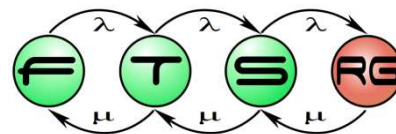


# Rendszermodellezés

## Hibamodellezés

(dr. Majzik István és Micskei Zoltán fóliái alapján)



# Tartalomjegyzék

- A szolgáltatásbiztonság fogalma
- A szolgáltatásbiztonságot befolyásoló tényezők
- A szolgáltatásbiztonság eszközei
- Szolgáltatásbiztonság analízise

# Motiváció: Hibamentes működés

- Szolgáltatási szint szerződések (SLA):
  - Ügyfél által elvárt jellemzők
  - Telekom szolgáltatások rendszerei („carrier grade”):  
„Öt kilences”: 99,999% (5 perc/év kiesés)
- Biztonságkritikus rendszerek:
  - Szabvány előírások a hibák gyakoriságára
  - Biztonságintegritási szintek (Safety Integrity Level)

SIL	Biztonságkritikus funkció hibája / óra
1	$10^{-6} \leq \text{THR} < 10^{-5}$
2	$10^{-7} \leq \text{THR} < 10^{-6}$
3	$10^{-8} \leq \text{THR} < 10^{-7}$
4	$10^{-9} \leq \text{THR} < 10^{-8}$

# Motiváció: Hibamentes működés

- Szolgáltatási szint szerződések (SLA):
  - Ügyfél által elvárt jellemzők
  - Telekom szolgáltatások rendszerei („carrier grade”):  
„Öt kilences”: 99,999% (5 perc/év kiesés)
- Biztonságkritikus rendszerek:
  - Szabvány előírások a hibák gyakoriságára
  - Biztonságintegritási szintek (Safety Integrity Level)

	Operáció hibája / óra	
Ha 15 év az élettartam, akkor ez alatt kb. 750 berendezésből 1-ben lesz hiba	$10^{-6} \leq \text{THR} < 10^{-5}$	Hiba nélküli működés ~ 11.000 év??
	$10^{-7} \leq \text{THR} < 10^{-6}$	
	$10^{-8} \leq \text{THR} < 10^{-7}$	
	$10^{-9} \leq \text{THR} < 10^{-8}$	

# Elkerülhetetlen: Hibahatások

**Fejlesztési folyamat**



**Működő termék**



- Tervezési hibák
- Implementációs hibák



- Hardver hibák
- Konfigurációs hibák
- Kezelői hibák

# Elkerülhetetlen: Hibahatások

## Fejlesztési folyamat



## Működő termék

- Tervezési hibák
- Implementációs hibák

- Hardver hibák
- Konfigurációs hibák
- Kezelői hibák

Fejlesztési folyamat jellemzői:

- Jobb minőségbiztosítás, jobb módszertanok
- De növekvő bonyolultság, nehezebb ellenőrzés

Szokásos becsült értékek 1000 kódsorra:

- Jó kézi fejlesztés és tesztelés: <10 hiba marad
- Automatizált fejlesztés: ~1-2 hiba marad
- Formális módszerek használata: <1 hiba marad

