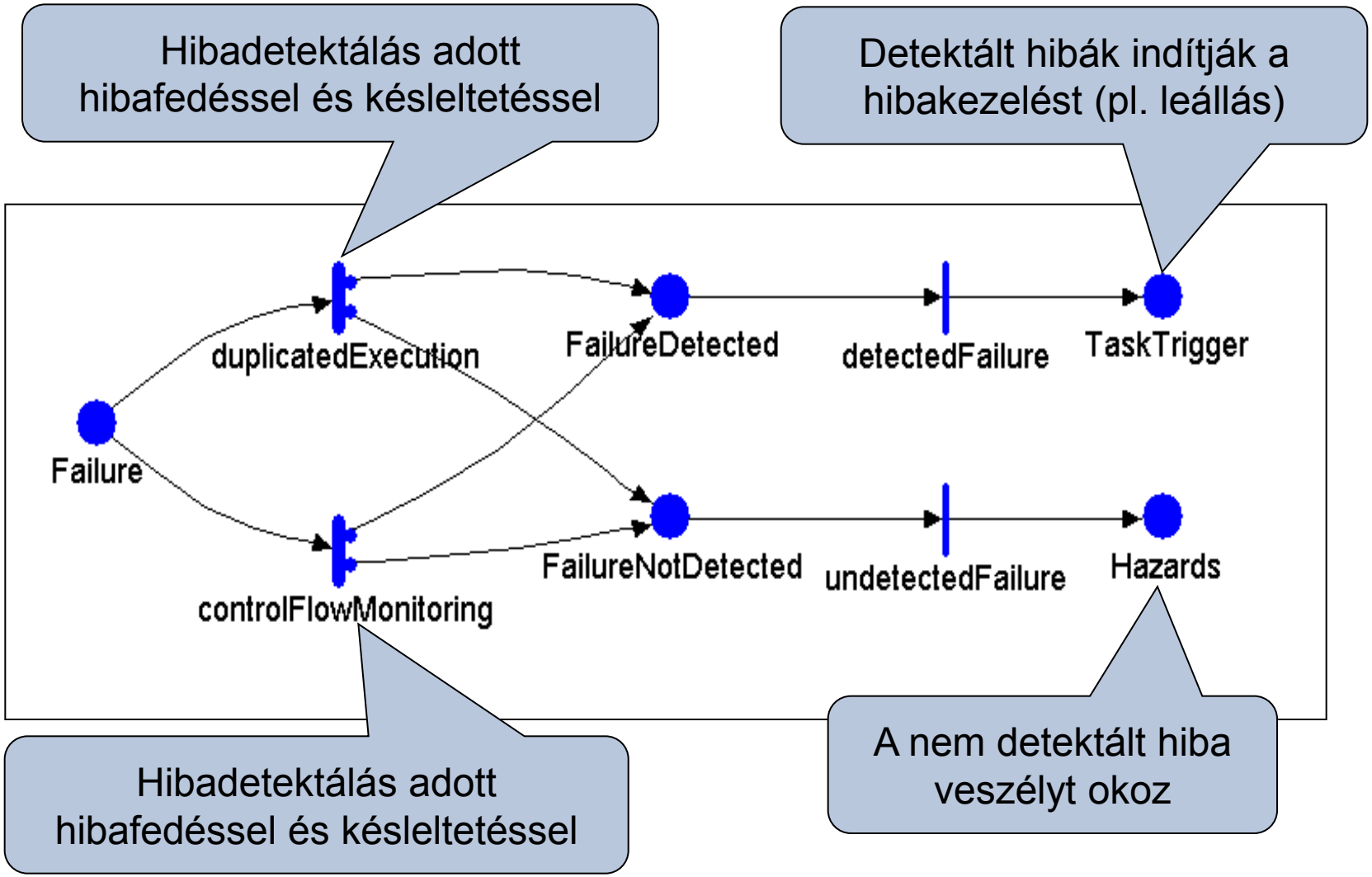
Hiba olyan erőforrásban, amit a taszk használ



