

Szoftver architektúra tervek ellenőrzése

Majzik István

Budapesti Műszaki és Gazdaságtudományi Egyetem

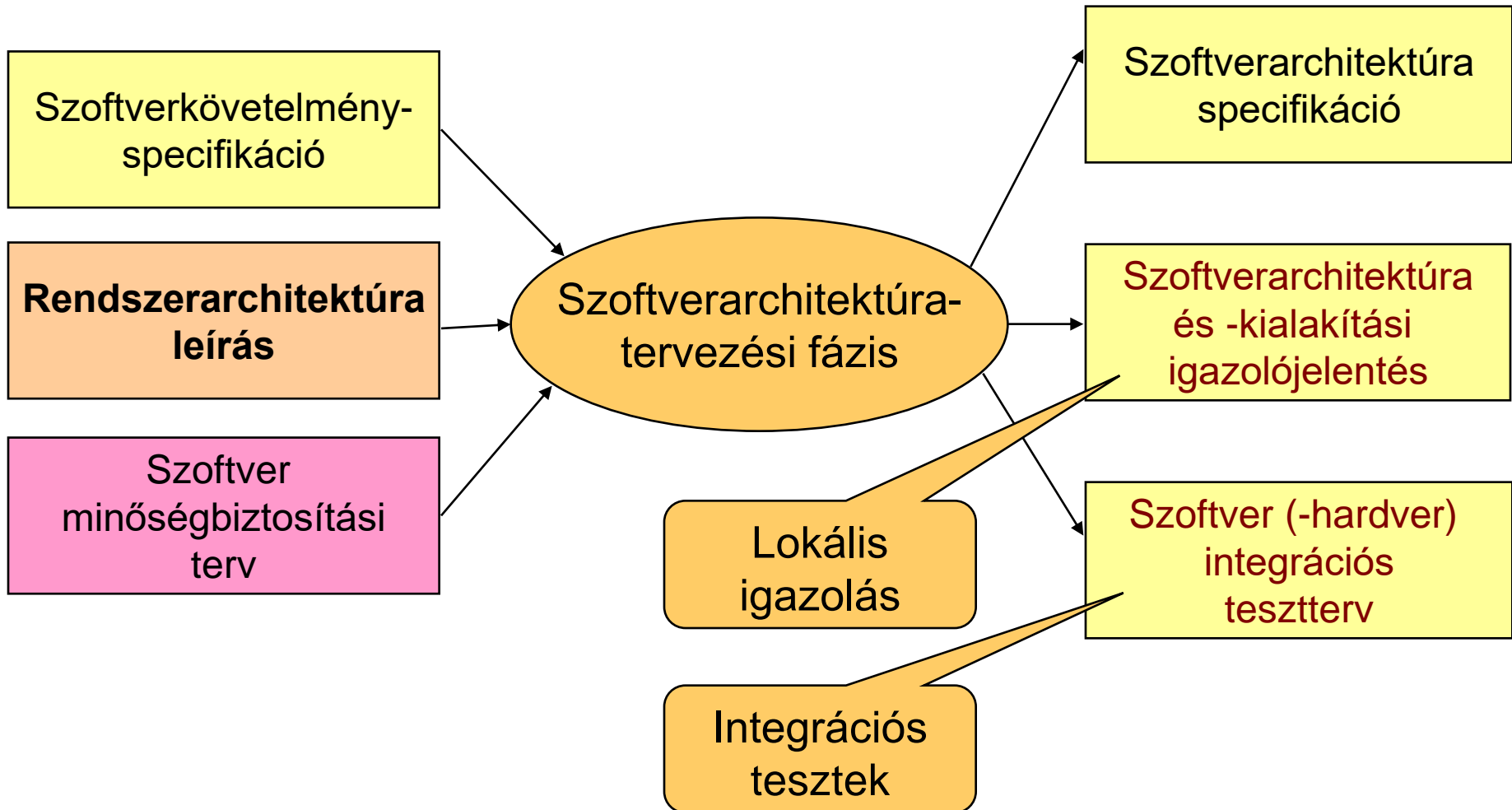
Méréstechnika és Információs Rendszerek Tanszék

<http://www.mit.bme.hu/~majzik/>

Tartalomjegyzék

- A fázis ki- és bemenetei
- Az architektúra tervek szerepe
 - Mit tartalmaz az architektúra terv?
 - Mik az ellenőrzési szempontok?
- Tipikus ellenőrzési feladatok
 - Követhetőségi elemzés
 - Szisztematikus statikus analízis
 - Kvantitatív analízis
 - Követelmények és architektúra összevetése
- Teszt tervezés: Ld. a tesztelési fázisban

Kimenetek és bemenetek



Tartalomjegyzék

- A fázis ki- és bemenetei
- Az architektúra tervek szerepe
 - Mit tartalmaz az architektúra terv?
 - Mik az ellenőrzési szempontok?
- Tipikus ellenőrzési feladatok
 - Követhetőségi elemzés
 - Szisztematikus statikus analízis
 - Kvantitatív analízis
 - Követelmények és architektúra összevetése
- Teszt tervezés: Ld. a tesztelési fázisban

Szoftverarchitektúra terv

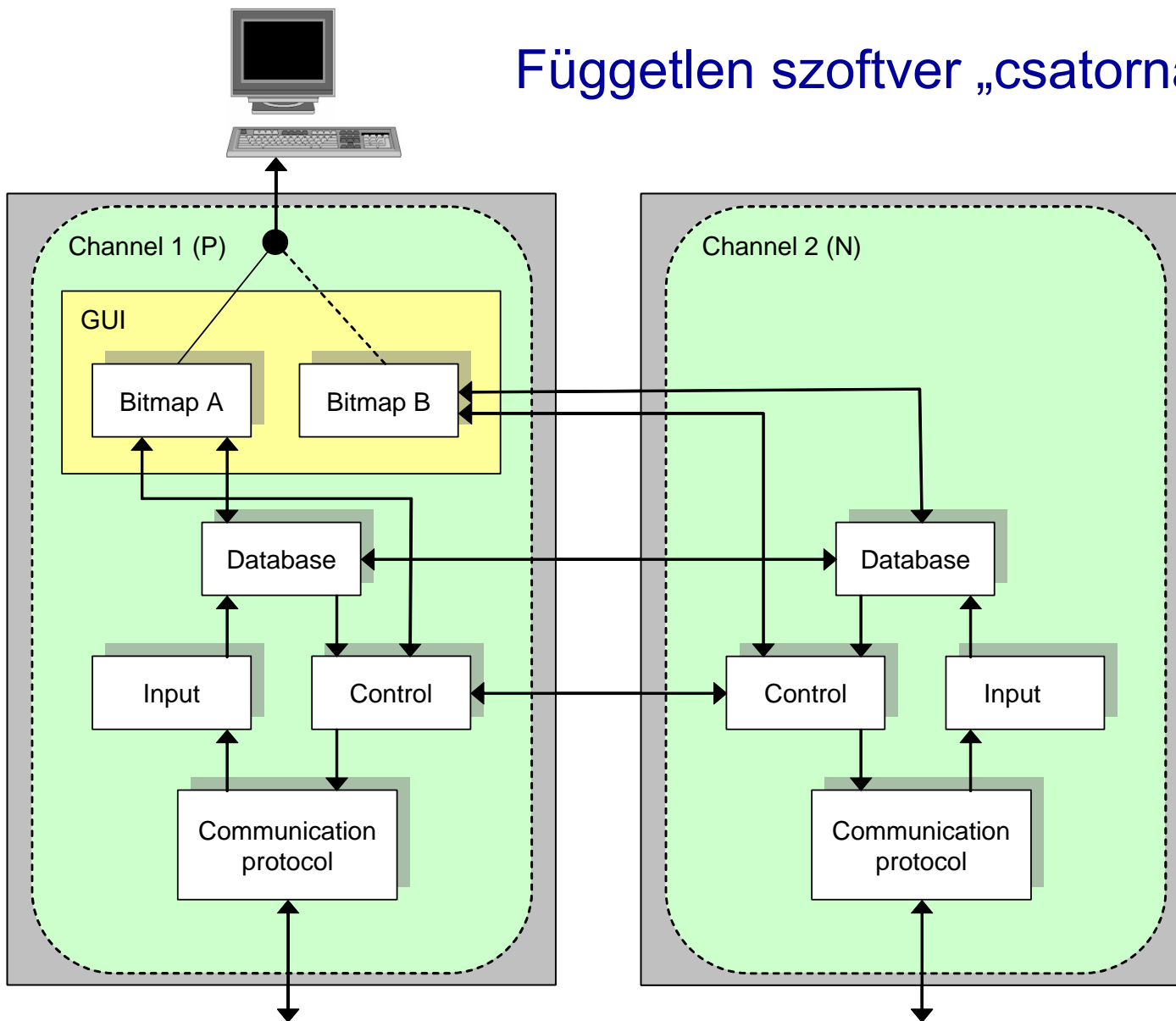
- Architektúra: Komponensek + közöttük lévő kapcsolatok
 - Hardver és szoftver komponensek
 - Kapcsolatok: pl. „uses the service of”, „deployed on”, „composed of”
- Tervezői döntések
 - Komponensek és kapcsolatok meghatározása
 - Hardver-szoftver együttműködés
 - Hibahatások figyelembe vétele
 - Méretezés: komponensek és kapcsolatok tulajdonságai
 - Teljesítmény, redundancia, biztonságosság, ...
 - Architektúra minták használata
 - Pl. redundancia, MVC, N-tier, ...
 - Újrafelhasználás (korábban fejlesztett, OTS komponensek)

Példa: OTS komponensek kritikus rendszerekben

- Korábban fejlesztett, érvényesített (validált) elemek: **alkalmasság igazolásával használhatók**
 - Szoftverkövetelményeknek való megfelelés
 - Azonos platform, kontextus
 - Változások esetén hatásvizsgálat
- Biztonságintegritási szintek
 - SIL 0: előfeltételek nélkül elfogadható az OTS komponens
 - SIL 1,2: **validációnak** tartalmaznia kell
 - SIL 3,4: validációnak tartalmaznia kell
 - + lehetséges **meghibásodások elemzése**
 - + **védelmi stratégia** és ennek tesztelése
 - + **hibanaplók** és kiértékelésük
- Szoftver modulok SIL szintje:
 - Alapértelmezés: Egyező a rendszer(modulok) SIL szintjével
 - **Csökkentés:** Van olyan mechanizmus, amely megakadályozza, hogy a szoftver meghibásodása a rendszer nem biztonságos állapotba kerülését okozhassa

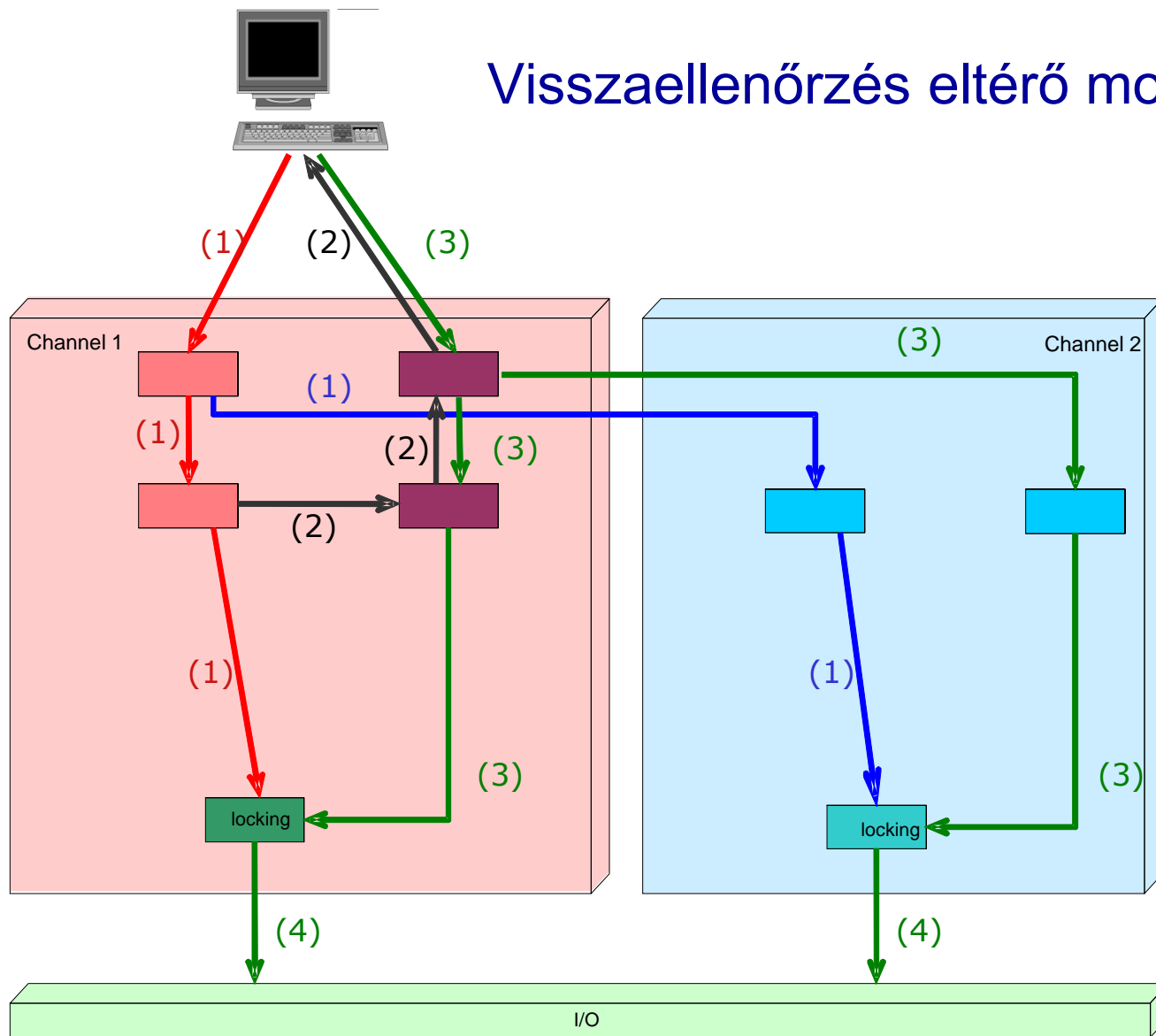
Példa: Architektúrális megoldás SIL csökkentésre