# Elkerülhetetlen: Hibahatások

**Fejlesztési folyamat**



**Működő termék**



- Tervezési hibák
- Implementációs hibák



- Hardver hibák
- Konfigurációs hibák
- Kezelői hibák

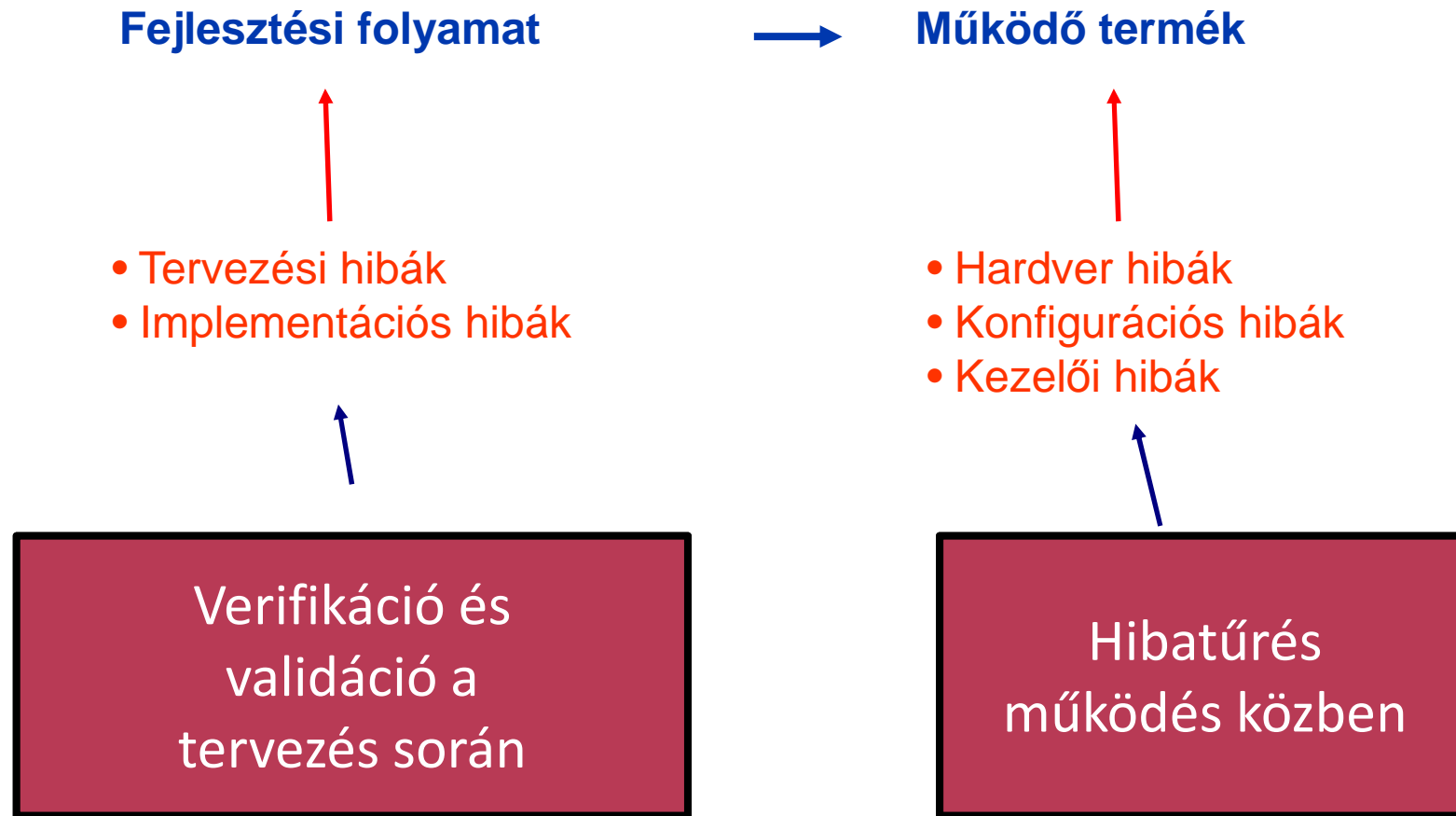
Technológia korlátai:

- Jobb paraméterek, jobb anyagok
- De növekvő bonyolultság (érzékenység)

Szokásos becsült értékek:

- CPU:  $10^{-5}$ ... $10^{-6}$  hiba/óra
- RAM:  $10^{-4}$ ... $10^{-5}$  hiba/óra
- LCD: ~ 2...3 év élettartam