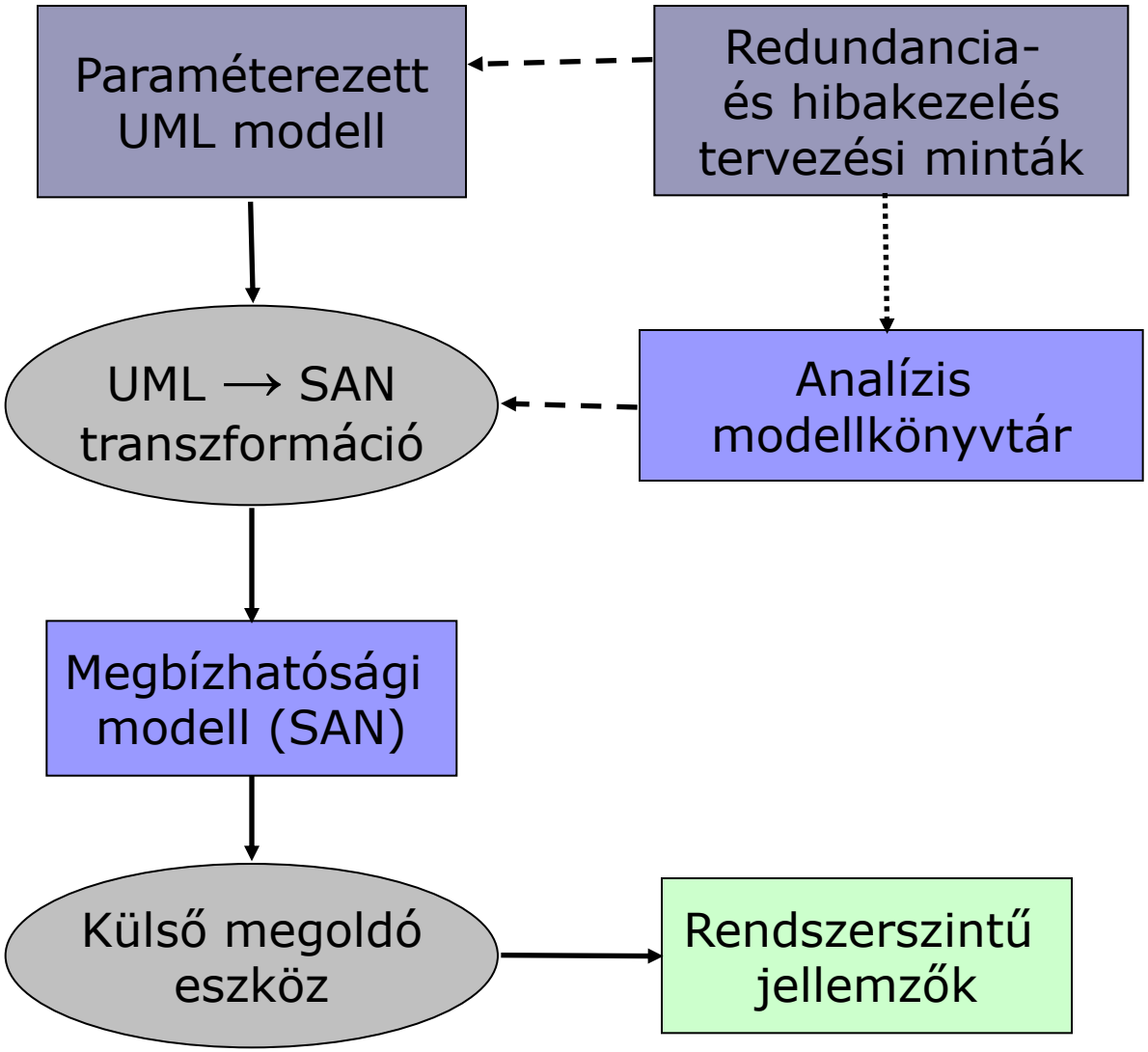
Taszk aktiválási gyakoriság, hiba aktiválási lehetőségek:

- aktivált hiba, ami a rendszerben marad
- aktivált hiba, felülíródik, de hatása van
- hatás nélkül felülírt hiba

