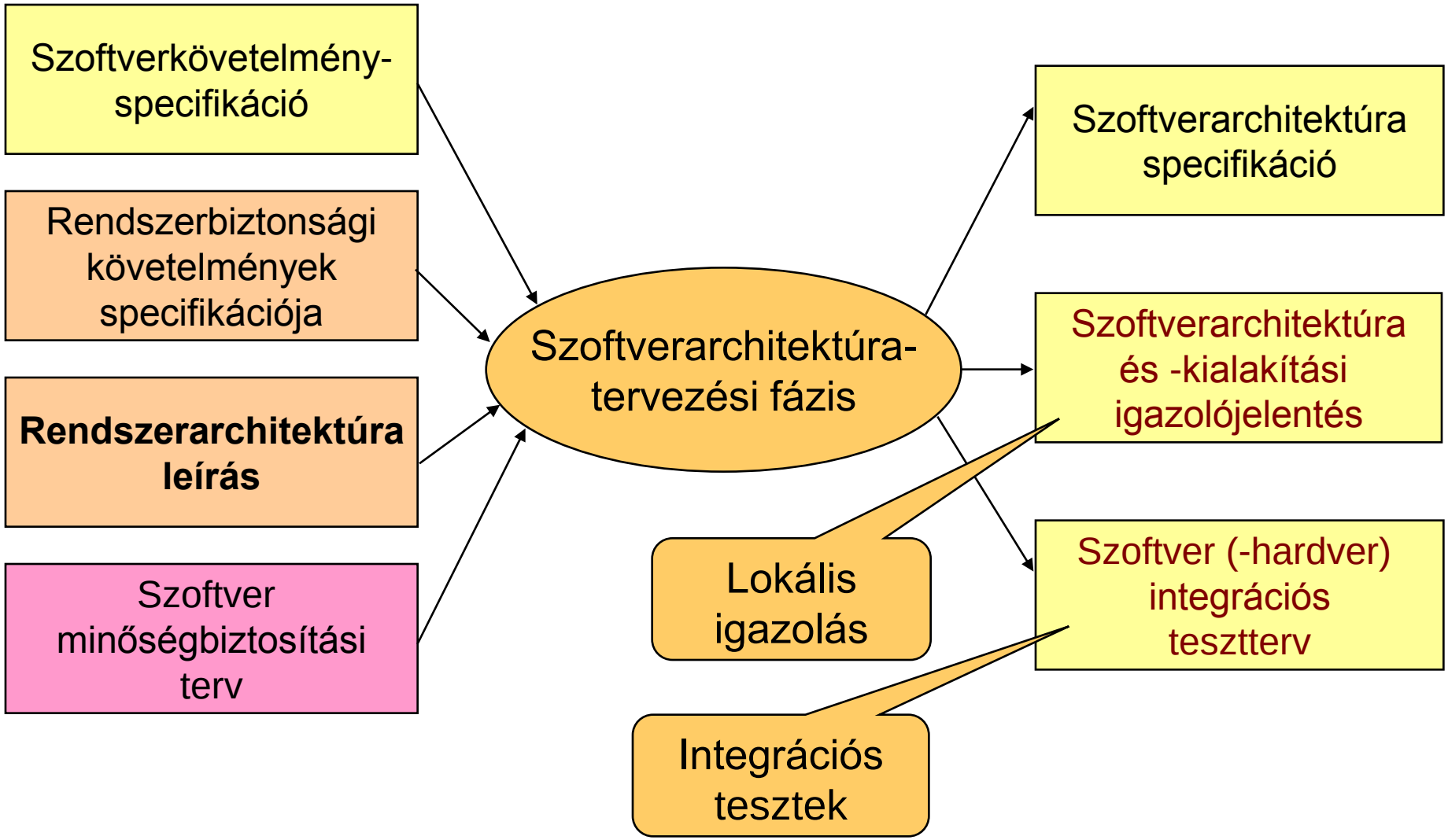
# Szoftver architektúra tervek ellenőrzése

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
- 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, OTS komponensek)

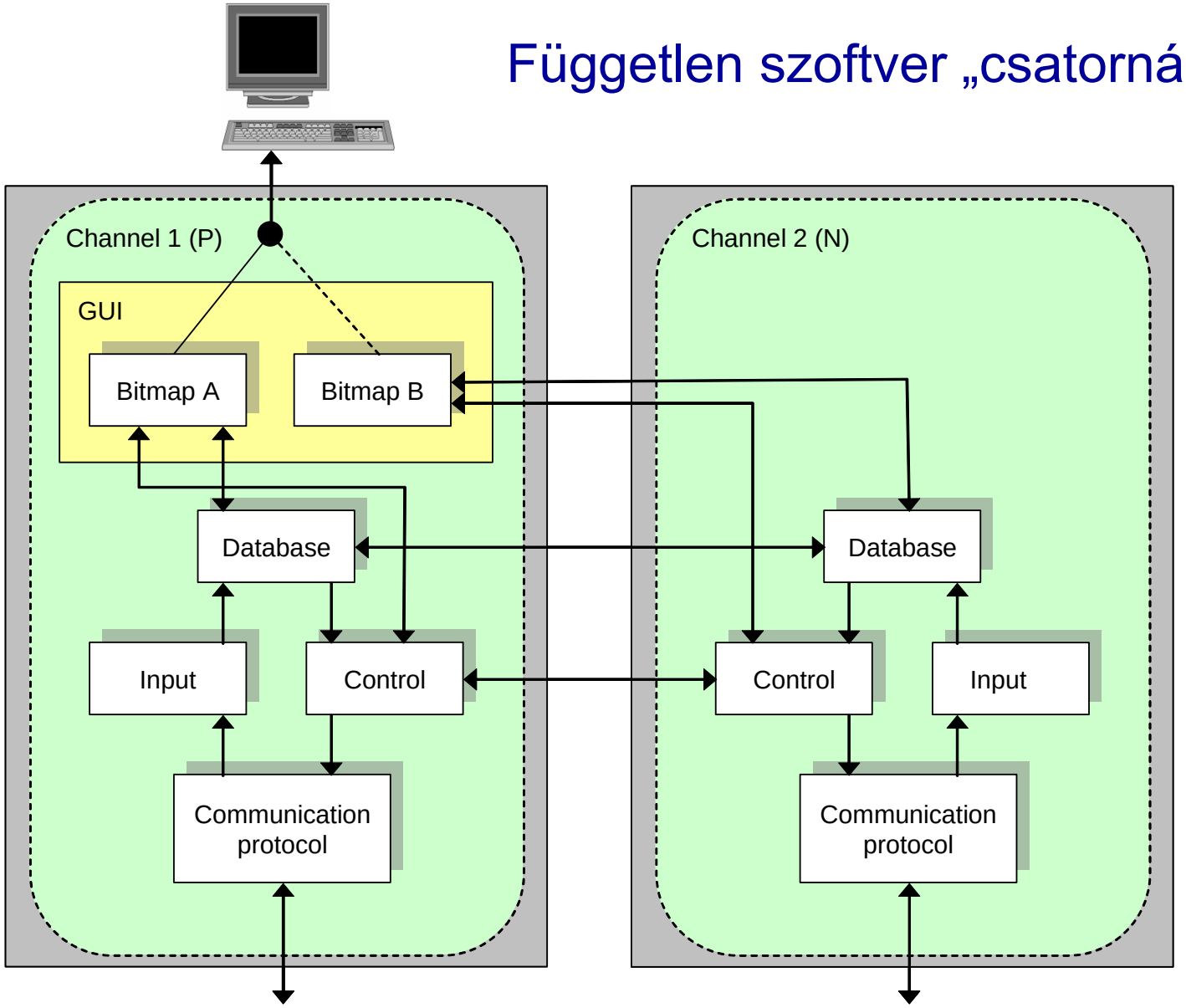
# Példa: OTS komponensek kritikus rendszerekben

- 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

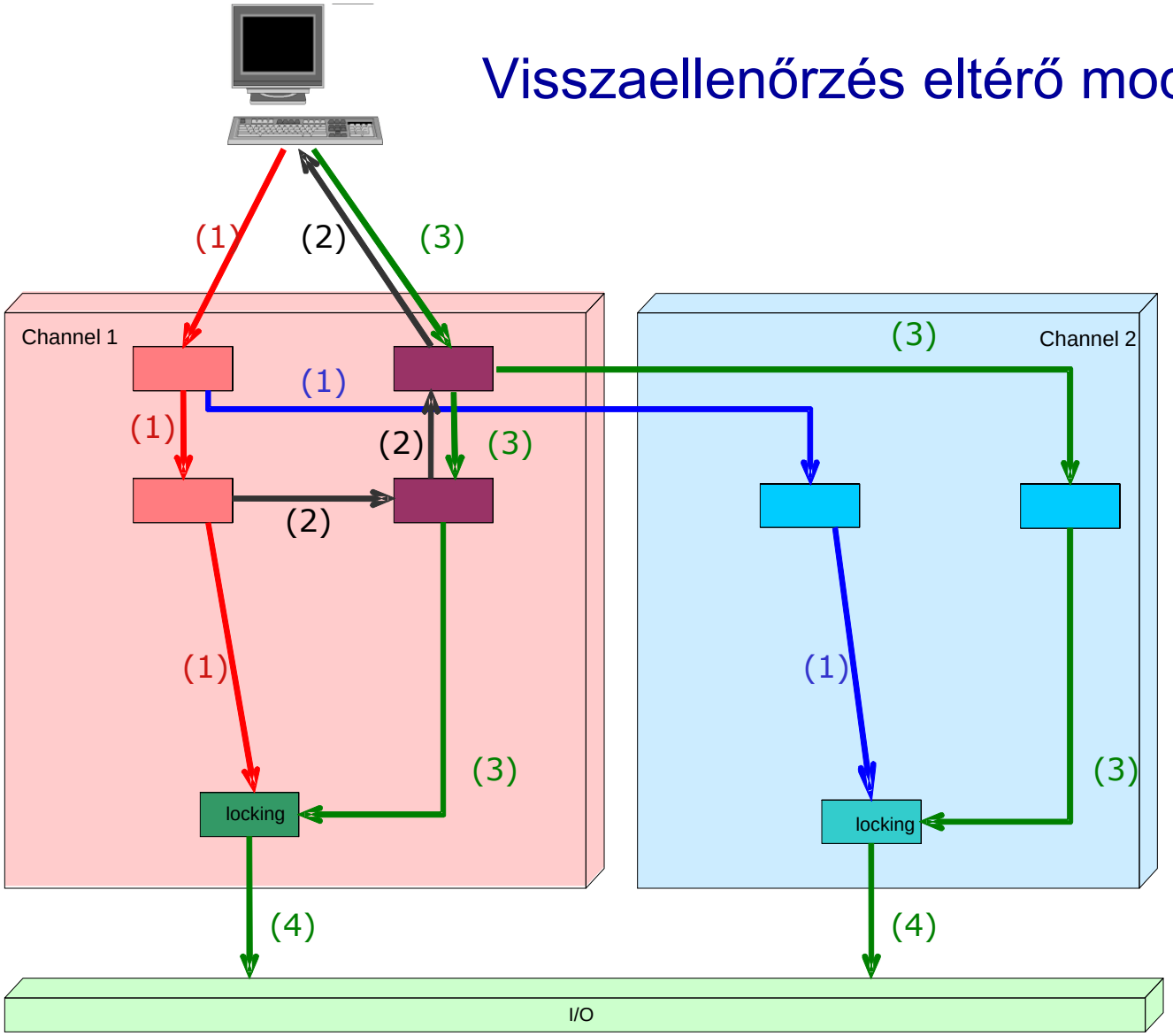
# Példa: Architekturális megoldás SIL csökkentésre

