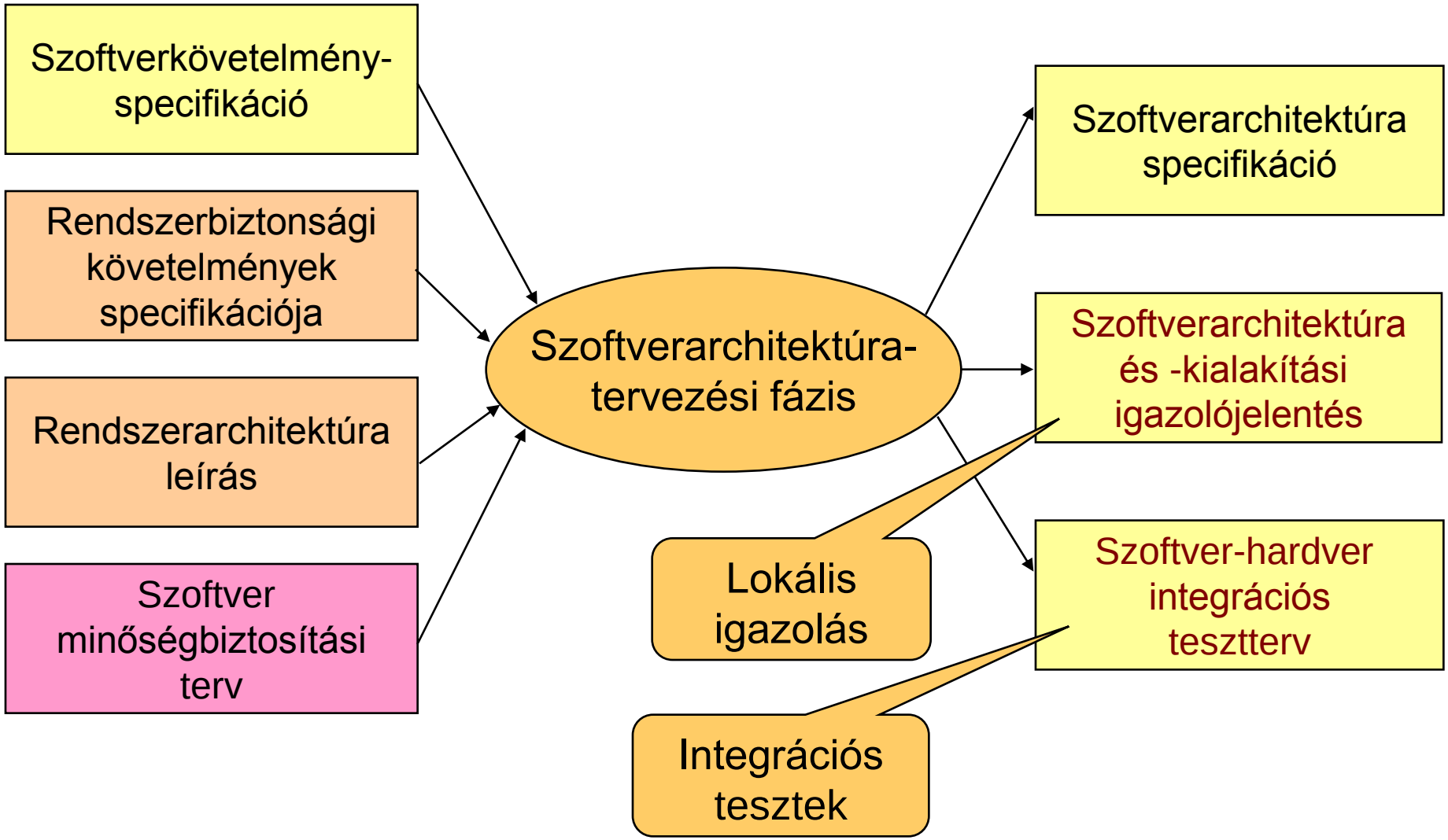
Szoftver architektúra tervek ellenőrzése

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 elkészítése
  - A komponensek azonosítása
  - Mit határoz meg az architektúra?
- Ellenőrzési feladatok
  - Követelményeknek való megfelelőség, követhetőség
  - Hibahatások vizsgálata
  - Extra-funkcionális követelmények vizsgálata
- Teszt tervezés

Kimenetek és bemenetek



Tartalomjegyzék

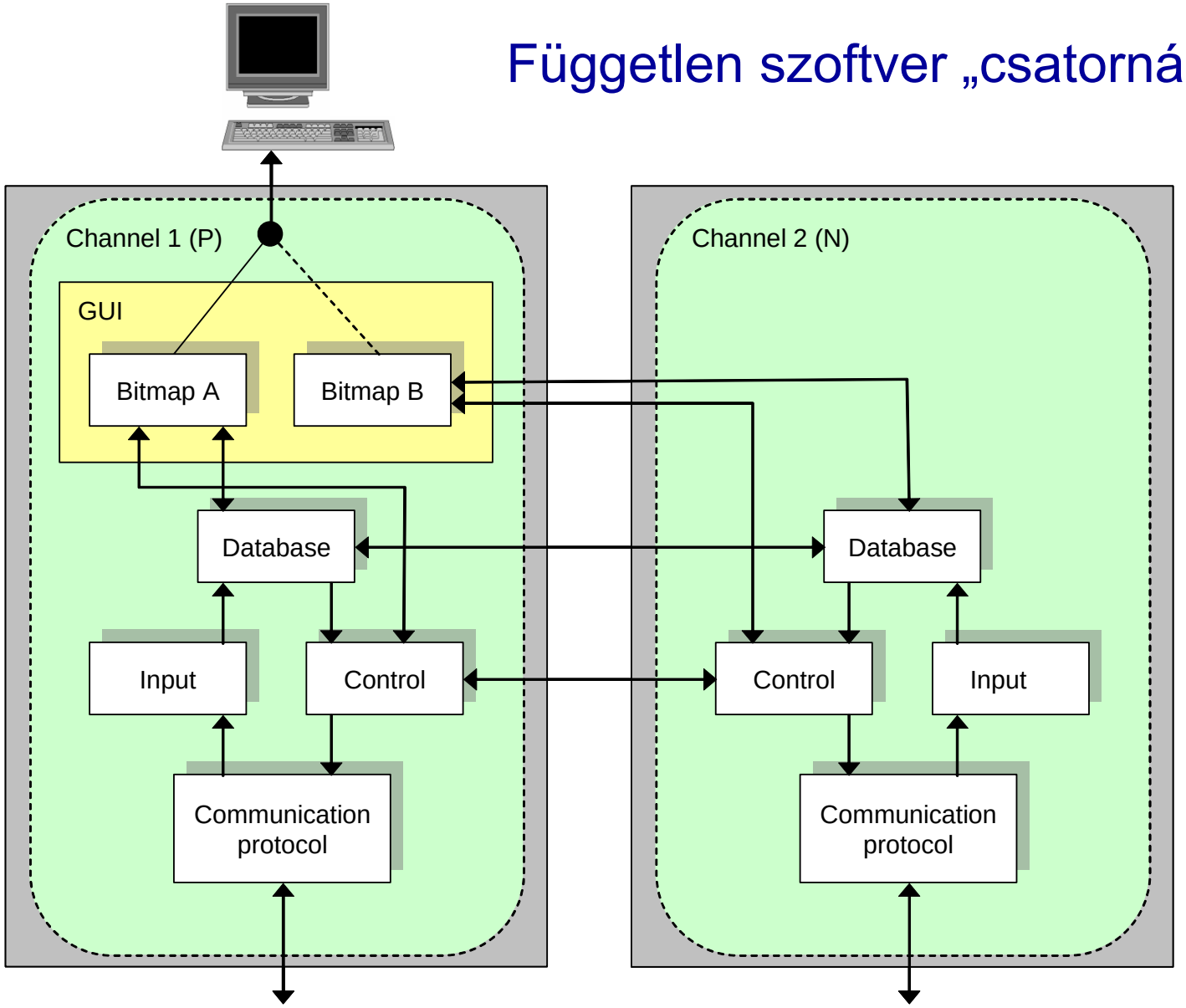
- A fázis ki- és bemenetei
- Az architektúra tervek elkészítése
  - A komponensek azonosítása
  - Mit határoz meg az architektúra?
- Ellenőrzési feladatok
  - Követelményeknek való megfelelőség, követhetőség
  - Hibahatások vizsgálata
  - Extra-funkcionális követelmények vizsgálata
- Teszt tervezés

Szoftverarchitektúra terv

- Architektúra: Komponensek + közöttük lévő kapcsolatok
  - Hardver-szoftver együttműködés azonosítása
  - Szoftverkomponensek (modulok) azonosítása
- Komponensek használata biztonságkritikus rendszerekben
  - Korábban fejlesztett, érvényesített (validált) elemek: alkalmasság igazolásával használhatók
    - Szoftverkövetelményeknek való megfelelés
    - Azonos platform, kontextus
    - Változások esetén hatásvizsgálat
  - (C)OTS komponensek használata
    - SIL 0: előfeltételek nélkül elfogadható
    - 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 (a függetlenség bizonyítható).

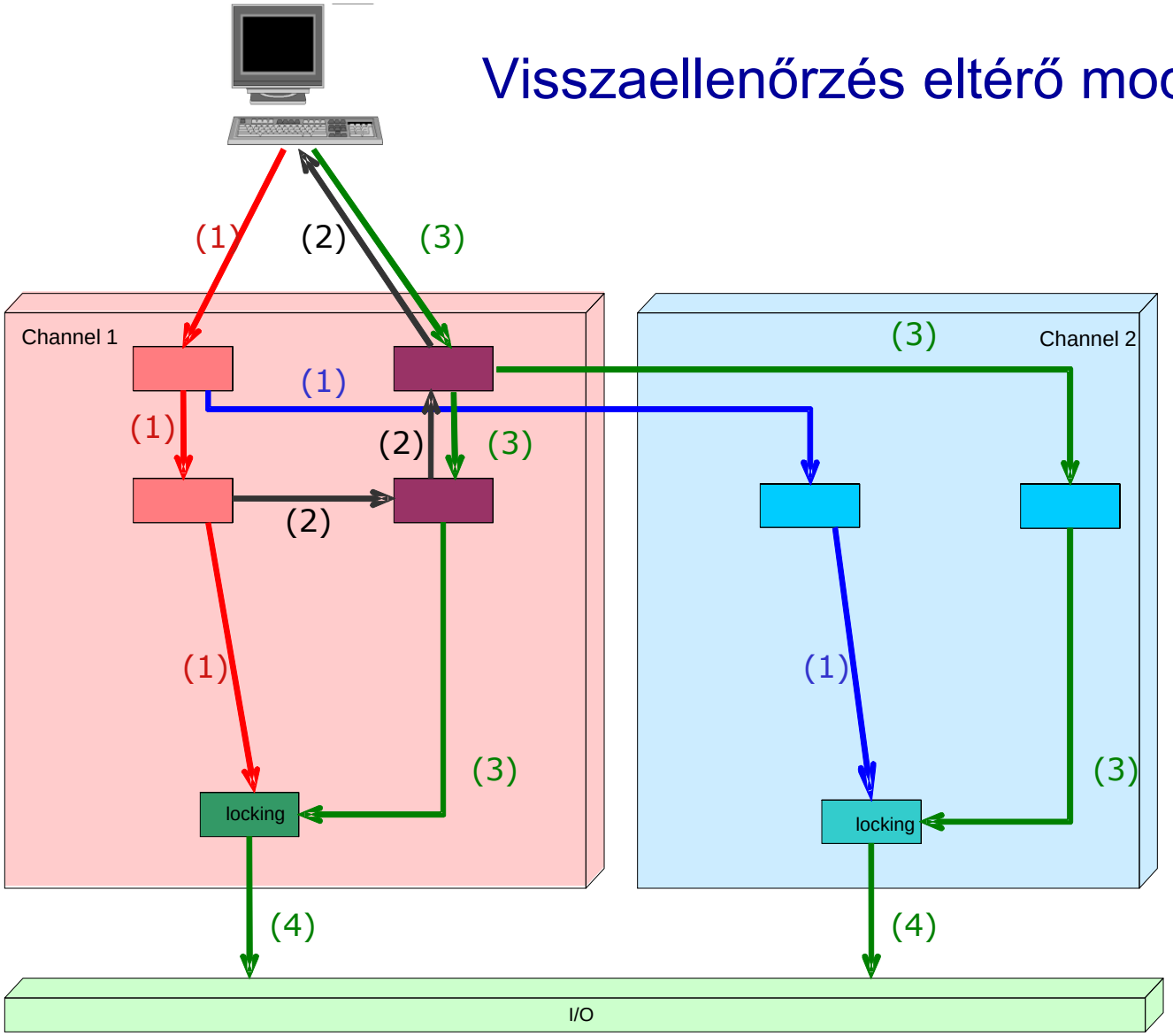
Példa: Architekturális megoldás SIL csökkentésre