# Elkerülhetetlen: Hibahatások





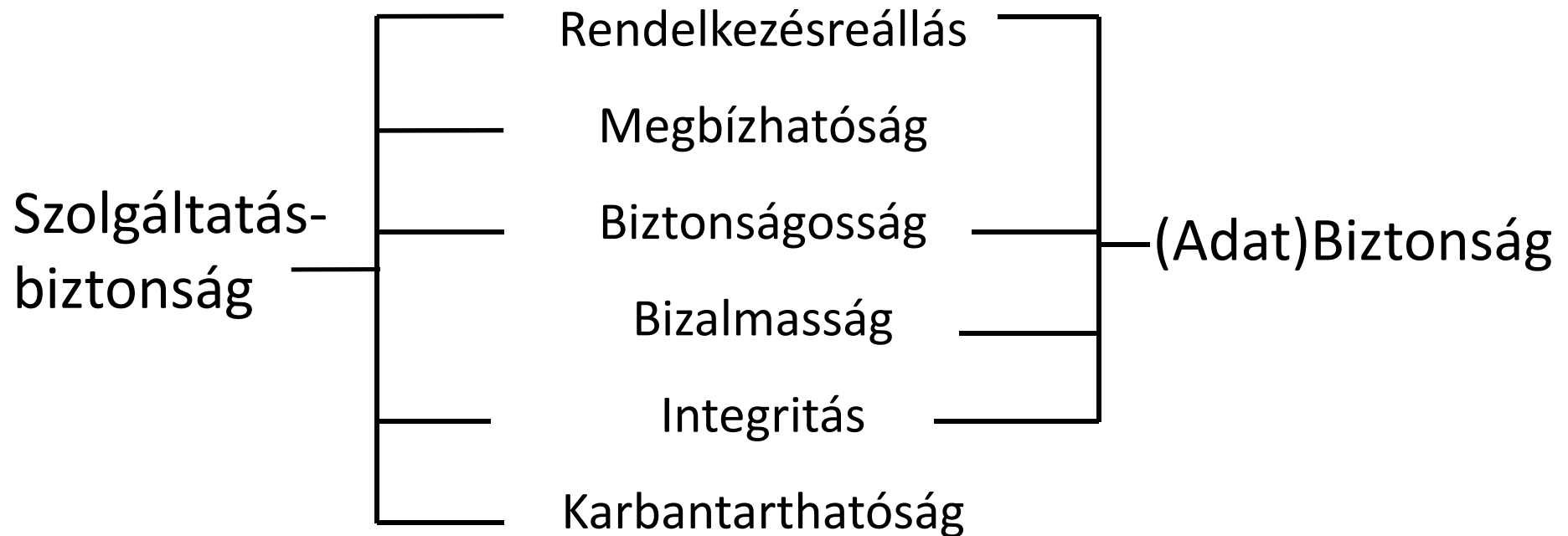
# Szolgáltatásbiztonság

**Szolgáltatásbiztonság** (dependability): a képesség, hogy igazoltan bízni lehet a szolgáltatásban

- *igazoltan*: elemzésen, méréseken alapul
- *bizalom*: szolgáltatás az igényeket kielégíti

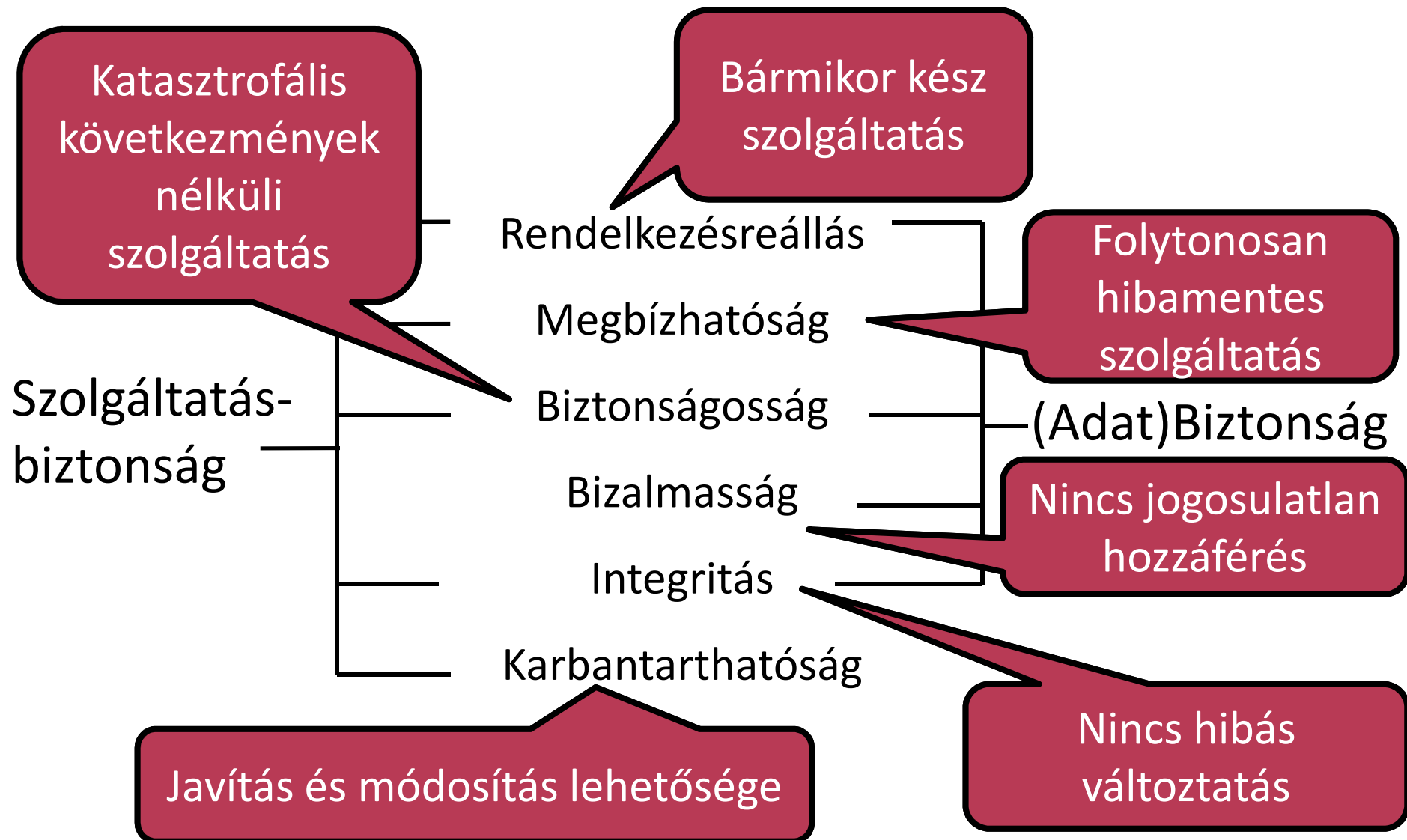
(nemfunkcionális követelmény)

