

Szoftverellenőrzési technikák

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

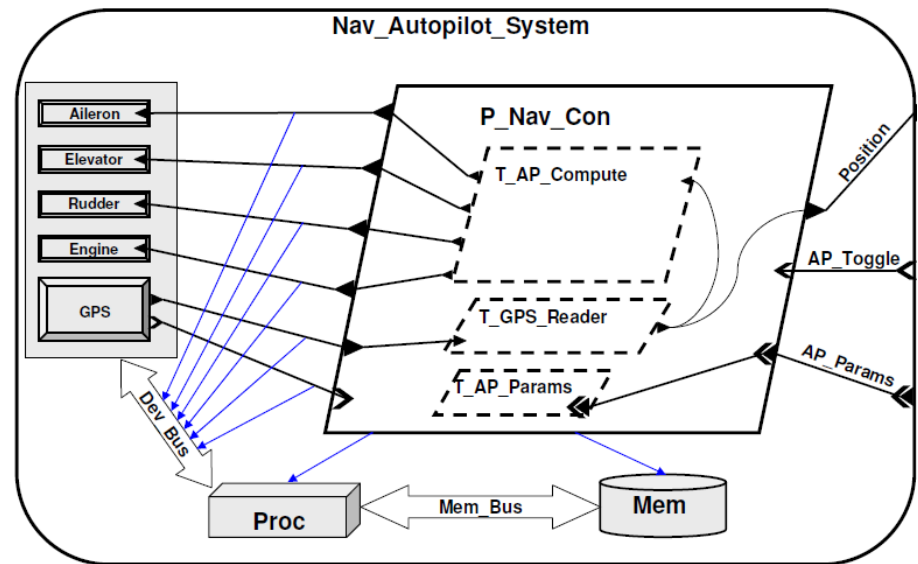
Szoftverarchitektúra

- Architektúra
 - Komponensek (tulajdonságokkal)
 - Köztük 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)

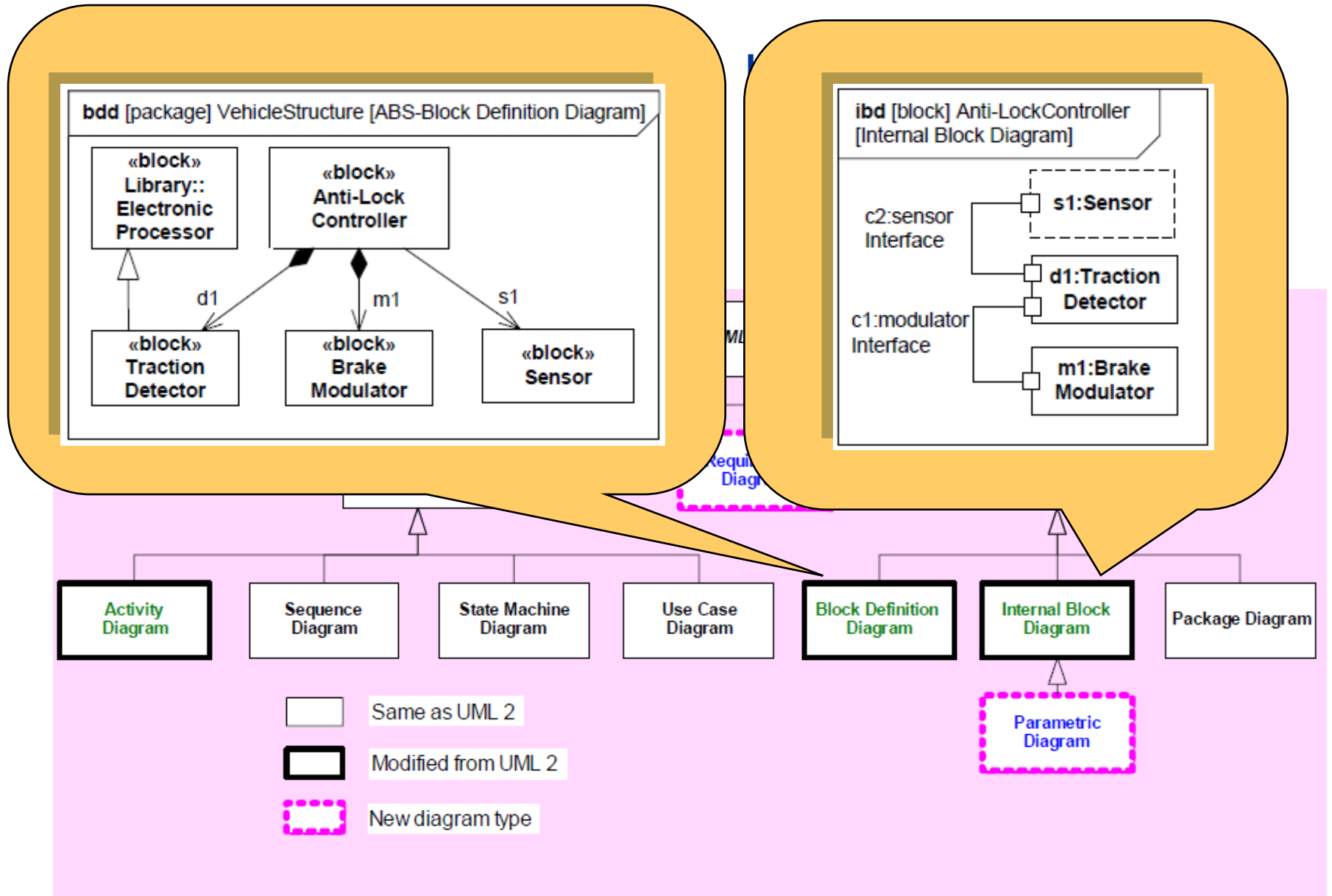
Jellegzetes nyelvek architektúra tervezéshez

- 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;
```



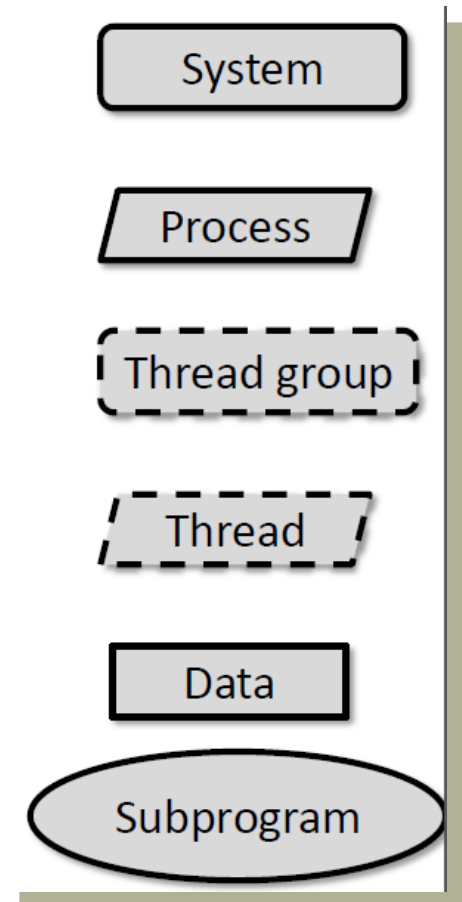
Jellegzetes nyelvek architektúra tervezéshez: SysML



Jellegzetes nyelvek architektúra tervezéshez: AADL

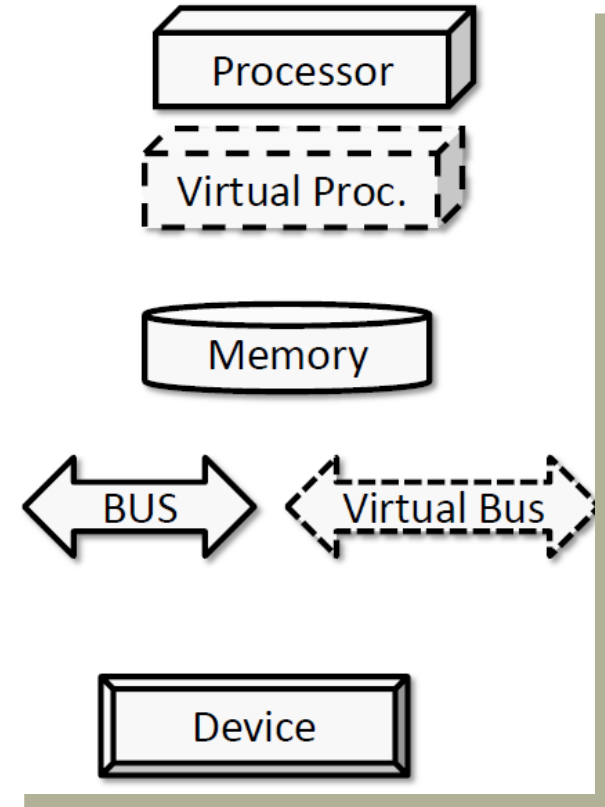
AADL: Architecture Analysis and Design Language (v2: 2009)

- 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



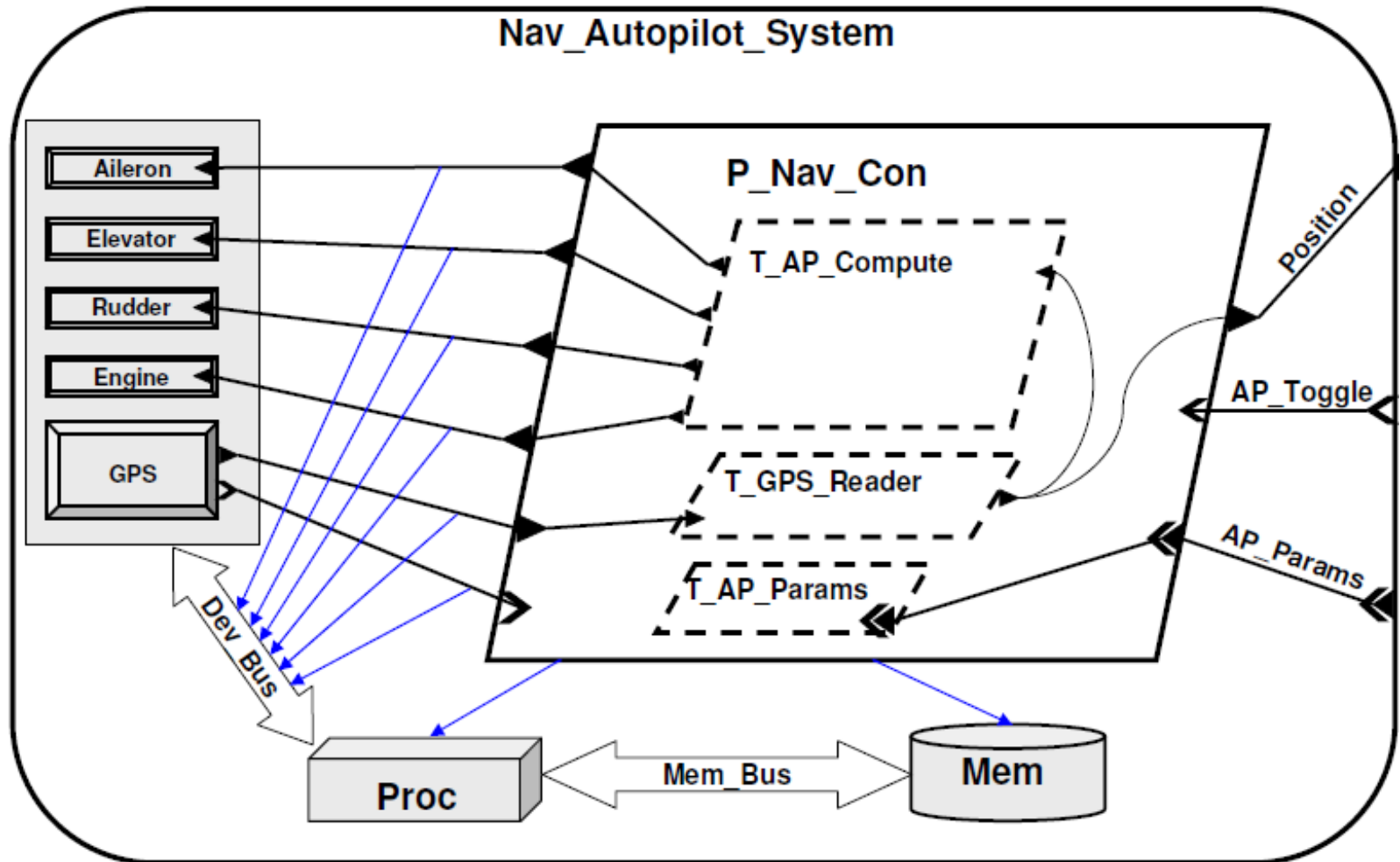
Jellegzetes nyelvek architektúra tervezéshez: AADL

- **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



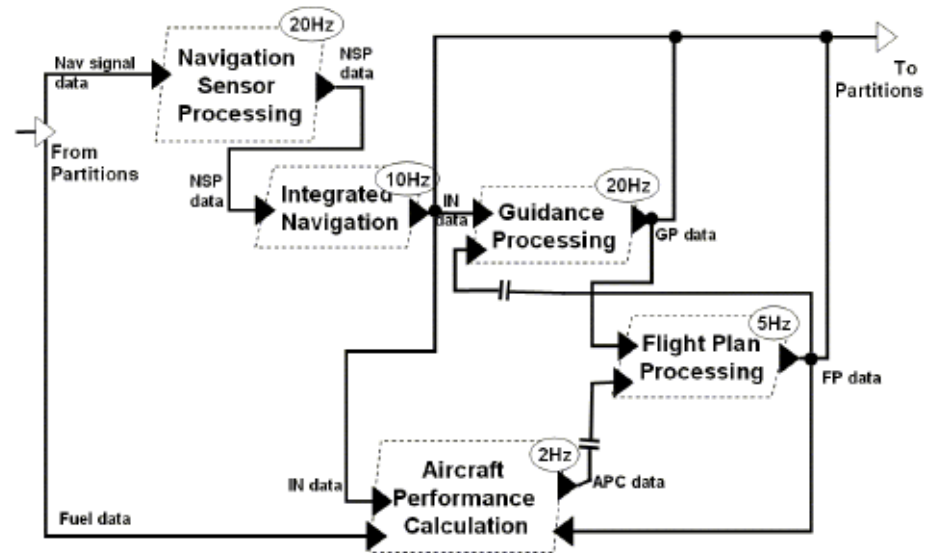
Jellegzetes nyelvek architektúra tervezéshez: AADL

- **Kapcsolatok**

- Adatok, események áramlása portokon

- **Tulajdonságok megadása analízishez**

- Időzítés
- Ütemezés
- Hibaterjedés (annex)
- ...

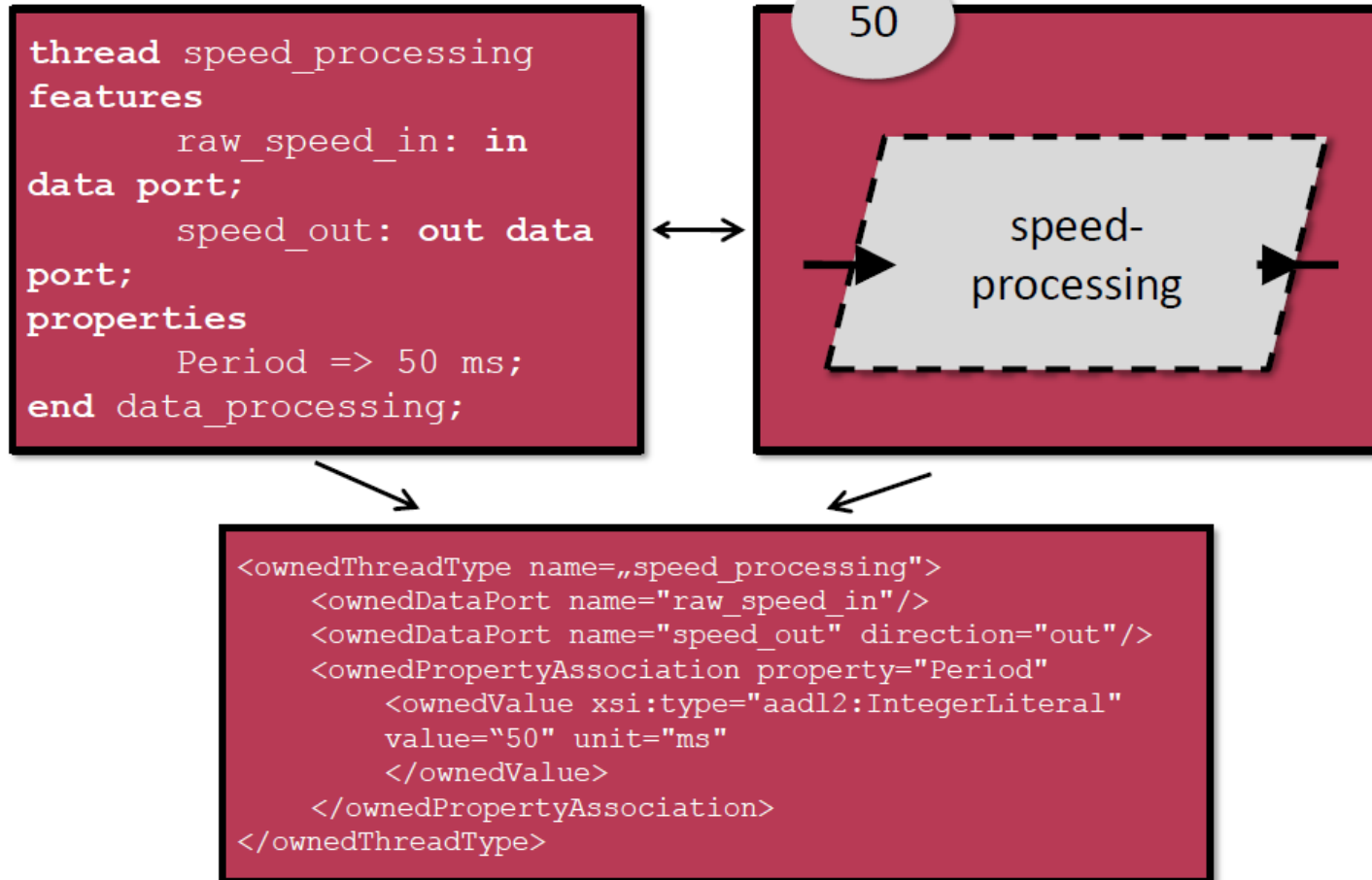