Független szoftver „csatornák”



Példa: Architekturális megoldás SIL csökkentésre

Visszaellenőrzés eltérő modulokon



Redundáns architektúrák hibakezeléshez

- 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

Mit határoz meg az architektúra?

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

Tartalomjegyzék

- A fázis ki- és bemenetei
- Az architektúra tervek elkészítése
  - A komponensek azonosítása
  - Mit határoz meg az architektúra?
- Ellenőrzési feladatok
  - Követelményeknek való megfelelőség, követhetőség
  - Hibahatások vizsgálata
  - Extra-funkcionális követelmények vizsgálata
- Teszt tervezés

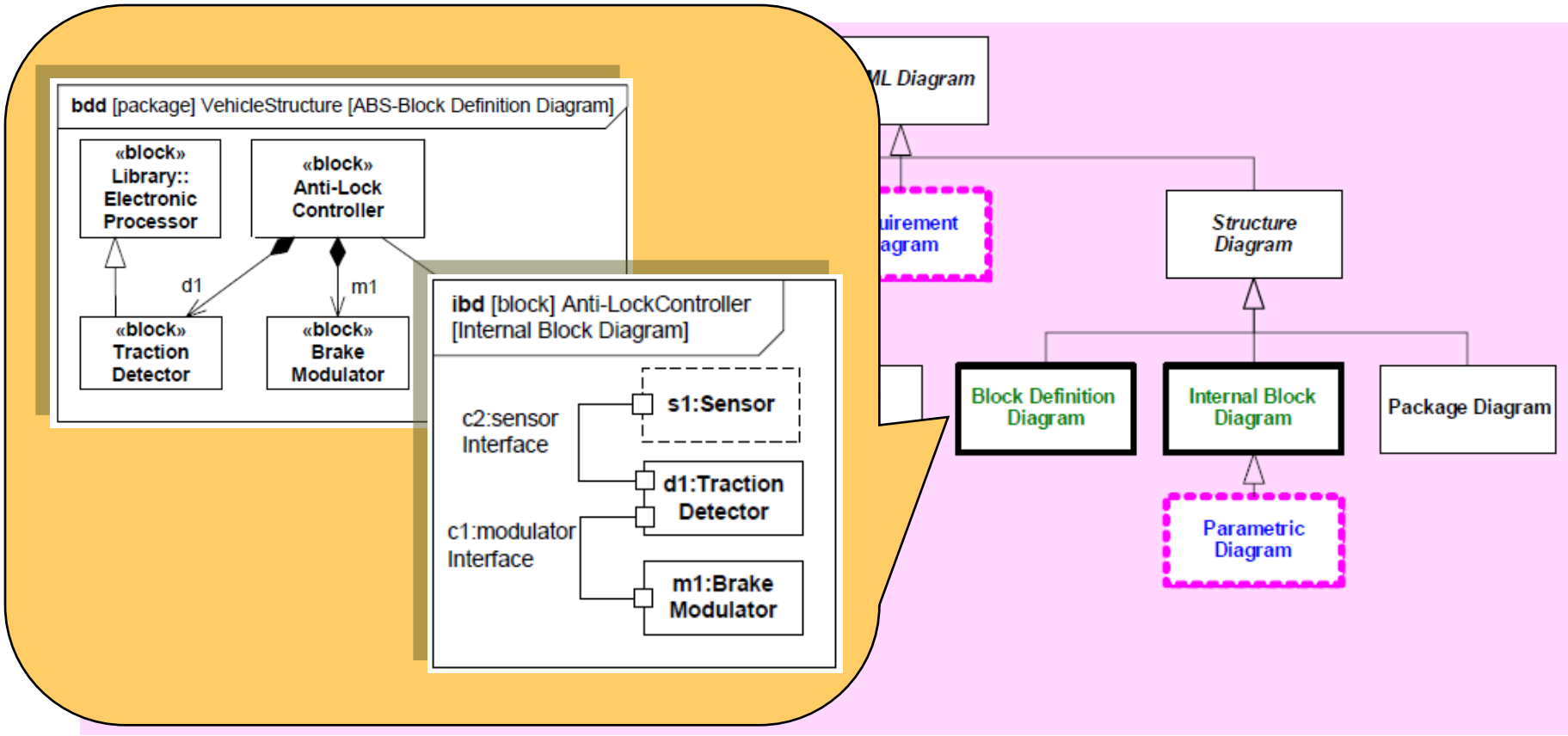
A szoftverarchitektúra igazolása (áttekintés)

Vizsgálati technikák áttekintése:

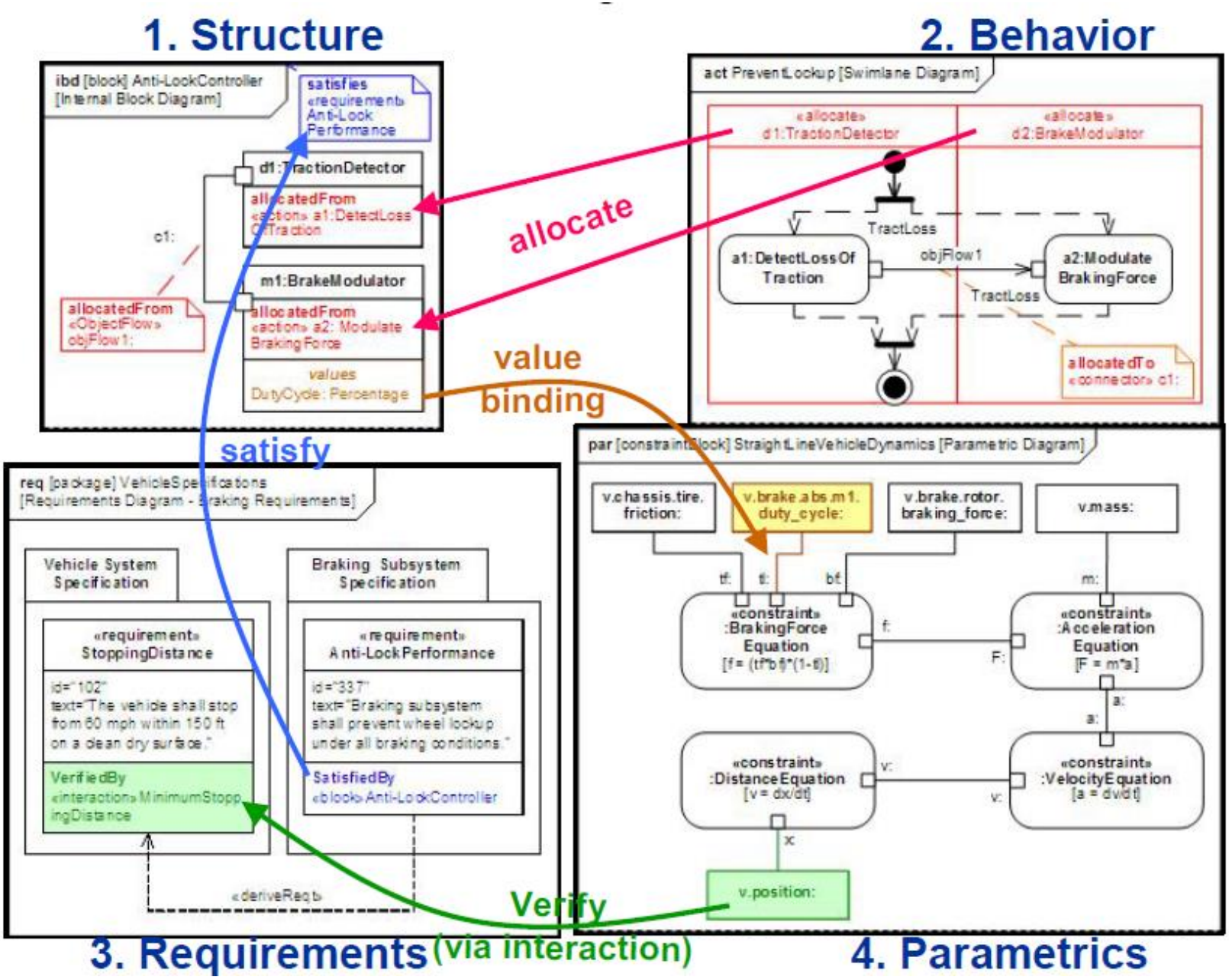
- Követhetőség
  - Követelményeknek való megfelelés ellenőrzése (követelmények „lefedése”
- Szisztematikus elemzés
  - Az architektúra alapján rendre végigellenőrizni az elvárásokat
  - Pl.: Hibák hatásainak felmérése kombinatorikus módszerekkel: Kapcsolat a lokális (komponens szintű) hibák és események és a rendszerszintű (veszélyes) állapotok között
- Modell alapú vizsgálatok
  - Az architektúra alapján analízis modell konstruálása
  - Pl.: Extra-funkcionális jellemzők meghatározása (tipikusan sztochasztikus) módszerekkel:
    Lokális (komponens/kapcsolati szintű) paraméterek alapján rendszerszintű extra-funkcionális jellemzők számítása

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



A szisztematikus hibaelemzés módszerei

1. Hibafa
2. Eseményfa
3. Ok-következmény analízis
4. Hibamód és -hatás analízis (FMEA)

1. Hibafa analízis

Rendszerszintű veszély okainak vizsgálata

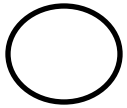
- Tipikusan felülről lefelé haladó analízis
- Felderíti a kezelendő hibaokokat és -kombinációkat

Hibafa konstrukció:

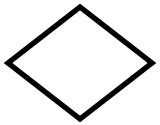
- (Rendszerszintű) veszély, veszélyes állapot:
  Azonosítás: környezet, követelmények, szabványok
- Közbenső események, pszeudo-események:
  Veszélyhez vezetnek,
  Boole-logikai kombinációi (AND, OR) alacsonyabb szintű eseményeknek
- Elsődleges események: további felbontás nincs