## Független szoftver „csatornák”



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

Sokféle kombináció megengedett

Failure assertion

Referencia

Bennmaradó hibák elleni védekezés

# 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, behatolás felderítés, helyreállítás |
| Tesztelhetőség | Vezérelhetőség, megfigyelhetőség |
| Karbantarthatóság | Elkülönítés (pl. MVC minta) |

# Tartalomjegyzék

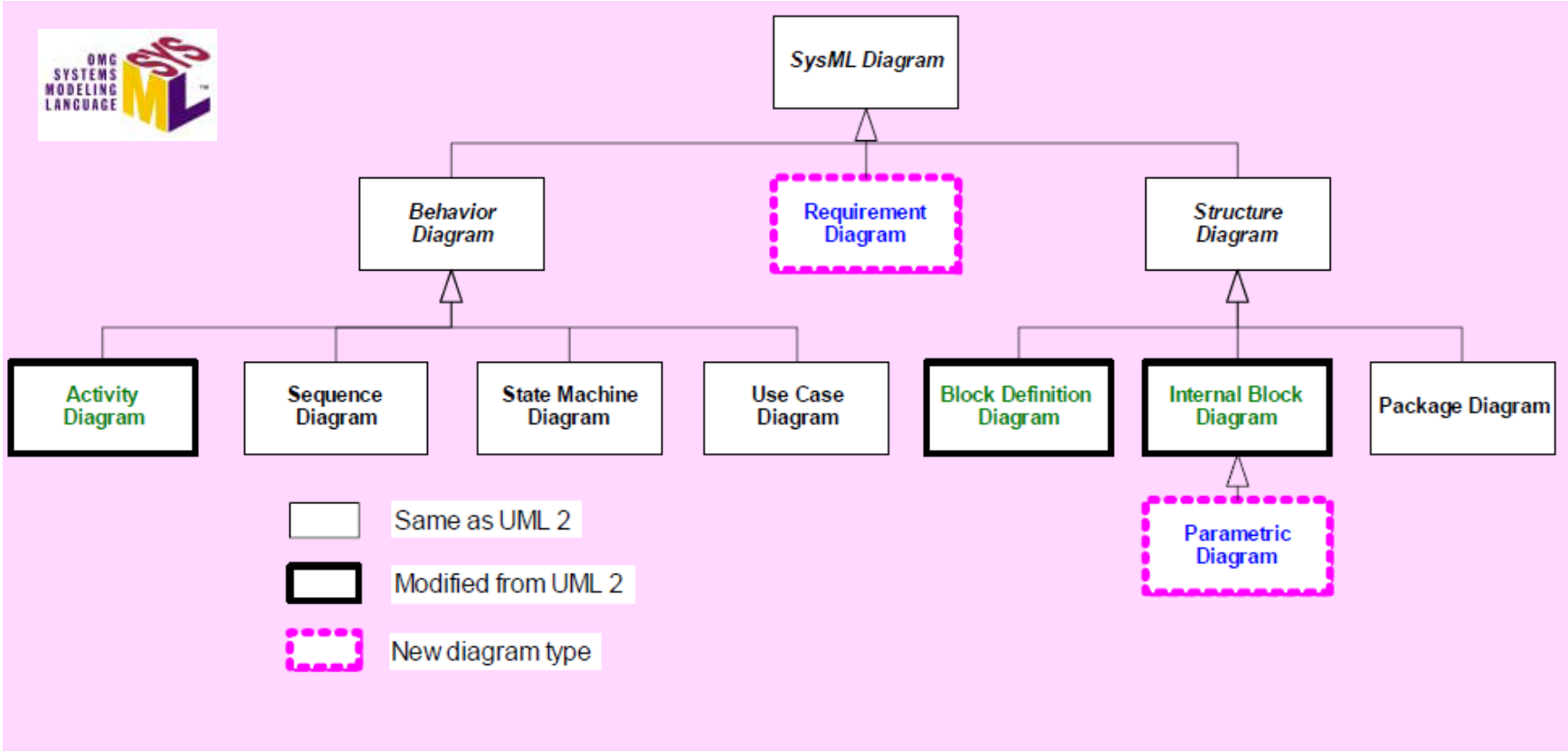
- A fázis ki- és bemenetei
- Az architektúra tervek elkészítése
  - A komponensek azonosítása
  - Mit határoz meg az architektúra?
- Ellenőrzési feladatok
  - Követelményeknek való megfelelőség, követhetőség
  - Hibahatások vizsgálata
  - Extra-funkcionális követelmények vizsgálata
- Teszt tervezés

# A szoftverarchitektúra ellenőrzési technikái (áttekintés)

- Követhetőség a követelmények felől
  - Követelmények „lefedése” architektúra elemekkel
- Szisztematikus elemzések: architektúra „bejárás”
  - Interfész analízis
    - Elvárt és nyújtott interfészek megfelelősége
  - Hibahatás analízis kombinatorikus módszerekkel
    - Komponens szintű hibák $\leftrightarrow$ Rendszerszintű hatások
- Modell alapú elemzések: 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

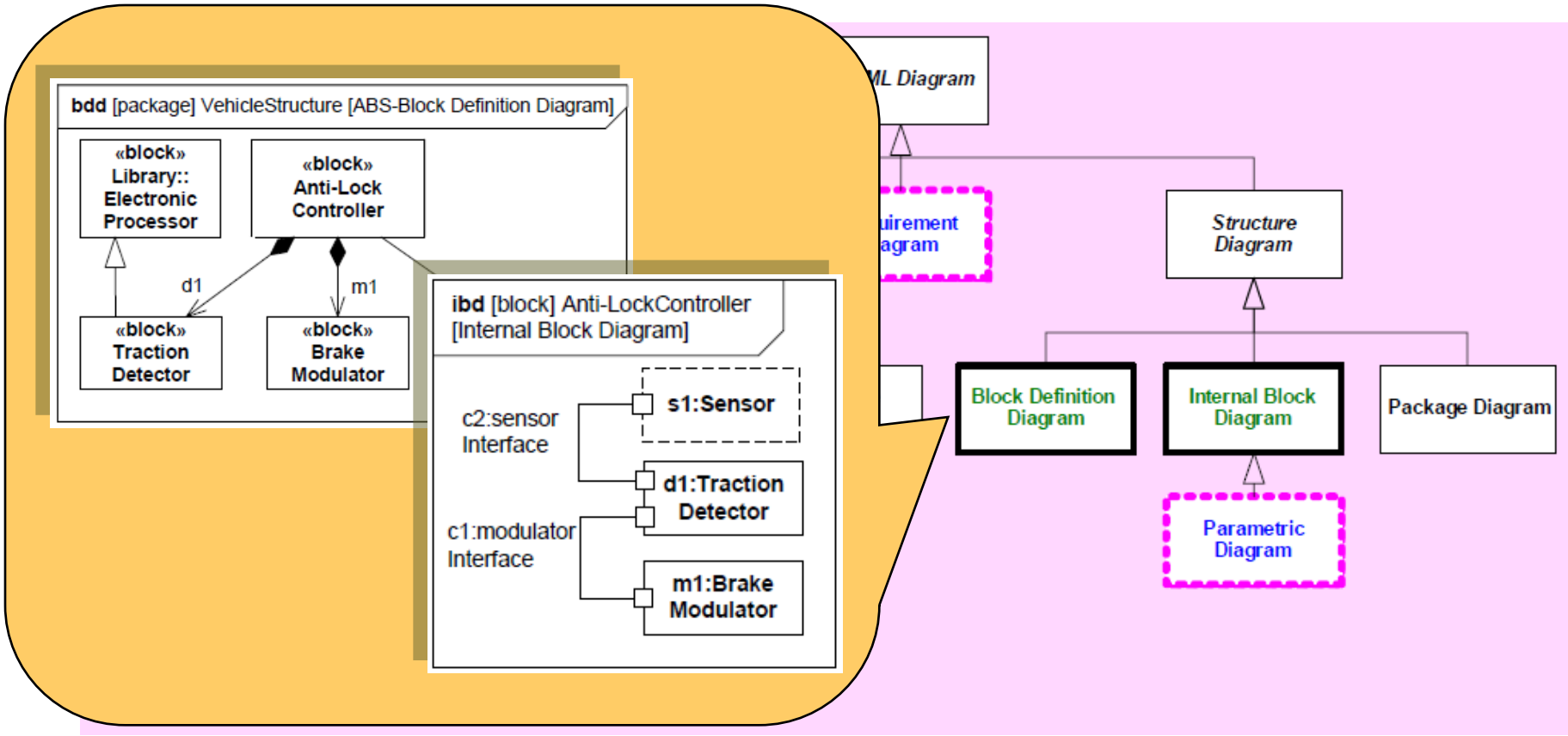
# Az architektúra terv igazolása: Követhetőség