```
thread implementation CoinPublisher.impl
  calls(u: subprogram updateTotal;);
  properties

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

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
 - Erőforrás hozzárendelés: Kritikus számítások kiszolgálása, várakozó kérések korlátozása
 - Erőforrás menedzsment: Párhuzamos feldolgozás, cache, erőforrás ütemezés, dinamikus hozzárendelés
- Adatbiztonság
 - Támadás kivédése: Elérés korlátozása, autentikáció, bizalmas adatok védelme
 - Támadás felderítése: Minták (változások) elemzése
 - Helyreállítás: Integritás karbantartása

Mit és hogyan határoz meg az architektúra? 2/2

- **Módosíthatóság**
 - Lokalizálás: Szemantikus koherencia
 - 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

Elérendő tulajdonságok és tervezési tér (áttekintés)

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

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

Sokféle kombináció megengedett

Failure assertion

Referencia

Bennmaradó hibák elleni védekezés

- **Ellenjavallt technikák (NR)**

- Előre / visszalépő helyreállítás
- Mesterséges intelligencia módszerek hibajavításra
- Dinamikus szoftver rekonfiguráció

Példa: OTS komponensek használata kritikus rendszerekben

- **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: 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

Áttekintés: Milyen vizsgálati technikák vannak?

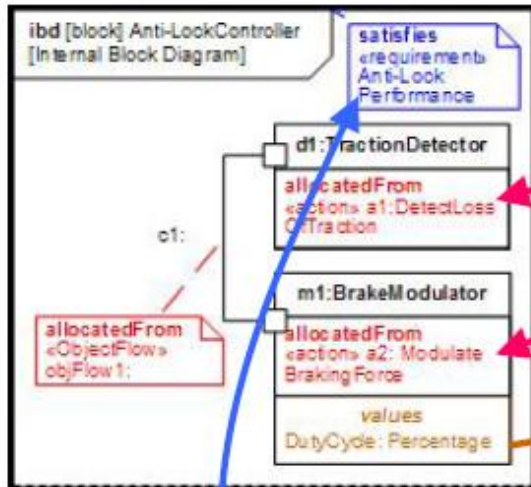
- Követhetőség a követelmények felől
 - Követelmények „lefedése” architektúra elemekkel
- Szisztematikus elemzések: az architektúra bejárása
 - **Interfész analízis**
 - Elvárt és nyújtott interfészek megfeleltetése
 - **Hibahatás analízis** kombinatorikus módszerekkel
 - Komponens szintű hibák ↔ Rendszerszintű hatások
- Modell alapú vizsgálatok: architektúra analízis
 - **Sztochasztikus analízis**: extra-funkcionális jellemzők meghatározása
 - Lokális (komponens illetve kapcsolati szintű) paraméterek alapján rendszerszintű jellemzők számítása
 - Az analízis modell konstruálása: Az architektúra alapján
- Trade-off analízis

Tartalomjegyzék

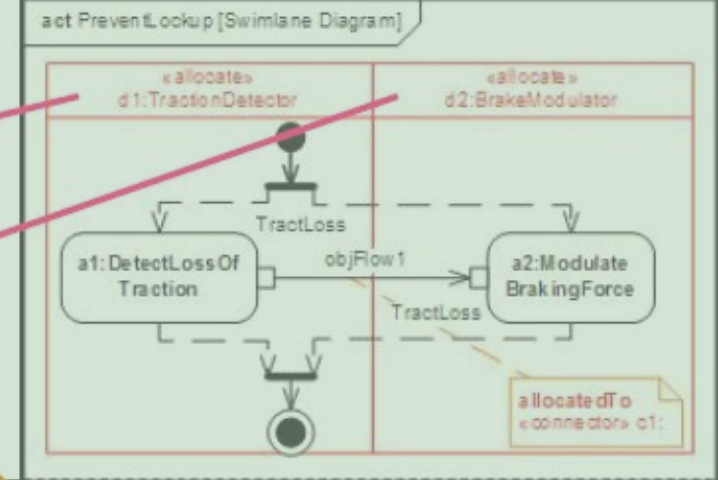
- Motiváció
 - Mit határoz meg az architektúra?
 - Milyen vizsgálati módszerek vannak?
- Követhetőség
- Szisztematikus vizsgálati módszerek
 - Interfész analízis
 - Hibahatás analízis
- Modell alapú vizsgálatok
 - Megbízhatósági modellezés
 - Teljesítmény modellezés
- ATAM architektúra elemzés