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

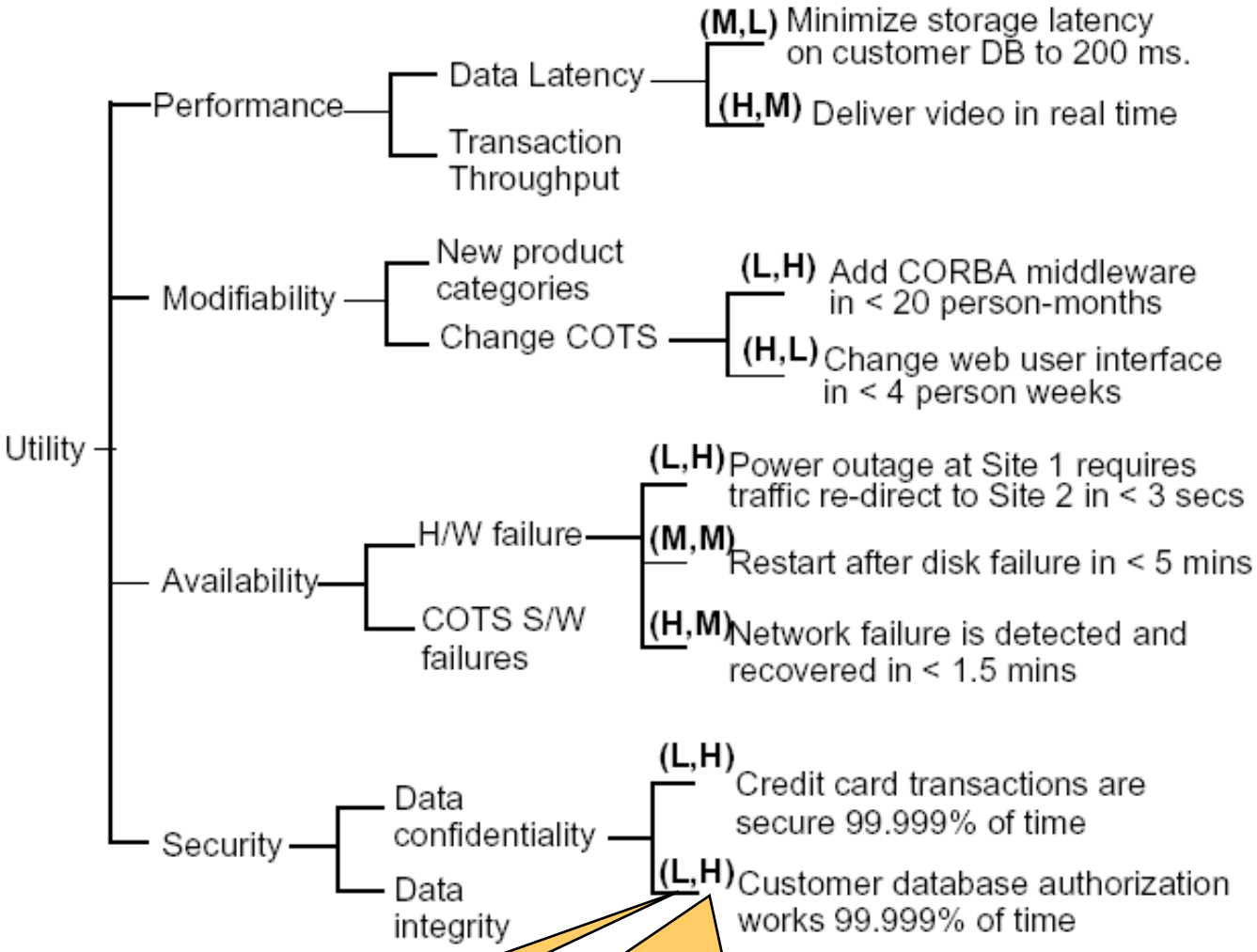
Architektúrák elemzése: ATAM

- Architecture Tradeoff Analysis Method (ATAM)
  - Milyen elérendő minőségi célok (jellemzők) vannak?
    - Hogyan viszonyulnak egymáshoz az elérendő célok?
  - Hogyan elégíti ki az architektúra az elérendő célokat?
    - Megfelelnek-e az architektúra szintű tervezési döntések a céloknak és ezek prioritásainak?
    - Milyen kockázatok maradtak?
- 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
  - Architektúra minták azonosítása (pl. rétegek, redundancia, …)
- Hasznossági fa (utility tree) elkészítése <- tervezők, elemzők
  - Gyökér: Általános megfelelőség
  - Második szint: Jellemzők (elérendő minőségi célok)
  - További szintek: Finomított jellemzők
  - Fa leveleihez forgatókönyvek rendelése: Jellemző szerepének bemutatása
    - Bemenetek, hatások, amik relevánsak a minőségi jellemzőhöz
    - Környezet (pl. tervezési vagy futásidőbeli)
    - Elvárt reakció (támogatás) az architektúra szempontjából
  - Prioritások felvétele a forgatókönyvek (azaz jellemzők) között
- Az architektúra analízise <- elemzők
  - Hogyan támogatja az architektúra a fontos forgatókönyveket?
  - Mik az érzékenységek, kompromisszumok, kockázatok?

Példa: Hasznossági fa



Fontosság
(Low, Medium, High)

Megvalósítási bonyolultság
(Low, Medium, High)

Példa: Az architektúra analízise

- A forgatókönyvek támogatása
  - Forgatókönyv: Diszk hiba esetén az adatok helyreállítása szükséges kevesebb, mint 5 perc alatt
  - Architektúra döntés: Replika adatbázis használata
- Érzékenységek
  - A replika adatbázis használata befolyásolja a rendelkezésre állást
  - A replika adatbázis használata befolyásolja a teljesítményt
    - Szinkron replika frissítés
    - Aszinkron replika frissítés
- Kompromisszumok optimalizálása (tradeoff)
  - A replika adatbázis szinkronitása hatással van mind a rendelkezésre állásra, mind pedig a teljesítményre
    - Kompromisszum (döntés): Aszinkron replika frissítés
- Kockázat
  - A replika frissítés kockázatot jelent, ha a teljesítményveszteségnek nagy a költsége
    - Adott várható terhelés és veszteség költség esetén megfelelő a döntés