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

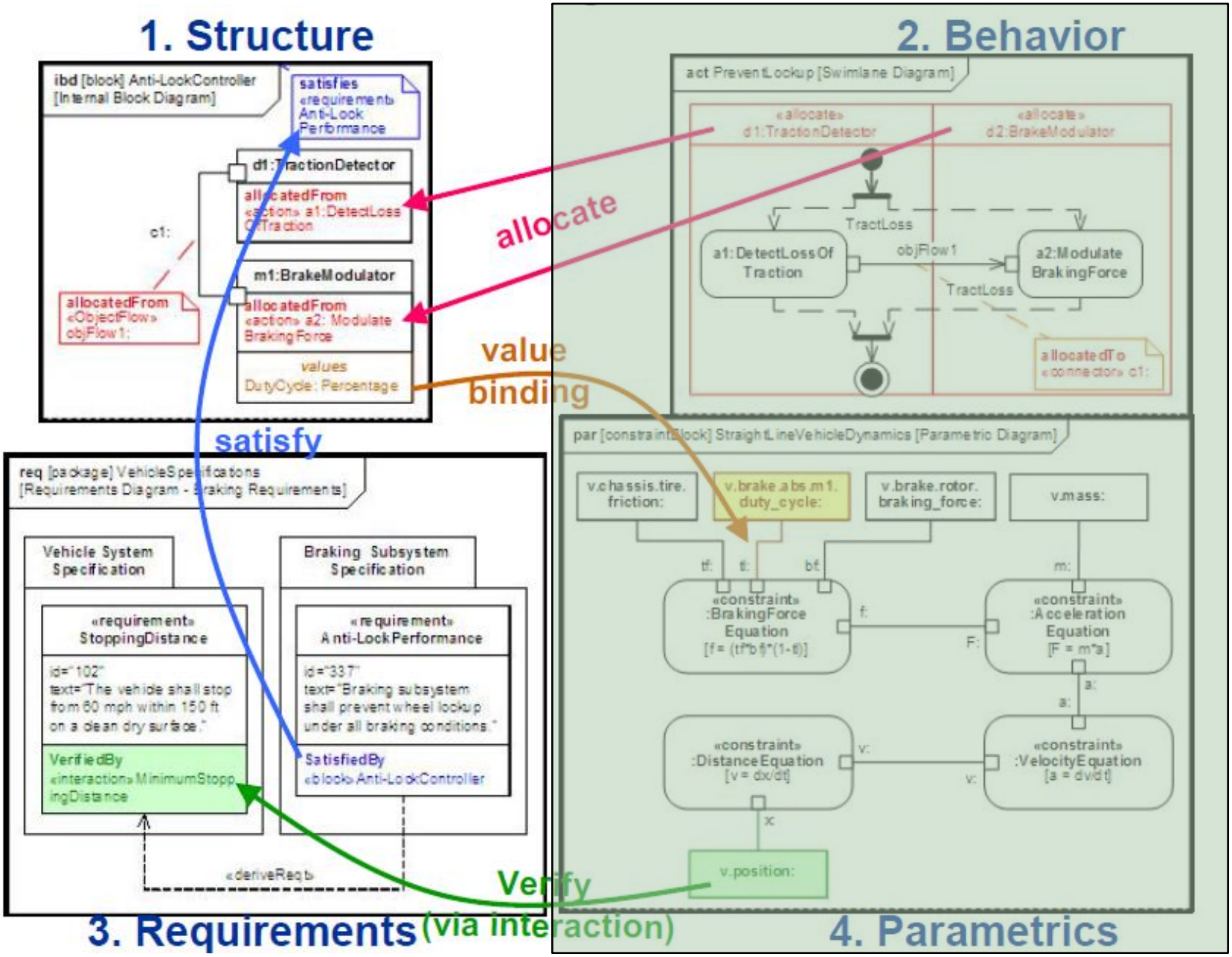


# Az architektúra terv igazolása: Követhetőség

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



# Relációk explicit megjelenítése és ellenőrzése



# 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

# 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)

# 1. Hibafa analízis

## Rendszerszintű veszély okainak vizsgálata

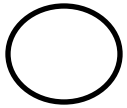
- Tipikusan felülről lefelé haladó analízis
- Felderíti a kezelendő hibaokokat és -kombinációkat

## Hibafa konstrukció:

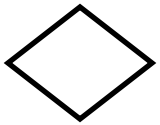
- (Rendszerszintű) veszély, veszélyes állapot:
  Azonosítás: környezet, követelmények, szabványok
- Közbenső események, pszeudo-események:
  Veszélyhez vezetnek,
  Boole-logikai kombinációi (AND, OR) alacsonyabb szintű eseményeknek
- Elsődleges események: további felbontás nincs

# Hibafa grafikus elemkészlet

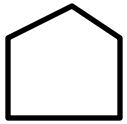
legfelső szintű vagy közbenső esemény



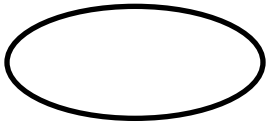
elsődleges (alapszintű) esemény



tovább nem vizsgált esemény



normál esemény (nem hiba vagy veszély)