Példa: Követhetőség SysML relációk alapján

1. Structure



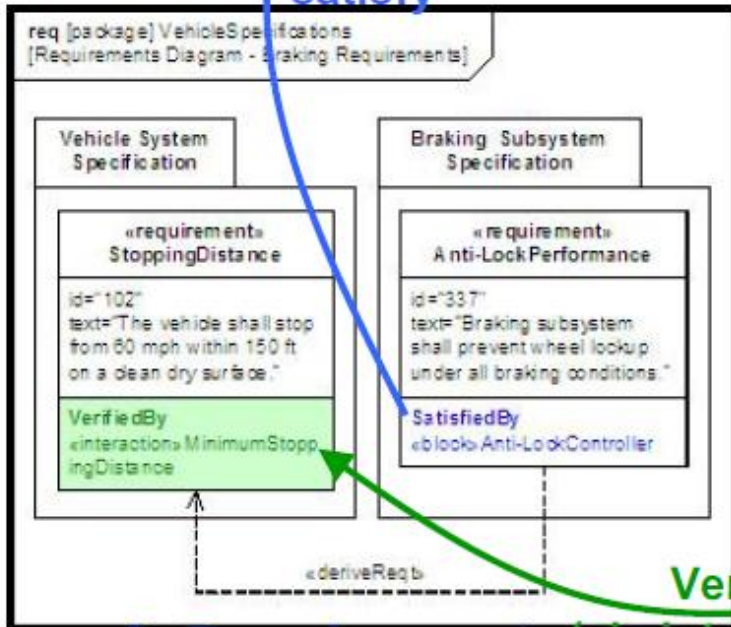
2. Behavior



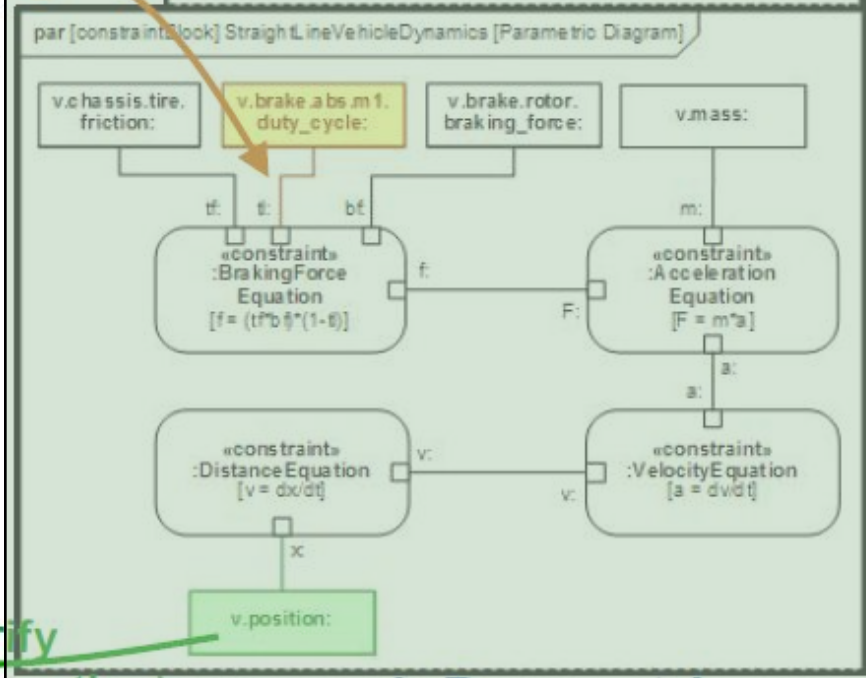
allocate

value binding

satisfy



Verify



3. Requirements (via interaction)

4. Parametrics

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

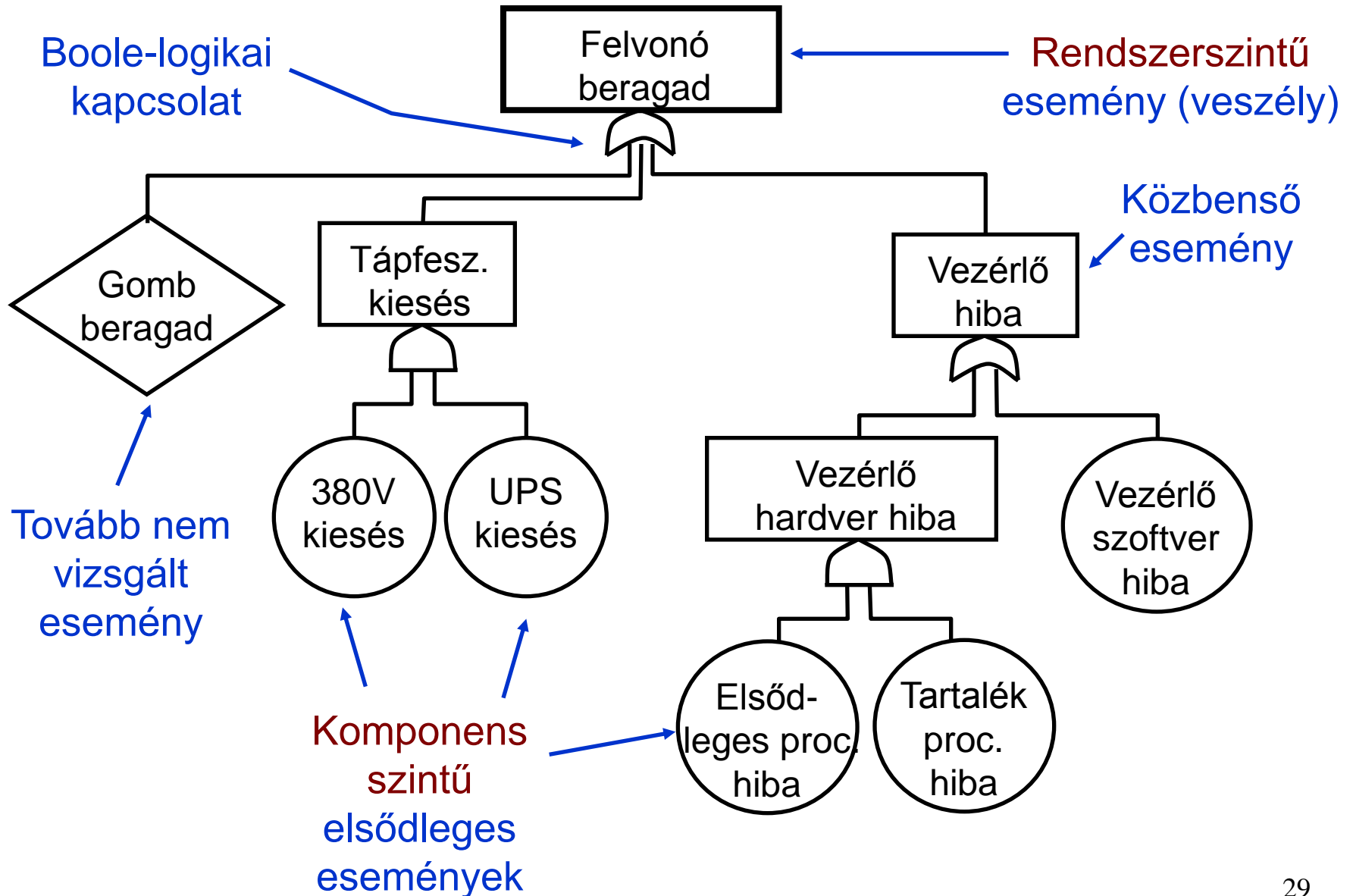
```
public class Purse {
    final int MAX_BALANCE;
    int balance;
    /*@ invariant pin != null && pin.length == 4 @*/
    byte[] pin;
    /*@ requires amount >= 0;
       @ assignable balance;
       @ ensures balance == \old(balance) - amount
           && \result == balance;
       @ signals (PurseException) 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

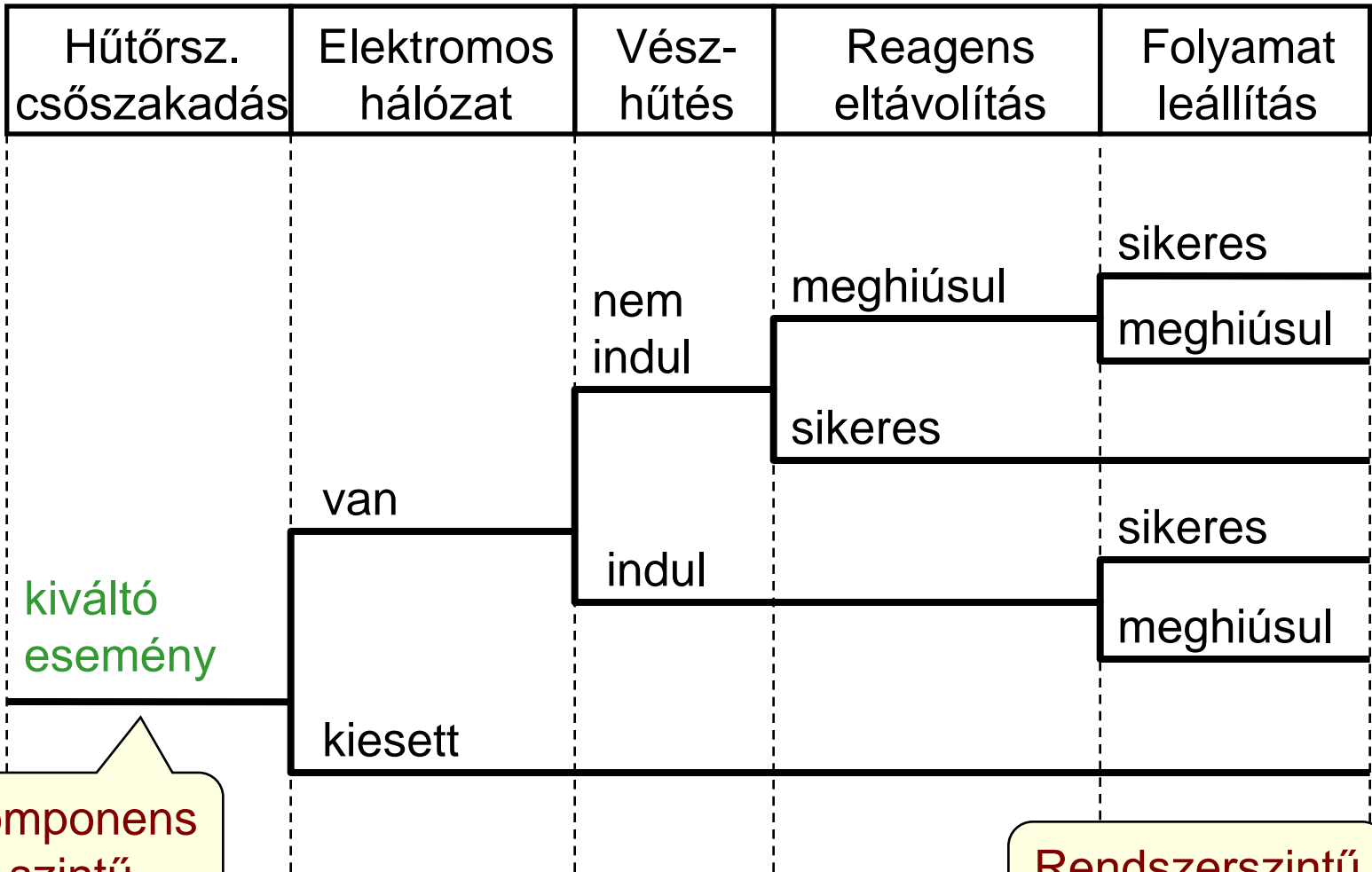
- Célkitűzés
 - **Komponens** jellemzők és **rendszerszintű** jellemzők közötti kapcsolatok megállapítása
 - Teljesség: Minden komponens hibamódjai szerepelnek
- Analízis megközelítése
 - Az architektúra szempontjából
 - **Alulról felfelé**: A komponensek (alrendszerek) felől
 - **Felülről lefelé**: 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 példája: Felvonó



Eseményfa példa: Reaktorhűtés

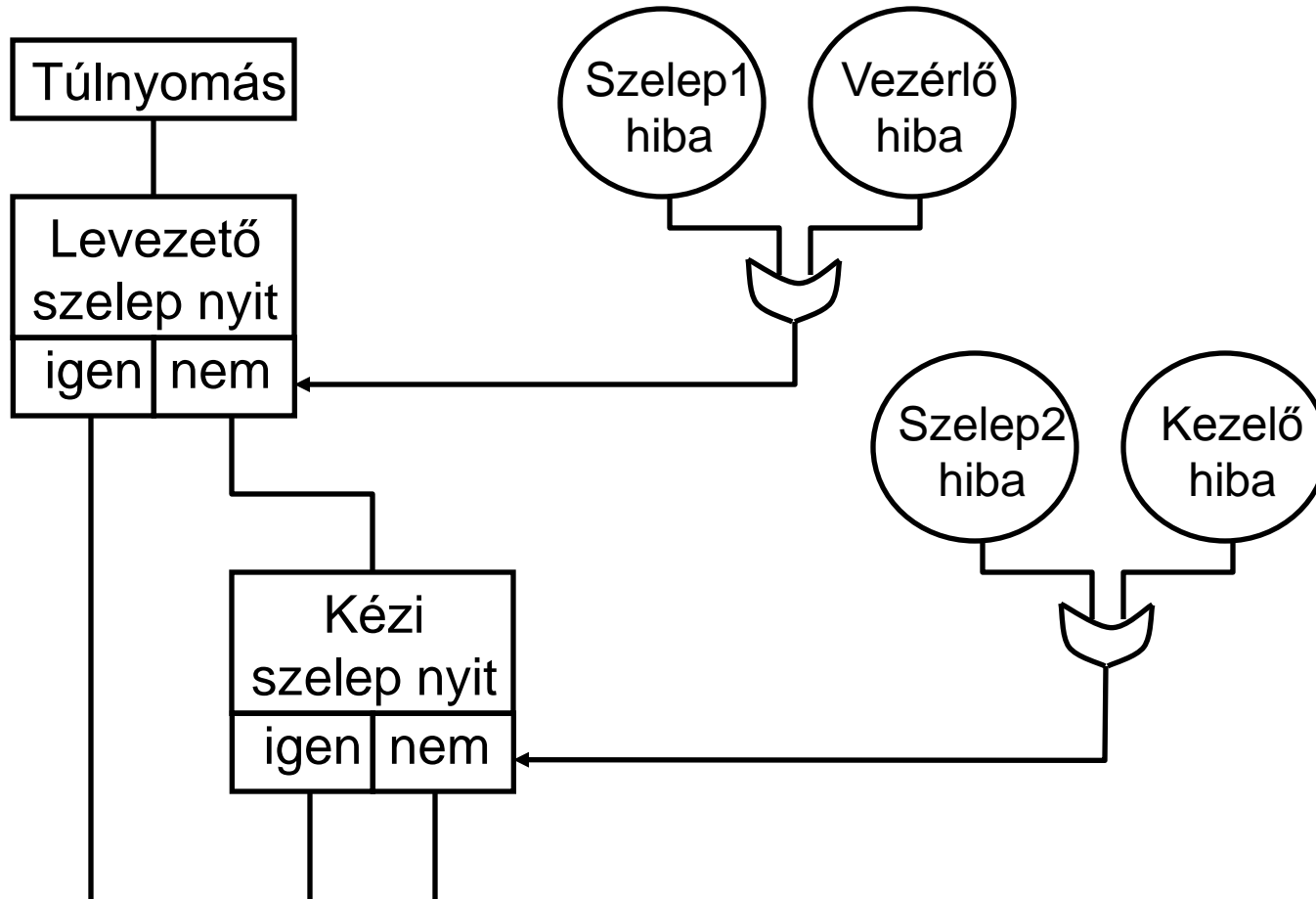
Releváns komponensek



Komponens szintű

Rendszerszintű hatások