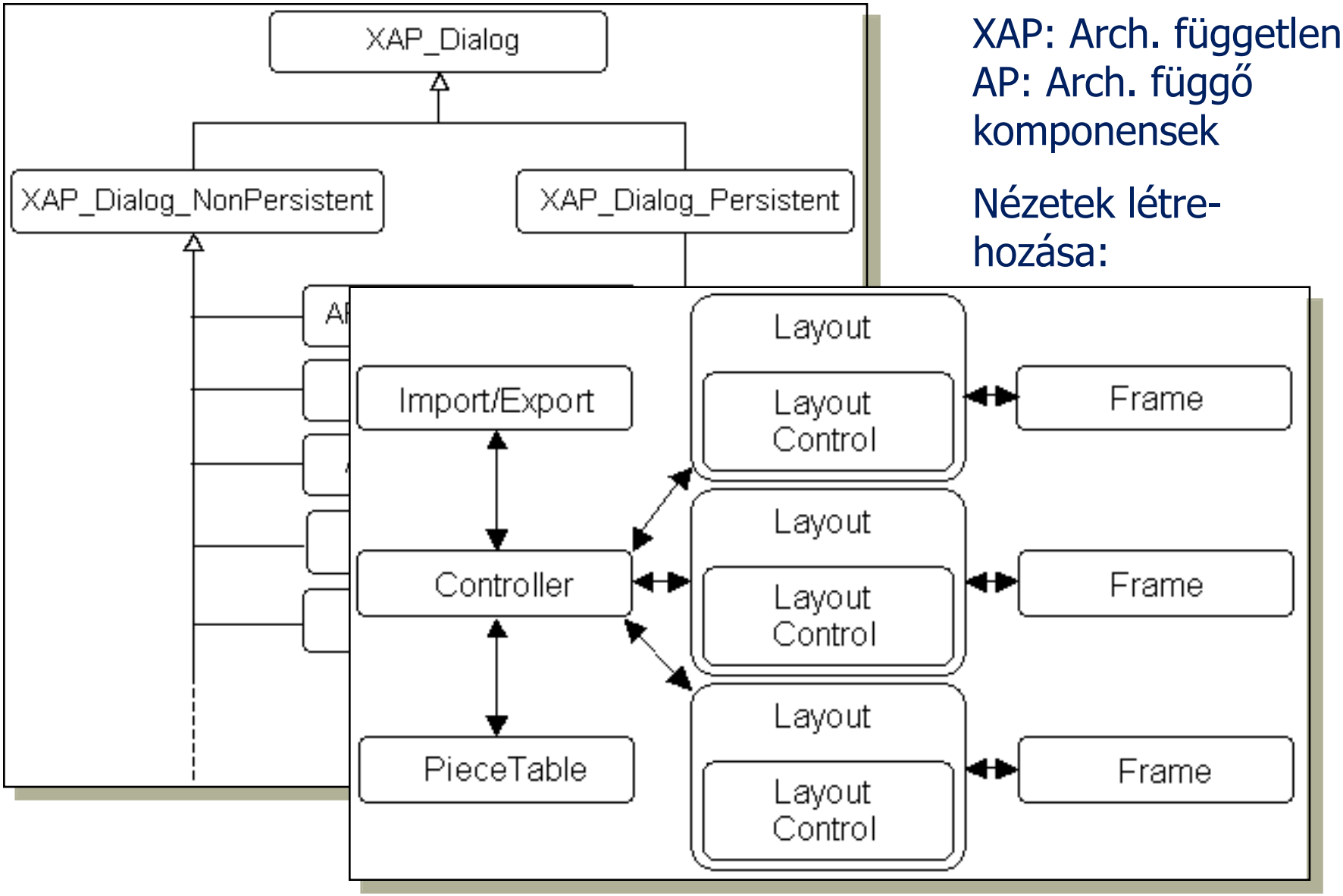
Kiértékelés a további érdekeltekkel

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

Mintapélda: AbiWord architektúra elemzése