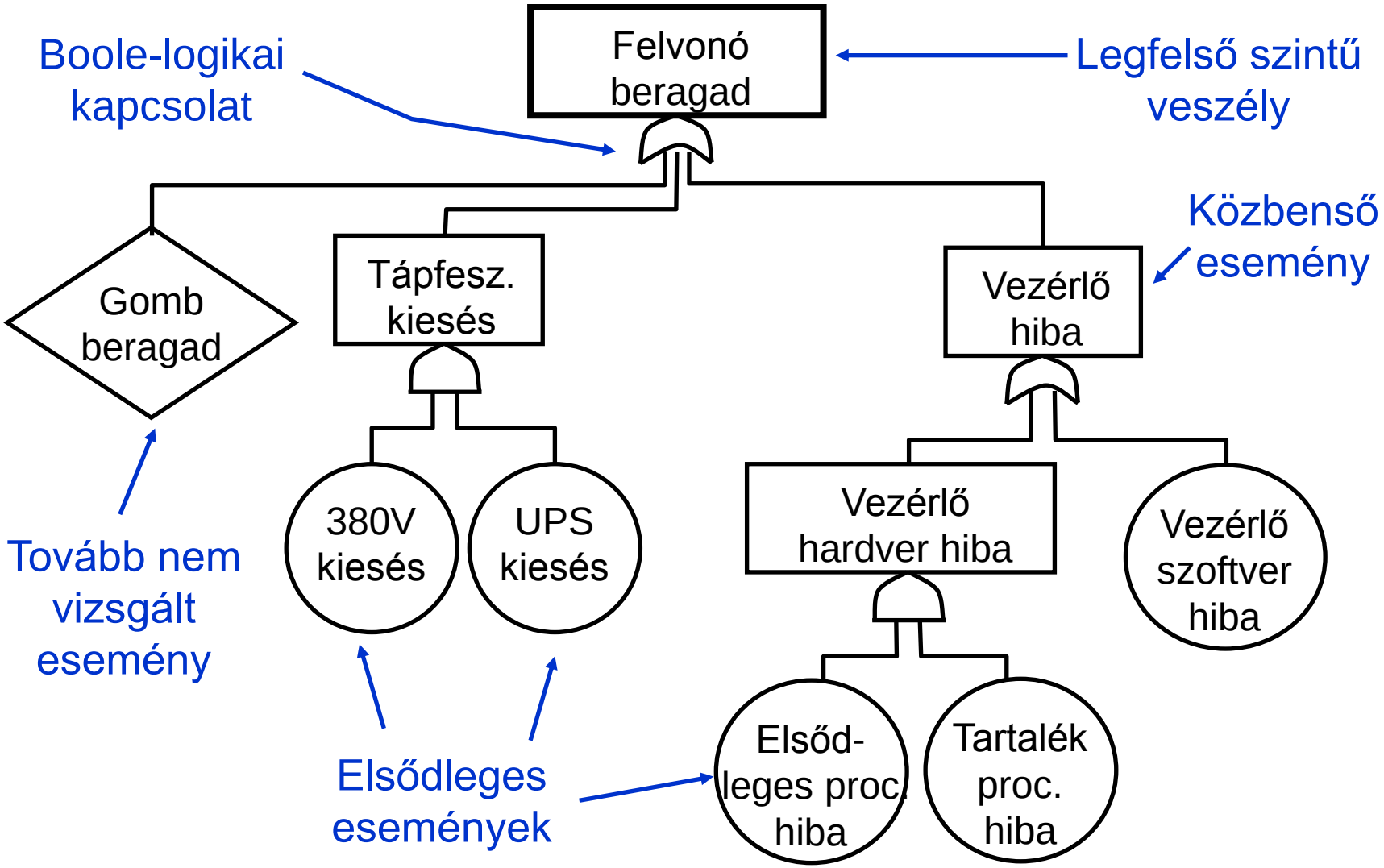


feltétel egy összetett esemény bekövetkezéséhez

ÉS kapu

VAGY kapu

# Hibafa példa: Felvonó



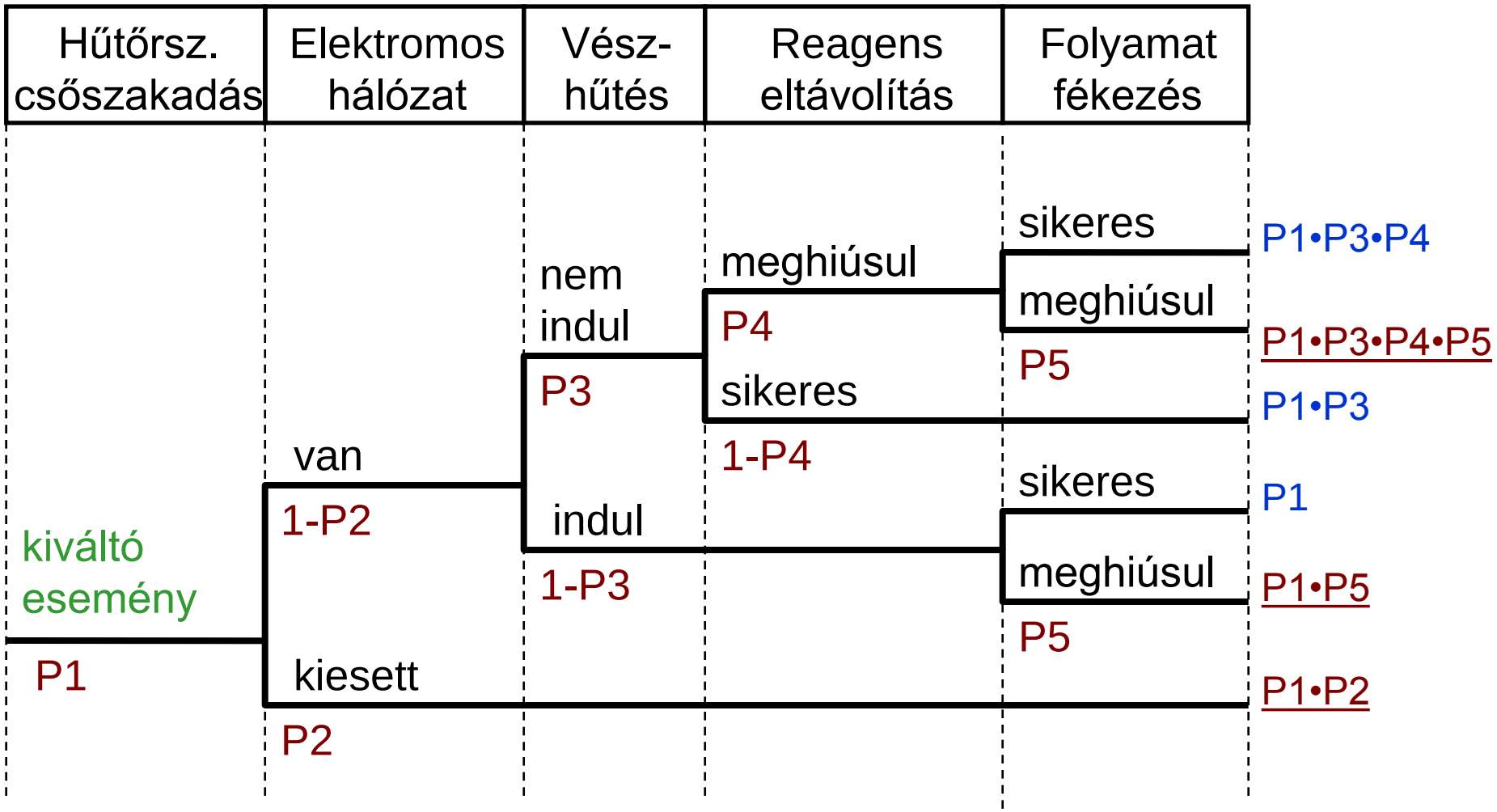
# Hibafa analízis

- 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

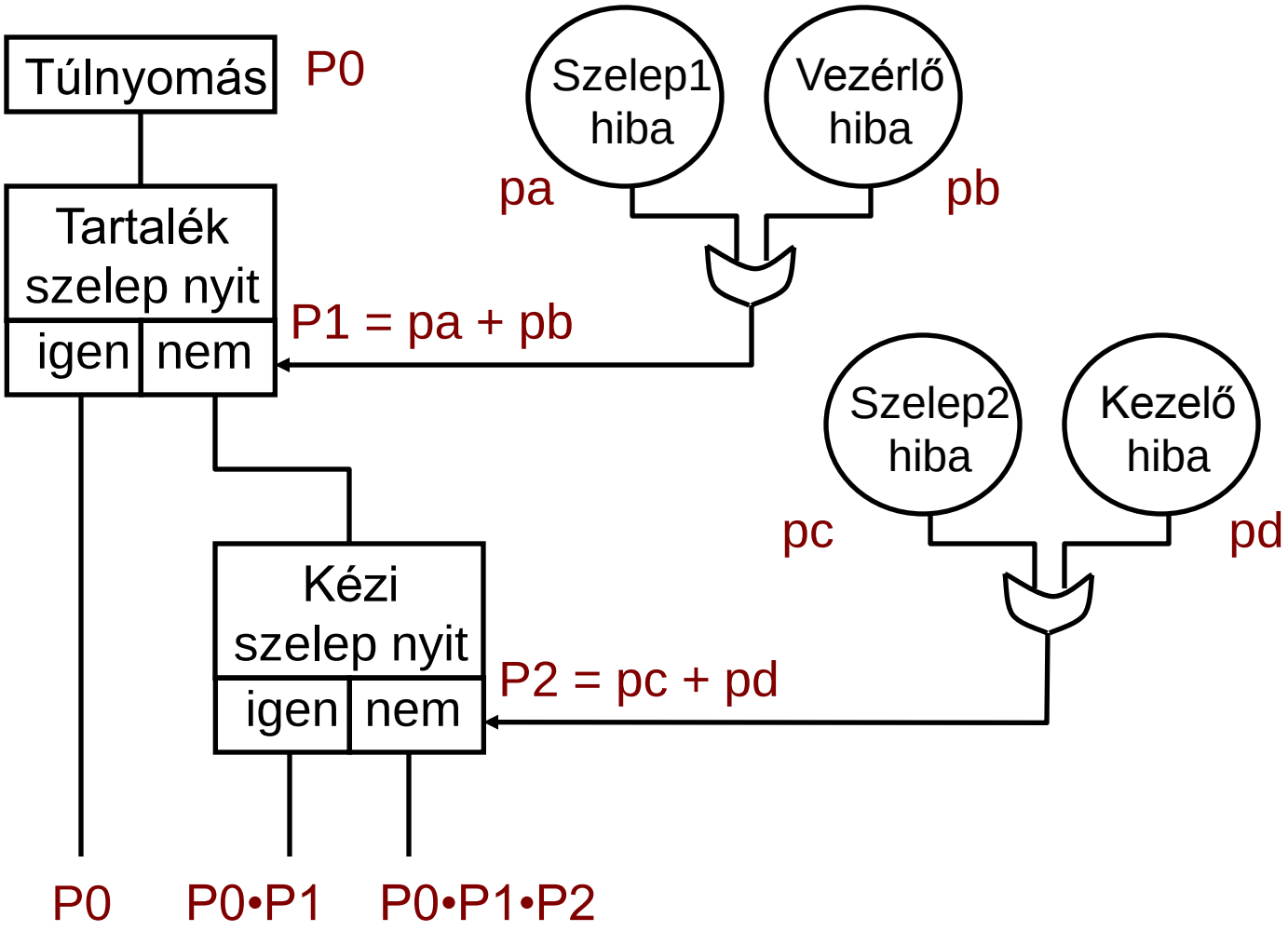
# Eseményfa példa: Reaktorhűtés



# 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<br>≤ L nem megy át | 65%<br>35% | - túlnyomás<br>- 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 |

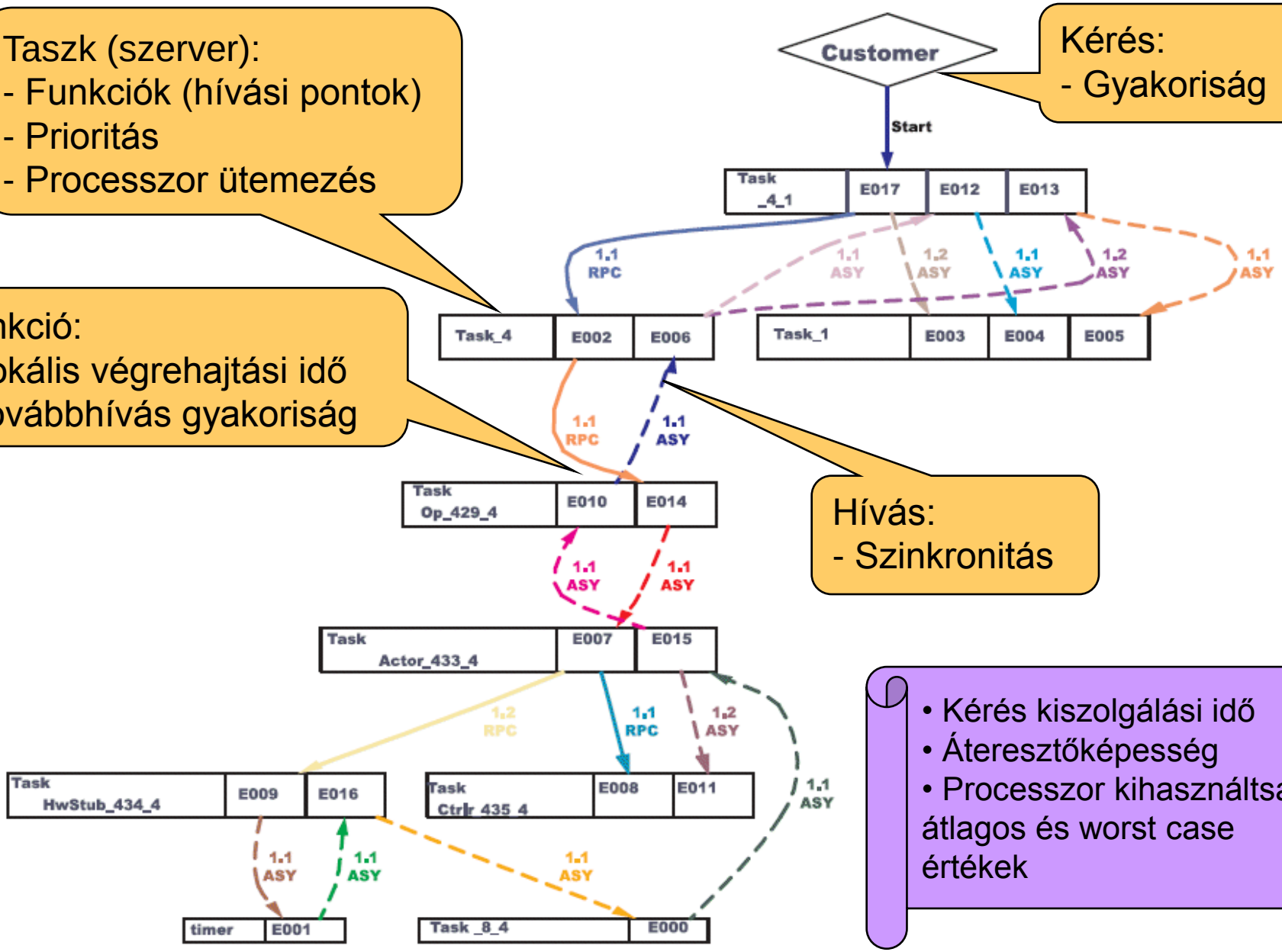
# Teljesítmény modellezés (LQN)