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

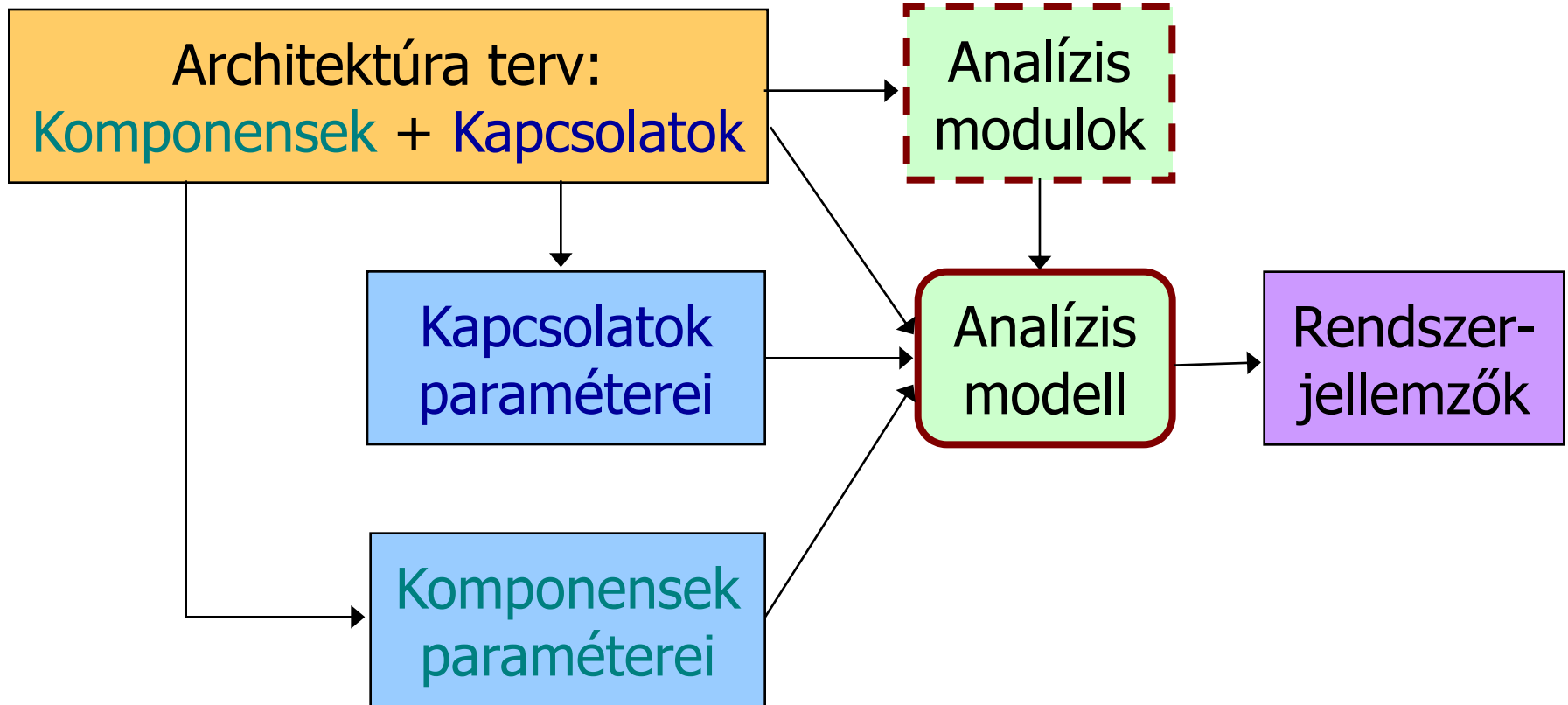
- Motiváció
 - Mit határoz meg az architektúra?
 - Milyen vizsgálati módszerek vannak?
- Követhetőség
- Szisztematikus vizsgálati módszerek
 - Interfész analízis
 - Hibahatás analízis
- **Modell alapú vizsgálatok**
 - **Megbízhatósági modellezés**
 - Teljesítmény modellezés
- ATAM architektúra elemzés

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

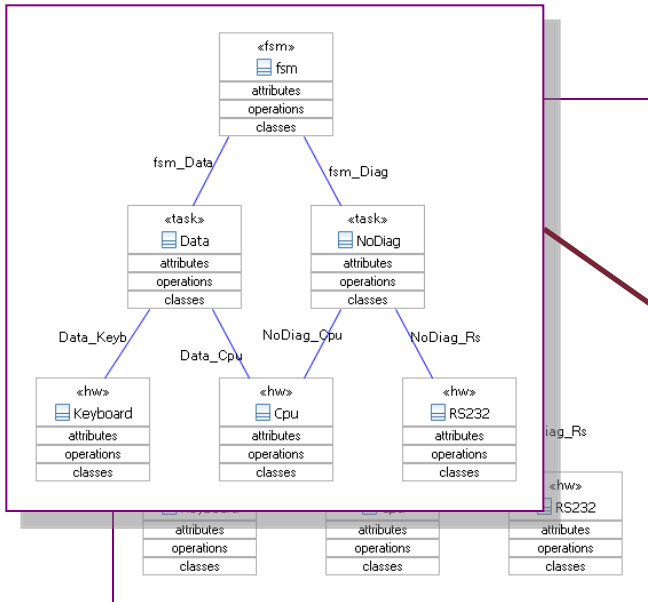


Rendszerszintű analízis modellek

	Megbízhatósági modell	Teljesítmény modell	Biztonsági modell
Komponens paraméterek	Meghibásodási tényező, lappangási idő, javítási tényező, hibafedés, ...	Funkció lokális végrehajtási idő, 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, készlet, MTTF, MTTR, MTBF	Kiszolgálási idő, áteresztő-képesség, processzor kihasználtság	Rendszerszintű veszély gyakoriság

Példa: UML alapú megbízhatósági modellezés

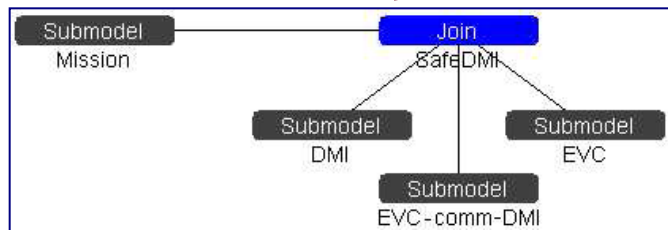
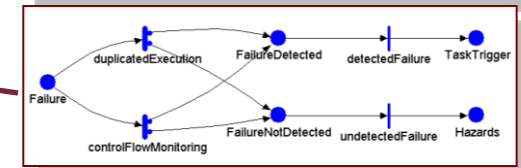
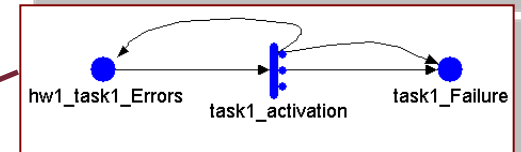
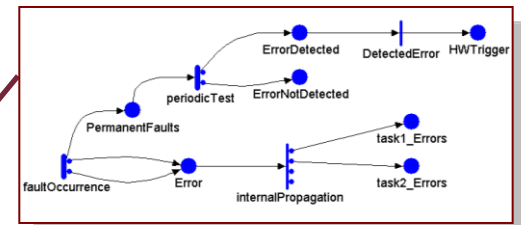
UML architektúra modell



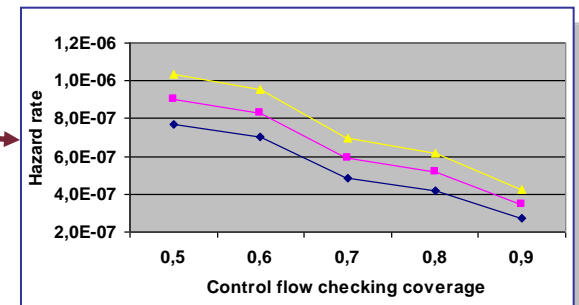
Megbízhatósági modell konstrukció



Analízis modellkönyvtár



Rendszerszintű megbízhatósági modell



Analízis eredmények