- Célok
- Architektúra
- Hasznossági fa, forgatókönyvek
- Analízis

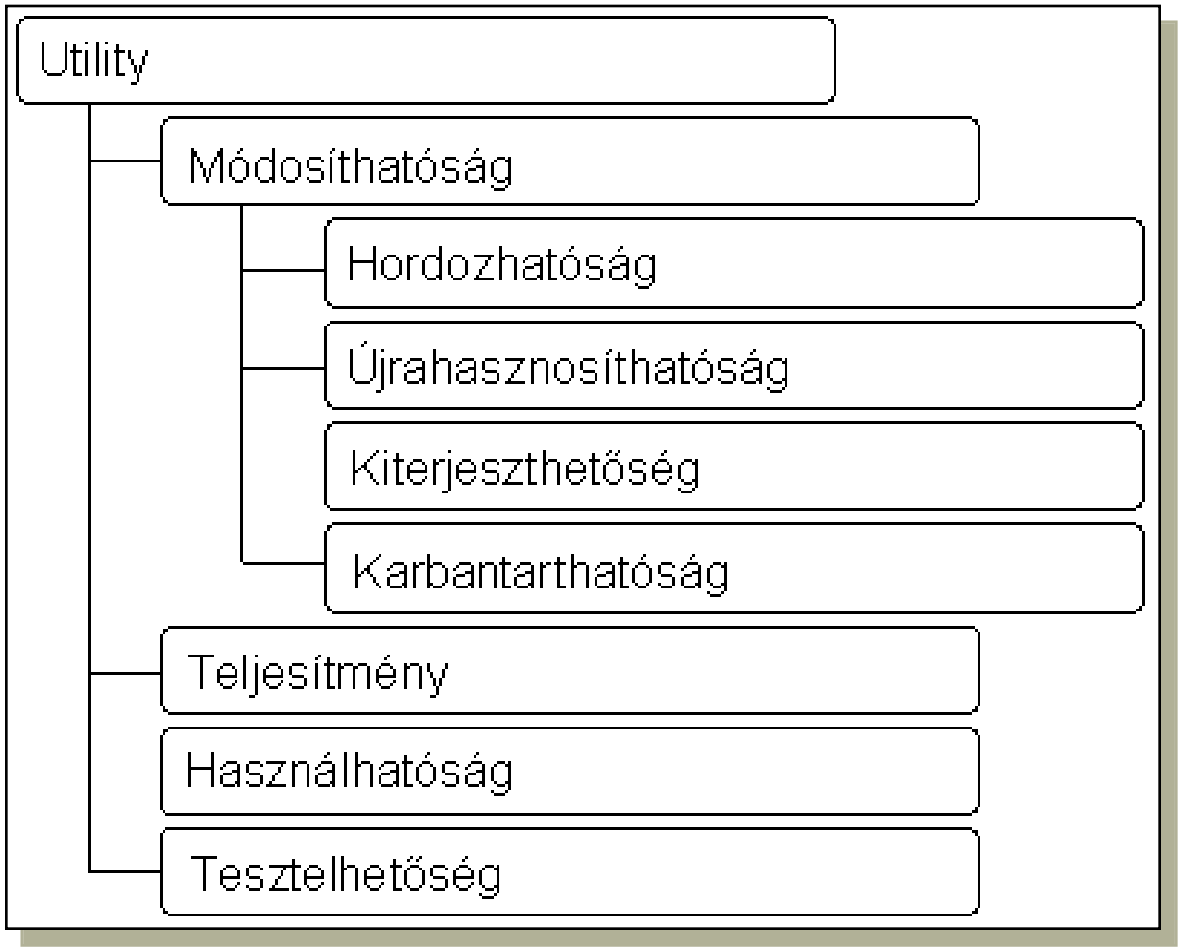
Mintapélda: Az architektúra



XAP: Arch. független,