# Szolgáltatásbiztonság jellemzői



Laprie et. al.: Basic Concepts and Taxonomy of Dependable and Secure Computing

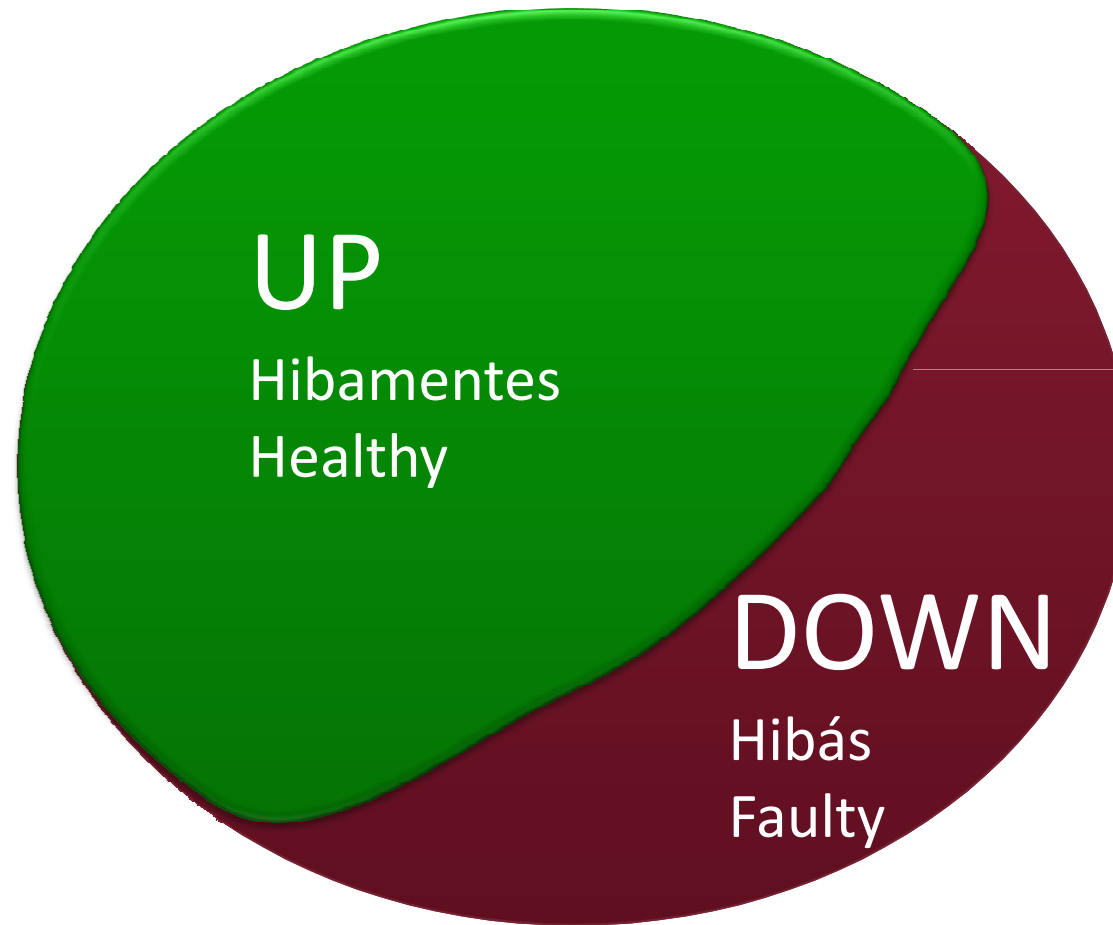
# Szolgáltatásbiztonság jellemzői



Laprie et. al.: Basic Concepts and Taxonomy of Dependable and Secure Computing

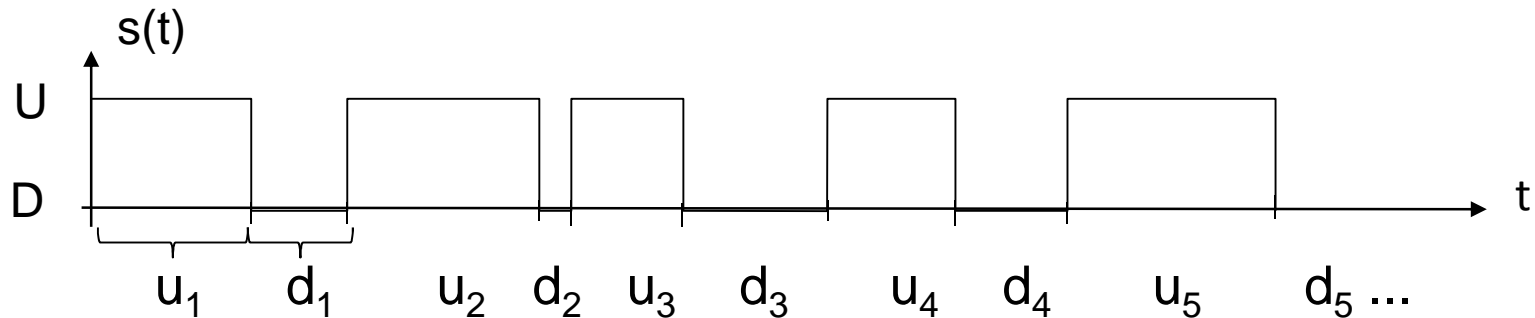
# Állapotpartícionálás

- S: a rendszer teljes állapottere



# Megbízhatósági mértékek

- Állapotparticionálás  $\rightarrow$   $s(t)$  rendszerállapot
  - Hibás (D) - Hibamentes (U) állapotpartíció