Hibafa grafikus elemkészlet

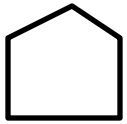
legfelső szintű vagy közbenső esemény



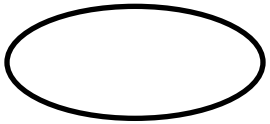
elsődleges (alapszintű) esemény



tovább nem vizsgált esemény



normál esemény (nem hiba vagy veszély)

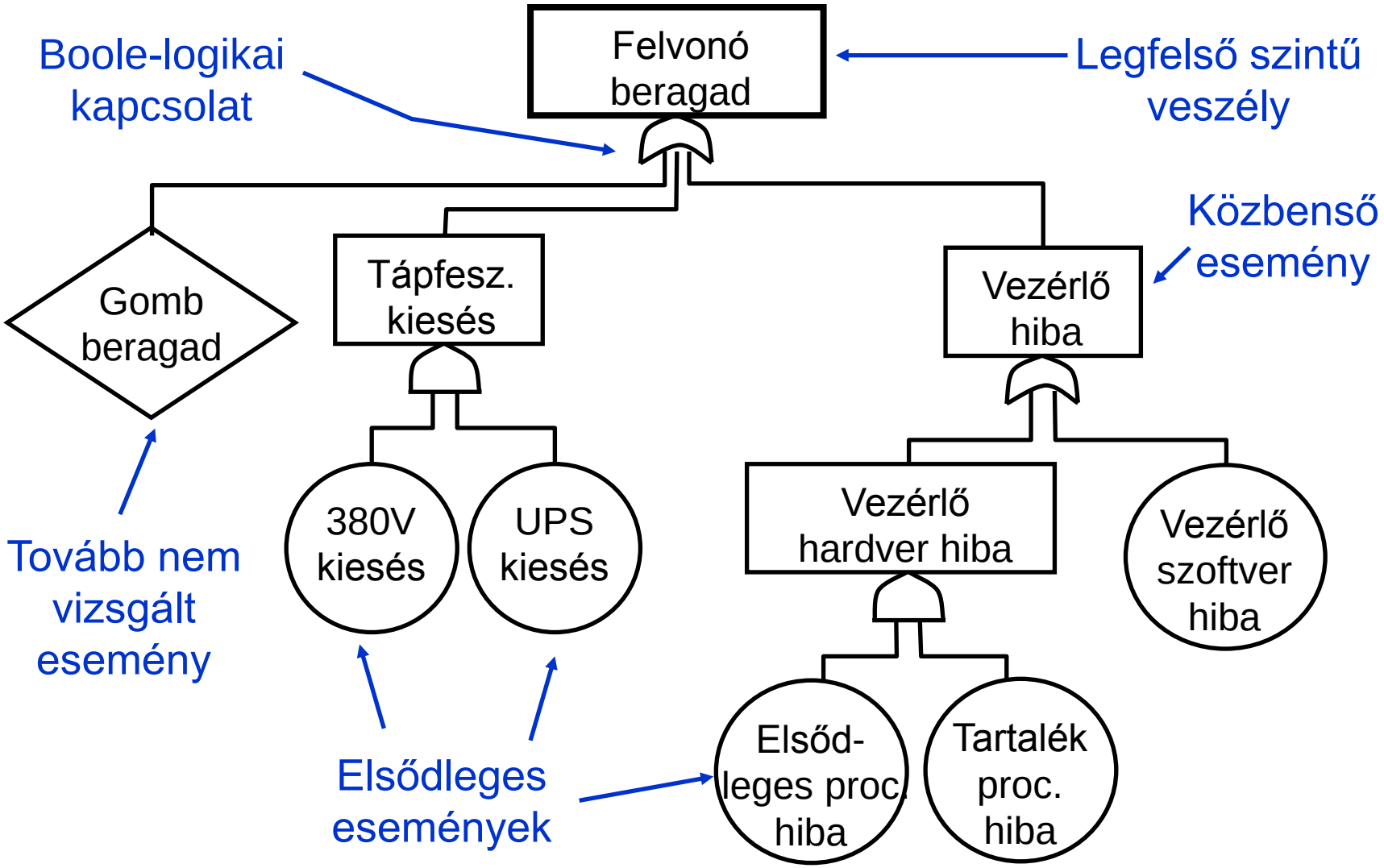


feltétel egy összetett esemény bekövetkezéséhez

ÉS kapu

VAGY kapu

Hibafa példa: Felvonó



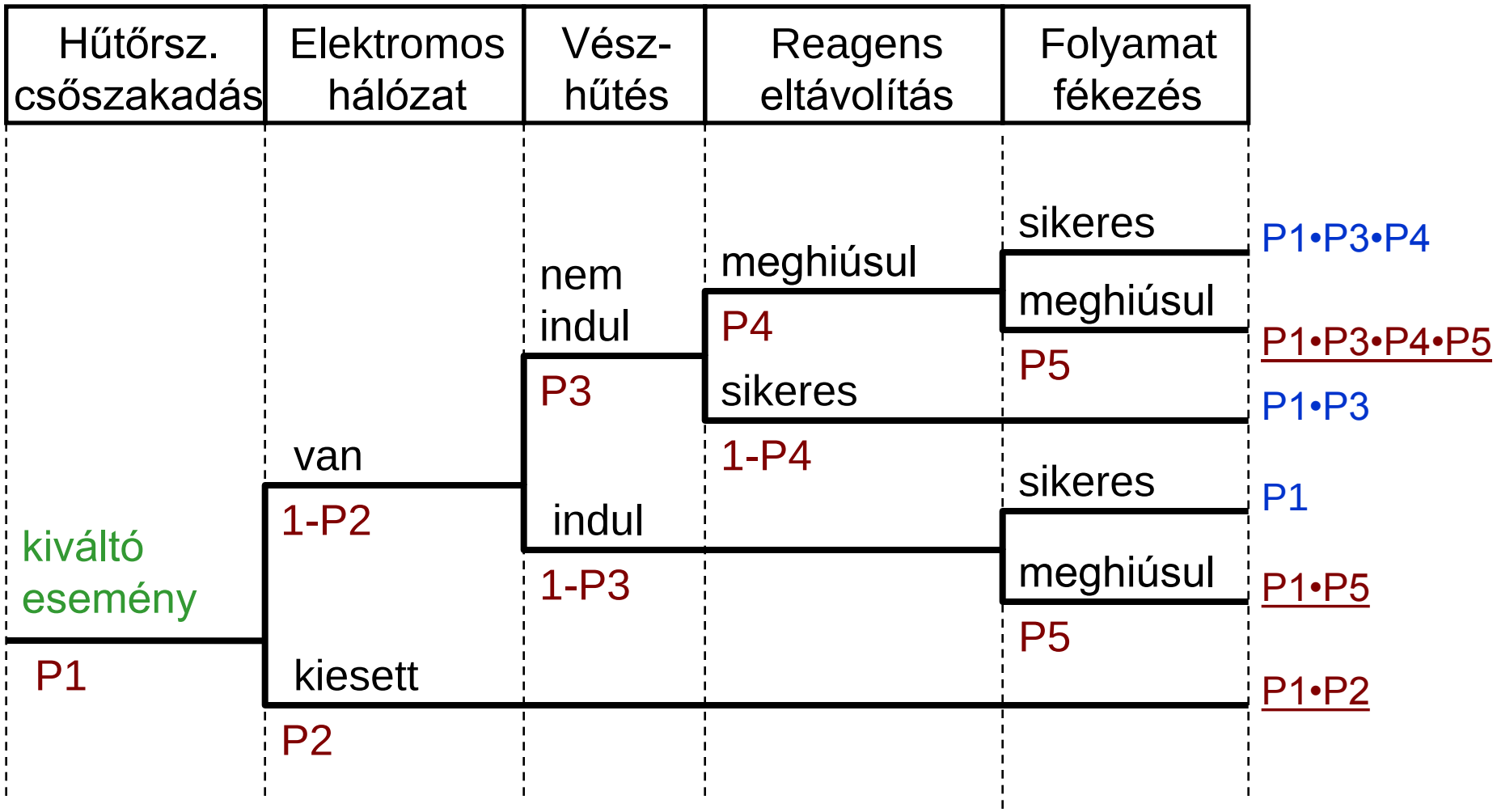
Hibafa analízis

- Minőségi (kvalitatív) analízis:
  - Hibafa redukció: Közbenső események feloldása
    $\rightarrow$ 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

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

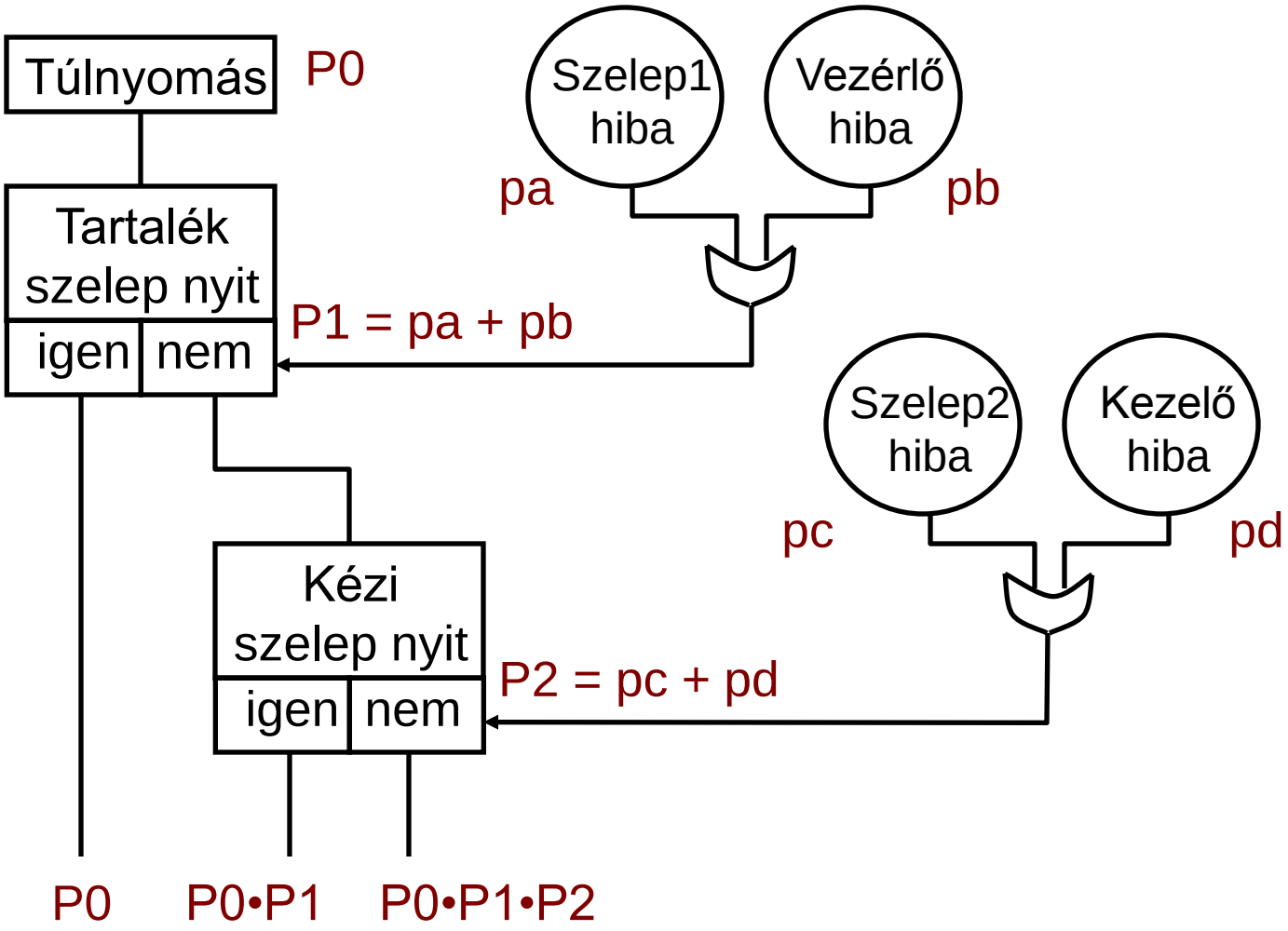
Eseményfa példa: Reaktorhűtés



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 ≤ L nem megy át	65% 35%	- túlnyomás - technológiai hiba
...	...	...	...