Független szoftver „csatornák”



Példa: Architektúrális megoldás SIL csökkentésre

Visszaellenőrzés eltérő modulokon



Jellegzetes architektúra minták: Redundancia

- **Hardver redundancia:**
 - Azonos komponensek:
Működés közbeni tranziens és állandósult hibák detektálása és (diagnosztika után) hibakezelés
 - Két komponens: Hibadetektálásra alkalmas, diagnosztika szükséges
 - Kettőnél több komponens (NMR): Hiba maszkolható (szavazás)
- **Szoftver redundancia:**
 - Eltérő tervezés („diverz programozás”):
A szisztematikus **szoftver tervezési hibák** detektálása és kezelése
 - Aktív redundancia: N-verziós programozás (NVP)
 - Passzív redundancia: Javító blokkok (RB)
 - Adaptív redundancia: Önkonfiguráló programozás (SCOP)
- **Információ redundancia:**
 - Hibafelismerő illetve hibajavító kódolás
- **Idő redundancia:**
 - Újrapróbálás: Tranziens hibák esetén lehet hatásos
 - Közvetett idő redundancia más technikák esetén is

Példa: Architektúra előírt megoldásai (EN50128)

- 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

Mit határoz meg az architektúra?

Elérendő tulajdonság	Tervezési tér (releváns döntések)
Teljesítmény	Erőforrás hozzárendelés, erőforrás menedzsment
Szolgáltatásbiztonság	Hibadetektálás, hibabehatárolás, hibakezelés (redundancia)
Biztonságosság	Veszély csökkentés, veszély kontroll (monitorozás)
Adatbiztonság	Integritás ellenőrzés, behatolás felderítés, helyreállítás
Tesztelhetőség	Vezérelhetőség, megfigyelhetőség
Karbantarthatóság	Elkülönítés (pl. MVC minta)

Tartalomjegyzék

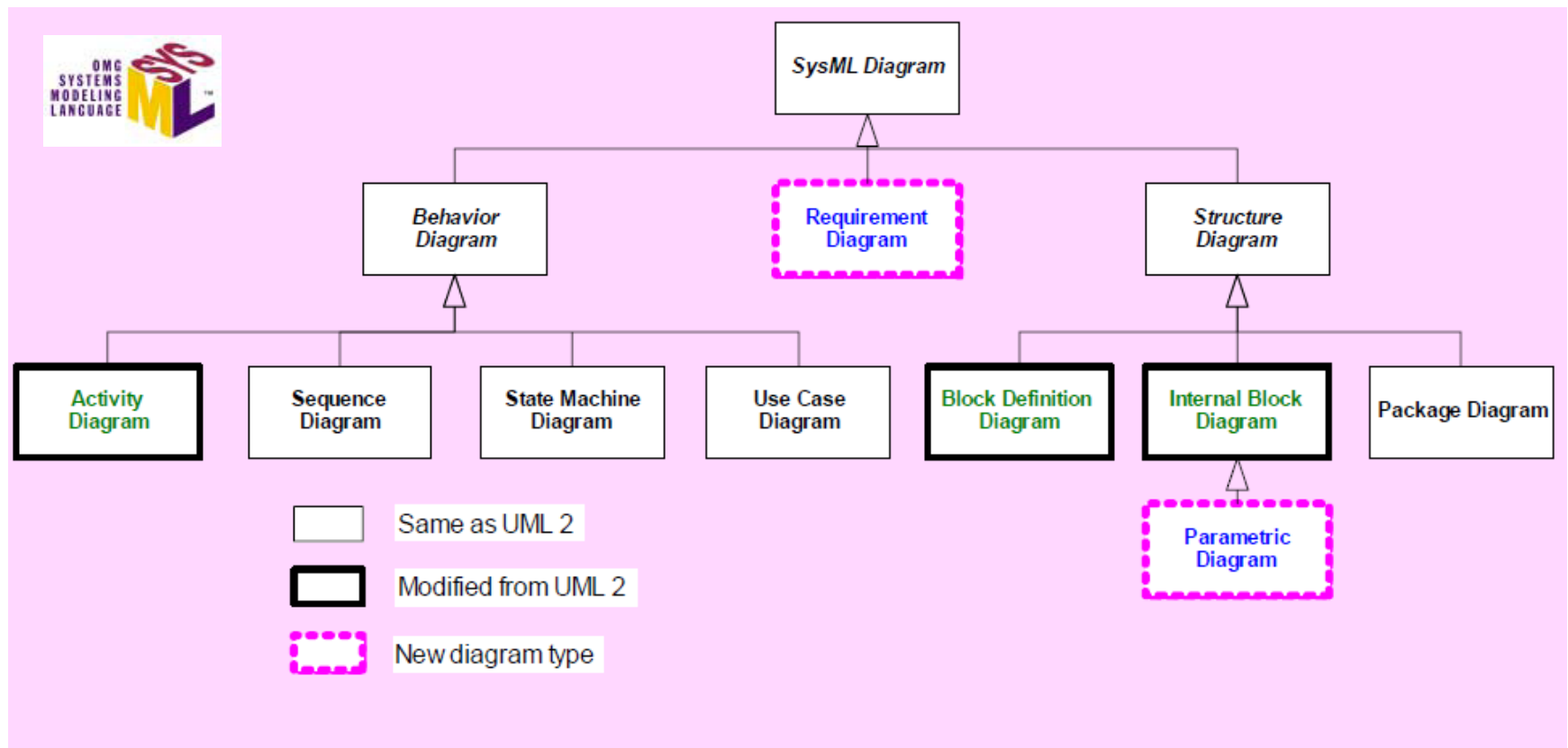
- A fázis ki- és bemenetei
- Az architektúra tervek szerepe
 - Mit tartalmaz az architektúra terv?
 - Mik az ellenőrzési szempontok?
- Tipikus ellenőrzési feladatok
 - Követhetőségi elemzés
 - Szisztematikus statikus analízis
 - Kvantitatív analízis
 - Követelmények és architektúra összevetése
- Teszt tervezés: Ld. a tesztelési fázisban

A szoftverarchitektúra ellenőrzési technikái (áttekintés)

- Követhetőségi elemzés
 - Követelmények lefedésének vizsgálata
- Statikus analízis: architektúra „bejárás”
 - Interfész analízis
 - Elvárt és nyújtott interfészek megfeleltetése
 - Hibahatás analízis kombinatorikus módszerekkel
 - Komponens szintű hibák ↔ Rendszerszintű hatások
- Kvantitatív analízis: modell alapú vizsgálatok
 - Extra-funkcionális jellemzők analízise sztochasztikus analízis modell készítésével
 - Lokális (komponens illetve kapcsolati szintű) paraméterek alapján rendszerszintű jellemzők számítása
- Követelmények és architektúra összevetése:
 - Architektúra trade-off analízis (ATAM)

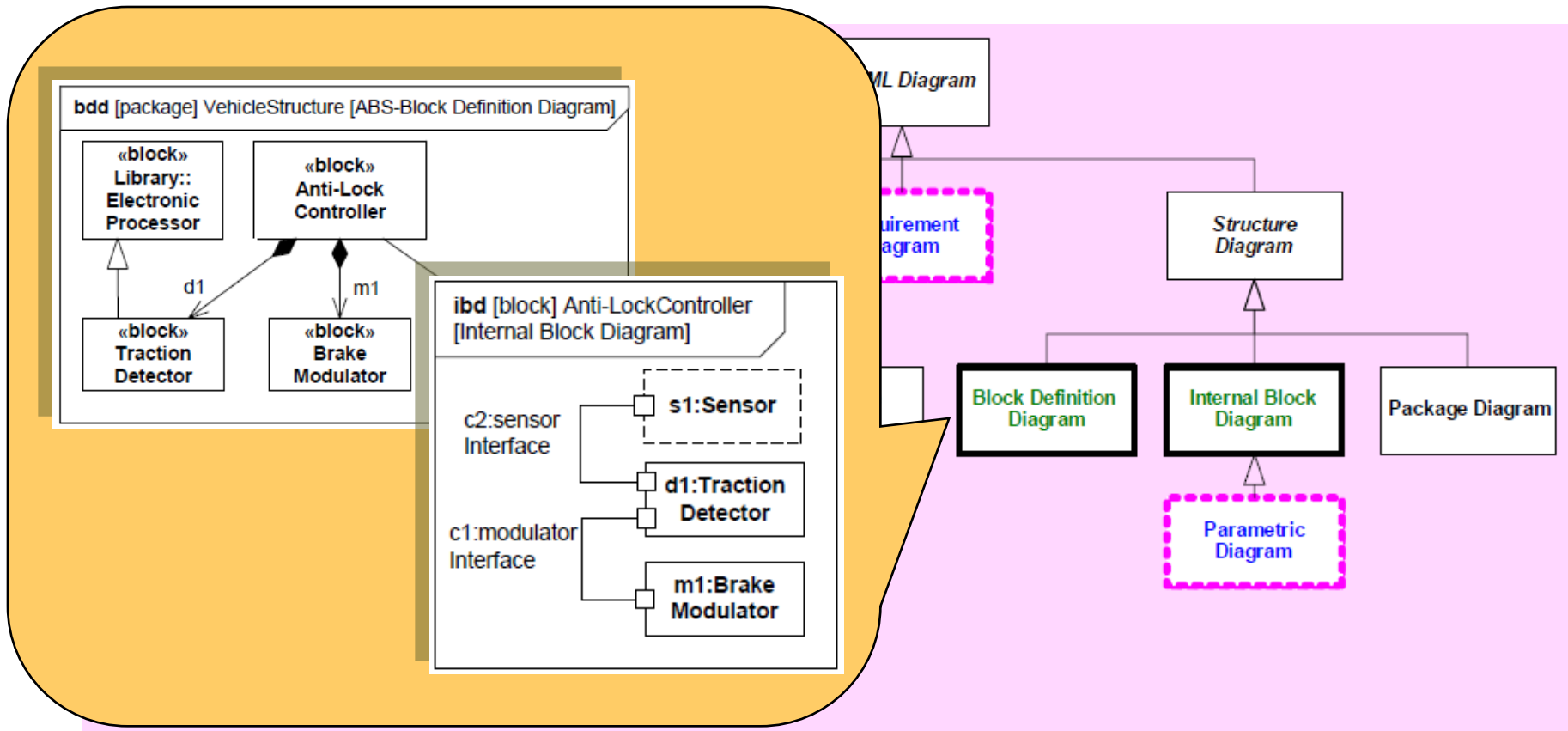
Az architektúra terv igazolása: Követhetőség

- Hogyan segítik ezt a félformális nyelvek?
- Példa: SysML (Systems Modelling Language)
 - Architektúra tervezés első lépése: Block diagram



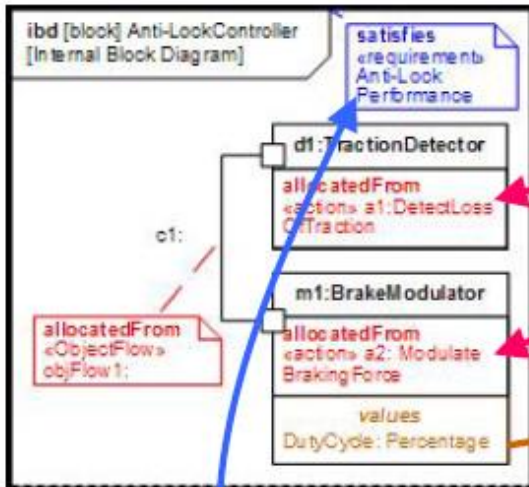
Az architektúra terv igazolása: Követhetőség