- Várható értékek:

- Első hiba bekövetkezése:  $MTFF = E\{u_1\}$   
(mean time to first failure, néha MTTF)
- Hibamentes működési idő:  $MUT = E\{u_i\}$
- Hibás állapot ideje:  $MDT = E\{d_i\}$
- Hibák közötti idő:  $MTBF = MUT + MDT$   
(mean time between failures)

# Valószínűség időfüggvények

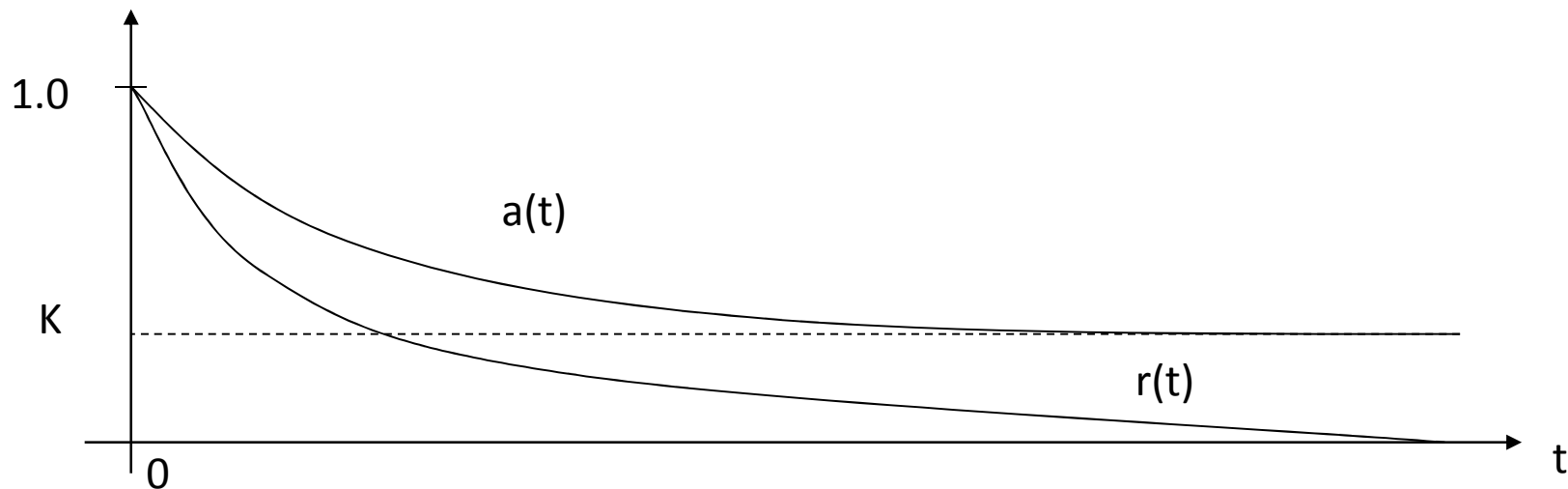
- megbízhatóság: (reliability)

$$r(t) = P\{\forall t' < t: s(t') \in U\} \quad (\text{nem hibásodhat meg})$$

- rendelkezésre állás: (availability)

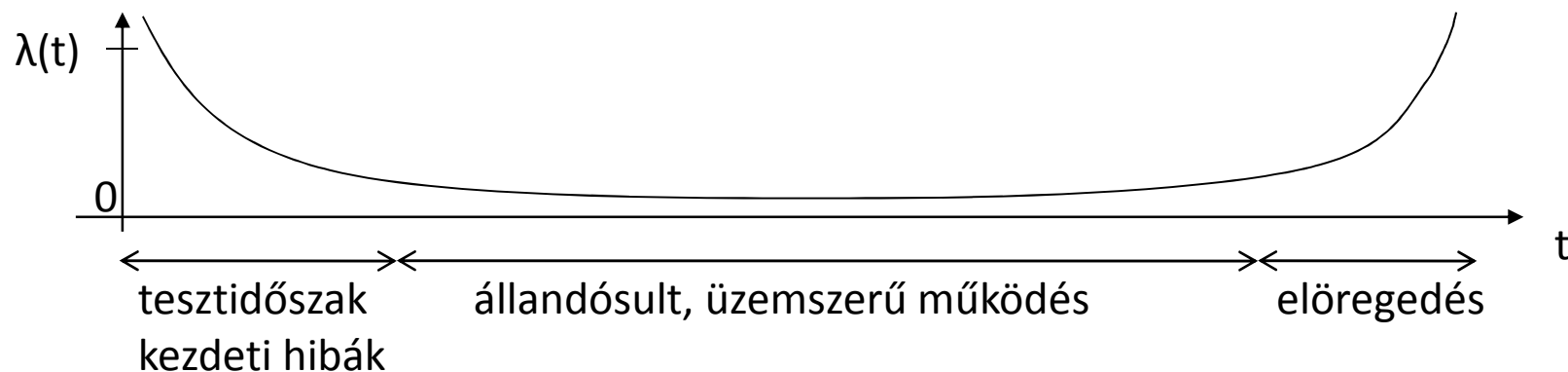
$$a(t) = P\{s(t) \in U\} \quad (\text{közben meghibásodhat})$$