A modell alapú elemzés módszerei

Cél: Architektúra változatok kiértékelése

- Analízis modellek készítése és paraméterezése az architektúra modellje alapján
  - Matematikai modell, jellemzői számíthatók („megoldható” 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 modellkönyvtár

Modell alapú architektúra vizsgálatok áttekintése

	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 áteresztő-képesség, processzor kihasználtság	Megbízhatóság, rendelkezésre állás, készenlét, MTTF, MTTR, MTBF	Rendszerszintű veszély gyakoriság

Első példa: Teljesítmény 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 áteresztő-képesség, processzor kihasználtság	Megbízhatóság, rendelkezésre állás, készenlét, MTTF, MTTR, MTBF	Rendszerszintű veszély gyakoriság

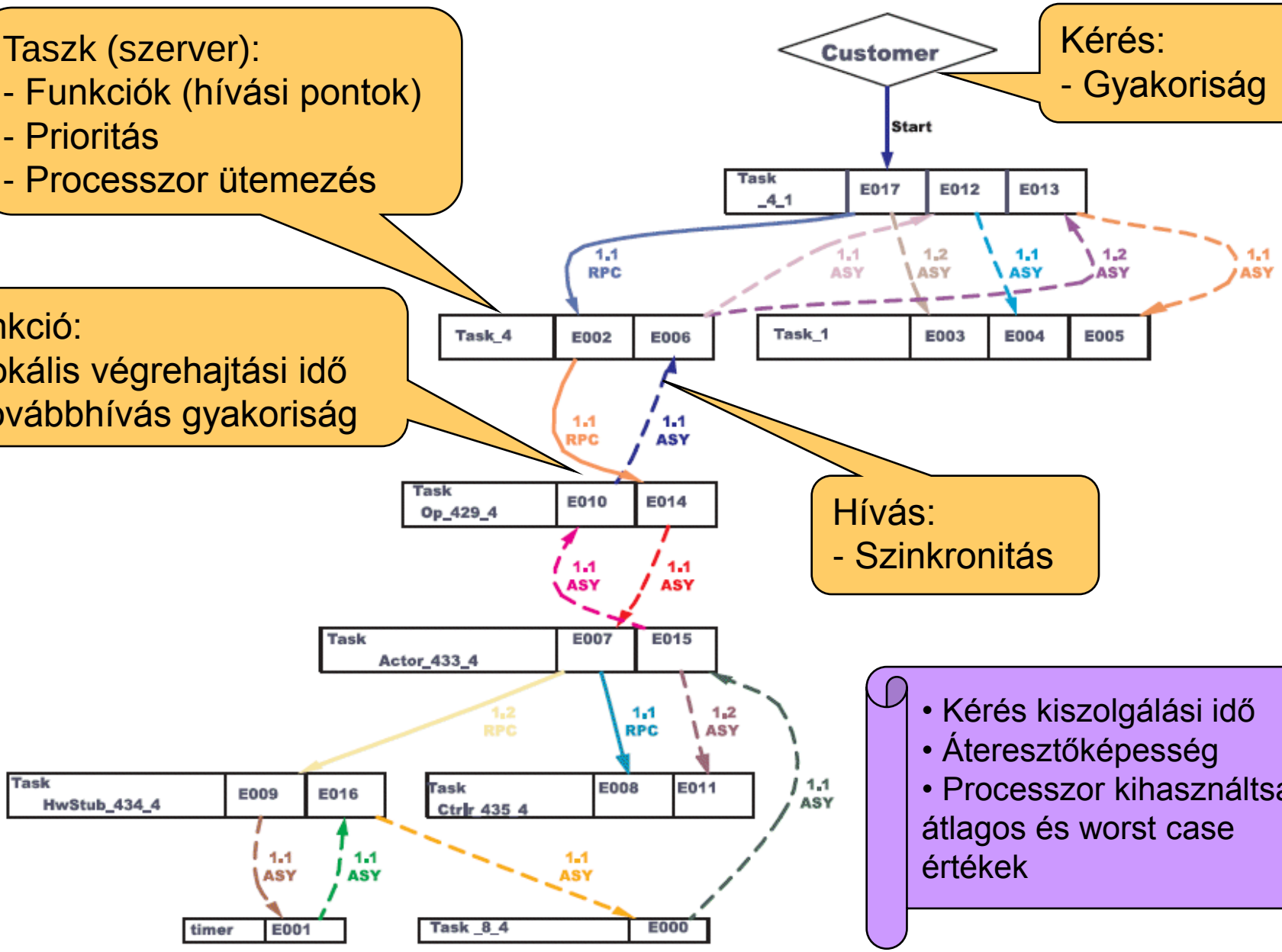
Teljesítmény modellezés (LQN)

Taszk (szerver):
- Funkciók (hívási pontok)
- Prioritás
- Processzor ütemezés

Kérés:
- Gyakoriság

Funkció:
- Lokális végrehajtási idő
- Továbbhívás gyakoriság

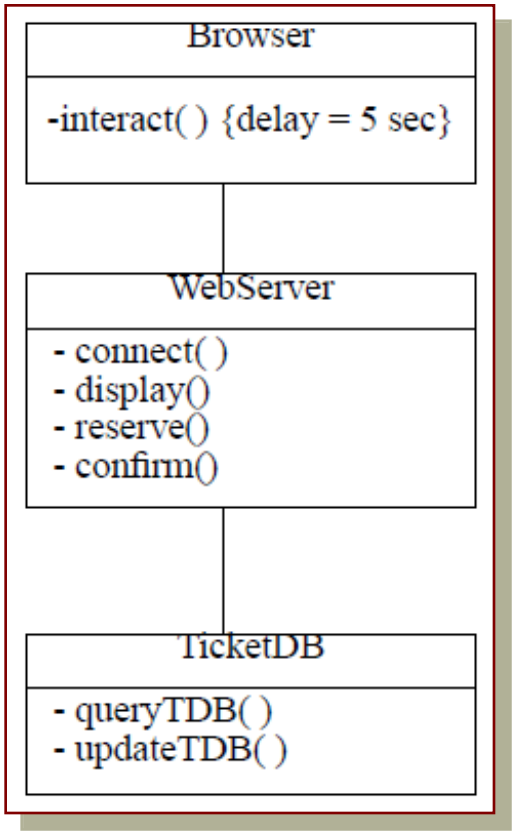
Hívás:
- Szinkronitás

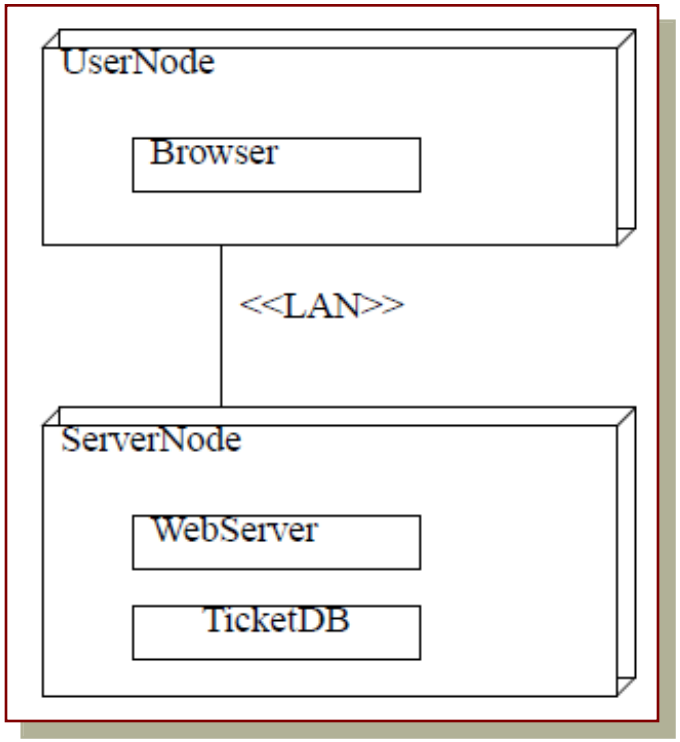


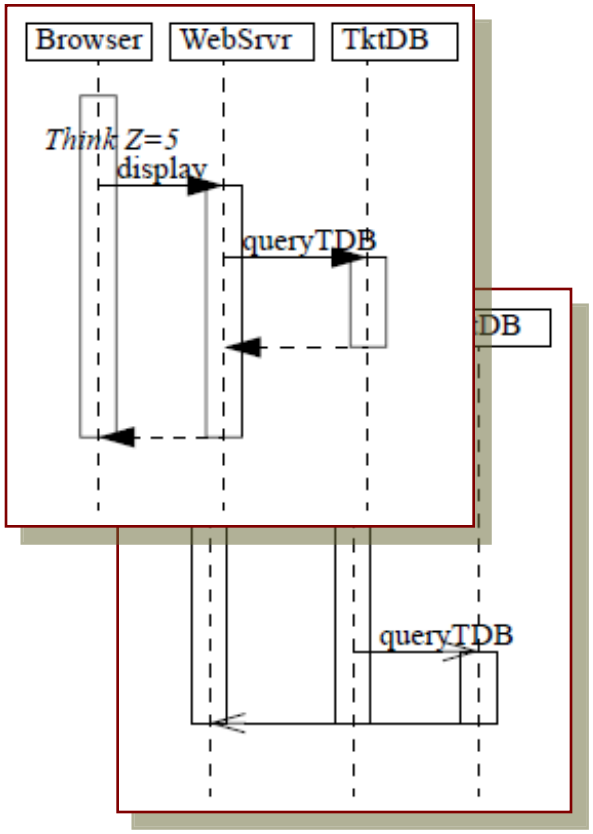
- Kérés kiszolgálási idő
- Áteresztőképesség
- Processzor kihasználtság

átlagos és worst case értékek

Az architektúra leképezése teljesítménymodellre





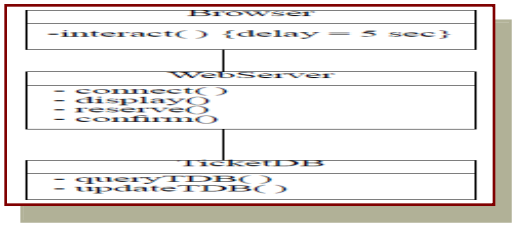


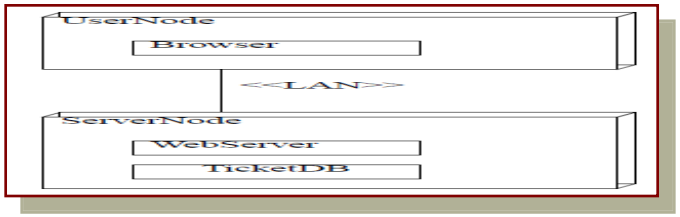
Osztályok Telepítés Interakciók

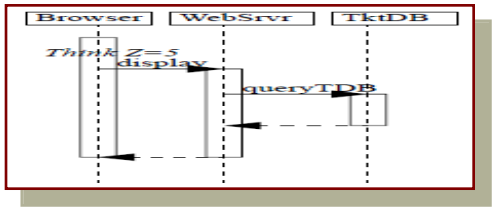
Lokális teljesítmény paraméterek megadása:

- Attribútumok (konvenció alapján értelmezhető)
- UML tagged value

Az architektúra leképezése teljesítménymodellre





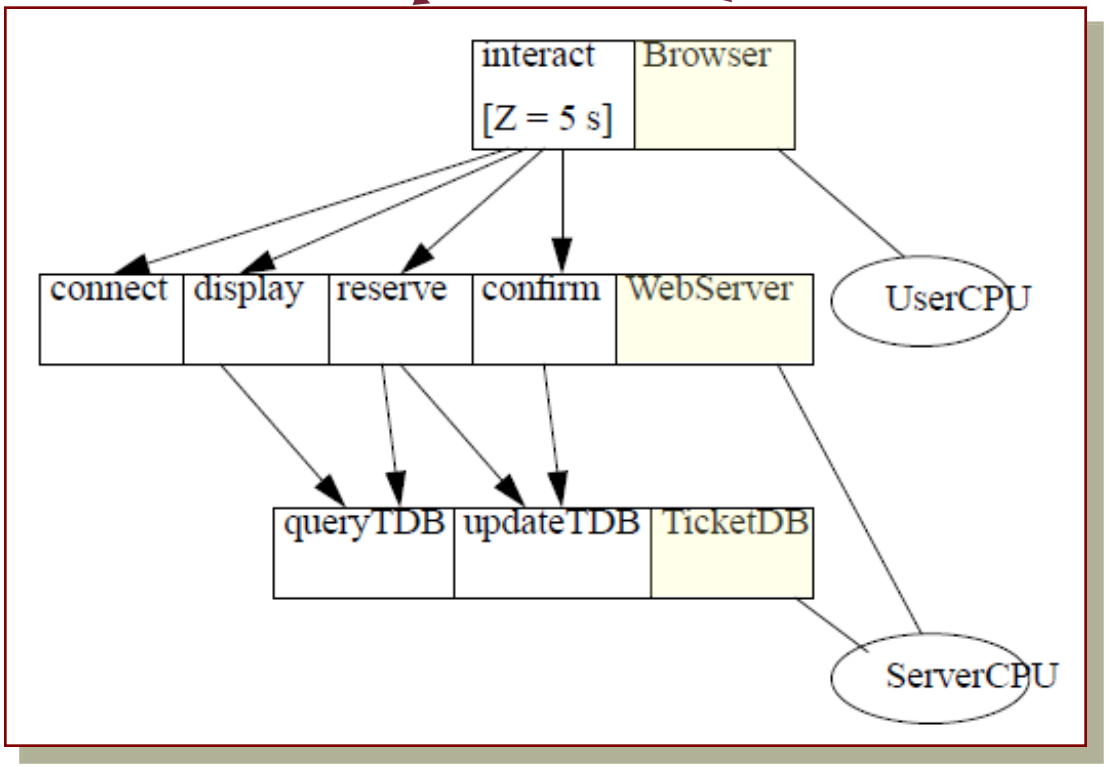


Osztályok

Telepítés

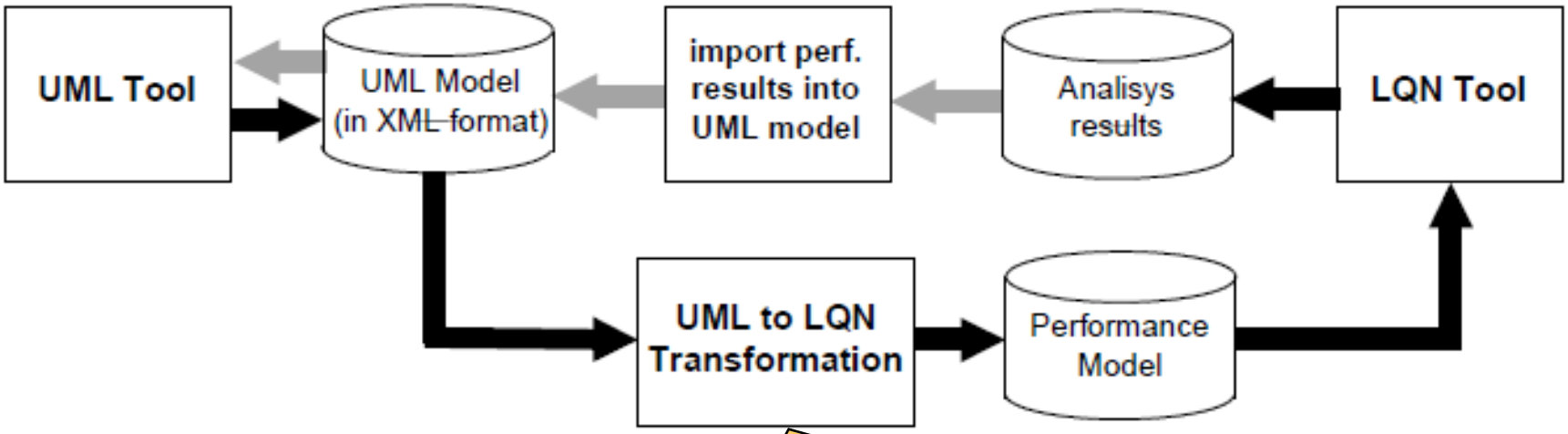
Interakciók

Modell-
transzformáció



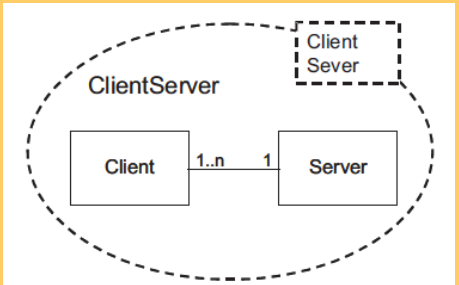
LQN
teljesítmény-
modell

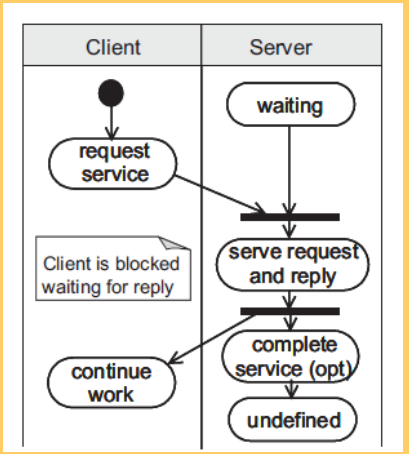
Az architektúra leképezése teljesítménymodellre



- Architektúra minták használata

Szinkron üzenetküldés



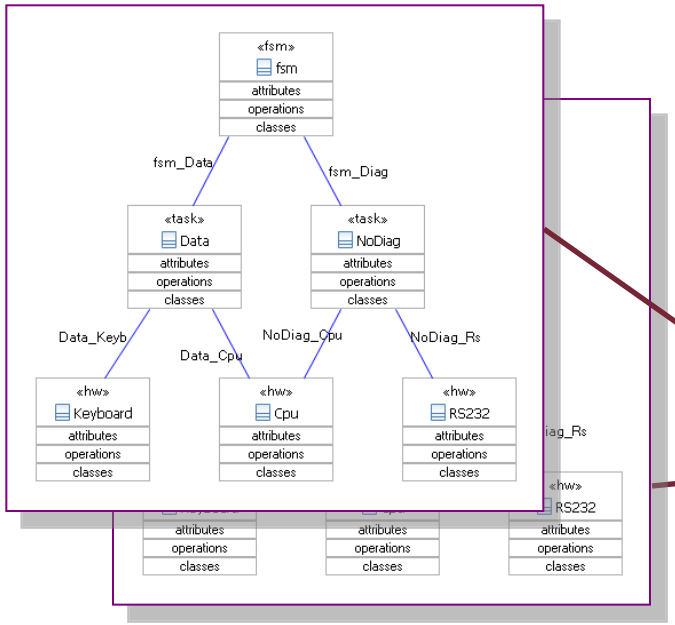


Második példa: Megbízhatósági modellezés

	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 áteresztő-képesség, processzor kihasználtság	Megbízhatóság, rendelkezésre állás, készenlét, MTTF, MTTR, MTBF	Rendszerszintű veszély gyakoriság

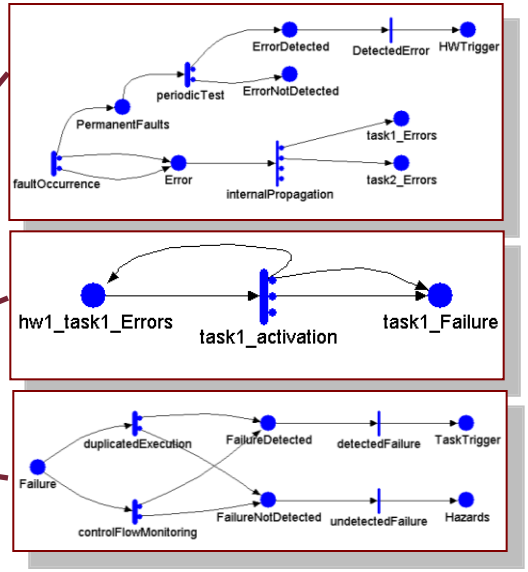
UML alapú megbízhatósági modellezés

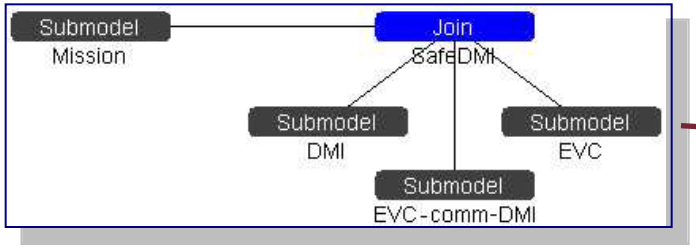
UML architektúra modell



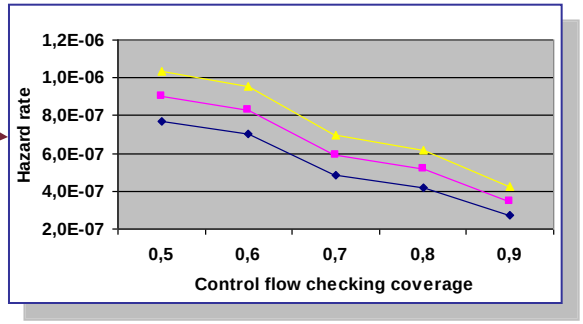
Megbízhatósági modell konstrukció

Analízis alhálók





Rendszerszintű megbízhatósági modell (sztochasztikus aktivitás hálózat)