Példa: Megbízhatósági modellezés – Architektúra modell

Komponens:

- Típus (HW, SW)
- Szerep (variáns, menedzser)

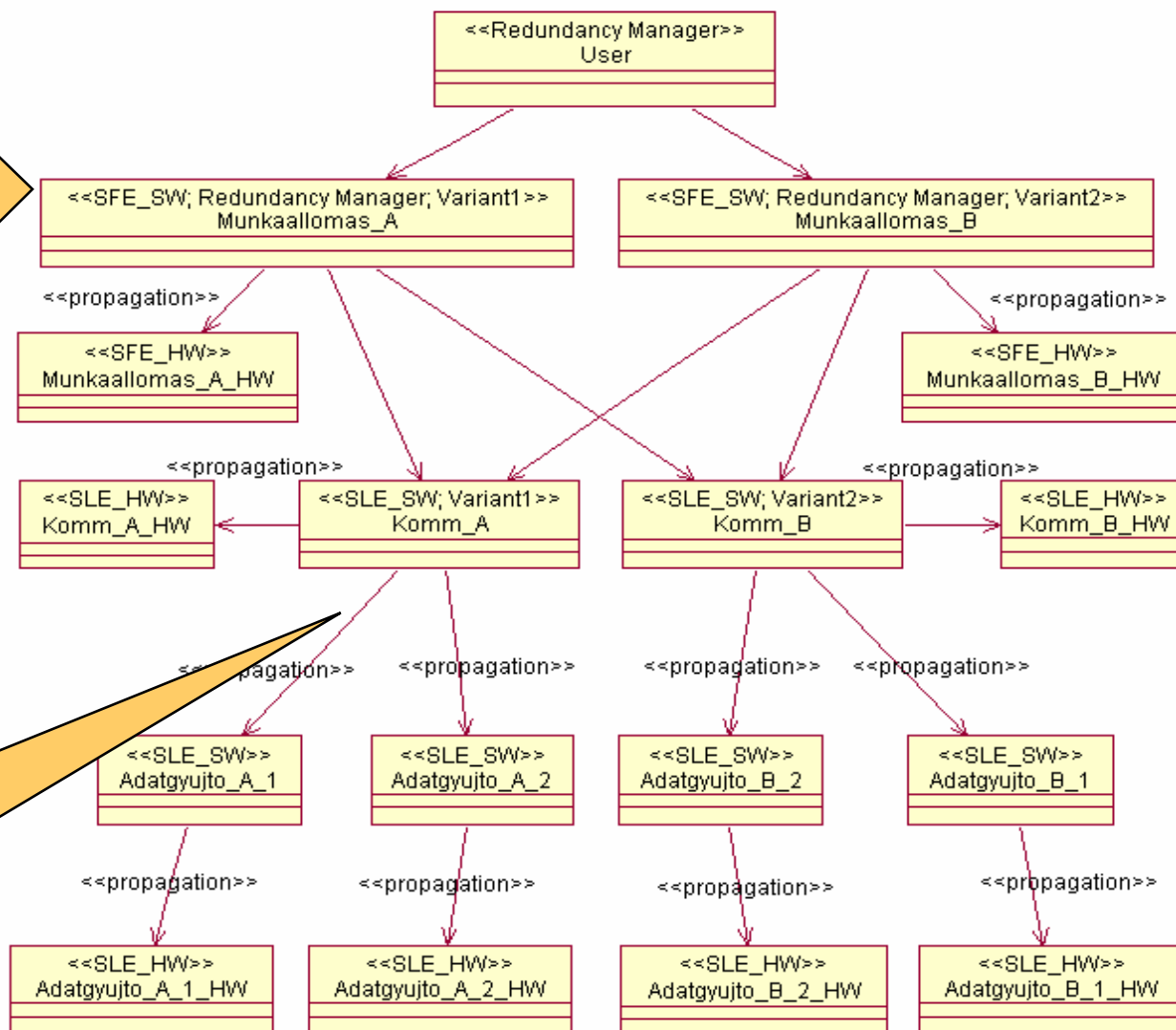
- Meghibásodási jellemzők:

- * meghibásodási gyakoriság,
- * lappangási idő,
- * javítási idő

Kapcsolat:

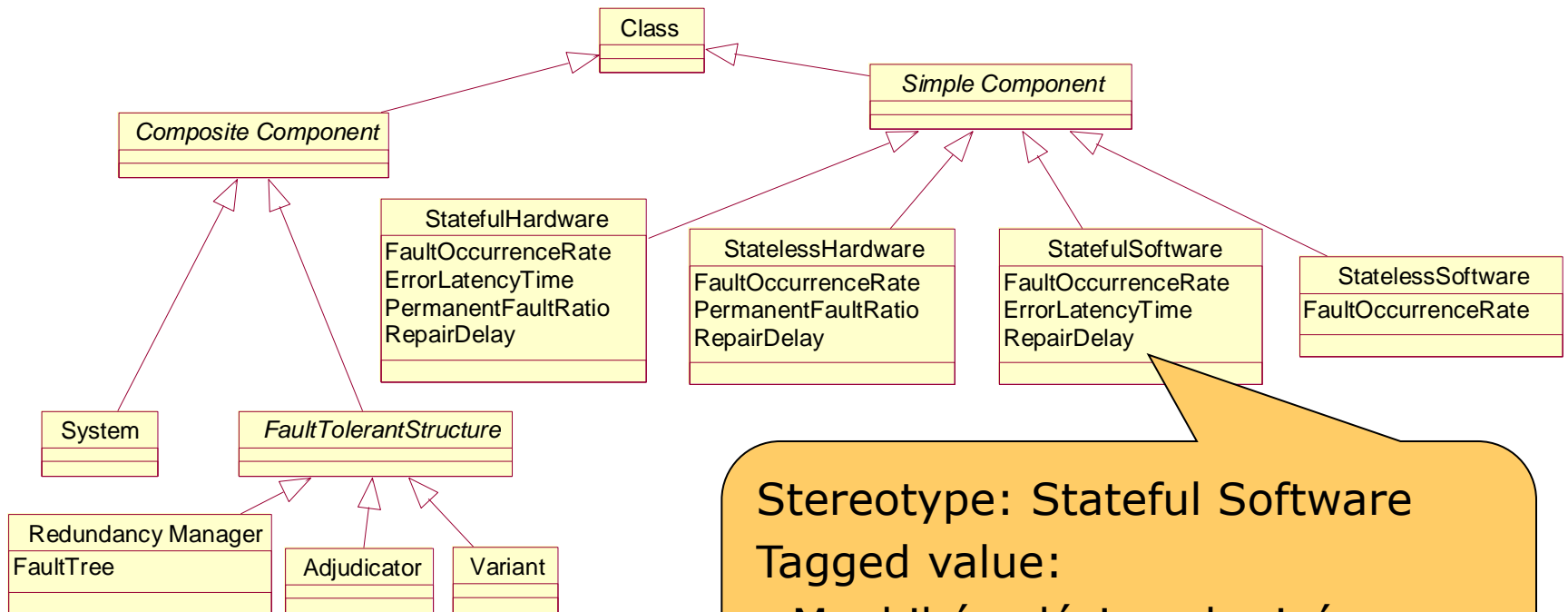
- Hibaterjesztési jellemzők:

- * hibaterjesztési valószínűség



Példa: Komponensek lokális tulajdonságai

- UML profil az extra-funkcionális tulajdonságokhoz

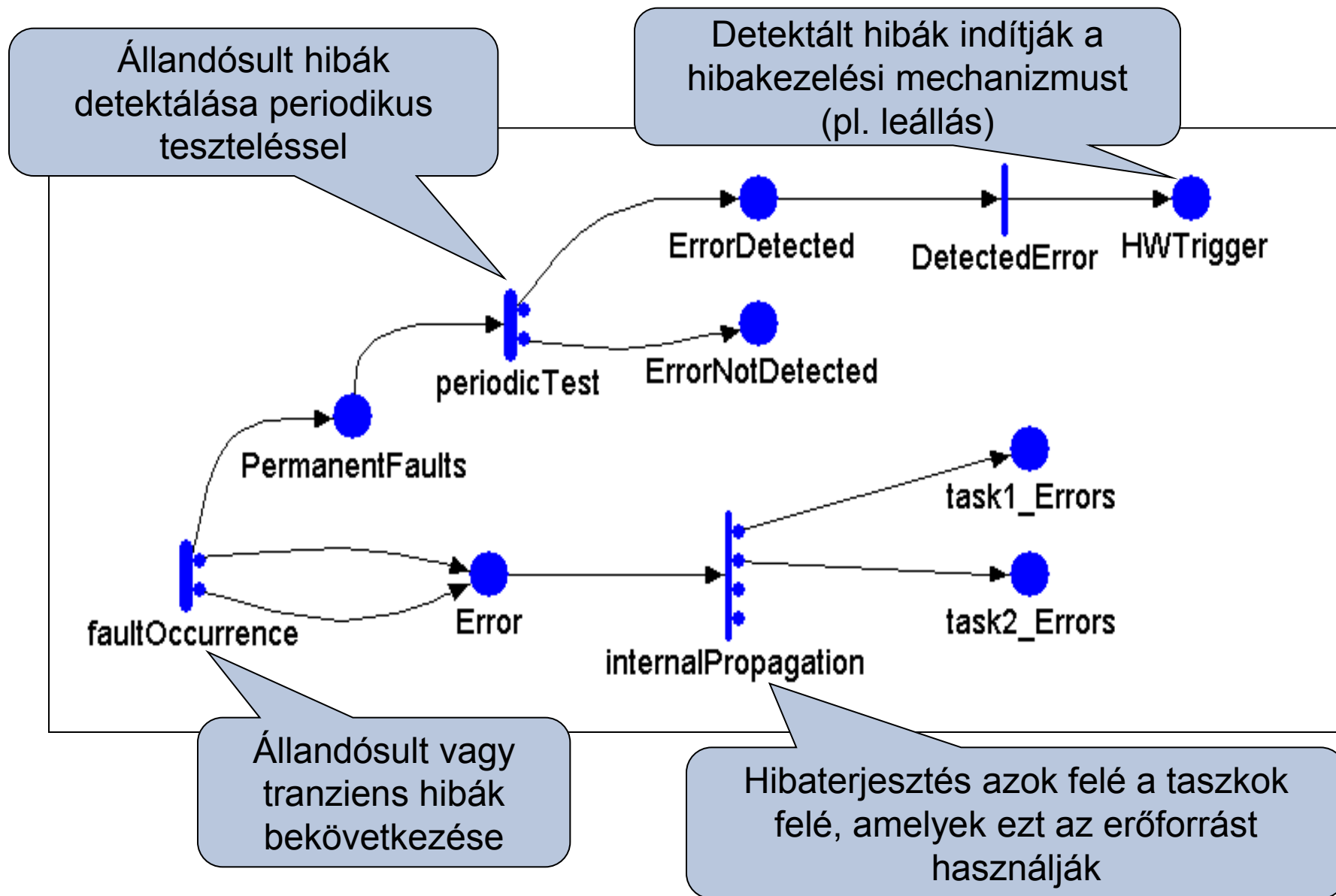


Stereotype: Stateful Software

Tagged value:

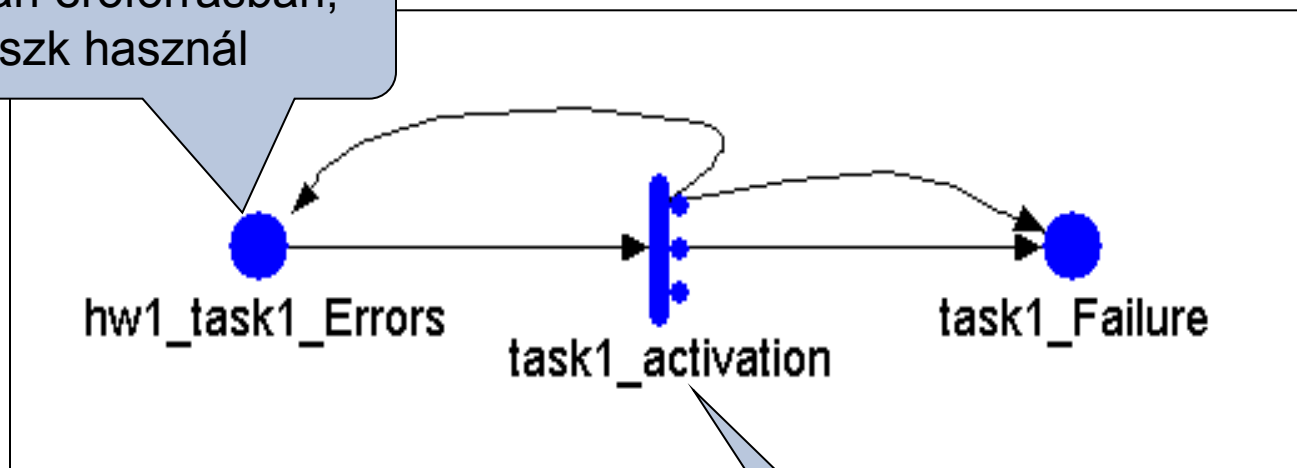
- Meghibásodási gyakoriság
- Hiba lappangási idő
- Javítási tényező

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



Példa: Kapcsolat analízis alháló: Hibaterjesztés

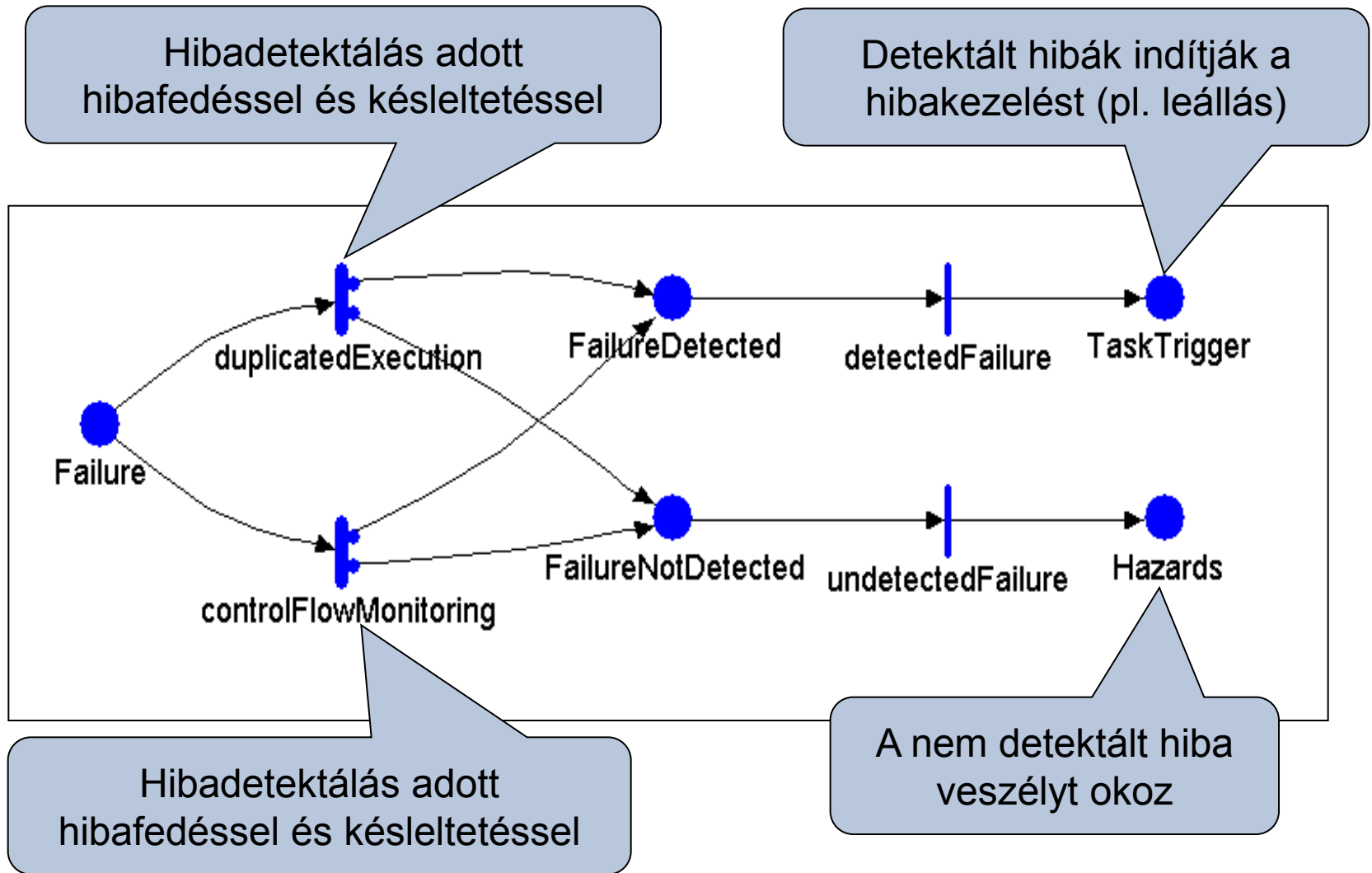
Hiba olyan erőforrásban,
amit a taszk használ



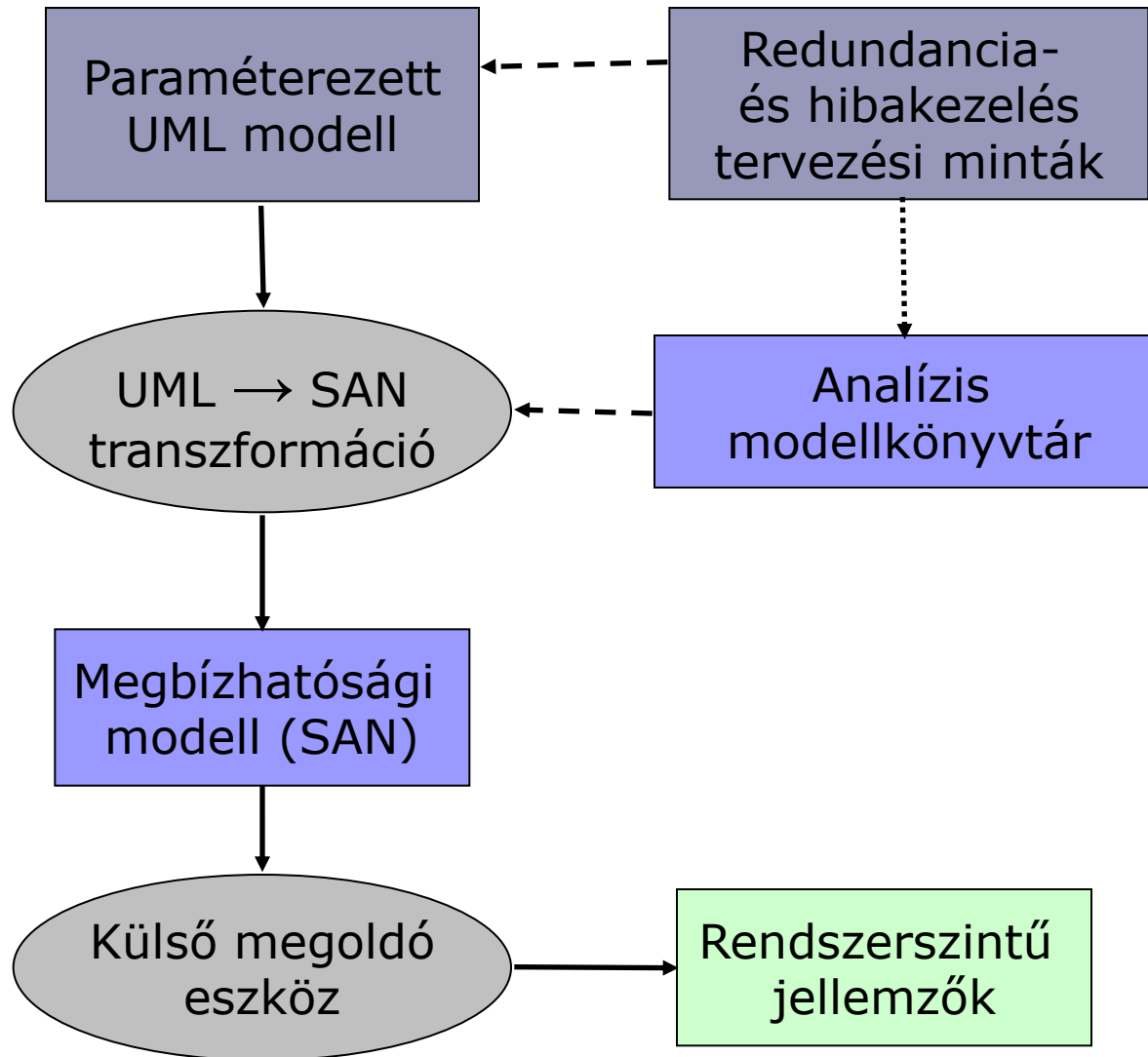
Taszk aktiválási gyakoriság,
hiba aktiválási lehetőségek:

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

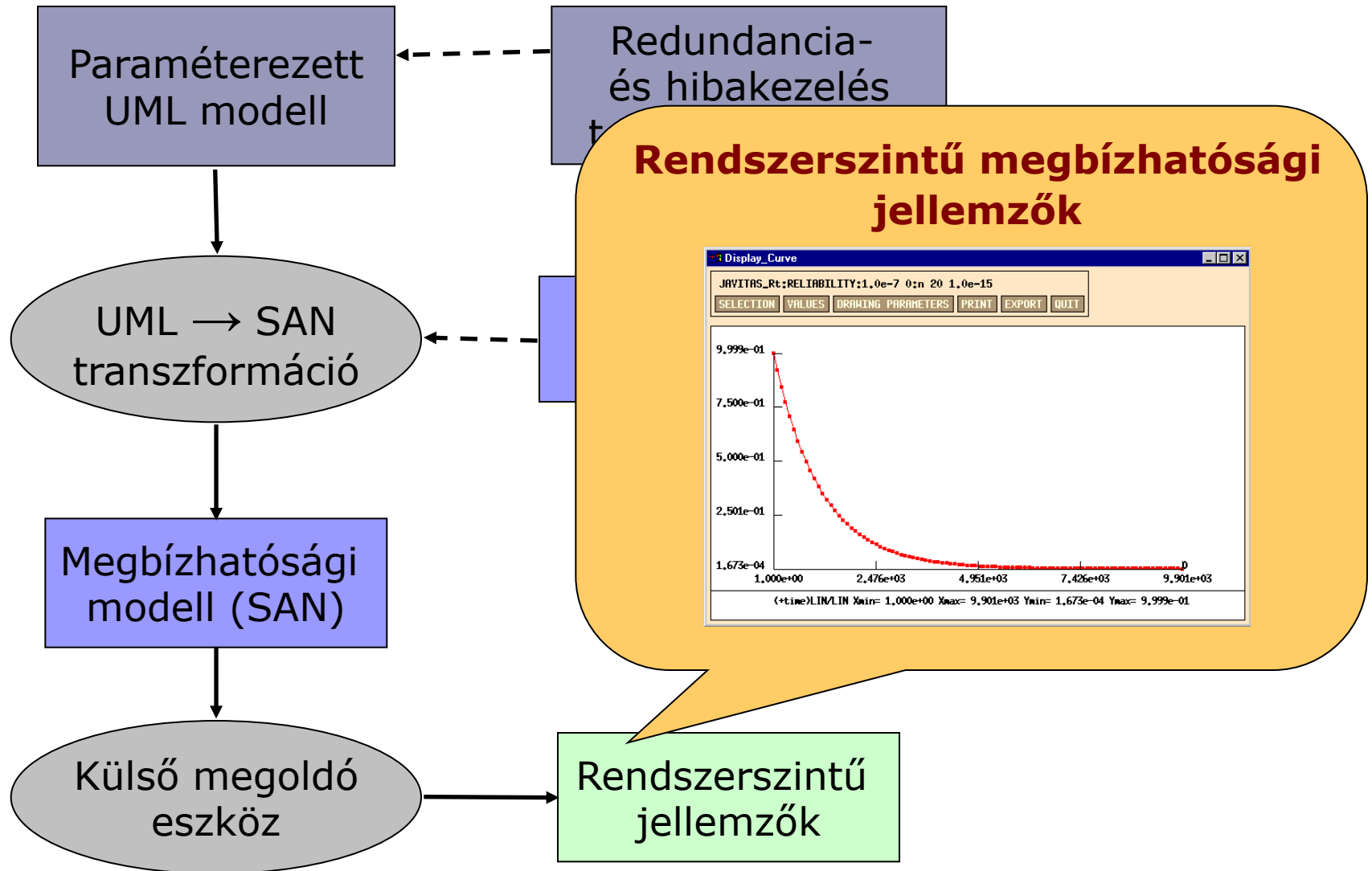
Példa: Komponens analízis alháló: Egy szoftver taszk



Példa: Automatikus analízis eszköz



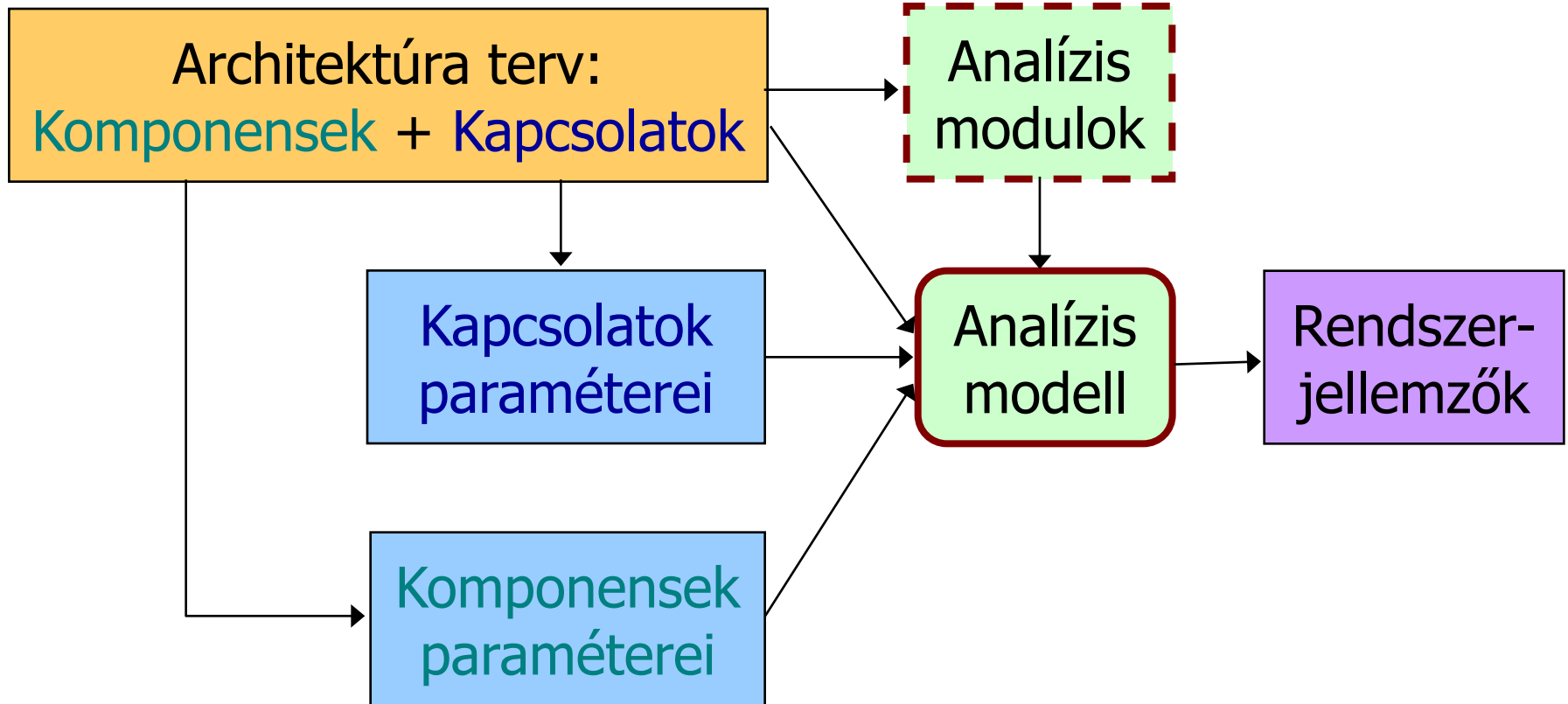
Példa: Automatikus analízis eszköz



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