- Hogyan segítik ezt a félformális nyelvek?
- Példa: SysML (Systems Modelling Language)
 - Architektúra tervezés első lépése: Block diagram

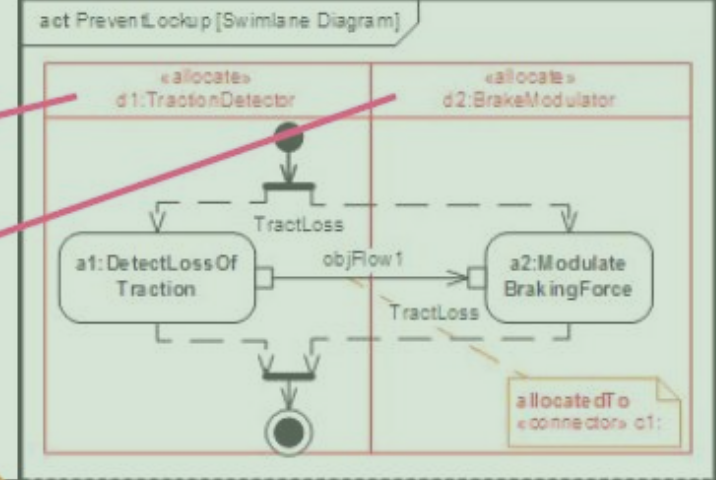


Relációk explicit megjelenítése és ellenőrzése

1. Structure



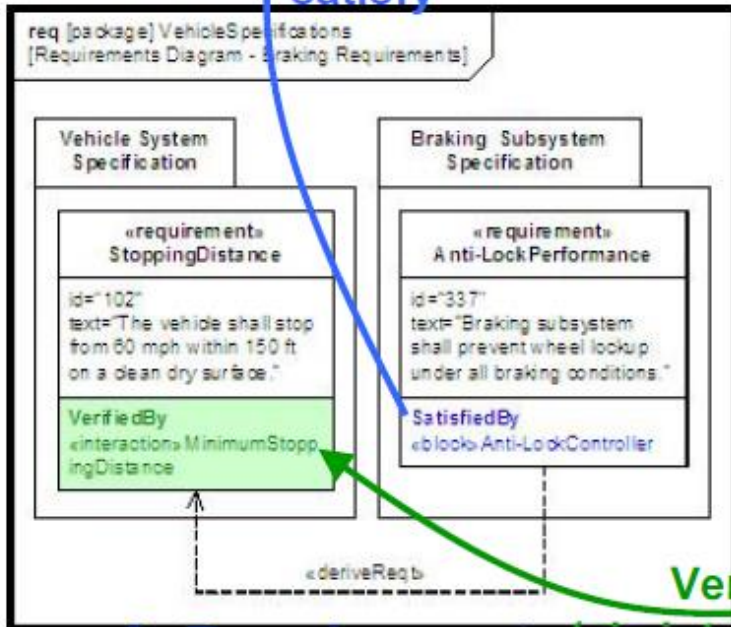
2. Behavior



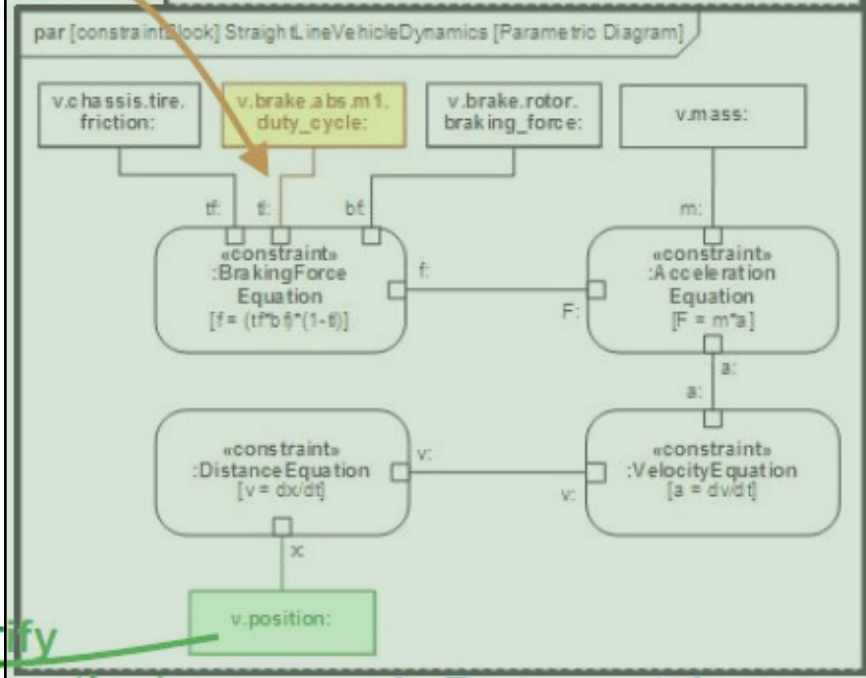
allocate

value binding

satisfy



Verify



3. Requirements (via interaction)

4. Parametrics

Statikus analízis: Interfész analízis

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

Statikus analízis: 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)

1. Hibafa analízis

Rendszerszintű veszély okainak vizsgálata

- Tipikusan felülről lefelé haladó analízis
- Felderíti a kezelendő hibaokokat és -kombinációkat

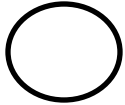
Hibafa konstrukció:

- (Rendszerszintű) veszély, veszélyes állapot:
Azonosítás: környezet, követelmények, szabványok
- Közbenső események, pseudo-események:
Veszélyhez vezetnek,
Boole-logikai kombinációi (AND, OR) alacsonyabb szintű eseményeknek
- Elsődleges események: további felbontás nincs

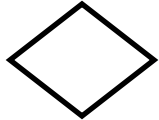
Hibafa grafikus elemkészlet



legfelső szintű vagy közbenső esemény



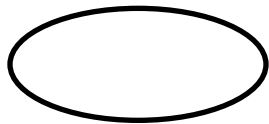
elsődleges (alapszintű) esemény



tovább nem vizsgált esemény



normál esemény (nem hiba vagy veszély)



feltétel egy összetett esemény
bekövetkezéséhez

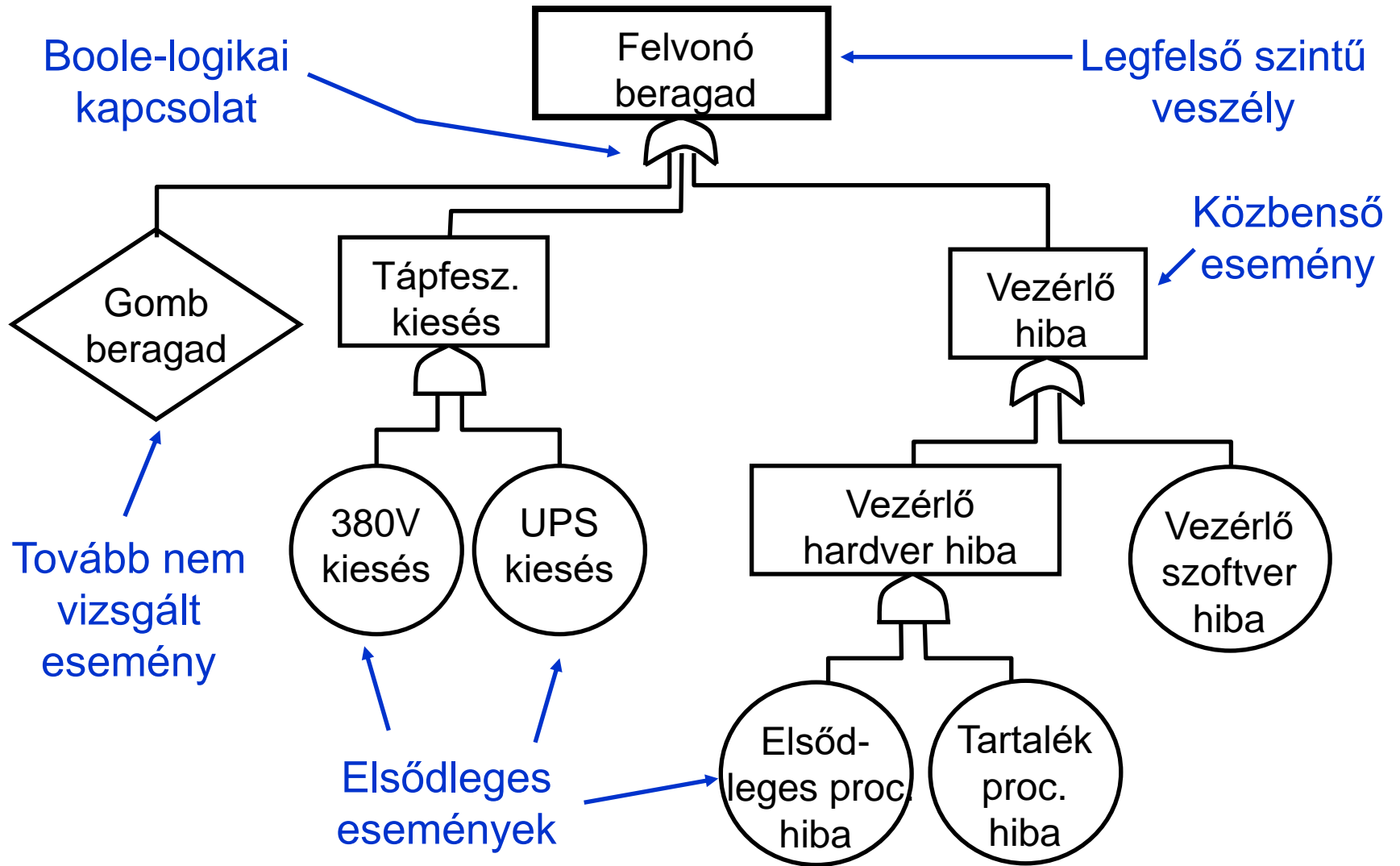


ÉS kapu



VAGY kapu

Hibafa példa: Felvonó



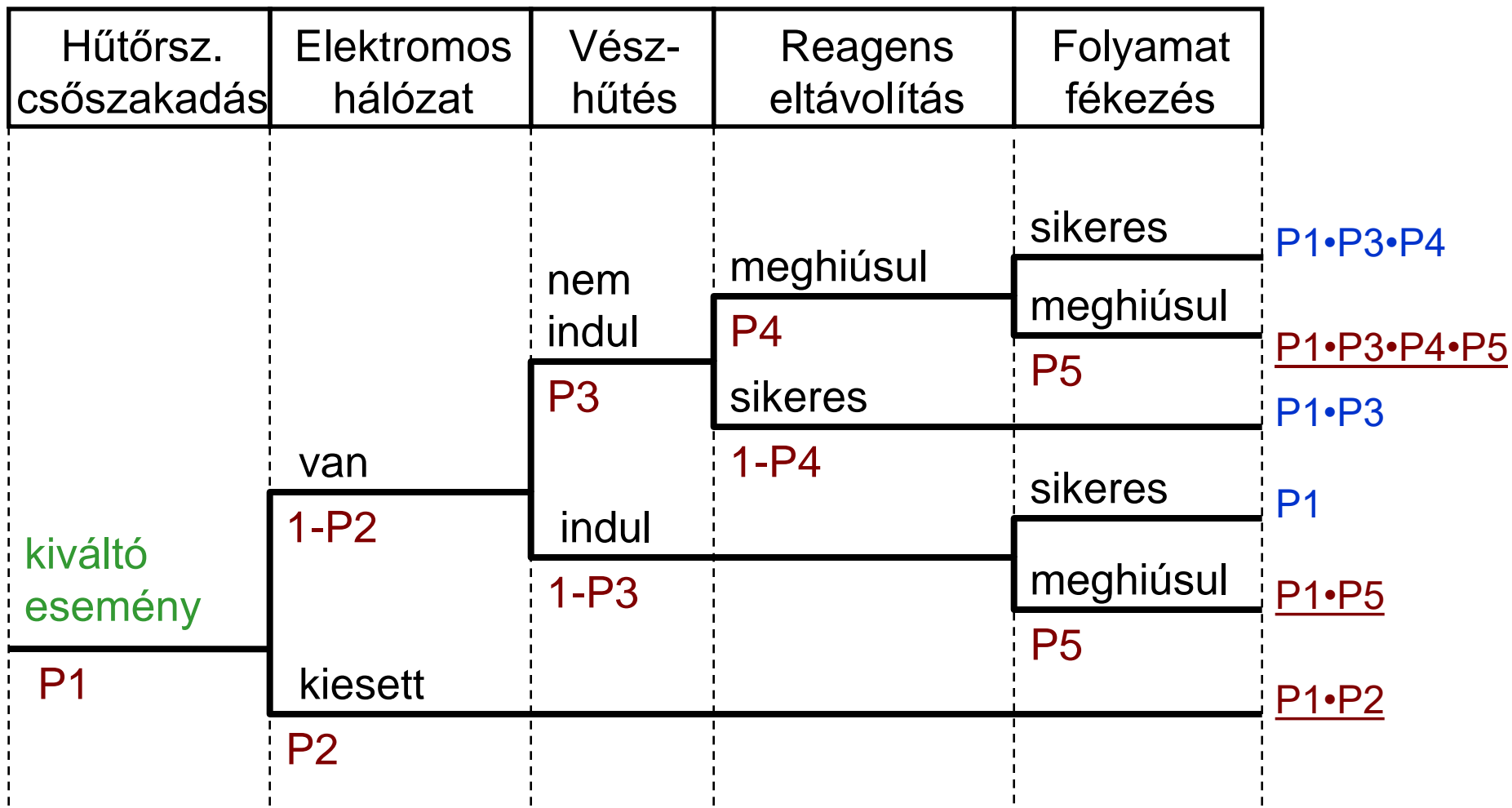
Hibafa analízis

- **Minőségi (kvalitatív) analízis:**
 - Hibafa **redukció**: Közbenső események feloldása
→ 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