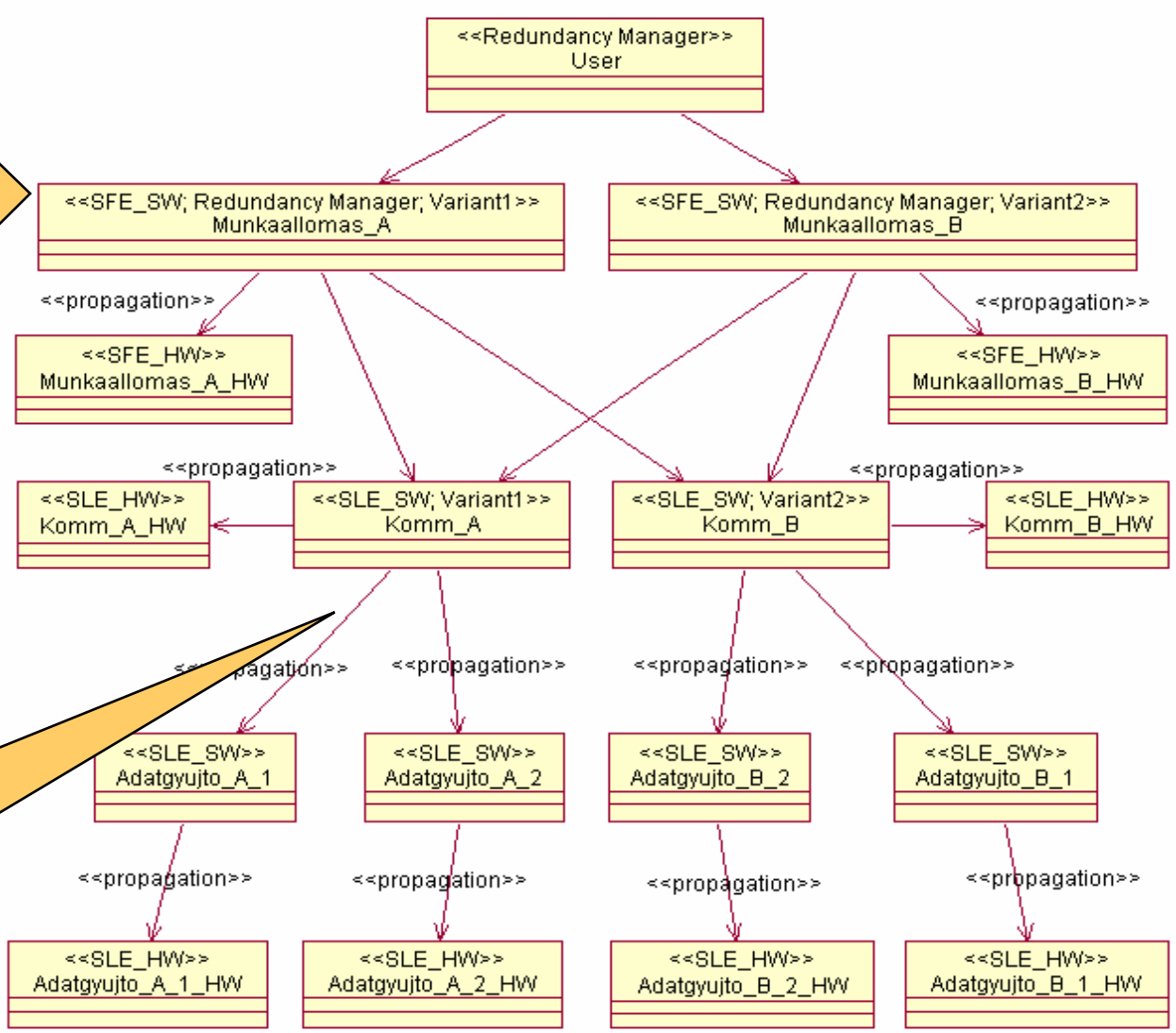


Analízis eredmények

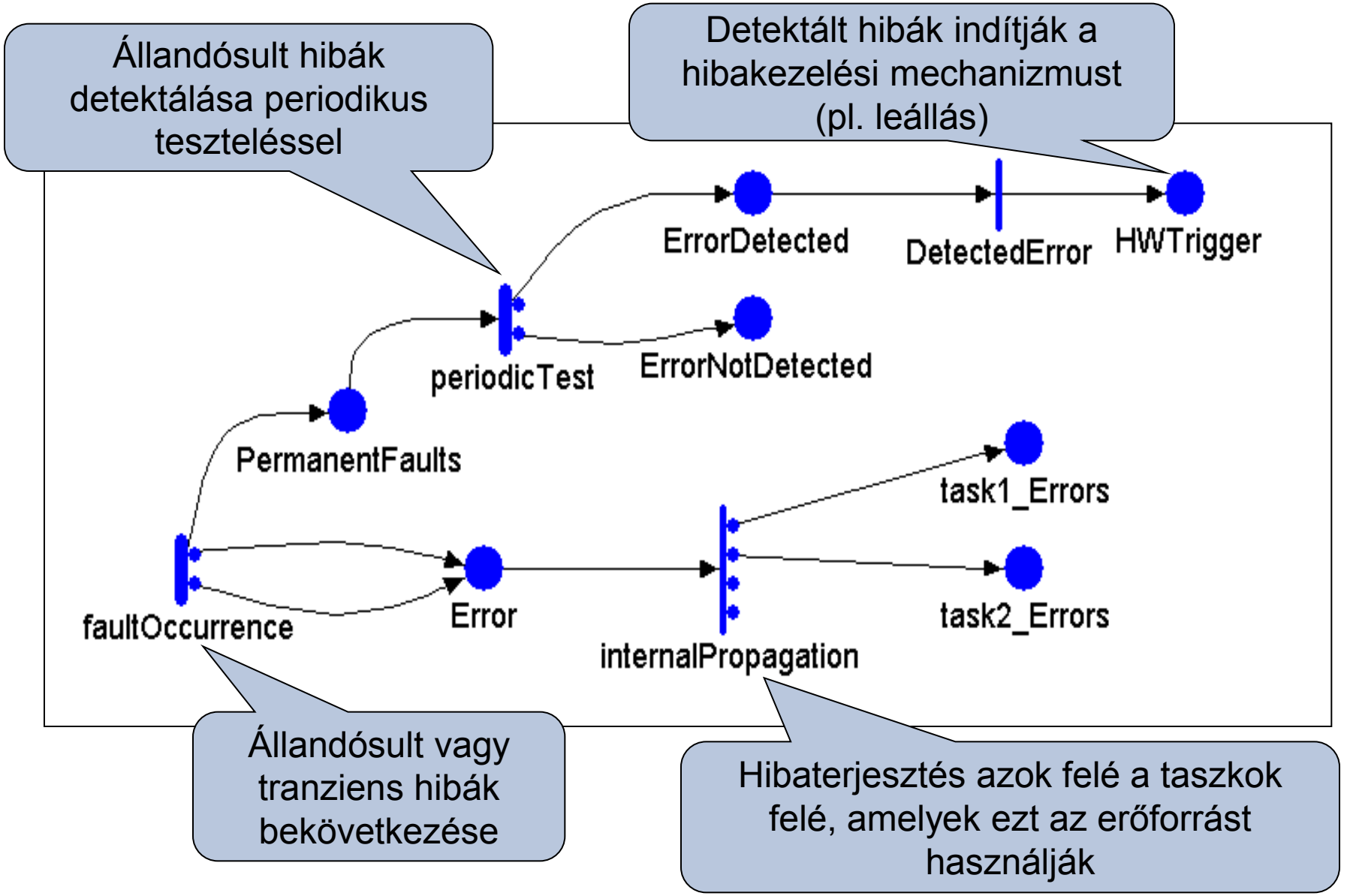
Megbízhatósági modellezés – Tervezői modell