Rendszerszintű analízis modellek

	Megbízhatósági modell	Teljesítmény modell	Biztonsági modell
Komponens paraméterek	Meghibásodási tényező, lappangási idő, javítási tényező	Funkció lokális végrehajtási idő, 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, készlet, MTTF, MTTR, MTBF	Kiszolgálási idő, áteresztő-képesség, processzor kihasználtság	Rendszerszintű veszély gyakoriság

Példa: Teljesítmény modellezés (LQN)

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

Kliens (kérés):
- Hívási gyakoriság

CPU



Processzor:
- Telepítés
- Ütemezés

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

Példa: Teljesítmény modellezés (LQN): Rétegek

Taszk (szerver):

- Funkciók (szolgáltatás hívási pontok)
- Prioritások

Funkció:

- Lokális végrehajtási idő
- Továbbhívás gyakoriság

Kliens (kérés):

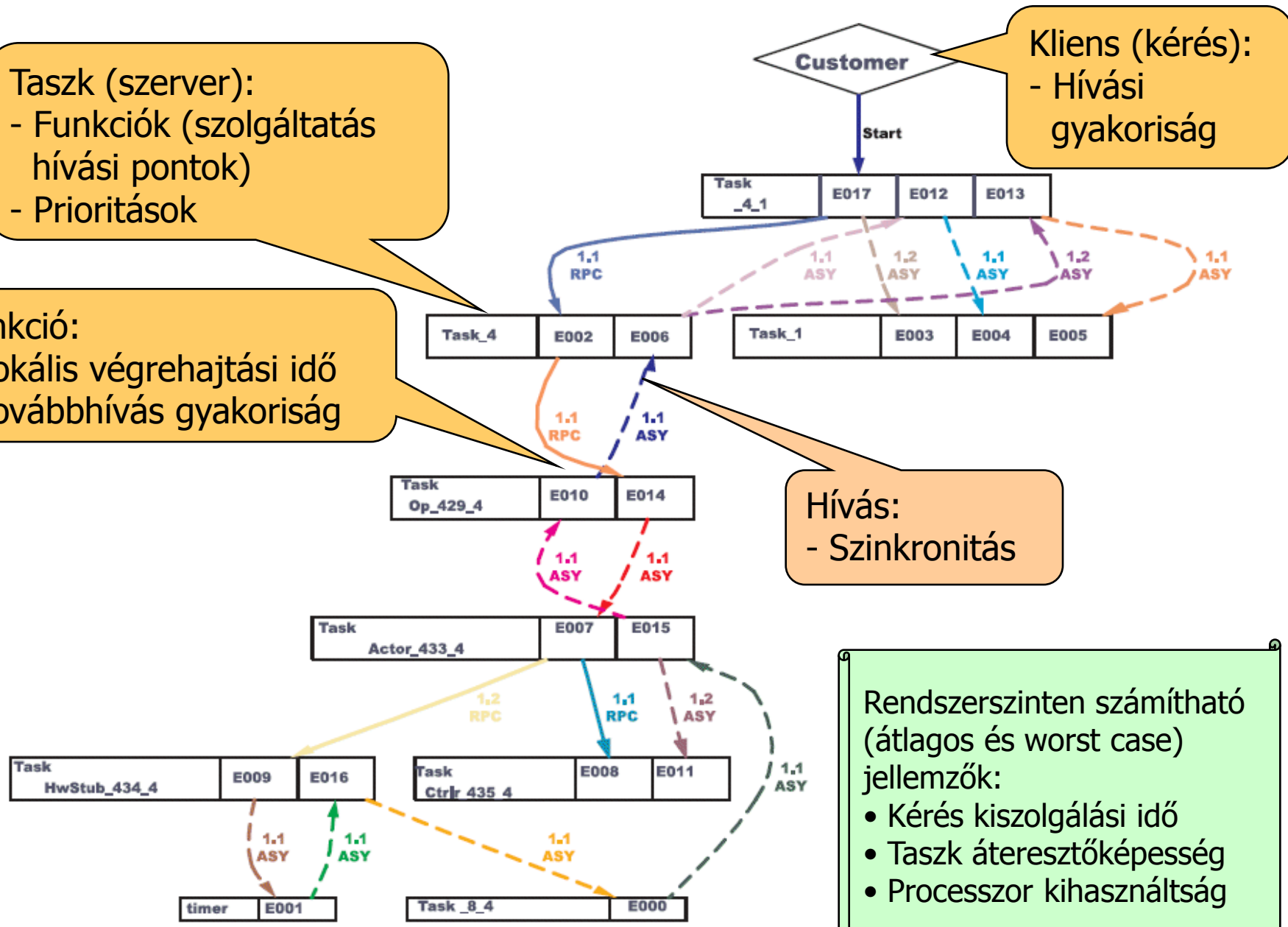
- Hívási gyakoriság

Hívás:

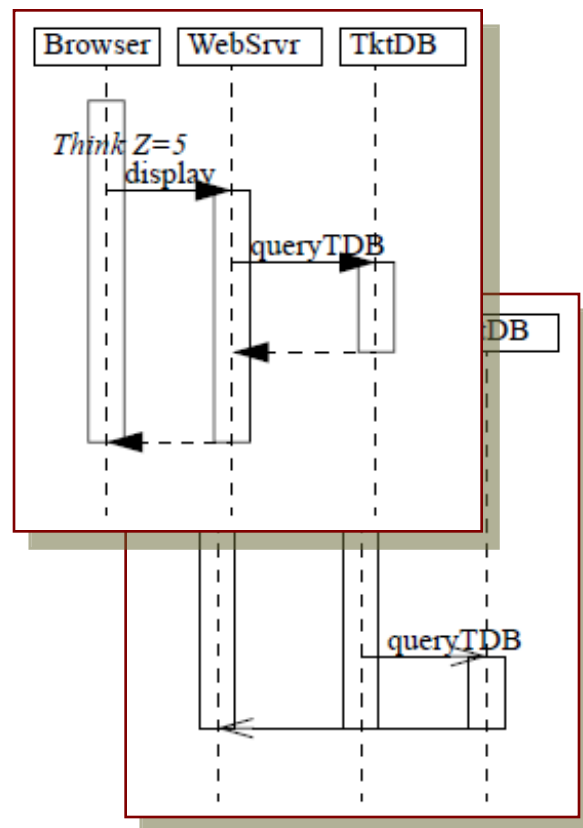
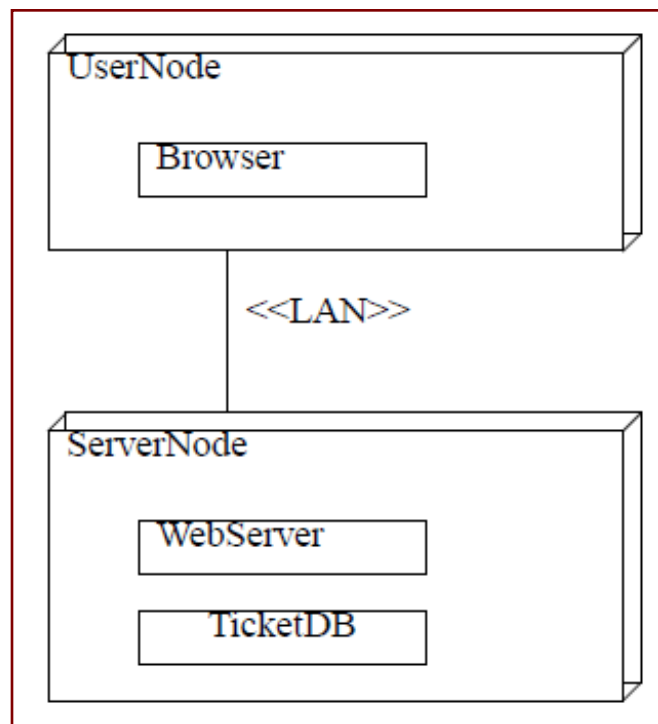
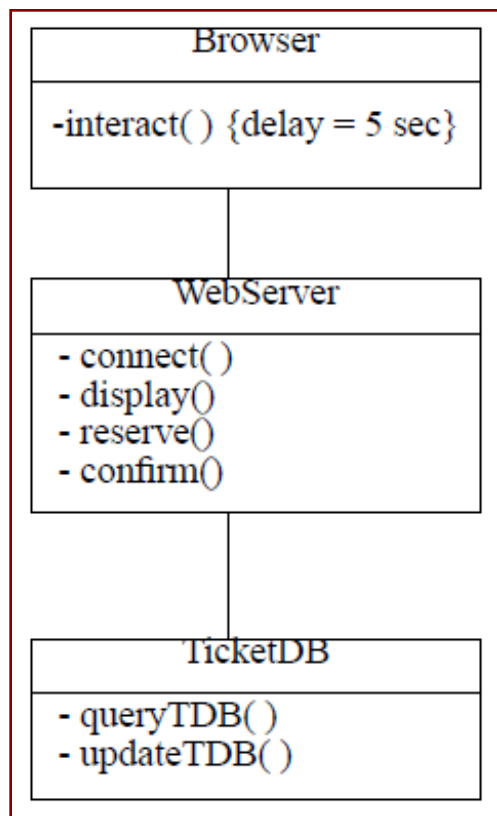
- Szinkronitás

Rendszerszinten számítható (átlagos és worst case) jellemzők:

- Kérés kiszolgálási idő
- Taszk áteresztőképesség
- Processzor kihasználtság



Példa: Az architektúra leképezése a teljesítmény modellre

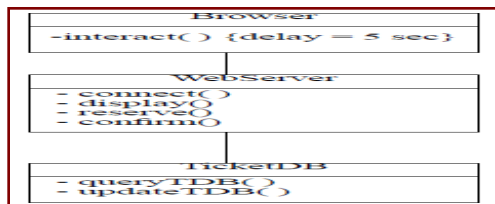