2. Eseményfa analízis

- **Előrelépő analízis:**
Elsődleges események következményeit vizsgálja
 - Kiváltó esemény: pl. egy komponens hibája
 - Következmények: más komponensek állapotától függ
 - Sorrendezés: oksági kapcsolat, időbeli viszony
 - Elágazások: események bekövetkezése
- **Hibázási „forgatókönyvek” vizsgálata**
 - Utak valószínűsége (elágazások valószínűsége alapján)
 - Védelmi rendszerek hatékonysága
- **Előnyök: Eseményszekvenciák vizsgálhatók**
 - Korlátok: Komplexitás, többszörös események, minden kiváltó eseményhez külön diagram

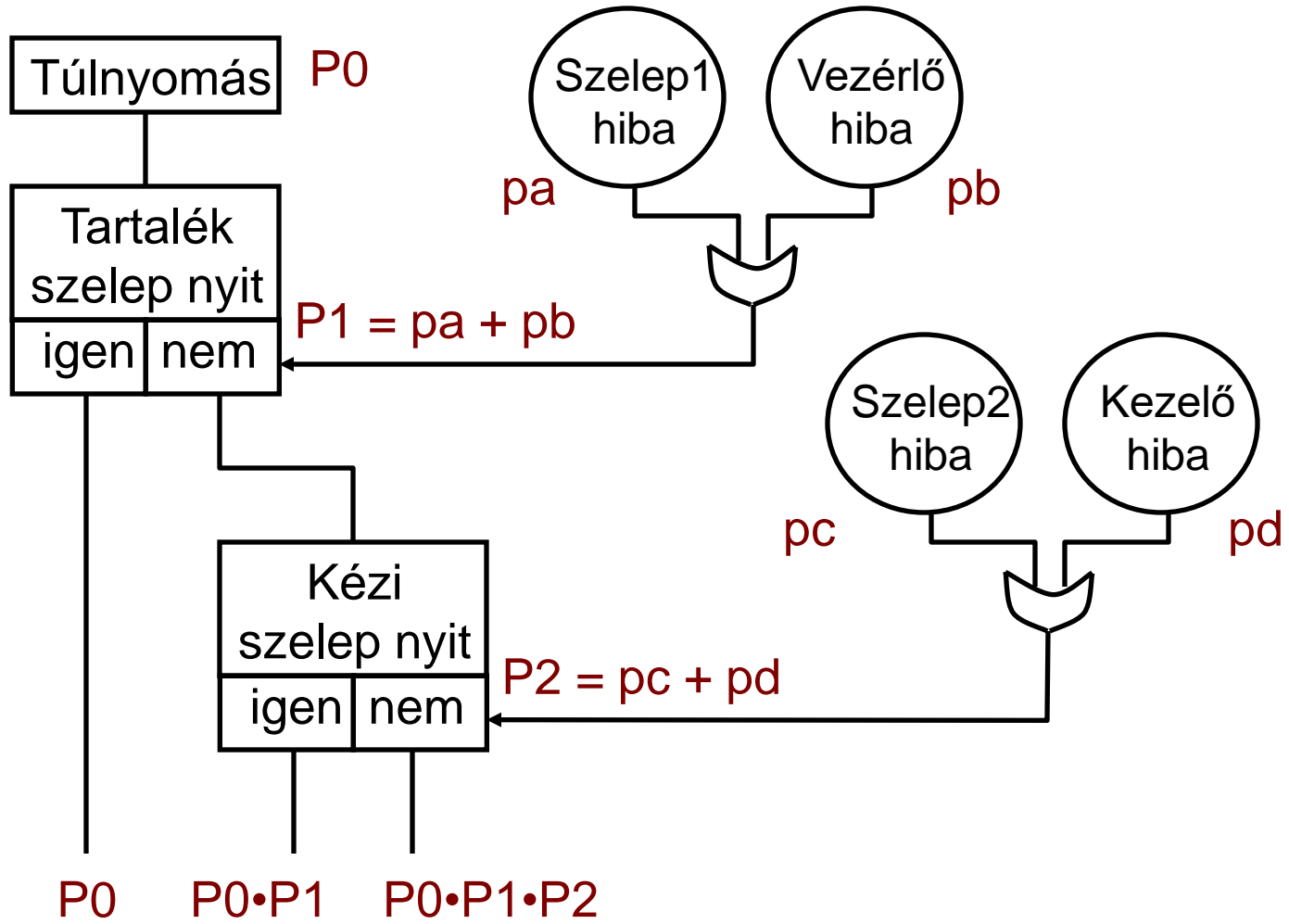
Eseményfa példa: Reaktorhűtés



3. Ok-következmény analízis

- **Eseményfa és hibafa összekapcsolása**
 - Eseményfa: forgatókönyvek (szekvencia)
 - Csatolt hibafa: esemény bekövetkezés „indoklása”, rendelkezésre állás számítása
- **Előnyök:**
 - Szekvenciák (előrelépő analízis) és ok-okozati kapcsolatok (hátralépő analízis) együtt
- **Korlátok:**
 - Minden kritikus eseményhez külön diagram szükséges

Példa ok-következmény analízisre



4. Hibamód és -hatás analízis (FMEA)

- Hibák és hatásaik felsorolása
- Előny:
 - Szisztematikus áttekintés
 - Redundancia felismerése

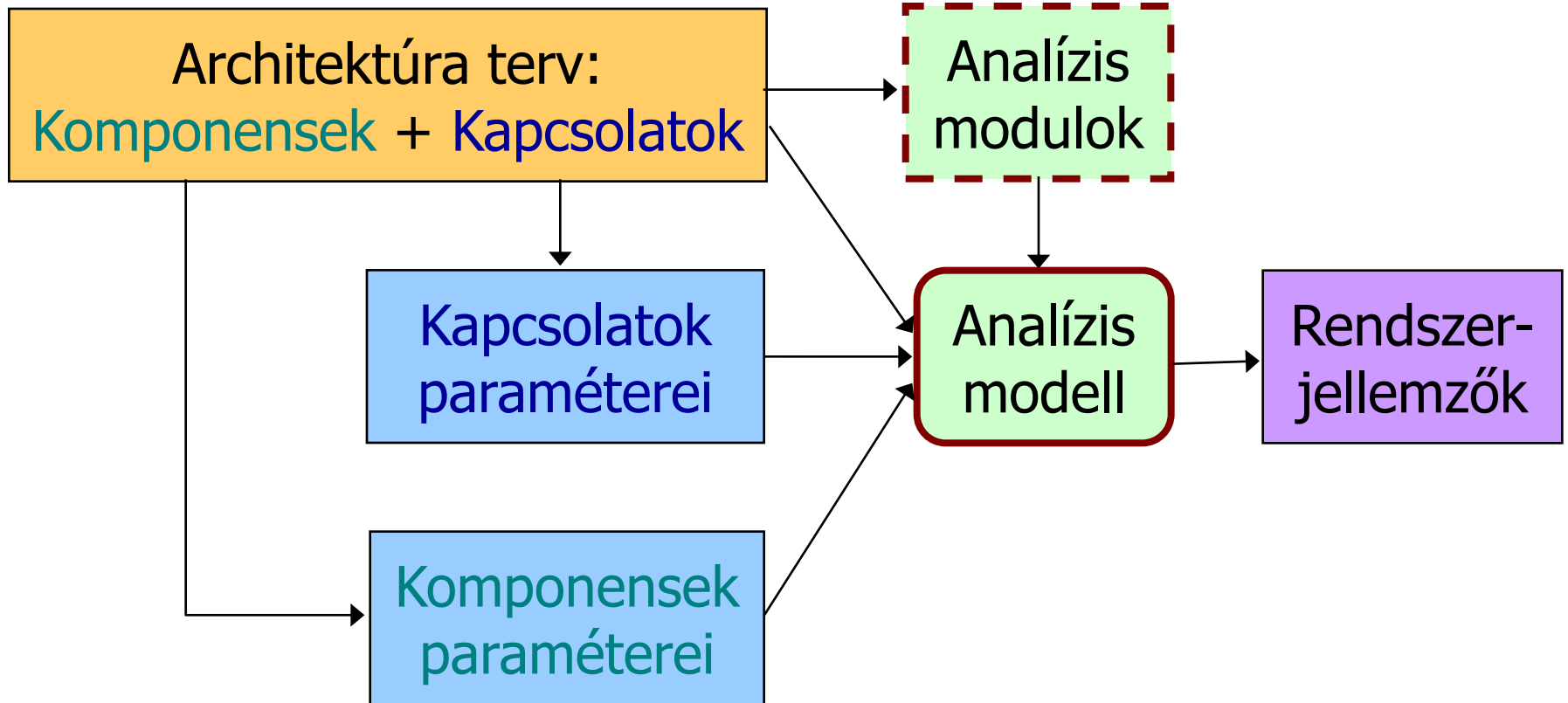
Komponens	Hibamód	Valószínűség	Hatás
L határérték-túllépés vizsgálat	$> L$ átmegy $\leq L$ nem megy át	65% 35%	- túlnyomás - technológiai hiba
...

A modell alapú kvantitatív analízis módszerei

Cél: Architektúra változatok kiértékelése

- **Analízis modellek készítése és paraméterezése az architektúra modellje alapján**
 - Matematikai modell, jellemzői számíthatók („megoldható” a modell)
- **Mit ír le az analízis modell?**
Komponens paraméterek -> Rendszerszintű jellemzők
 - Teljesítmény
 - Megbízhatóság
 - Biztonság
 - Adatbiztonság
- **Moduláris modellalkotás a tipikus**
 - **Architektúra:** Komponensek és kapcsolatok
 - **Analízis modell:** Ehhez analízis modell modulok

Modell alapú vizsgálatok



Modell alapú architektúra vizsgálatok áttekintése

	Teljesítmény modellezés	Megbízhatósági modellezés	Biztonsági modellezés
Komponens paraméterek	Funkció lokális végrehajtási idő, taszk prioritás, processzor 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, taszk áteresztőképesség, processzor kihasználtság	Megbízhatóság, rendelkezésre állás, készenlét, MTTF, MTTR, MTBF	Rendszerszintű veszély gyakoriság

Első példa: Teljesítmény modellezés

	Teljesítmény modellezés	Megbízhatósági modellezés	Biztonsági modellezés
Komponens paraméterek	Funkció lokális végrehajtási idő, taszk prioritás, processzor 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, taszk áteresztőképesség, processzor kihasználtság	Megbízhatóság, rendelkezésre állás, készenlét, MTTF, MTTR, MTBF	Rendszerszintű veszély gyakoriság

Teljesítmény modellezés (LQN)

Taszk (szerver):

- Funkciók (hívási pontok)
- Prioritás
- Processzor ütemezés

Funkció:

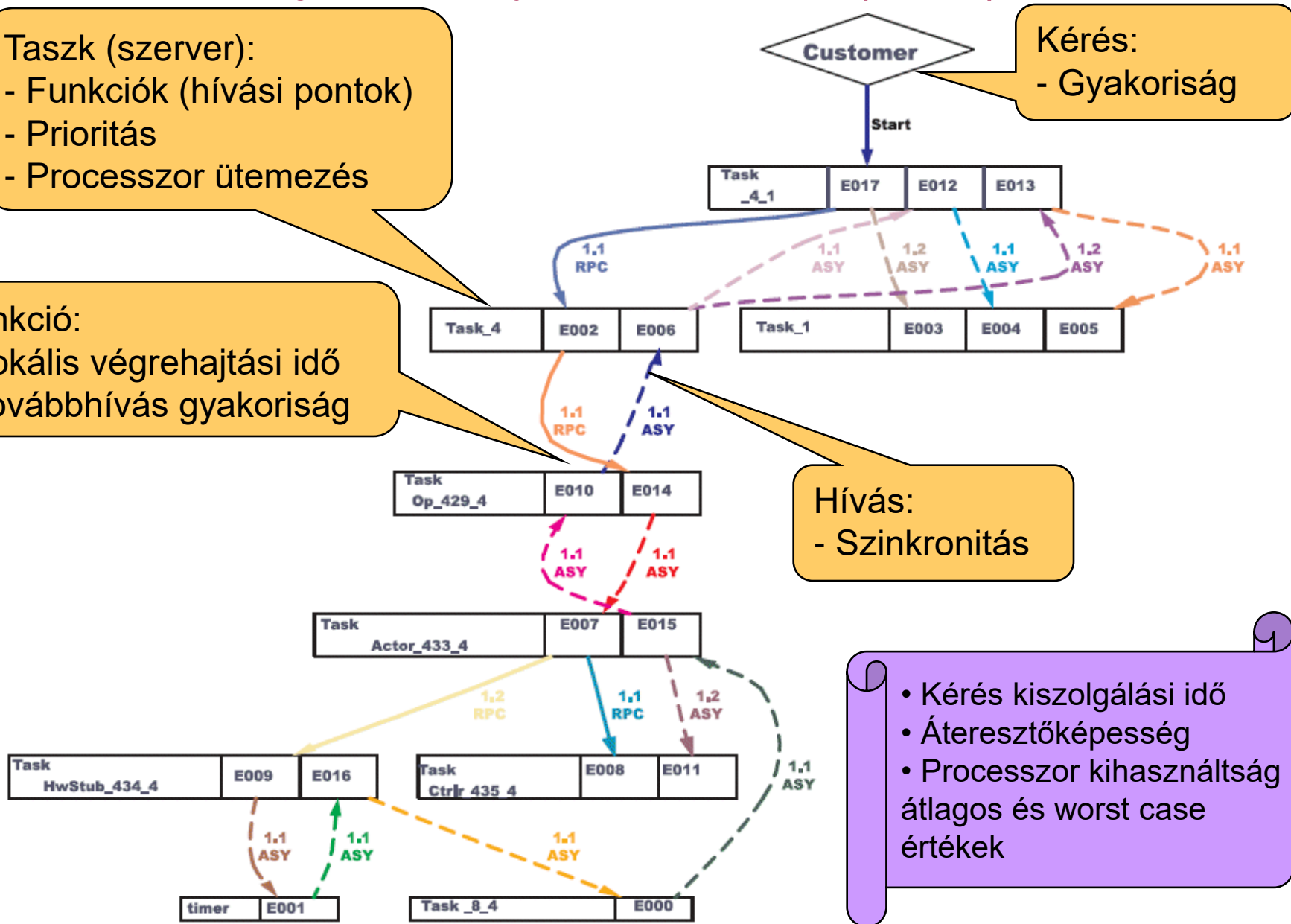
- Lokális végrehajtási idő
- Továbbhívás gyakoriság

Kérés:

- Gyakoriság

Customer

Start

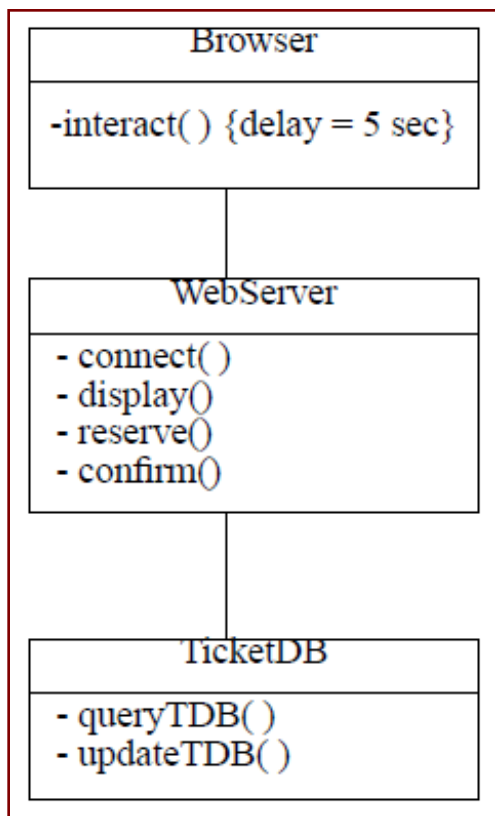


Hívás:

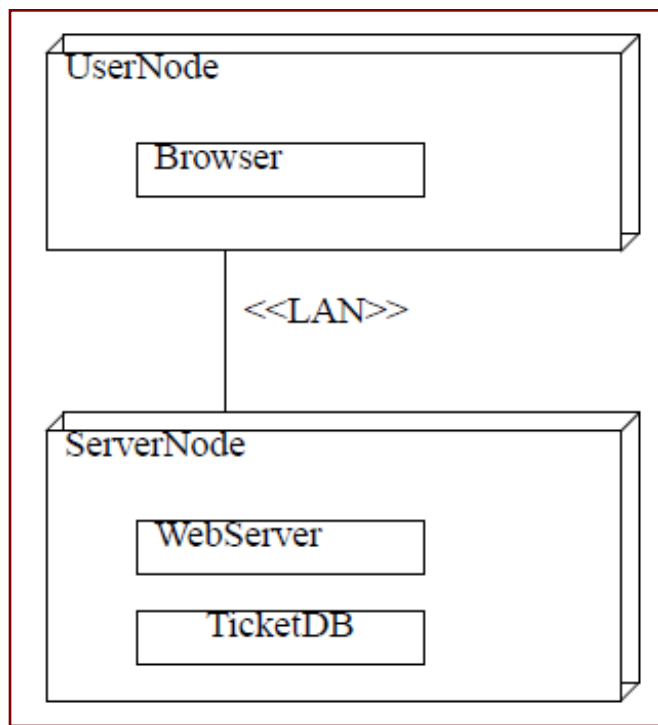
- Szinkronitás

- Kérés kiszolgálási idő
- Áteresztőképesség
- Processzor kihasználtság
átlagos és worst case értékek

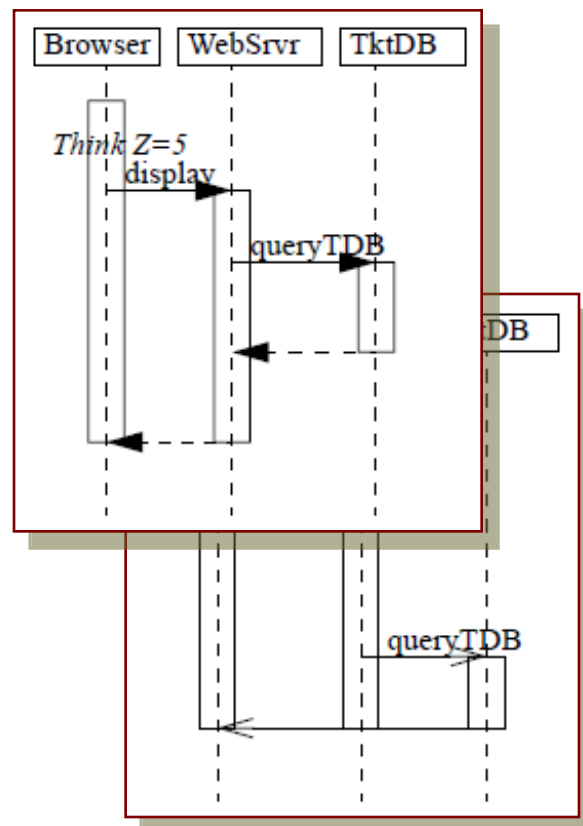
Az architektúra leképezése teljesítménymodellekre



Osztályok



Telepítés

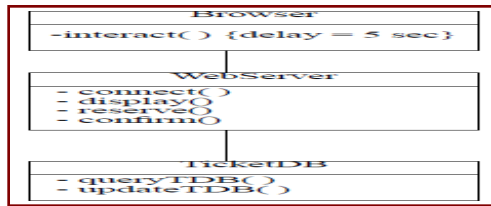


Interakciók

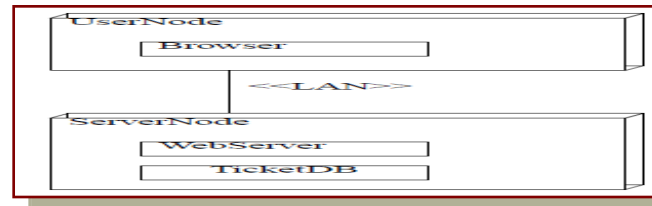
Lokális teljesítmény paraméterek megadása:

- Attribútumok (konvenció alapján értelmezhető)
- UML tagged value

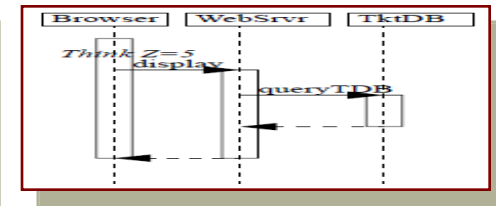
Az architektúra leképezése teljesítménymodellekre