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

Kapcsolat:
- Hibaterjesztési jellemzők:
  * hibaterjesztési valószínűség

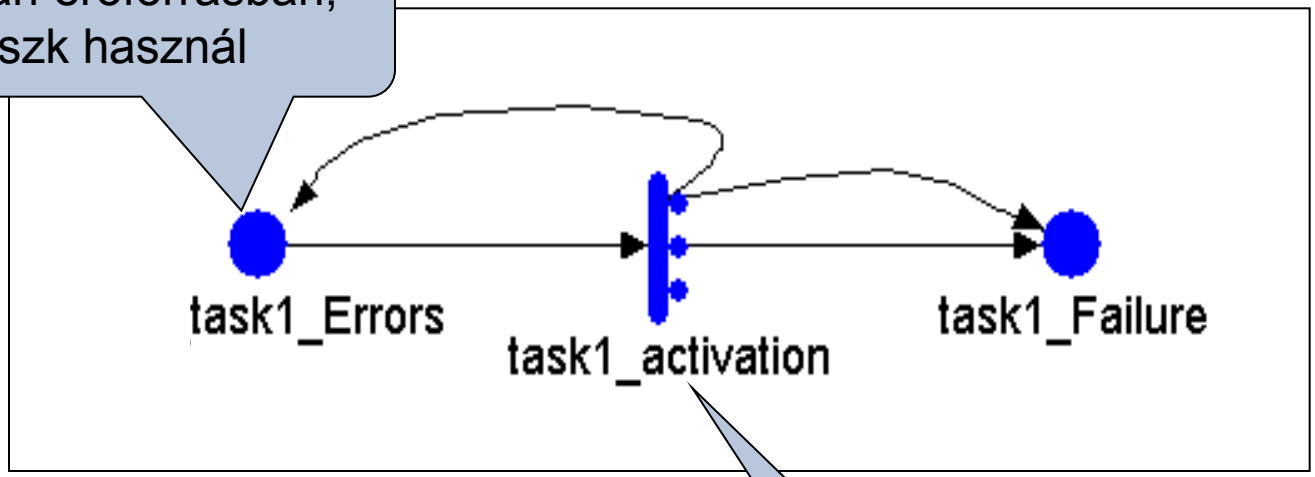


Egy HW erőforrás (pl. memória) analízis modellje



A hibaterjesztés analízis modellje

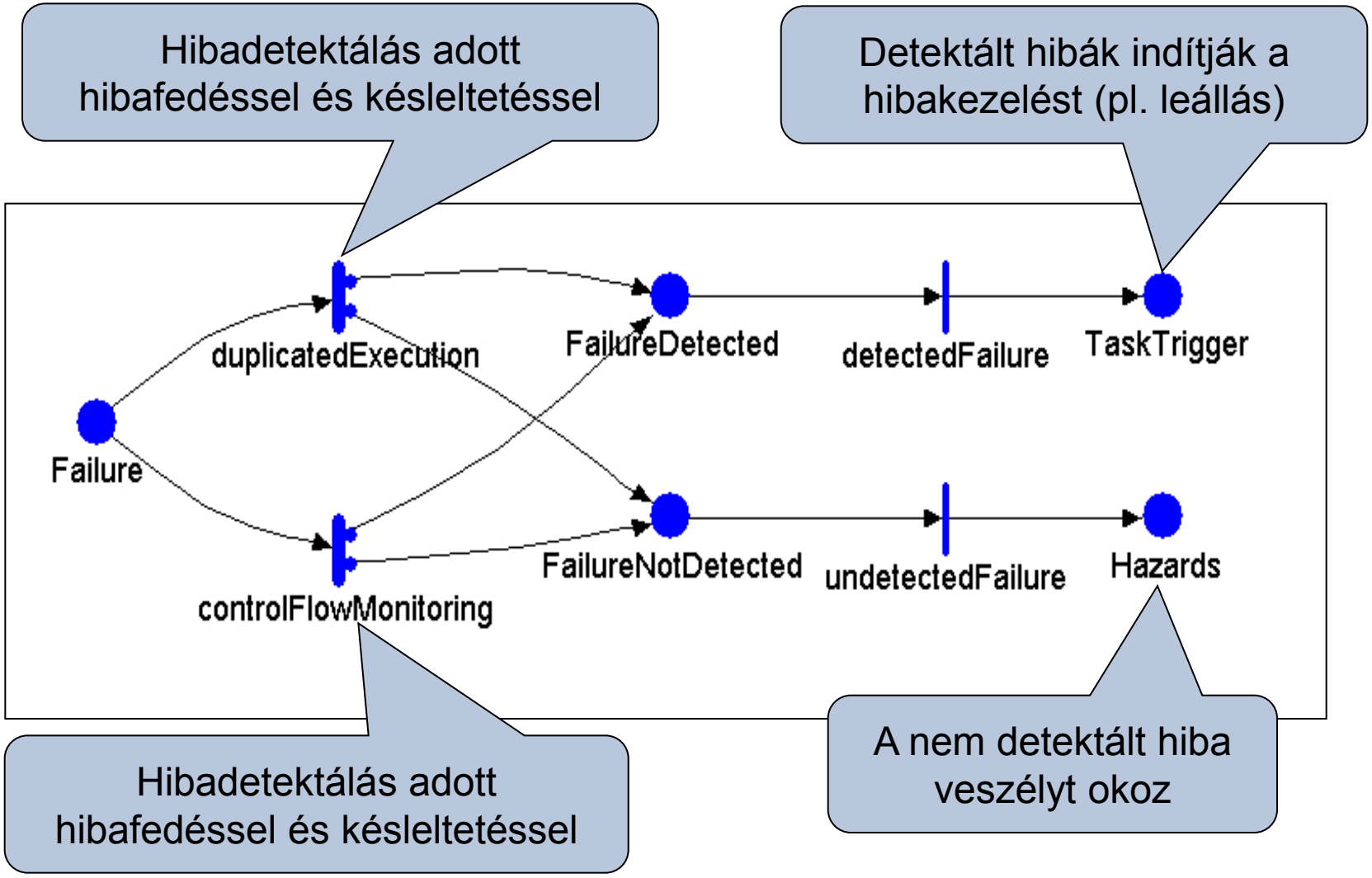
Hiba olyan erőforrásban, amit a taszk használ



Taszk aktiválási gyakoriság, hiba aktiválási lehetőségek:

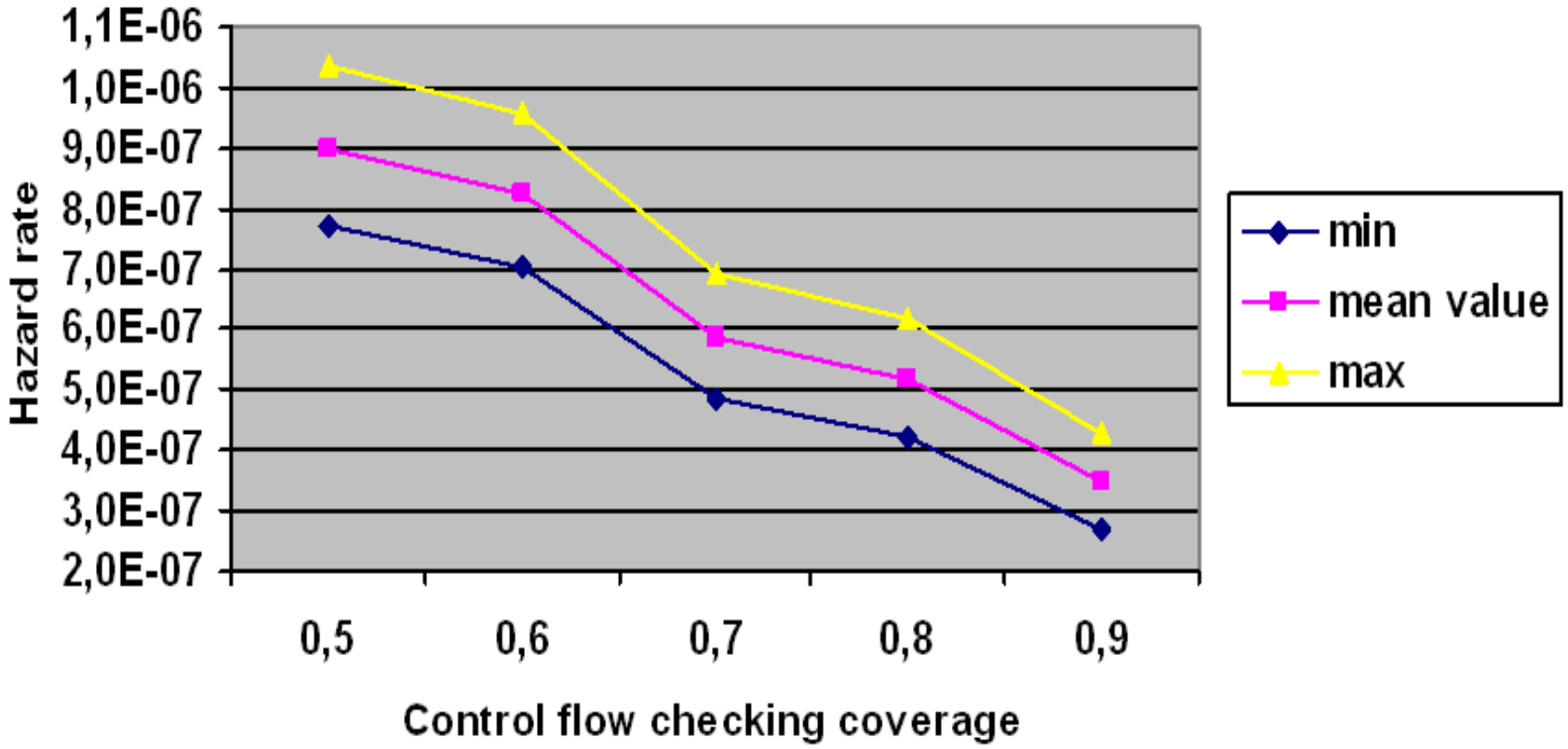
- aktivált hiba, ami a rendszerben marad
- aktivált hiba, felülíródik, de hatása van
- hatás nélkül felülírt hiba

Egy szoftver taszk analízis modellje

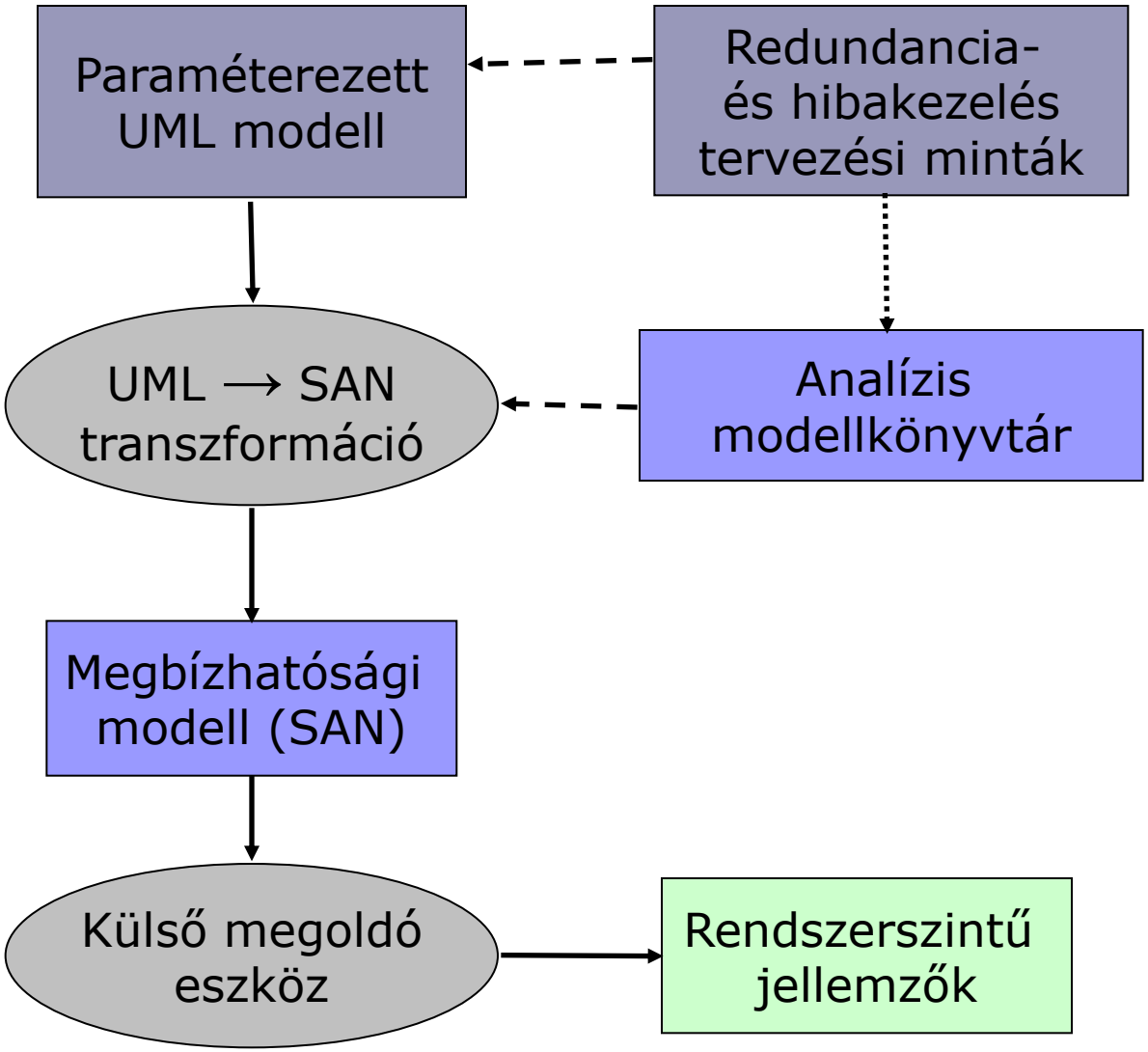


Analízis eredmény (példa)

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



Automatikus analízis eszköz



Automatikus analízis eszköz

Paraméterezett UML modell

UML ⟶ SAN transzformáció

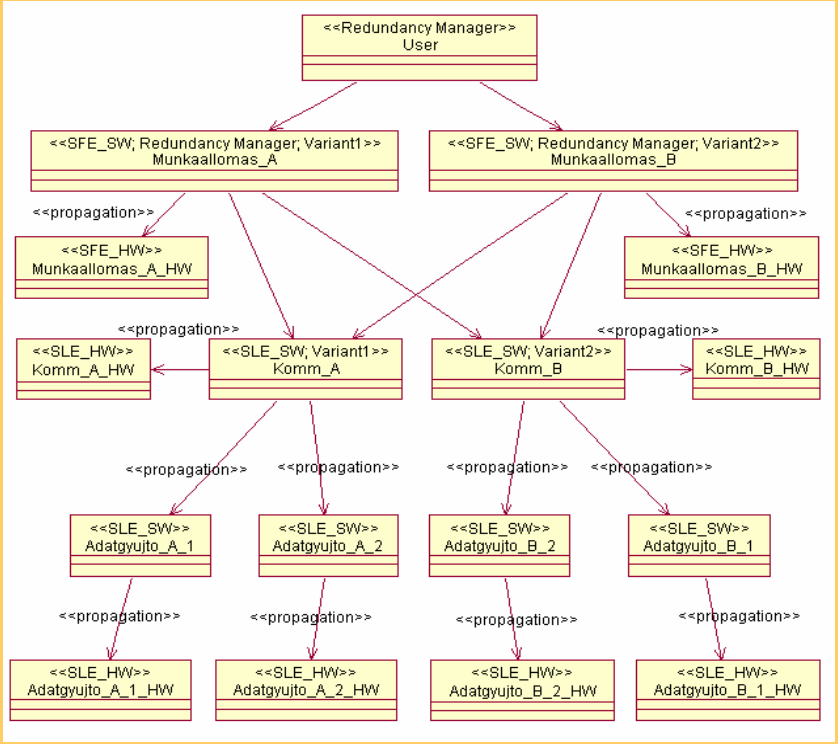
Megbízhatósági modell (SAN)