Osztályok és
lokális jellemzők

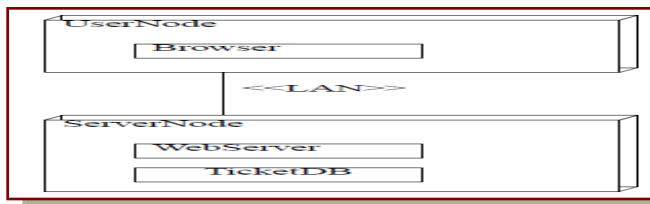
Processzorok
és telepítés

Interakciók
(hívások)

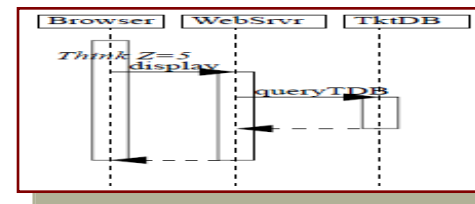
Példa: Az architektúra leképezése a teljesítmény modellre



Osztályok

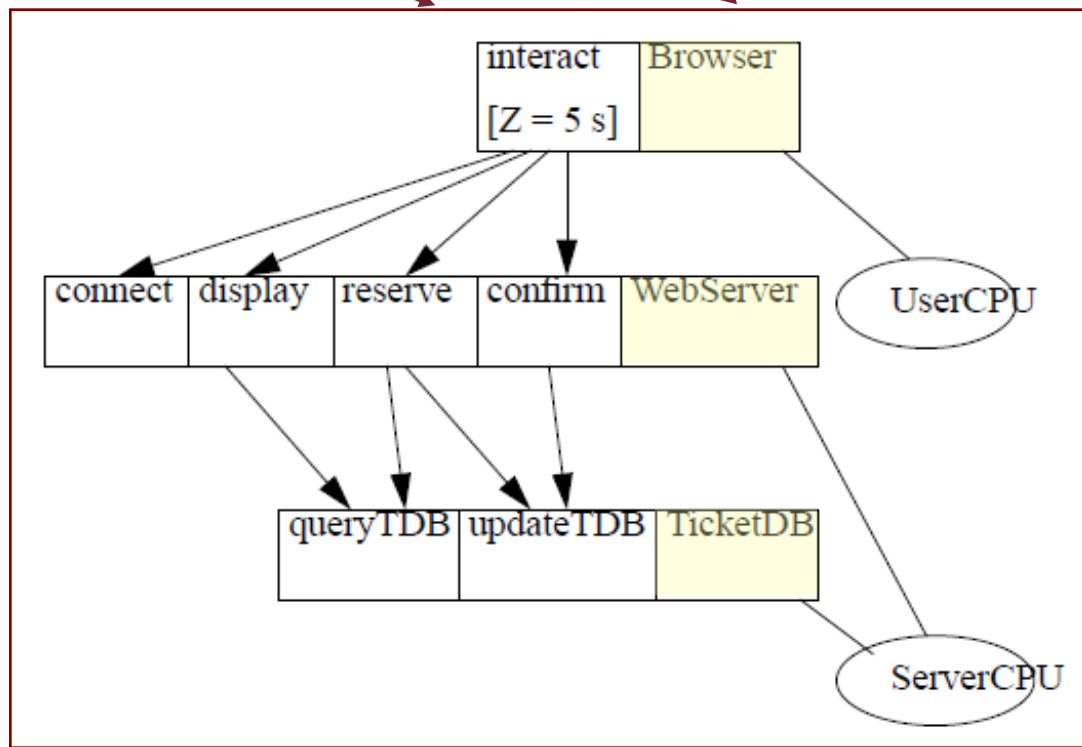


Telepítés



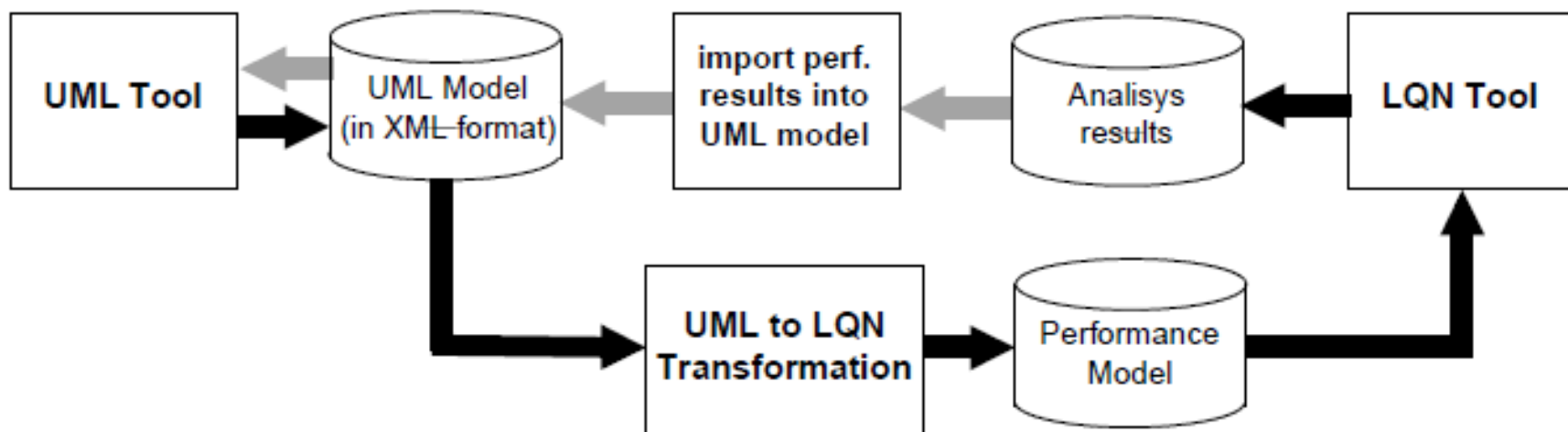
Interakciók

Modell-
transzformáció



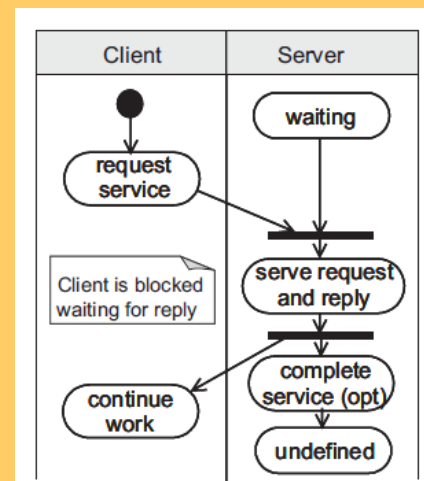
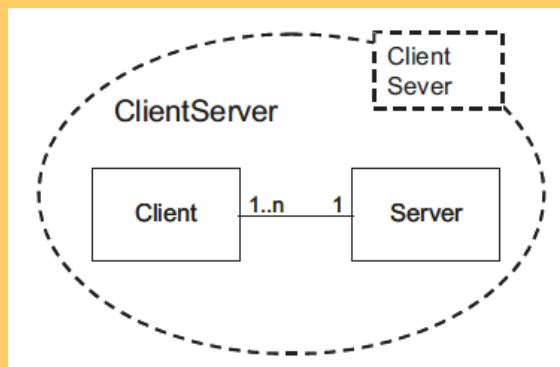
LQN
teljesítmény-
modell

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
 - Megbízhatósági modellezés
 - Teljesítmény modellezés
- **ATAM architektúra elemzés**

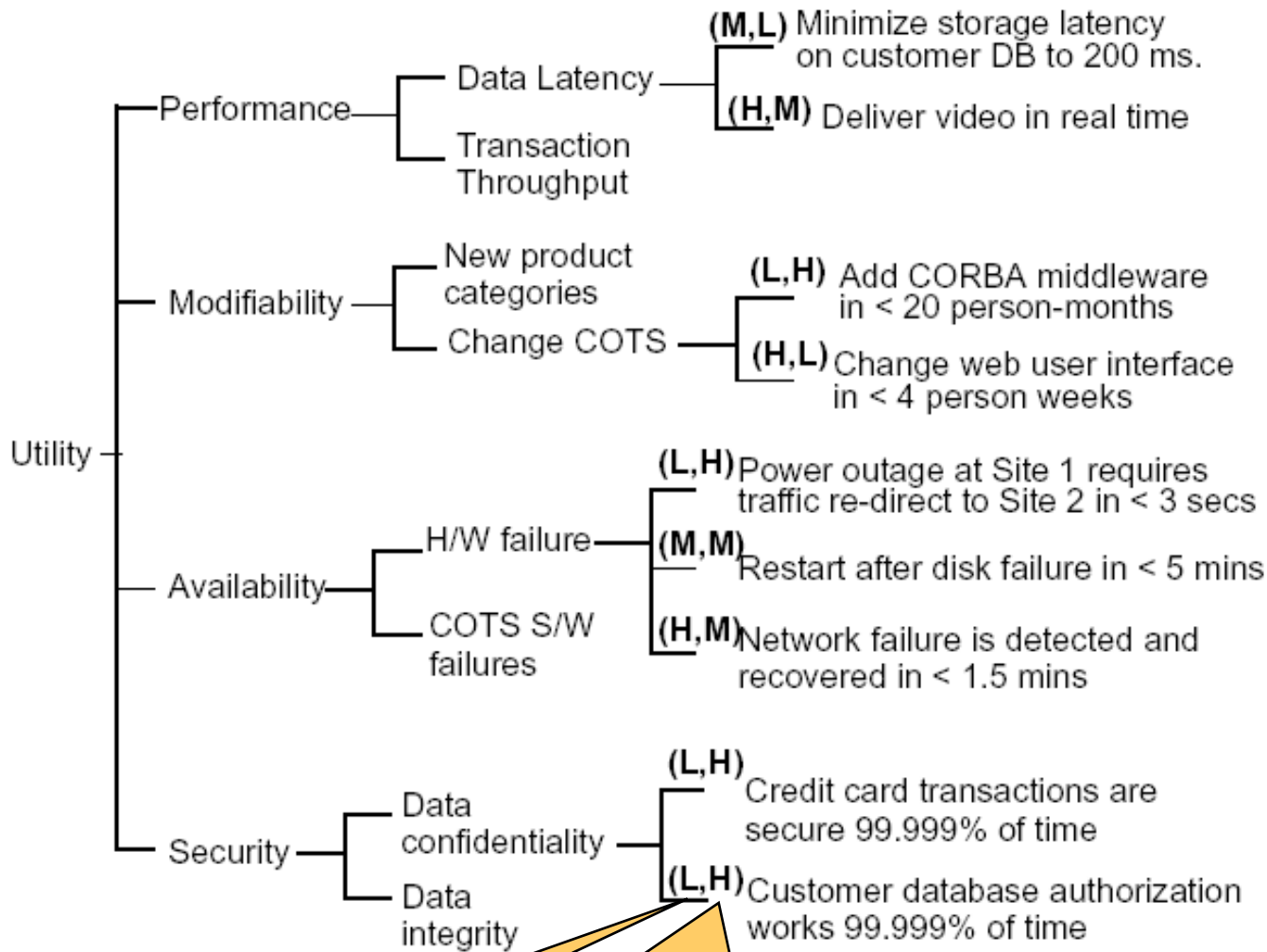
Architektúrák elemzése: ATAM

- Architecture Tradeoff Analysis Method (ATAM)
 - Milyen minőségi célok vannak?
 - Hogyan viszonyulnak egymáshoz a minőségi célok?
 - Hogyan elégíti ki az architektúra a minőségi 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 felismerése (pl. rétegek, redundancia, ...)
- **Hasznossági fa (utility tree) elkészítése** <- tervezők, elemzők
 - Gyökér: Általános megfelelés
 - 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 a **kockázatok**, erős és gyenge pontok?

Példa: Hasznossági fa



Fontosság
(Low, Medium, High)

Megvalósítási bonyolultság
(Low, Medium, High)