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

Kérés:
- Gyakoriság

Funkció:
- Lokális végrehajtási idő
- Továbbhívás gyakoriság

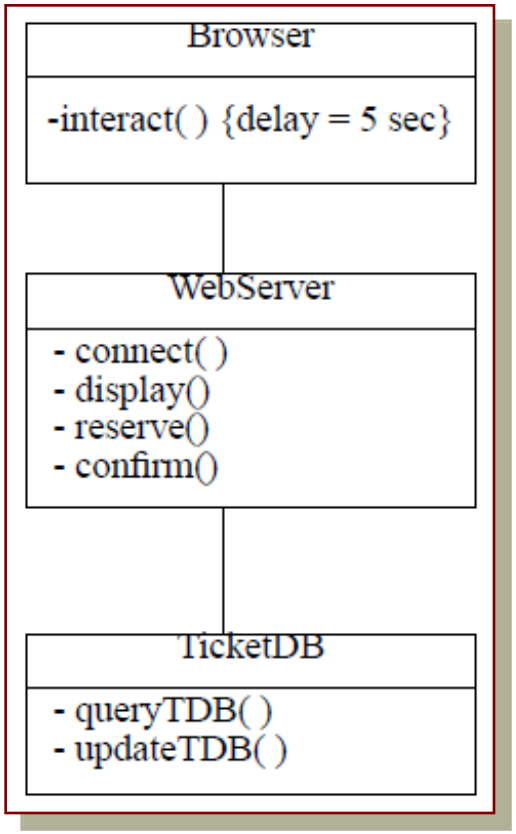
Hívás:
- Szinkronitás

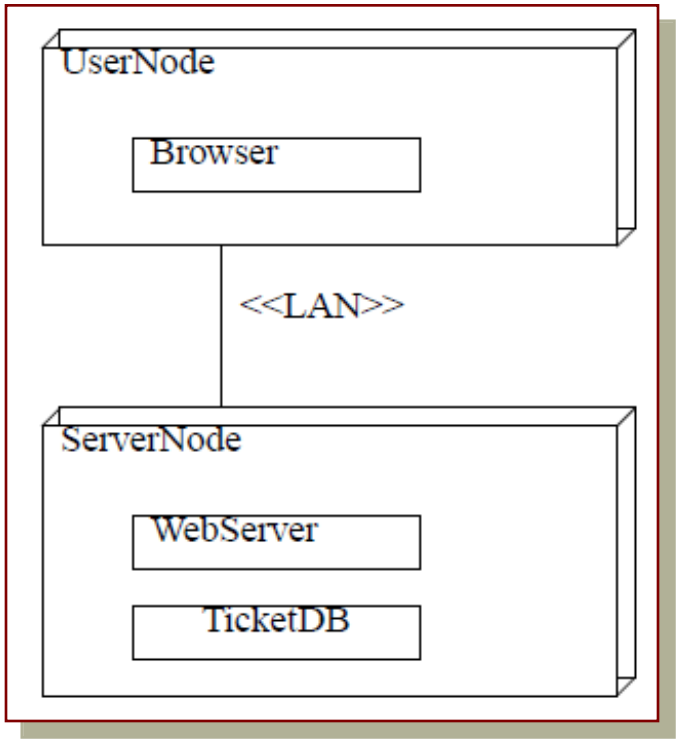


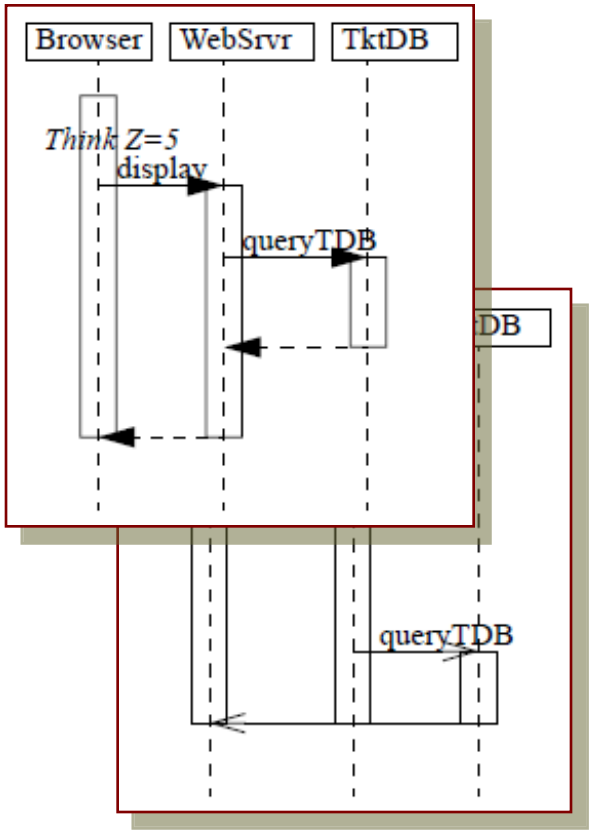

- Kérés kiszolgálási idő
- Áteresztőképesség
- Processzor kihasználtság

átlagos és worst case értékek

# Az architektúra leképezése teljesítménymodellre



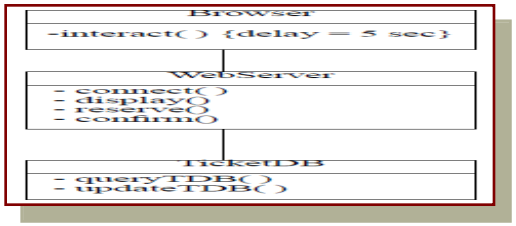



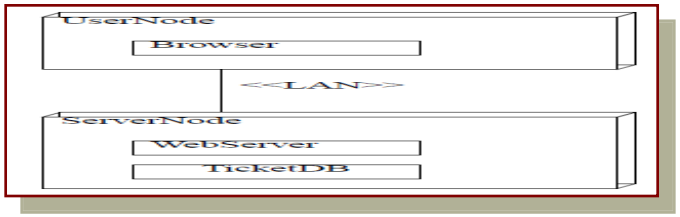

Osztályok Telepítés Interakciók

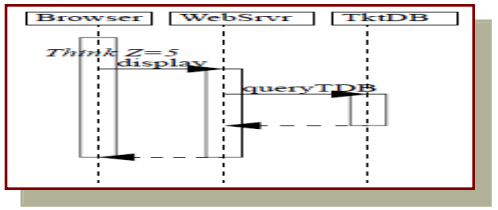
Lokális teljesítmény paraméterek megadása:

- Attribútumok (konvenció alapján értelmezhető)
- UML tagged value

# Az architektúra leképezése teljesítménymodellre



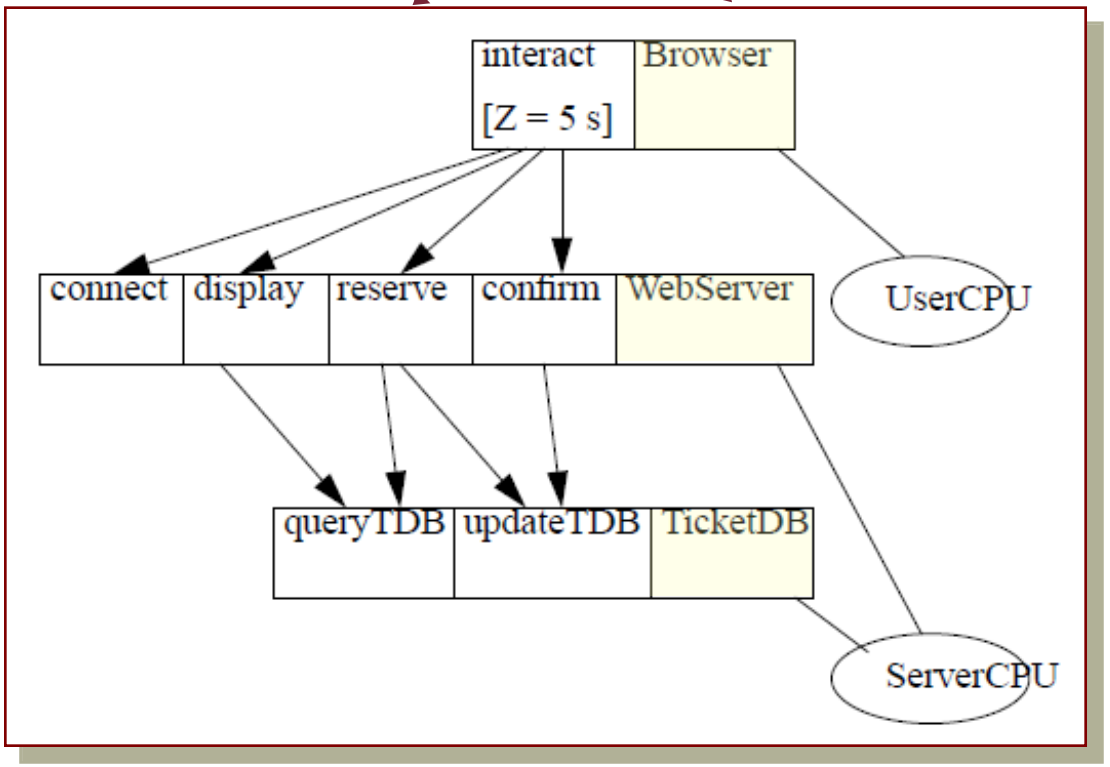




Osztályok

Telepítés

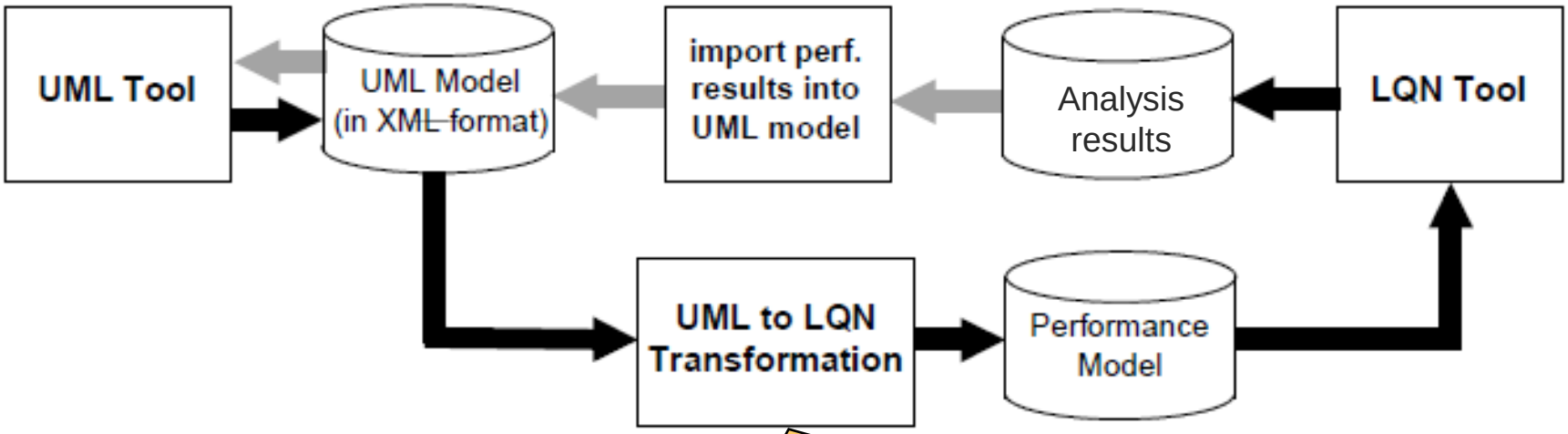
Interakciók

Modell-
transzformáció



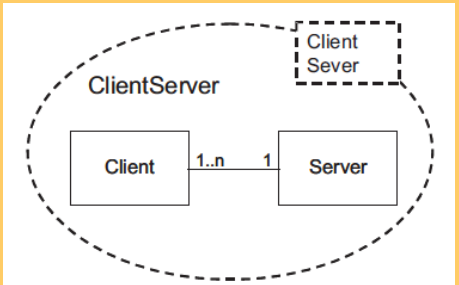

LQN
teljesítmény-
modell

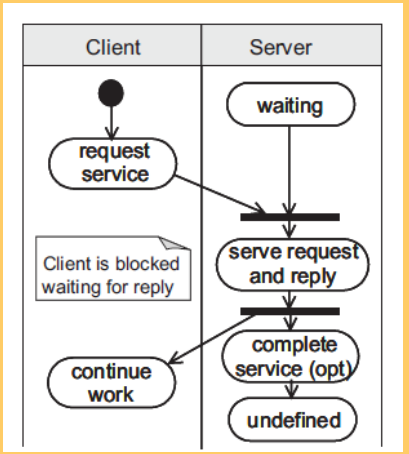
# Az architektúra leképezése teljesítménymodellre