- készenléti tényező:  $K = \lim_{t \rightarrow \infty} a(t) = \frac{MUT}{MUT + MDT}$



# Megbízhatóság

- **megbízhatóság**:  $r(t) = P\{s(t') \in U, \forall t' < t\}$
- (első) meghibásodások gyakorisága:  $-r'(t)$ 
  - Ez az „első meghibásodás ideje” valószínűségi változó eloszlásának sűrűségfüggvénye!
  - Várható érték: MTTF
- **meghibásodási ráta**:  $\lambda(t) = -r'(t) / r(t)$  (meghibásodás valószínűsége időegységenként és darabonként)
  - „kádgörbe”:



# Megbízhatóság

- **Közelítés:** állandósult állapot,  $\lambda(t) = \lambda$  (konst.)
  - „Örökifjú” / „memoryless” tulajdonság
  - Jól tesztelt informatikai rendszerre igaz lehet: előbb avul el, minthogy elöregedne
- Következmény:  $r(t) = e^{-\lambda t}$
- $-r'(t)$ : meghibásodás ideje exponenciális eloszlás lesz
  - $\lambda$  paraméterrel
  - $1/\lambda$  várható értékkel
  - Vagyis: **MTTF =  $1/\lambda$**  !



# Rendelkezésre állás követelményei

A rendelkezésre állási tényező	Maximális kiesés idő 1 év alatt
2 db 9-es (99%)	3,5 nap
3 db 9-es (99,9%)	9 óra
4 db 9-es (99,99%)	1 óra
5 db 9-es (99,999%)	5 perc
6 db 9-es (99,9999%)	32 másodperc
7 db 9-es (99,99999%)	3 másodperc

## Elosztott rendszerek (hibatűrés nélkül, irányadó számok):

- 1 szgép: 95%
- 2 szgép: 90%
- 5 szgép: 77%
- 10 szgép: 60%

# Tartalomjegyzék

- A szolgáltatásbiztonság fogalma
- A szolgáltatásbiztonságot befolyásoló tényezők
- A szolgáltatásbiztonság eszközei
- Szolgáltatásbiztonság analízise

# Befolyásoló tényezők

- **Hibajelenség** (failure):  
A specifikációnak nem megfelelő szolgáltatás
  - értékbeli / időzítésbeli, katasztrofális / „jóindulatú”
- **Hiba** (error):  
Hibajelenséghez vezető rendszerállapot
  - lappangó → detektált
- **Meghibásodás** (fault):  
A hiba feltételezett oka
  - hatás: alvó → aktív
  - fajta: véletlen vagy szándékos, időleges vagy állandósult
  - eredet: fizikai/emberi, belső/külső, tervezési/működési

# Szoftver hibák

- Szoftver hiba: **Állandósult, tervezési** hiba
- **Aktiválás** a működési profil függvénye
  - Bemeneti tartományok, trajektória
- **Megbízhatóság arányos:**  
Tesztelés után **benntmaradó hibák száma**
- **Benntmaradó hibák száma arányos:**  
**Időegység alatt detektált hibák száma** a tesztelés végén
  - Statisztikai tesztelés: Megbízhatóság mérése
- Statisztikai módszerekkel becsülhető, meddig kell a tesztelést folytatni adott megbízhatóság eléréséhez

# Hatáslánc

- **Meghibásodás → Hiba → Hibajelenség**
  - pl. szoftver:
    - meghibásodás: progr. hiba: csökkentés helyett növel
    - hiba: vezérlés ráfut, változó értéke hibás lesz
    - hibajelenség: számítás végeredménye rossz
  - pl. hardver:
    - meghibásodás: kozmikus sugárzás egy bitet átbillent
    - hiba: hibás pozícióérték a memóriában
    - hibajelenség: robotkar a falnak ütközik
- **Rendszer hierarchiaszint függvénye**
  - alsó szintű hibajelenség felsőbb szinten meghibásodás
    - kimenet beragadás egy chip szintjén hibajelenség
    - rendszer szintjén meghibásodás (chip a cserélhető komponens)