AP: Arch. függő komponensek

Nézetek létre-hozása:

Mintapélda: Hasznossági fa



Prioritások

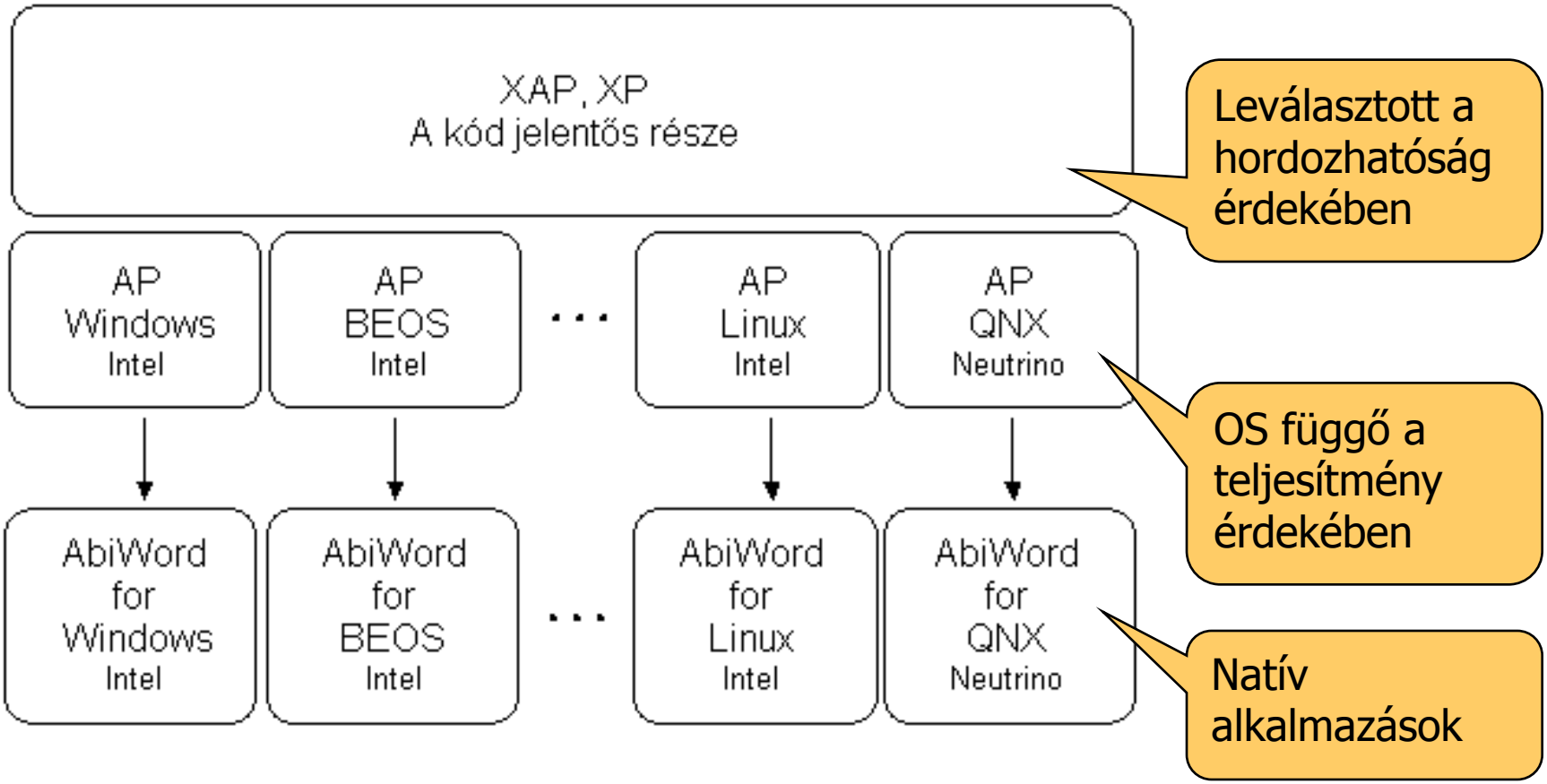
1.
4.
5.
6.
2.
3.
7.