Külső megoldó eszköz

Ren… jellemzők

Rendszer architektúra modell

UML osztály/objektum diagram



Automatikus analízis eszköz

Paraméterezett UML modell

Redundancia- és hibakezelés tervezési minták

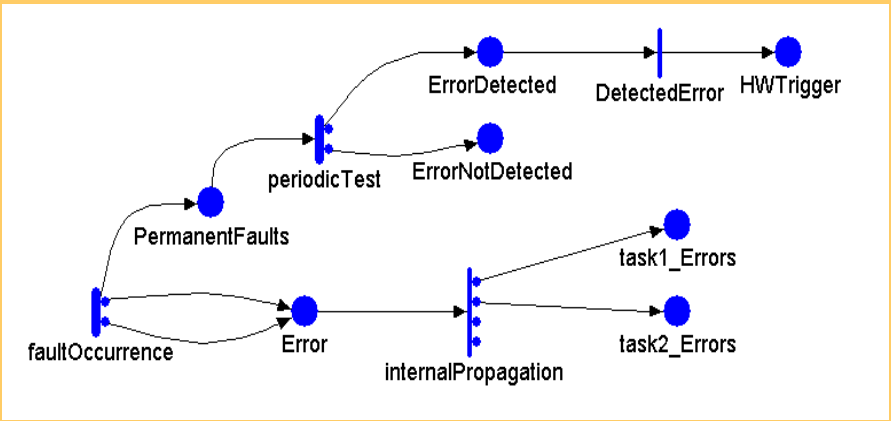
UML ⟶ SAN transzformáció

Analízis modellkönyvtár

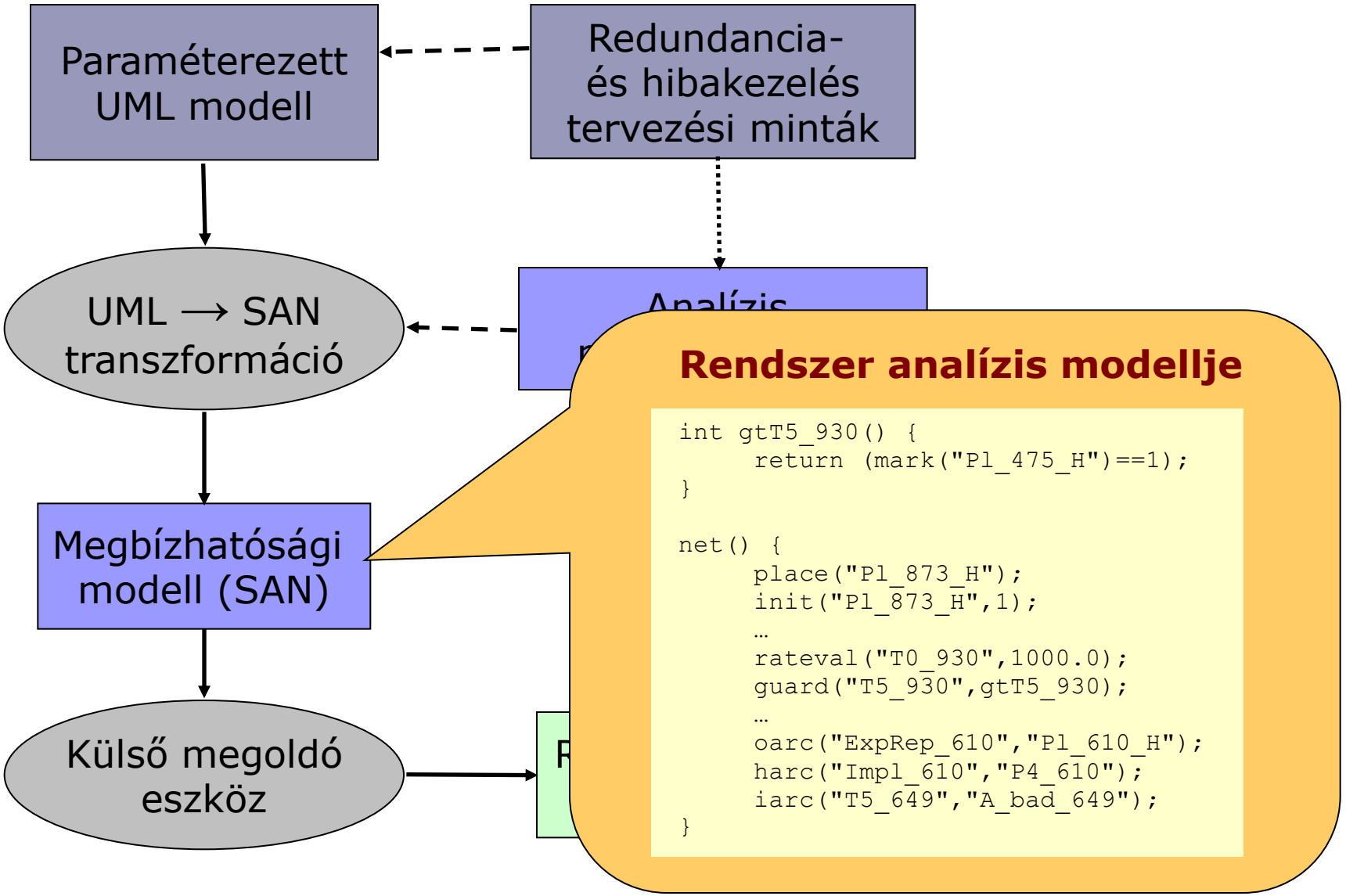
Megbízhatósági modell (SAN)

Külső megoldó eszköz

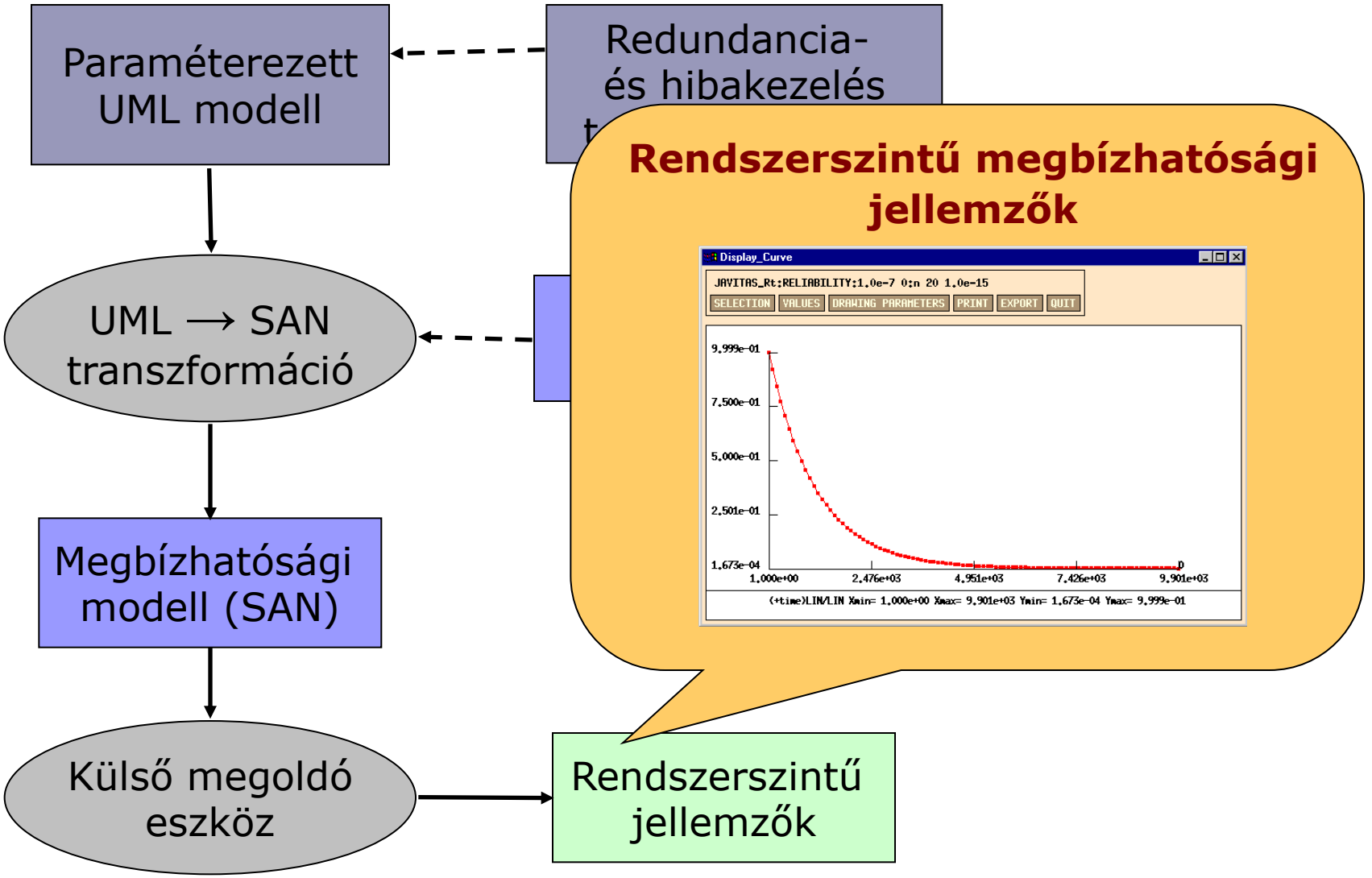
Hardver analízis modellje



Automatikus analízis eszköz



Automatikus analízis eszköz



Tartalomjegyzék

- A fázis ki- és bemenetei
- Az architektúra tervek elkészítése
  - A komponensek azonosítása
  - Mit határoz meg az architektúra?
- Ellenőrzési feladatok
  - Követelményeknek való megfelelőség, követhetőség
  - Hibahatások vizsgálata
  - Extra-funkcionális követelmények vizsgálata
- Teszt tervezés

Szoftver-hardver integrációs tesztterv

- Dokumentálandó:
  - Teszt esetek, típusok
  - Teszt környezet (eszközök, konfiguráció leírás)
  - Teszt kritériumok (teljes elvégzés megítélhető)
- Tevékenységek
  - Telepítés és rendszerintegráció megkülönböztetve
  - Telephelyi és alkalmazói tesztek elkülönítése
  - Teszt esetek és eredmények gépi rögzítése
- Teszt módszerek: Ld. az integrációs fázistól!
  - Integrációs tesztek
  - Rendszertesztek

Összefoglalás

- A fázis ki- és bemenetei
- Az architektúra tervek elkészítése
  - A komponensek azonosítása
  - Mit határoz meg az architektúra?
- Ellenőrzési feladatok
  - Követelményeknek való megfelelőség, követhetőség
  - Hibahatások vizsgálata
  - Extra-funkcionális követelmények vizsgálata
- Teszt tervezés