# Hatáslánc

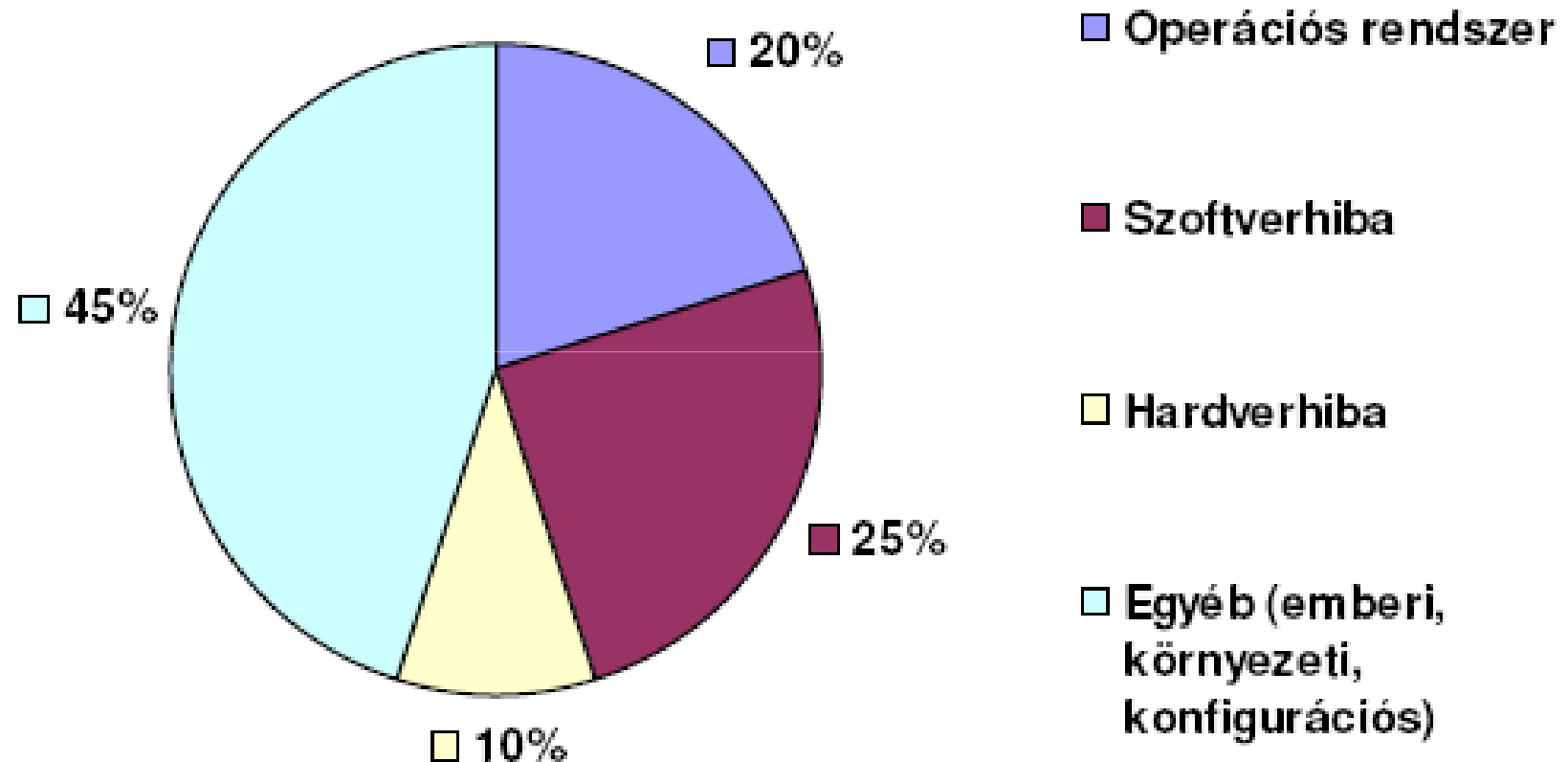
- A hatáslánc befolyásolása
  - meghibásodási tényező csökkentése
    - jobb minőségű komponensek
    - szigorúbb fejlesztési folyamat (ellenőrzés, tesztelés)

A meghibásodás-mentesség nem garantálható  
(csökkenő chipméretek, bonyolultabb programok)
  - hibajelenség kialakulásának megakadályozása
    - rendszerstruktúra kialakítása: redundancia
- Hibatípusok:
  - előre figyelembe vehető hibák:  
optimális **kezelés a tervezési folyamat során**
  - előre figyelembe nem vehető hibák:  
megfelelő **rendszerstruktúra** kialakítása szükséges

# Meghibásodások kategorizálása

- Hardverhibák
  - alaprendszer (alaplap, processzor, memória)
  - tápellátás (tápegység, szünetmentes táp)
  - adattároló alrendszer
  - hálózat
- Szoftverhibák
  - az operációs rendszer hibái
  - alkalmazáshibák
  - illesztőprogram-hibák
- ...
- Emberi hibák
  - rendszergazdai hibák
  - illetékes felhasználók nem rosszindulatú hibái
  - illetékes felhasználók rosszindulatú hibái
  - illetéktelen felhasználók támadásai
- Környezeti hatások
  - üzemeltetési környezet rendellenességei, például a légkondicionálás leállása, bombariadó, csőtörés
  - természeti katasztrófák