Mintapélda: Forgatókönyvek

1. Hordozhatóság
   - Bemeneti hatás: Új operációs rendszert is támogatni kell
   - Környezet: Fejlesztési idő
   - Reakció: A leválasztott OS függőségek (ld. AP*) miatt a támogatás hetekben mérhető idő alatt megvalósítható alacsony költséggel
3. Használhatóság
   - Bemeneti hatás: A felhasználó HTML dokumentum készítése közben szeretné ellenőrizni az eredményt
   - Környezet: Futási idő
   - Reakció: A szoftver számos nézet lehetőségét biztosítja
5. Kiterjeszthetőség
   - Bemeneti hatás: Nem áll a felhasználó rendelkezésére valamely funkció
   - Környezet: Futási idő
   - Reakció: A felhasználó számos előre elkészített plug-in segítségével tudja bővíteni a szoftver funkcionalitását

Mintapélda: Analízis

- Tradeoff: Hordozhatóság és teljesítmény



Az ATAM előnyei

- Érdekeltek bevonása
  - Fejlesztő, tesztelő, felhasználó, elemző szempontjai
  - Hatékony kommunikáció az érdekeltek között
- Tisztázott minőségi követelmények
  - Jellemzők és scenariok azonosítása
  - Prioritások meghatározása
- Kockázatok korai azonosítása
  - Architektúra döntések hatásának explicit elemzése (modell alapú analízis is bevonható)
  - Kompromisszumok vizsgálata
- Architektúrához kapcsolódó döntések dokumentálása

Kitekintés: Több szempontú optimalizálás

- Alapötlet: Pontozási rendszer alternatívák (itt: architektúra változatok) közötti választáshoz
  - Több kritérium az alternatívák közötti választáshoz
  - Több teljesítési szint minden kritérium esetén
  - Pontszámok (értékelés, súlyozás) az egyes teljesítési szintekhez
    - A pontszámok hozzárendelése a legnehezebb része az értékelésnek
  - Az egyes alternatívák esetén összegezhetők a kritériumok teljesítési szintjeihez tartozó pontszámok, ami a döntés alapja
- PAPRIKA: Potentially all pairwise rankings of all possible alternatives
  - Pontszámok hozzárendelése a kritériumok teljesítési szintjeihez, az alternatívák páronkénti rangsorolása (fontossági sorrendje) alapján
  - Egy megadott rangsor által már meghatározott (dominált) további párokat nem kell értékelni (így nincs kombinatorikus robbanás)
    - Pl.: 25 páronkénti rangsorolás elég, ha 5 kritérium és 3-3 szint van
  - A pontszámok a rangsorolás alapján lineáris programozással hozzárendelhetők

Miről volt szó?

- 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