- Architektúra minták használata

Szinkron üzenetküldés



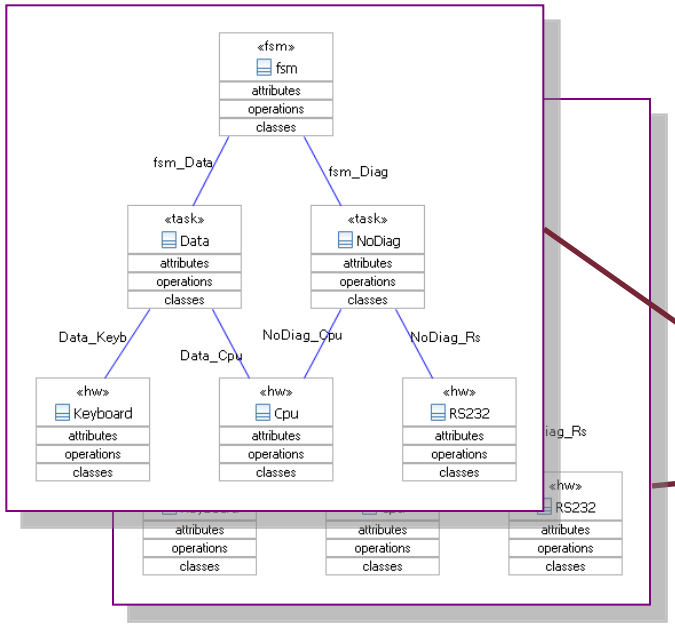



# Második példa: Megbízhatósági modellezés

| | **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 |

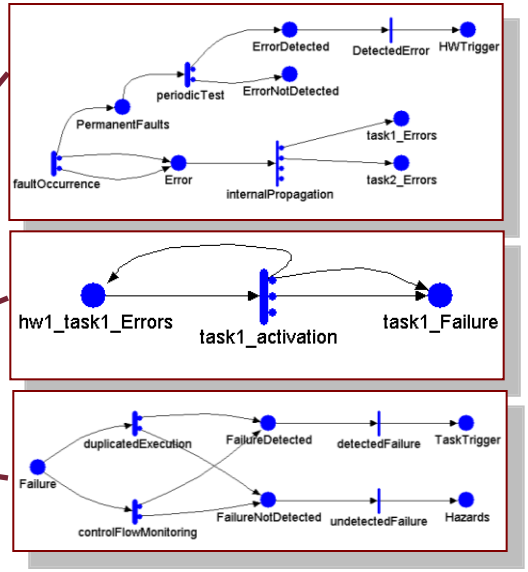
# UML alapú megbízhatósági modellezés

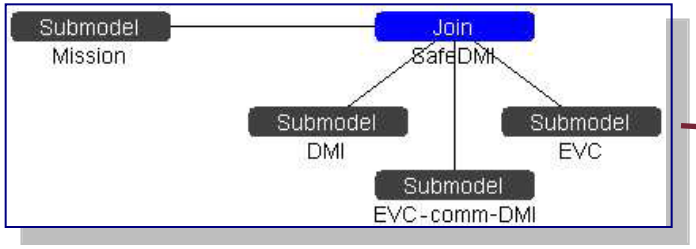
## UML architektúra modell



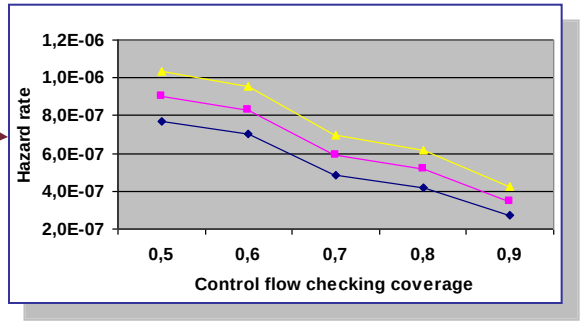

## Megbízhatósági modell konstrukció

## Analízis alhálók




## Rendszerszintű megbízhatósági modell (sztochasztikus aktivitás hálózat)

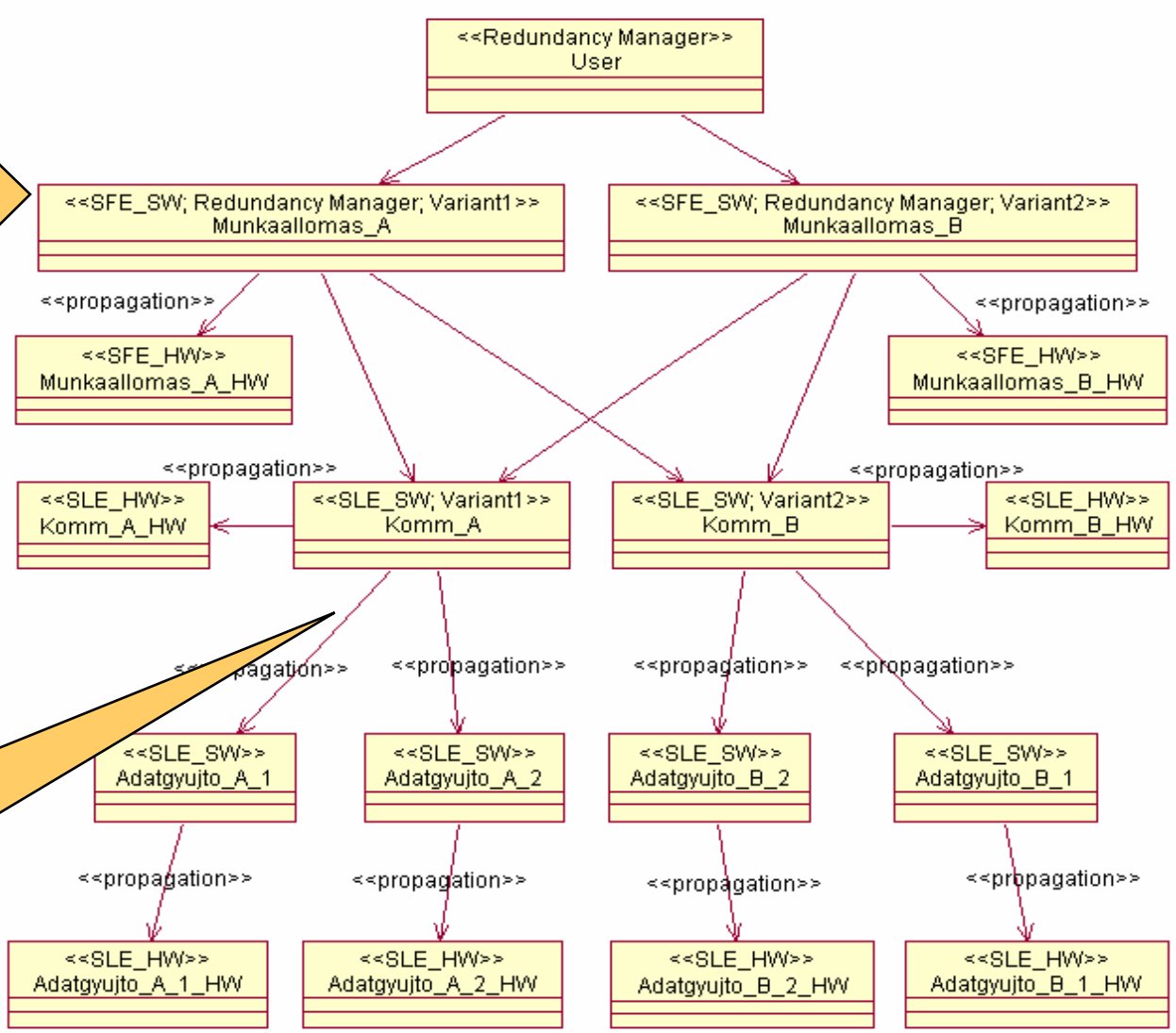



## Analízis eredmények

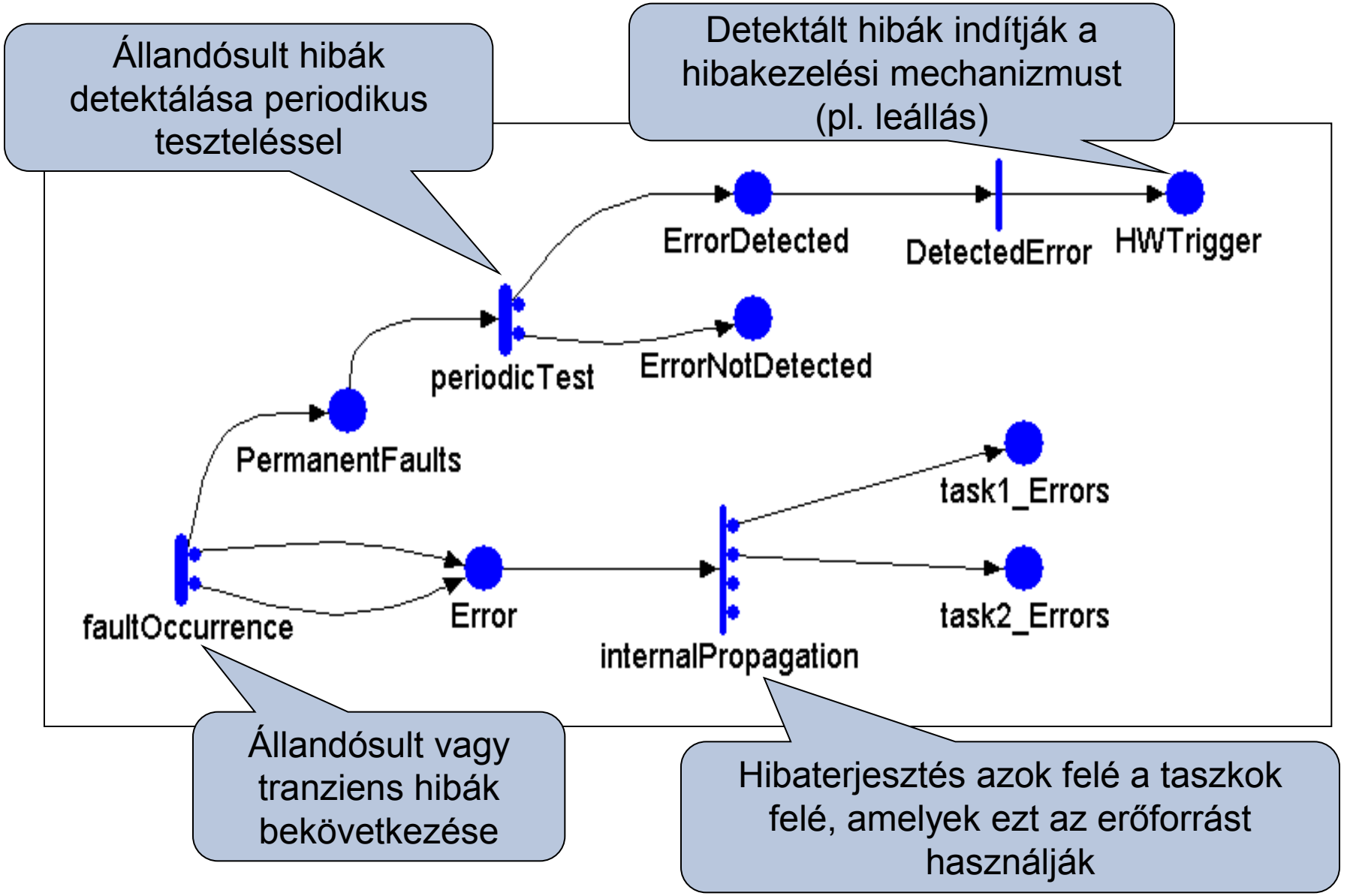
# Megbízhatósági modellezés – Tervezői modell