Osztályok

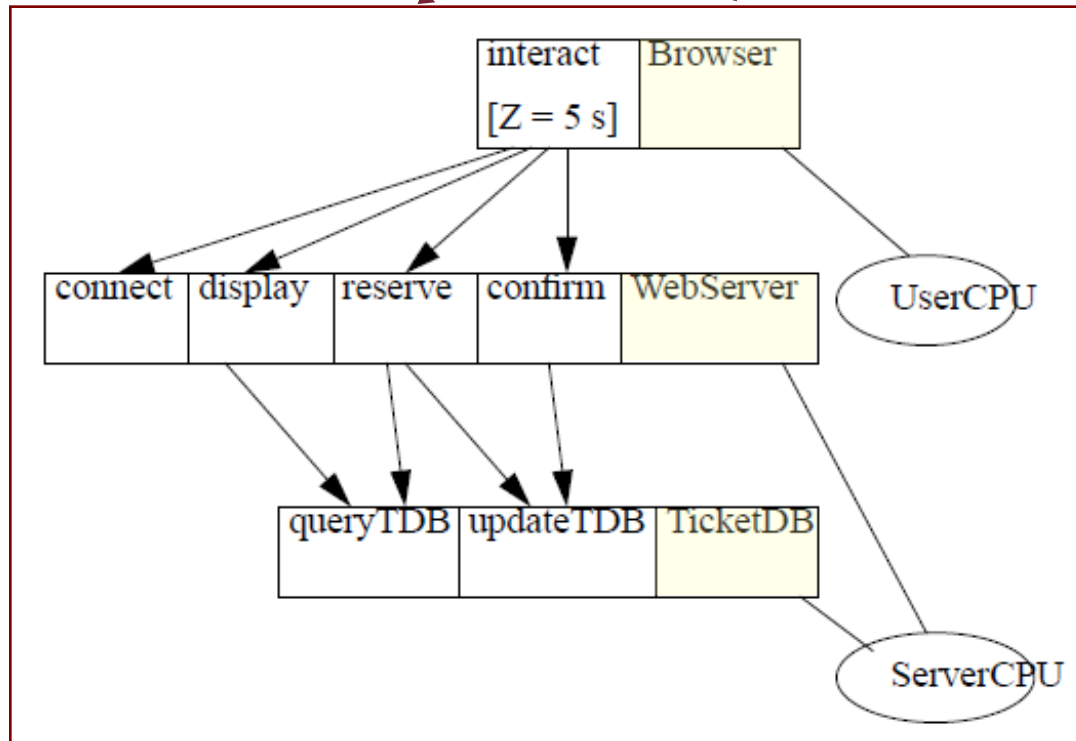


Telepítés



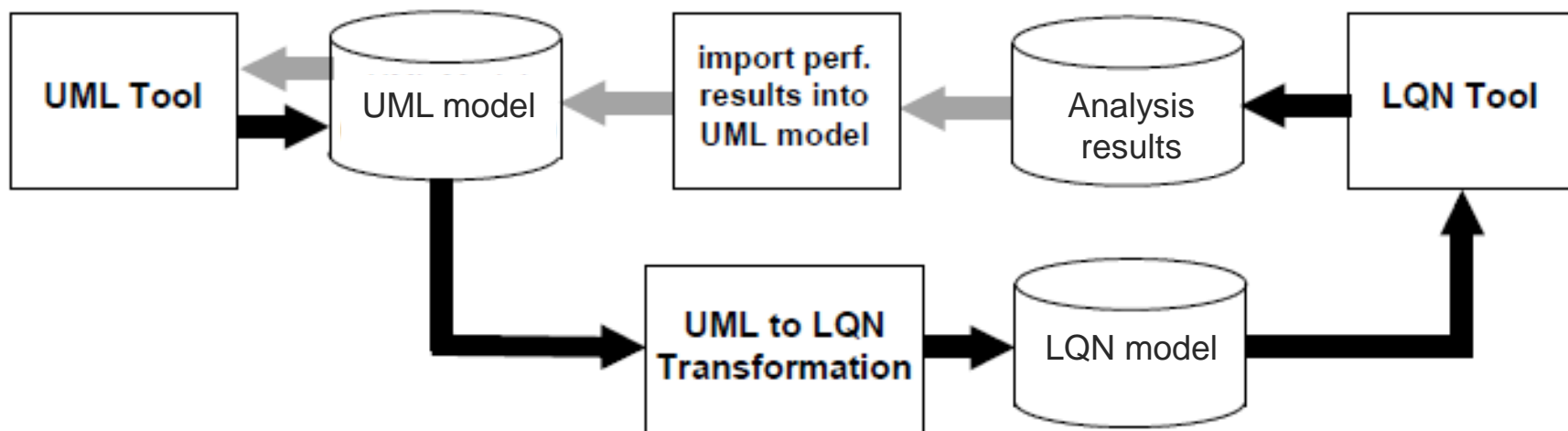
Interakciók

Modell-
transzformáció



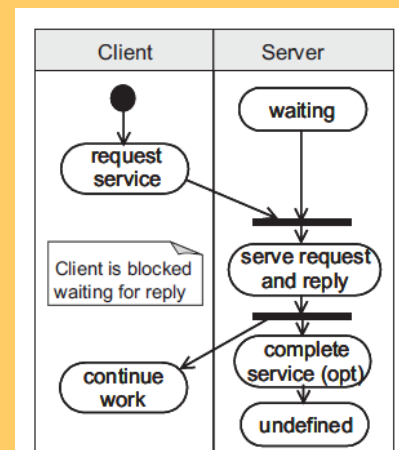
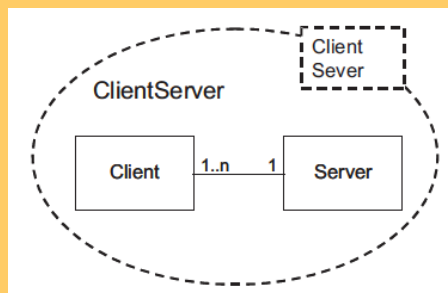
LQN
teljesítmény-
modell

Az architektúra leképezése teljesítménymodellekre



- Architektúra minták használata

Szinkron üzenetküldés

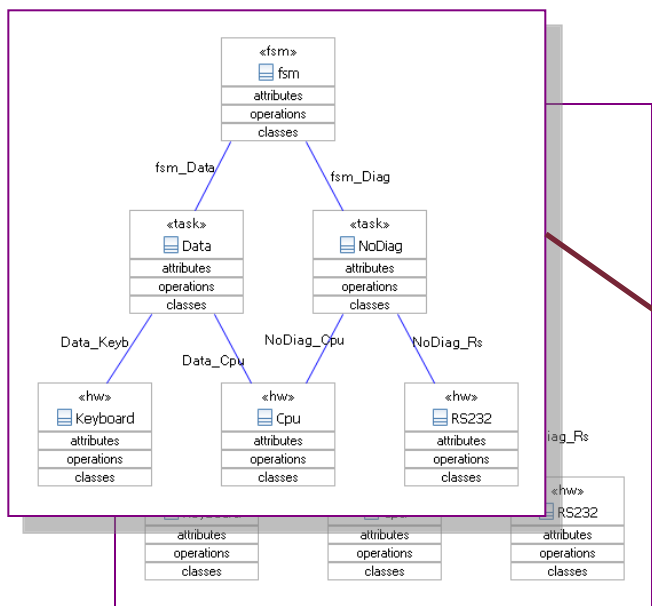


Második példa: Megbízhatósági modellezés

	Teljesítmény modell	Megbízhatósági modell	Biztonsági modell
Komponens paraméterek	Funkció lokális végrehajtási idő, taszk prioritás, processzor 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, taszk áteresztőképesség, processzor kihasználtság	Megbízhatóság, rendelkezésre állás, készenlét, MTTF, MTTR, MTBF	Rendszerszintű veszély gyakoriság

UML alapú megbízhatósági modellezés

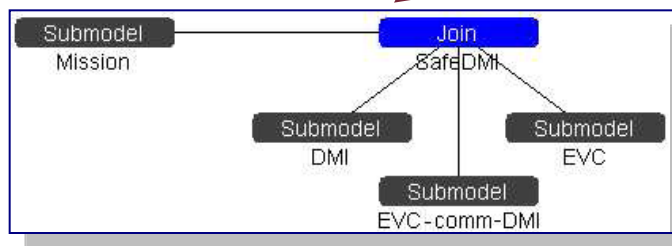
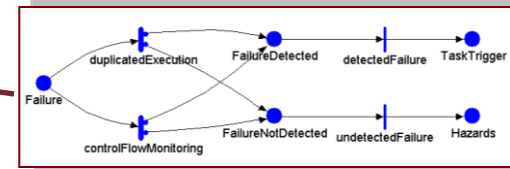
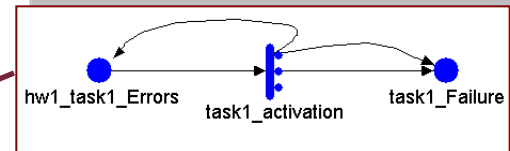
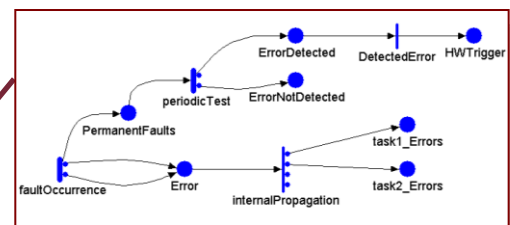
UML architektúra modell



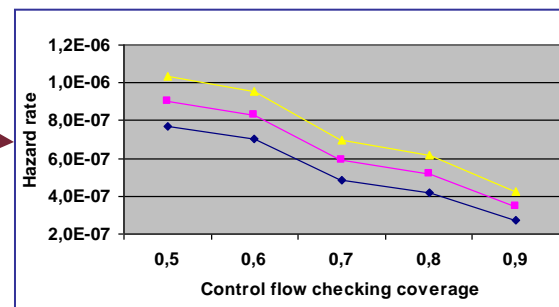
Megbízhatósági modell konstrukció



Analízis alhálók



Rendszerszintű megbízhatósági modell (sztochasztikus aktivitás hálózat)



Analízis eredmények

Megbízhatósági modellezés – Tervezői modell

Komponens:

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

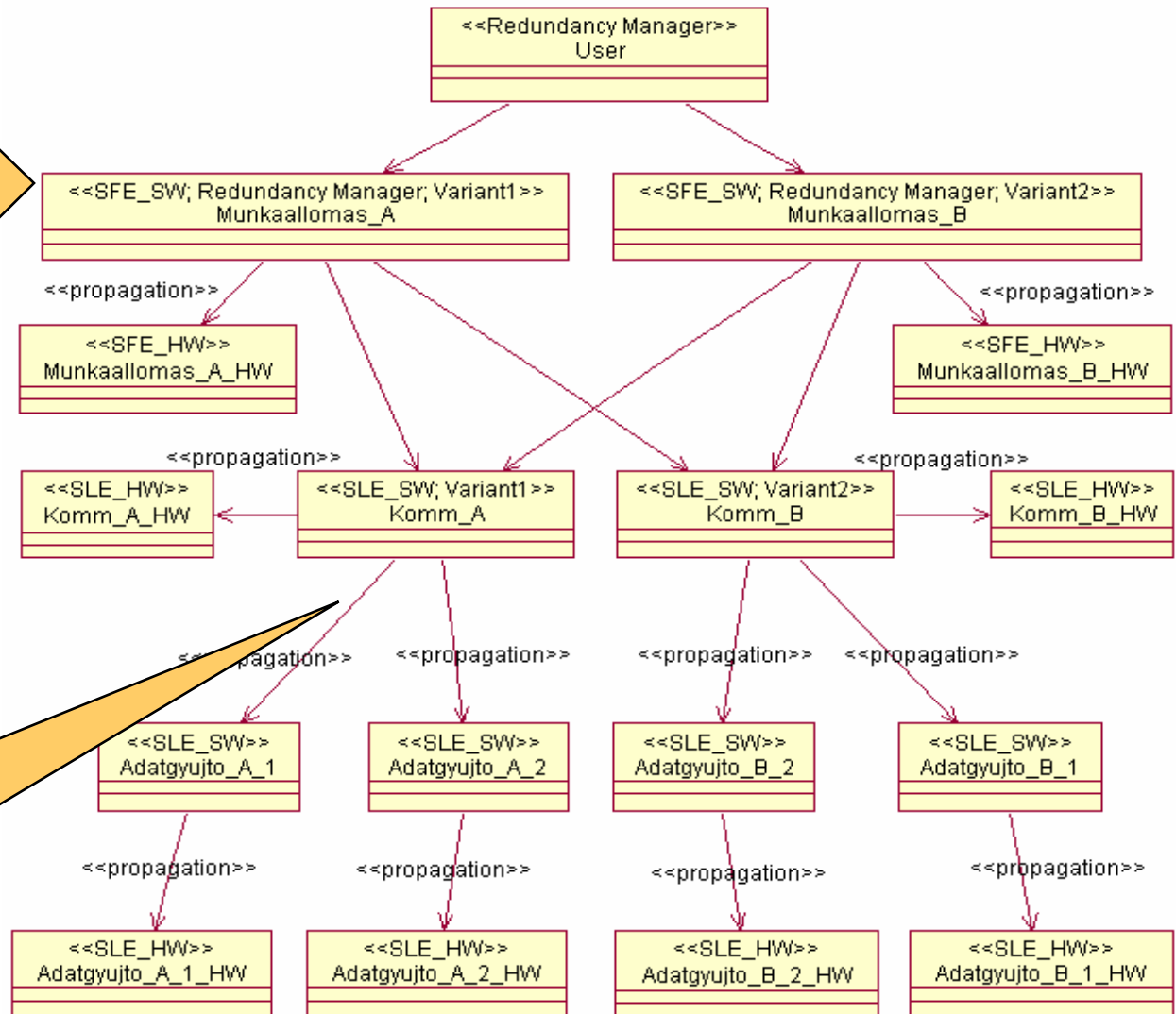
- Meghibásodási jellemzők:

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

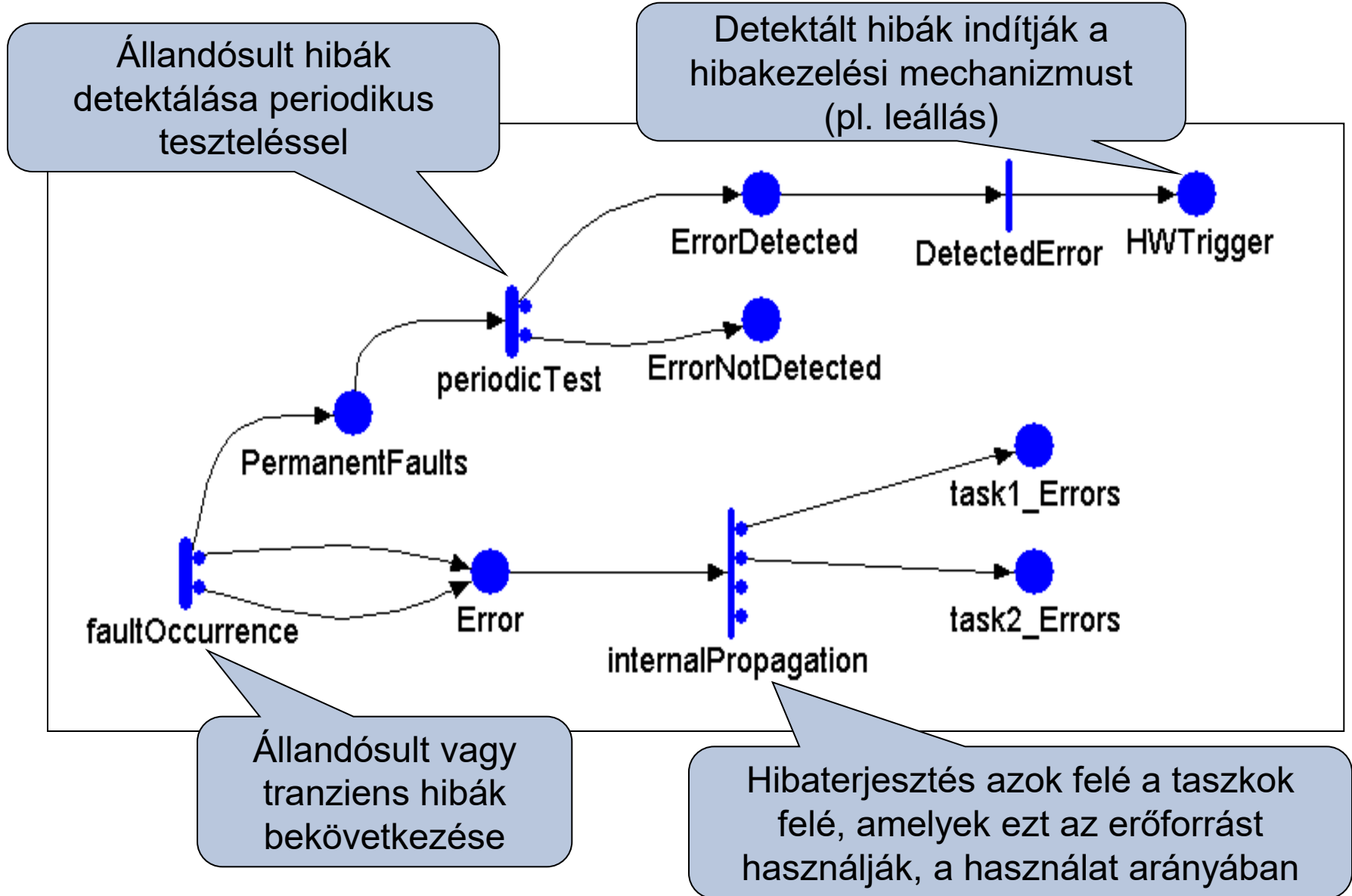
Kapcsolat:

- Hibaterjesztési jellemzők:

- * hibaterjesztési valószínűség

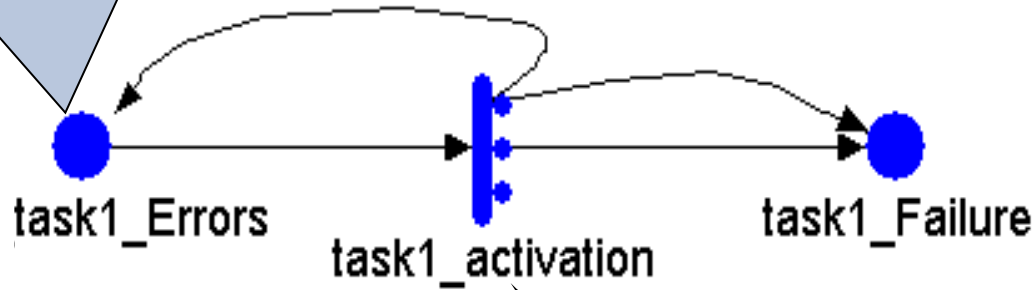


Egy HW erőforrás (pl. memória) analízis modellje



A hibaterjesztés analízis modellje

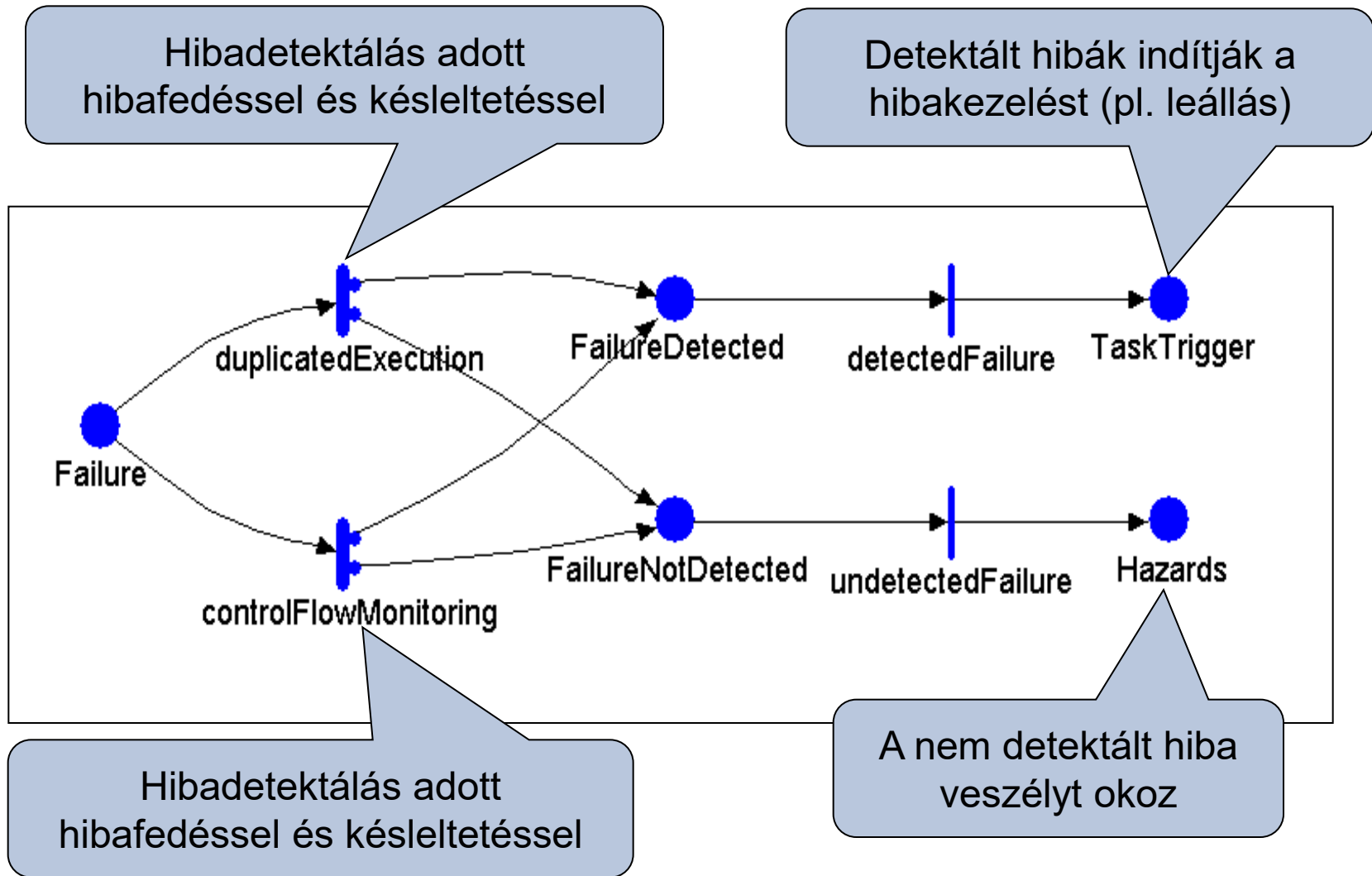
Hiba olyan erőforrásban,
amit a taszk használ



Taszk aktiválási gyakoriság,
hiba aktiválási lehetőségek:

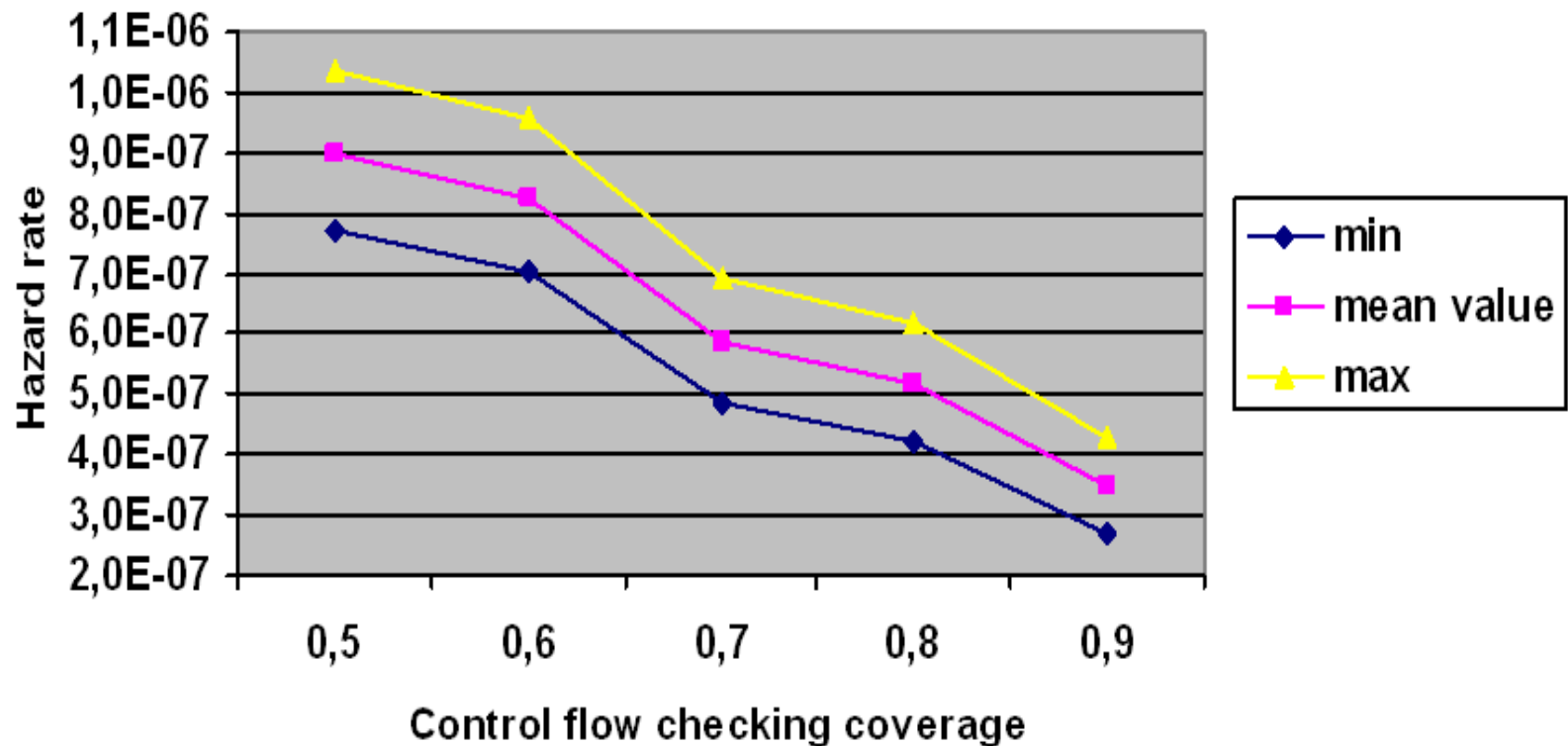
- aktivált hiba,
ami a rendszerben marad
- aktivált hiba, felülíródik,
de hatása van
- hatás nélkül felülírt hiba

Egy szoftver taszk analízis modellje

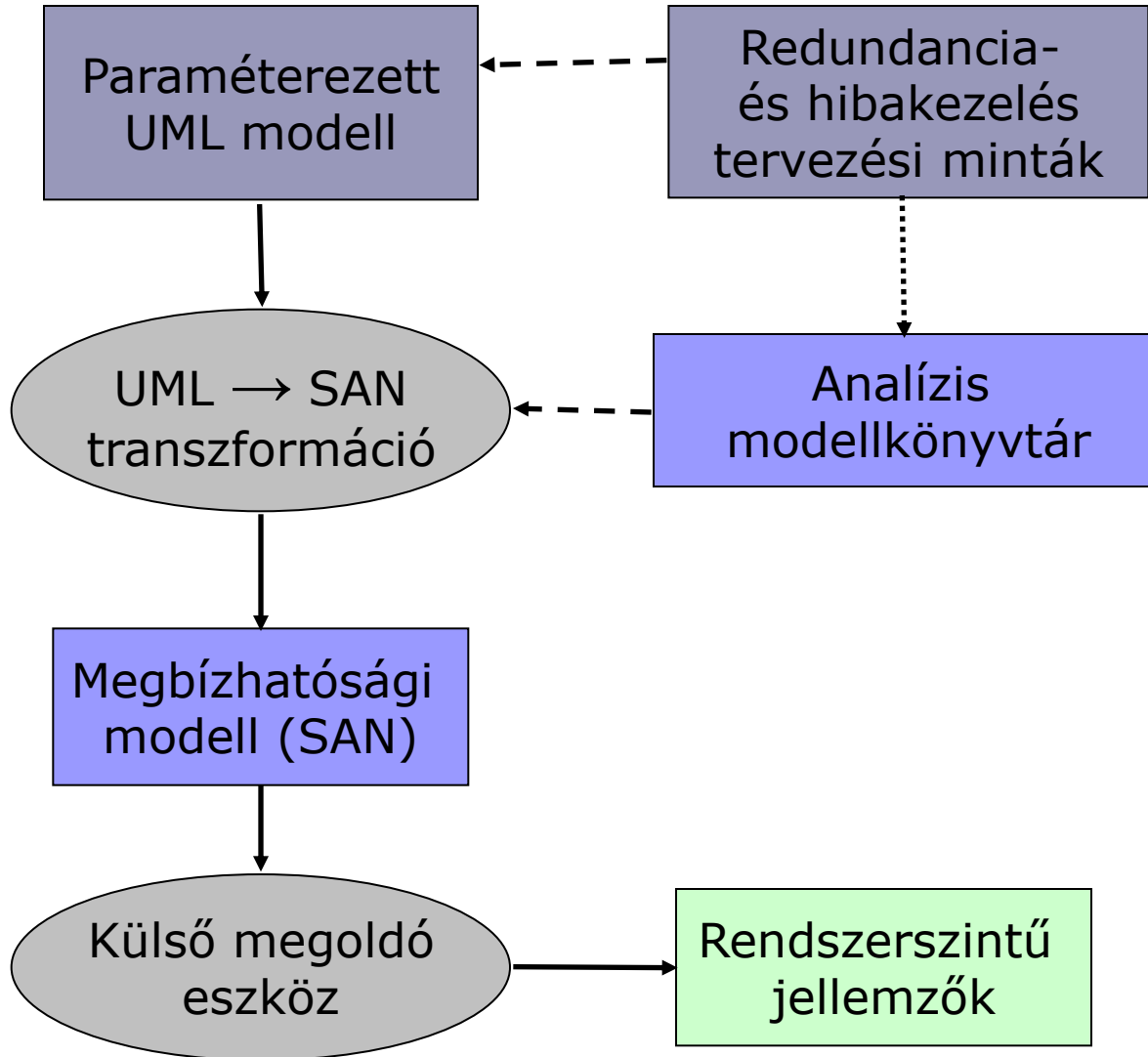


Analízis eredmény (példa)