Példák: Az architektúra analízise

- A forgatókönyvek támogatása
 - Forgatókönyv: Diszk hiba esetén az adatok helyreállítása szükséges <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
- Tradeoff (kompromisszumok optimalizálása)
 - A replika adatbázis szinkronitása hatással van mind a rendelkezésre állásra, mind pedig a teljesítményre
 - Döntés: Aszinkron replika frissítés
- Kockázat
 - A replika frissítés kockázatot jelent, ha a teljesítményveszteségnek nagy a költsége
 - Adott várható terhelés esetén megfelelő a döntés

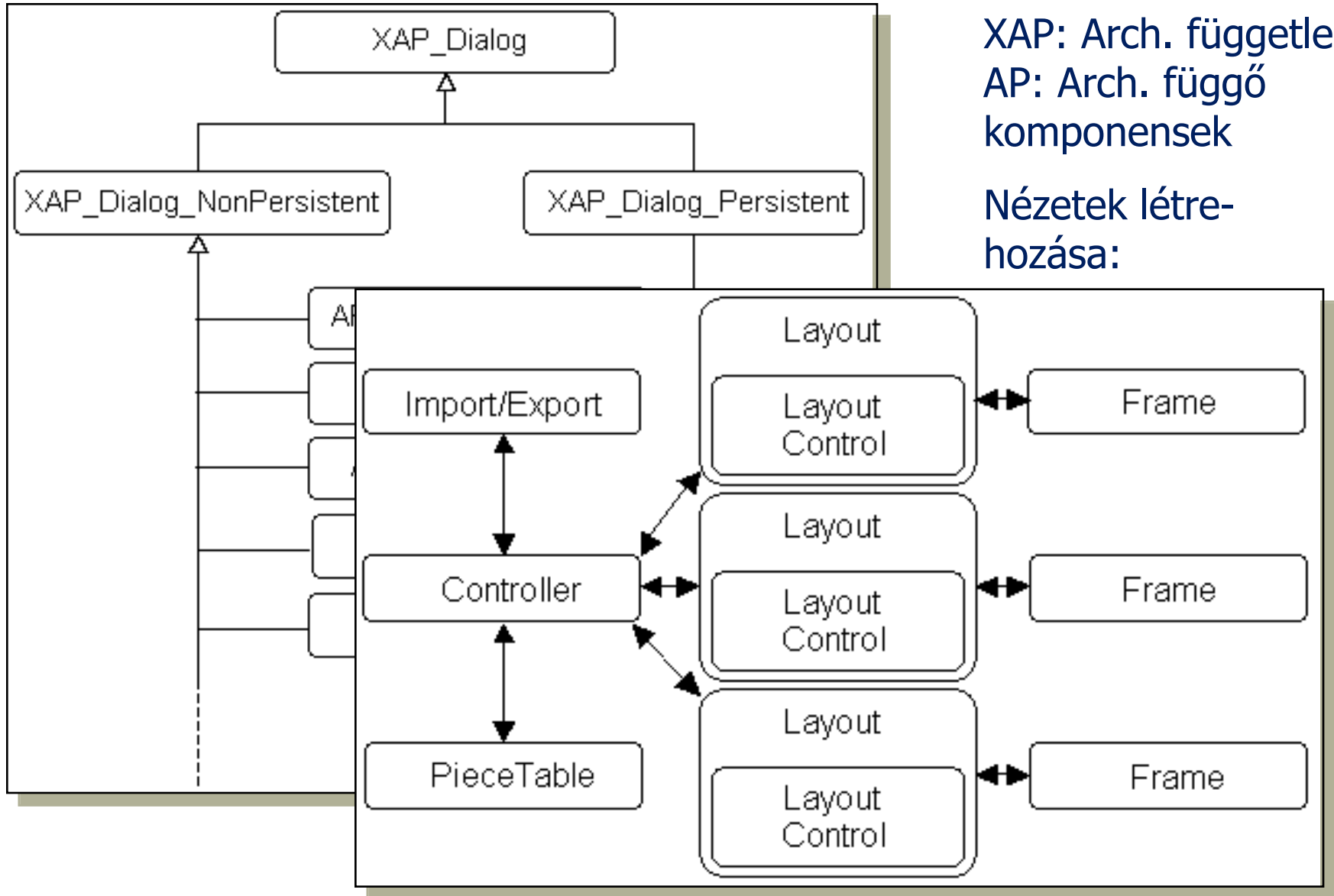
Kiértékelés a további érdekeltekkel

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

Mintapélda: AbiWord architektúra elemzése

- Célok
- Architektúra
- Hasznossági fa, forgatókönyvek
- Analízis

Mintapélda: Az architektúra

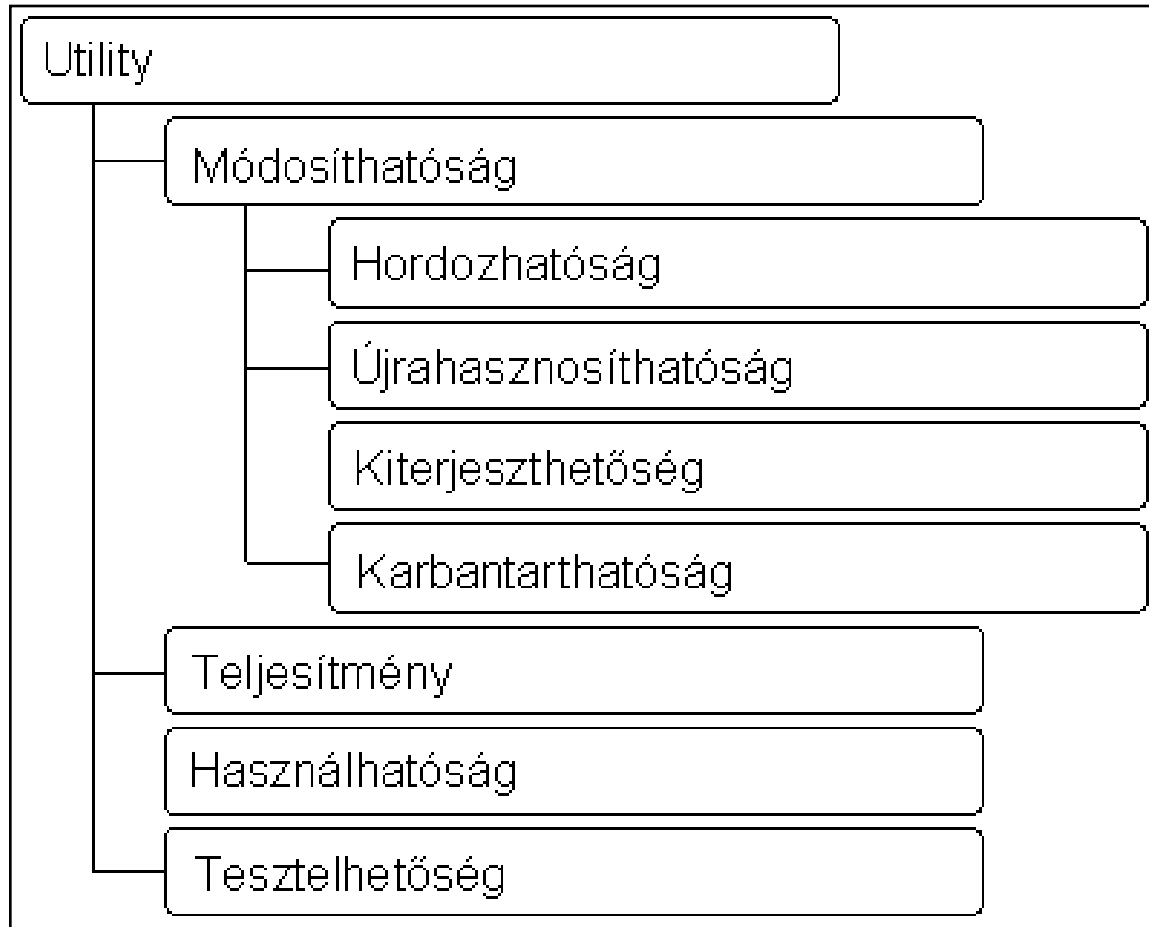


XAP: Arch. független,
AP: Arch. függő
komponensek

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

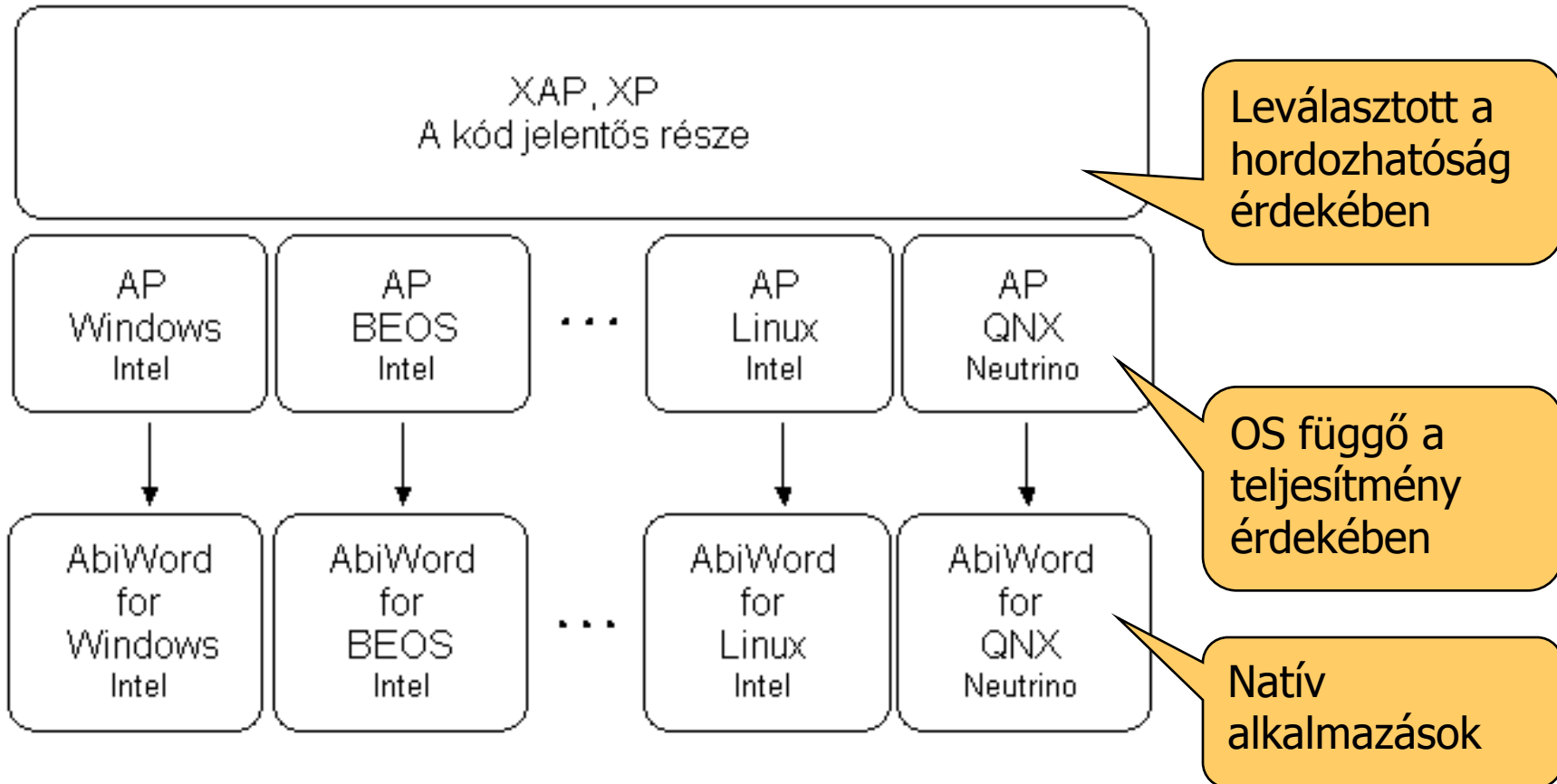
- **Bemeneti hatás:** A felhasználó HTML dokumentum készítése közben szeretné ellenőrizni az eredményt
- **Környezet:** Futási idő
- **Reakció:** A szoftver számos nézet lehetőségét biztosítja

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



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