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

Kapcsolat:
- Hibaterjesztési jellemzők:
  * hibaterjesztési valószínűség

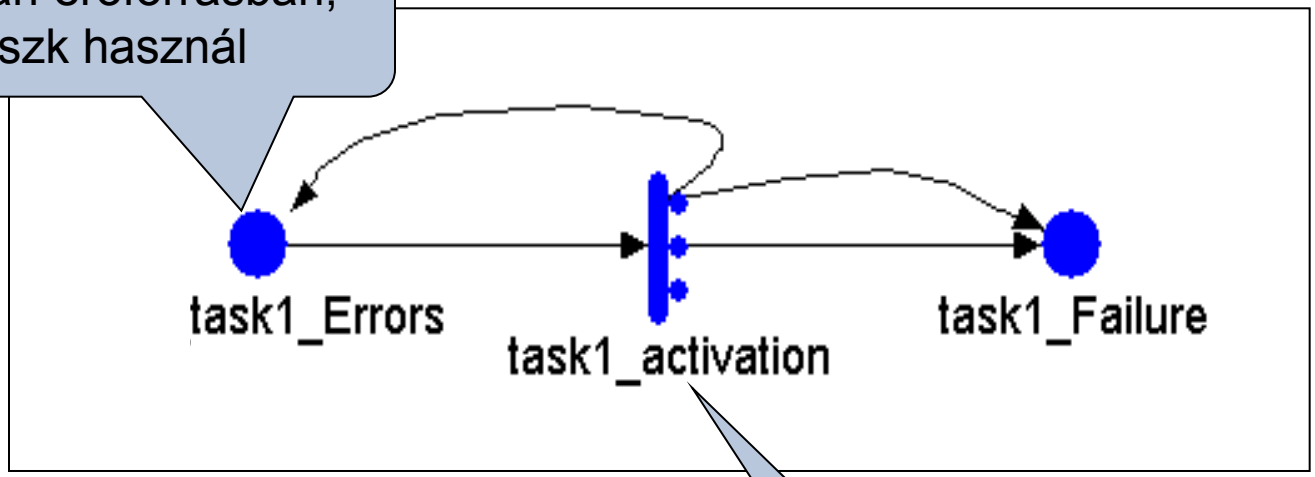


# Egy HW erőforrás (pl. memória) analízis modellje



# A hibaterjesztés analízis modellje

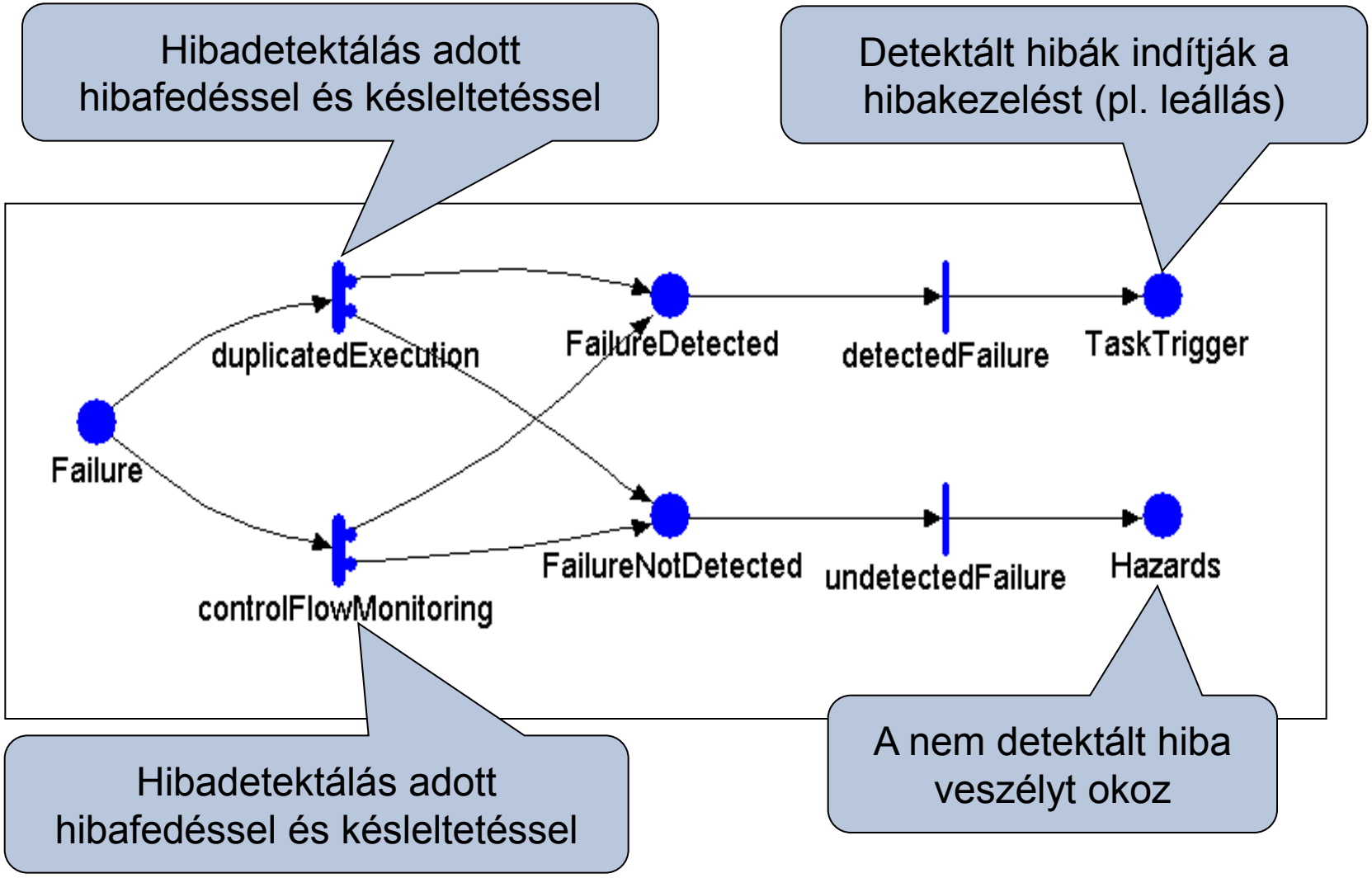
Hiba olyan erőforrásban, amit a taszk használ




Taszk aktiválási gyakoriság, hiba aktiválási lehetőségek:

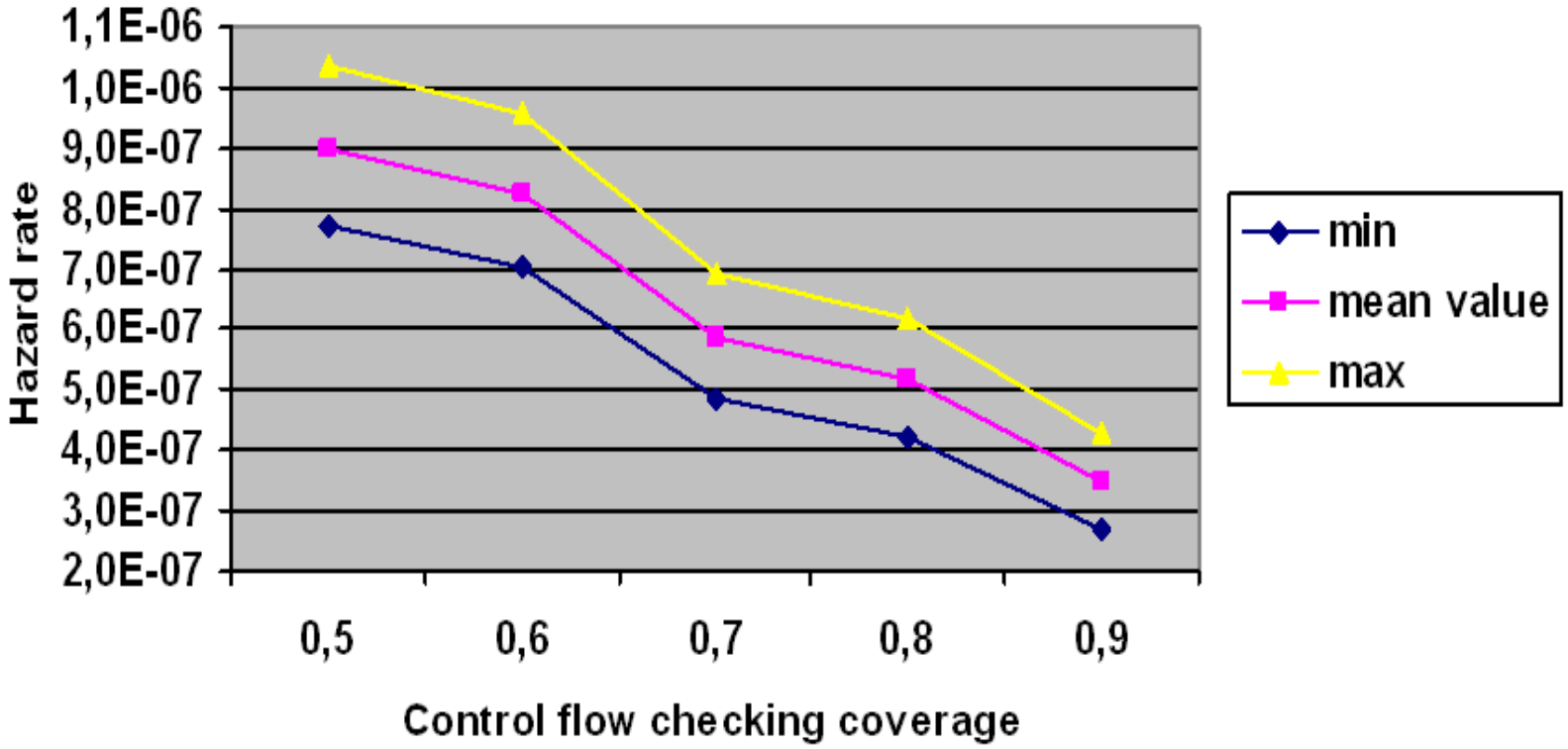
- aktivált hiba, ami a rendszerben marad
- aktivált hiba, felülíródik, de hatása van
- hatás nélkül felülírt hiba

# Egy szoftver taszk analízis modellje

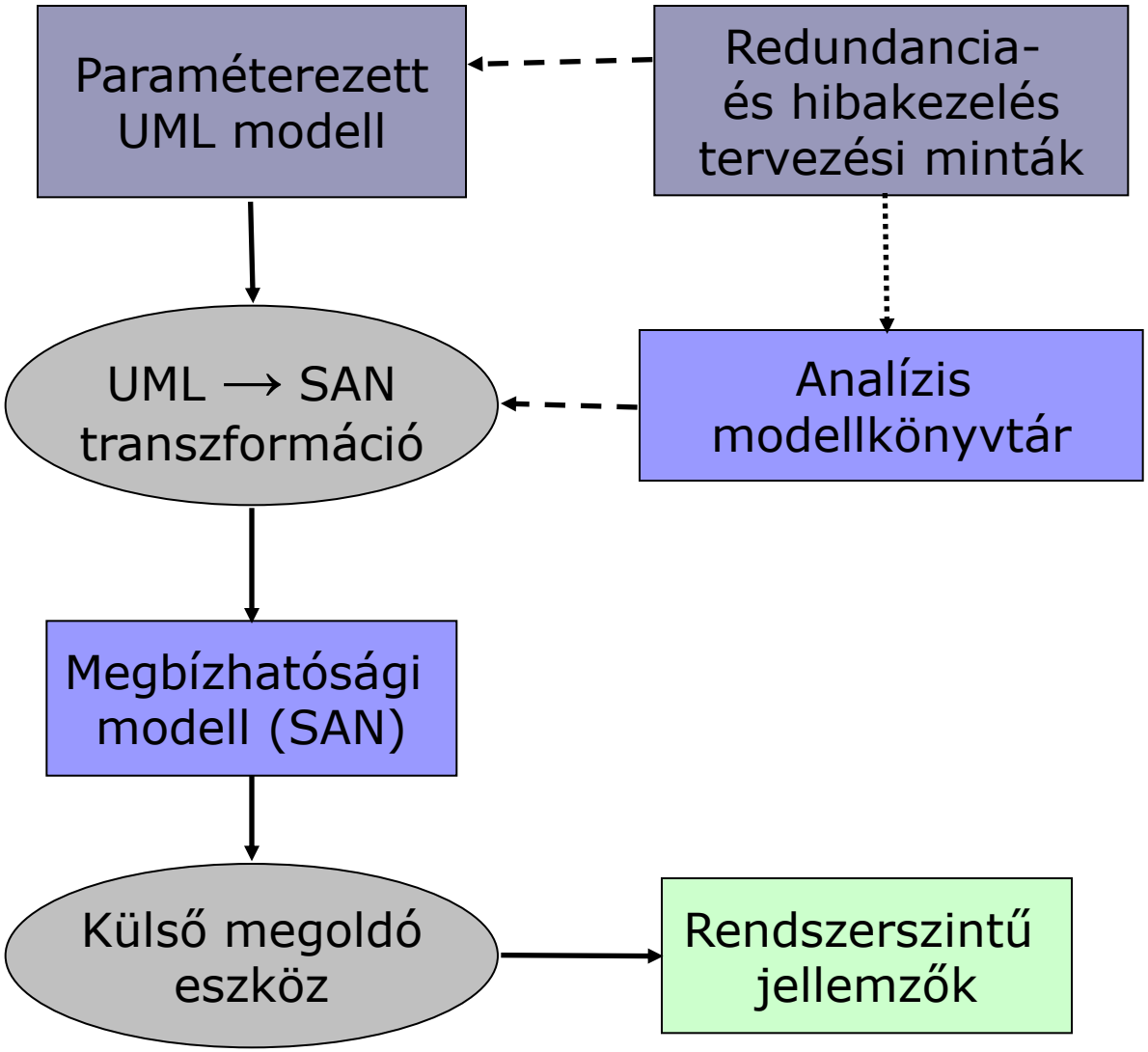


# Analízis eredmény (példa)

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



# Automatikus analízis eszköz



# 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