- Ha a hibadetektálás hibafedése 50% alá csökken, akkor a SIL2 követelmény ($10^{-7} < \text{THR} < 10^{-6}$) nem teljesül



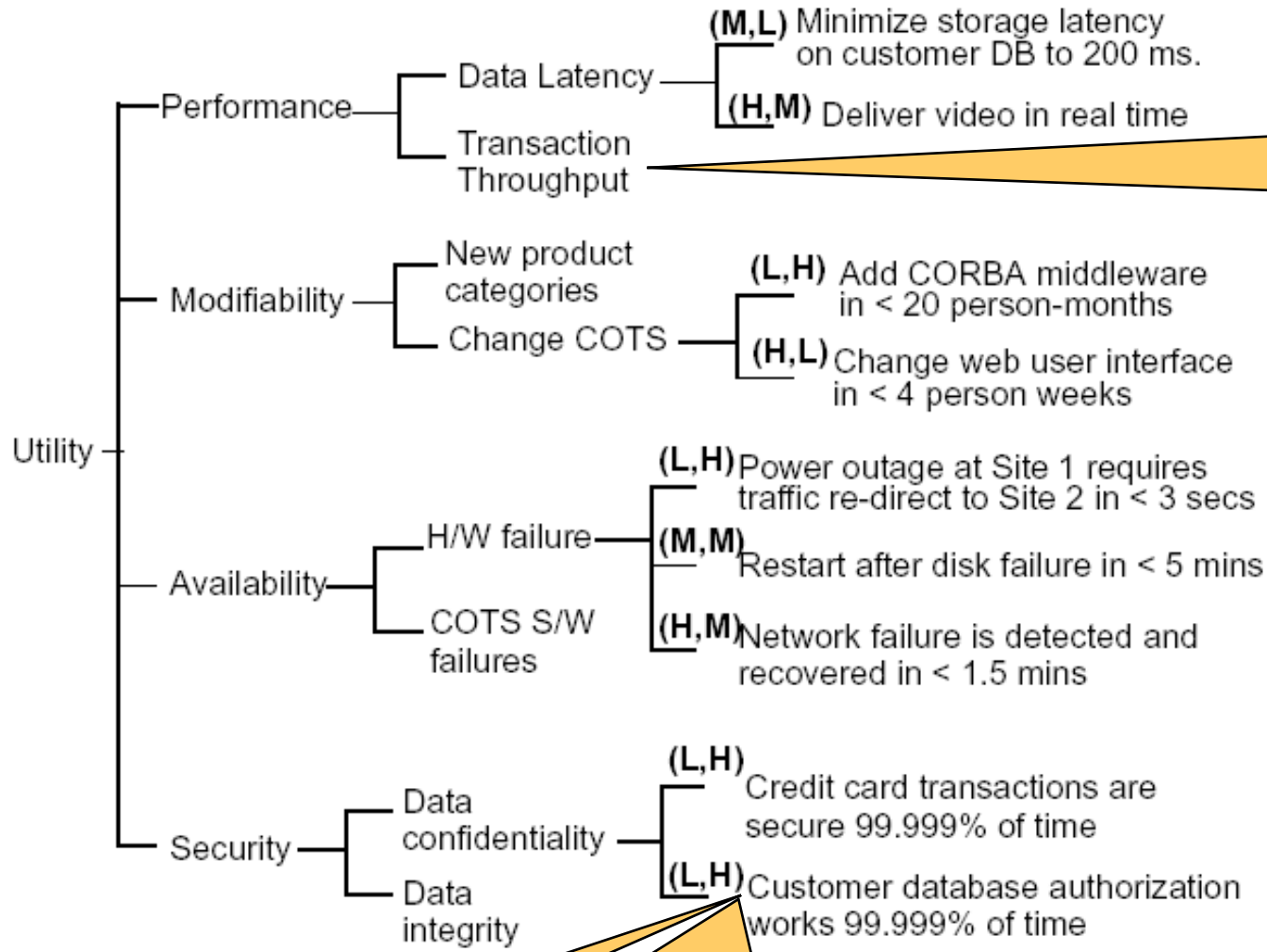
Automatikus analízis eszköz



Követelmények és architektúra összevetése

- 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?
- Alapötletek
 - Elérendő célok szisztematikus összegyűjtése:
Hasznossági fa (utility tree) a célok prioritásaival
 - Célok megfoghatóvá és érthetővé tétele:
Forgatókönyvek csatolása (scenario)
 - Architektúra kiértékelés: Mi volt a döntés, mik az érzékenységek, kompromisszumok, kockázatok?

Elérendő célok összegyűjtése: Hasznossági fa



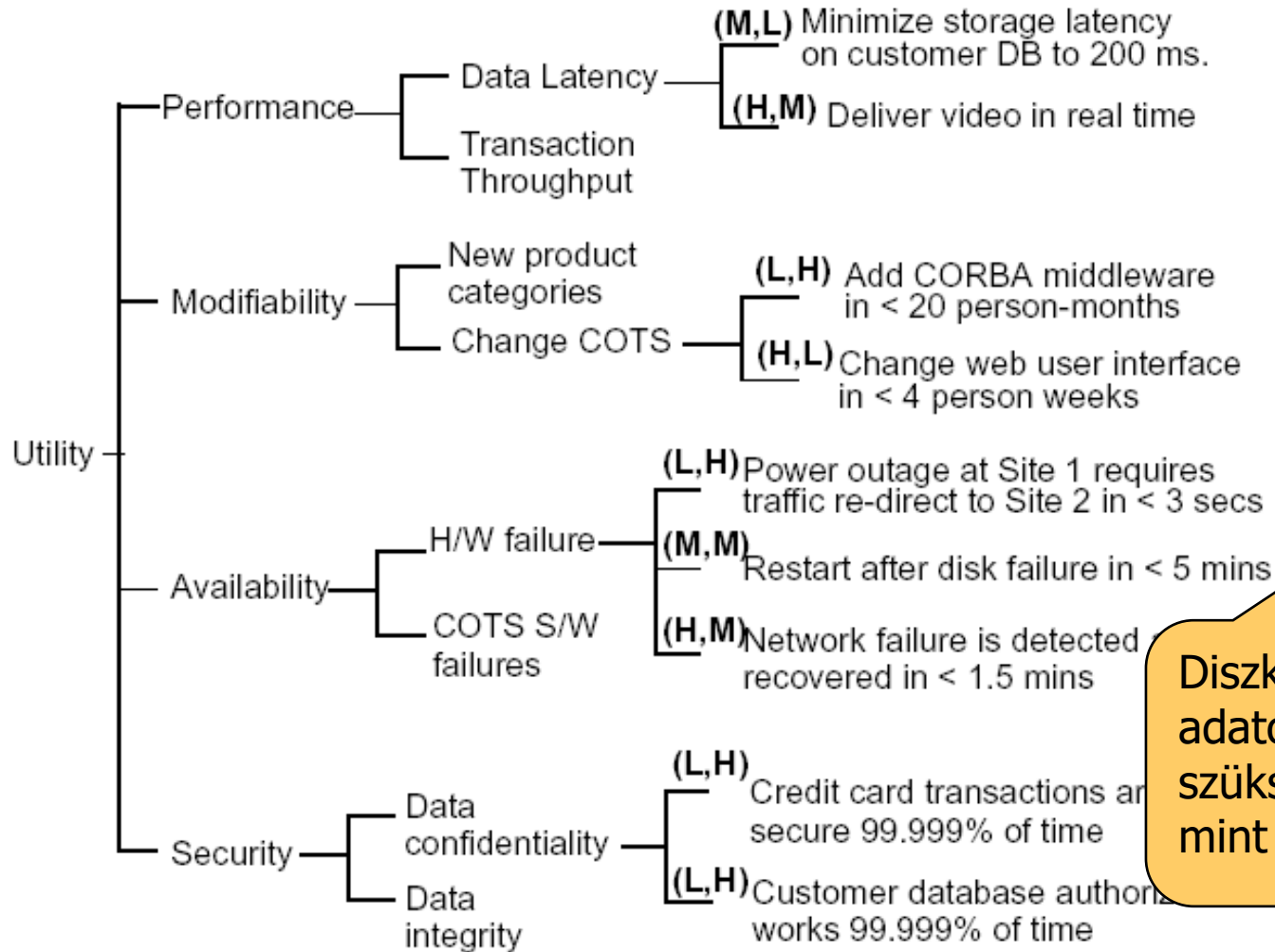
Elérendő illetve finomított célok

Forgatókönyvek jelenítik meg a célokat

Fontosság (prioritás): Low, Medium, High

Megvalósítási bonyolultság: Low, Medium, High

Forgatókönyvek hozzárendelése



Diszk hiba esetén az adatok helyreállítása szükséges, kevesebb, mint 5 perc alatt

Az architektúra analízisének lépései (példával)

- A forgatókönyvek támogatásának vizsgálata
 - **Forgatókönyv**: Diszk hiba esetén az adatok helyreállítása szükséges kevesebb, mint 5 perc alatt
 - Reakció mint architektúra döntés: **Replika adatbázis használata**
- **Érzékenységek vizsgálata**
 - 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 szinkronizálása hatással van mind a rendelkezésre állásra, mind pedig a teljesítményre
 - **Kompromisszum (architektúra döntés)**: Aszinkron replika frissítés
- **A kompromisszumok kockázatának elemzése**
 - Az aszinkron replika frissítés (mint architektúra döntés) kockázatot jelent, ha nagy a költsége a teljesítményvesztésnek
 - A döntés csak adott várható terhelés és veszteségi költség esetén megfelelő!

A kiértékelés rögzített folyamata

- **Üzleti célok bemutatása** <- fejlesztés vezetője
 - Funkciók, üzleti célok, elérendő minőségi jellemzők, érdekelték
 - 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
 - Elérendő minőségi célok és 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?

Az ATAM előnyei

- Érdekeltek bevonása
 - Fejlesztő, tesztelő, felhasználó, elemző szempontjai
 - Hatékony kommunikáció az érdekelték között
- Tisztázott minőségi követelmények
 - Jellemzők és scenariók 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 **összegezhető** a kritériumok teljesítési szintjeihez tartozó pontszámok, ami a döntés alapja
- **PAPRIKA**: Potentially all pairwise rankings of all possible alternatives
 - Kezdőlépés: az alternatívák **páronkénti rangsorolása** (fontossági sorrend megadása)
 - Egy megadott rangsor által már **meghatározott** (dominált) további párokat nem kell értékelni (így nincs kombinatorikus robbanás)
 - Példa: 25 páronkénti rangsorolás elég, ha 5 kritérium és 3-3 szint van
 - A **pontszámok hozzárendelhetők** a rangsorolás alapján lineáris programozással

Összefoglalás

- A fázis ki- és bemenetei
- Az architektúra tervek szerepe
 - Mit tartalmaz az architektúra terv?
 - Mik az ellenőrzési szempontok?
- Tipikus ellenőrzési feladatok
 - Követhetőségi analízis
 - Szisztematikus statikus analízis
 - Kvantitatív analízis
 - Követelmények és architektúra összevetése
- Teszt tervezés: Ld. a tesztelési fázisban