# A hibajelenségek okai IT rendszerek esetén





# Tartalomjegyzék

- A szolgáltatásbiztonság fogalma
- A szolgáltatásbiztonságot befolyásoló tényezők
- A szolgáltatásbiztonság eszközei
- Szolgáltatásbiztonság analízise

# A szolgáltatásbiztonság eszközei

- **Hiba megelőzés:** Meghibásodás megakadályozása
  - fizikai hibák: jó minőségű alkatrészek, árnyékolás,...
  - tervezési hibák: **verifikáció**
- **Hiba megszüntetés:** Hibaállapotból kimosztítás
  - prototípus fázis: **tesztelés**, diagnosztika, javítás
  - működés közben: **monitorozás**, javítás
- **Hibatűrés:** Szolgáltatást nyújtani hiba esetén is
  - működés közben: **hibakezelés**, **redundancia**
- **Hiba előrejelzés:** Hibák és hatásuk becslése
  - mérés és „jóslás”, megelőző karbantartás

# Hibatűrő rendszerek

- Akármilyen jó is az ellenőrzés a tervezés során, a szolgáltatásbiztonság nem garantálható:
  - időleges hardver hibák (ld. zavarérzékenység)
  - teszteletlen szoftver hibák
  - figyelembe nem vett komplex interakciók
- Fel kell készülni a működés közbeni hibákra!
- **Hibatűrés**: Szolgáltatást nyújtani hiba esetén is
  - működés közbeni autonóm hibakezelés
  - beavatkozás a meghibásodás → hibajelenség láncba
  - rendszertechnikai megoldások (+ megbízható alkatrész)
- Alapfeltétel: Redundancia (tartalékolás)
  - többlet erőforrások a hibás komponensek kiváltására

# Redundancia megjelenése

## 1. Hardver redundancia

- többlet hardver erőforrások
  - eleve a rendszerben lévők (elosztott rendszer)
  - hibatűréshez betervezett (tartalék)

## 2. Szoftver redundancia

- többlet szoftver modulok

## 3. Információ redundancia

- többlet információ a hibajavítás érdekében
  - hibajavító kódolás (ECC)

## 4. Idő redundancia

- ismételt végrehajtás, hibakezelés többlet ideje

Együttes megjelenés!

# Redundancia típusai

- **Hidegtartalék** (passzív redundancia):
  - normál üzemmódban **passzív**, hiba esetén aktiválva
  - lassú átkapcsolás (elindítás, állapot frissítés,...)
  - pl. tartalék számítógép
- **Langyos** tartalék:
  - normál üzemmódban **másodlagos funkciók**
  - gyorsabb átkapcsolás (indítást nem kell várni)
  - pl. naplózó gép átveszi a kritikus funkciókat
- **Meleg** tartalék (aktív redundancia):
  - normál üzemmódban **aktív**, ugyanazt a feladatot végzi
  - azonnal átkapcsolható
  - pl. kettőzés, többszörözés

# 1. Hardver redundancia

- **Többszörözés**
  - Kettőzés
    - hibadetektálás: pl. master-checker kialakítás
    - hibatűrés csak diagnosztikai támogatással és átkapcsolással
  - TMR: Triple-modular redundancy
    - hiba maszkolás szavazással
    - szavazó kritikus elem (de egyszerű)
  - NMR: N-modular redundancy
    - hibamaszkolás többségi szavazással
    - MTFF kisebb, de missziós idő túlélése nagyobb esélyű
    - repülőgép fedélzeti eszközök: 4MR, 5MR
- **Tipikus: nagy rendelkezésre állású fűrtök**

## 2. Szoftver redundancia

Használat:

### 1. Szoftver tervezési hibák esetén:

- ismételt végrehajtás nem segít...
- redundáns modulok: **eltérő tervezés** szükséges  
**variánsok**: azonos specifikáció, de
  - eltérő algoritmus, adatstruktúrák
  - más fejlesztési környezet, programnyelv
  - elszigetelt fejlesztés

### 2. Időleges (hardver) hibák esetén:

- ismételt végrehajtás esetén a hiba nem jelentkezik
- hibahatások kiküszöbölése a fontos

# 3. Információ redundancia

- Hibajavító kódolás
  - memóriák. háttértárak, adatátvitel
  - pl. Hamming-kód, Reed-Solomon kódok
- Korlátozott hibajavító képesség
  - hosszú idejű adatstabilitás rossz lehet (“felgyűlnek” a hibák)
  - háttértárak: “memory scrubbing”  
folyamatos olvasás és javítva visszaírás
- Redundáns (többpéldányos) adatbázisok:
  - hozzáférések konzisztenciájának biztosítása
  - egypéldányos sorosíthatóság



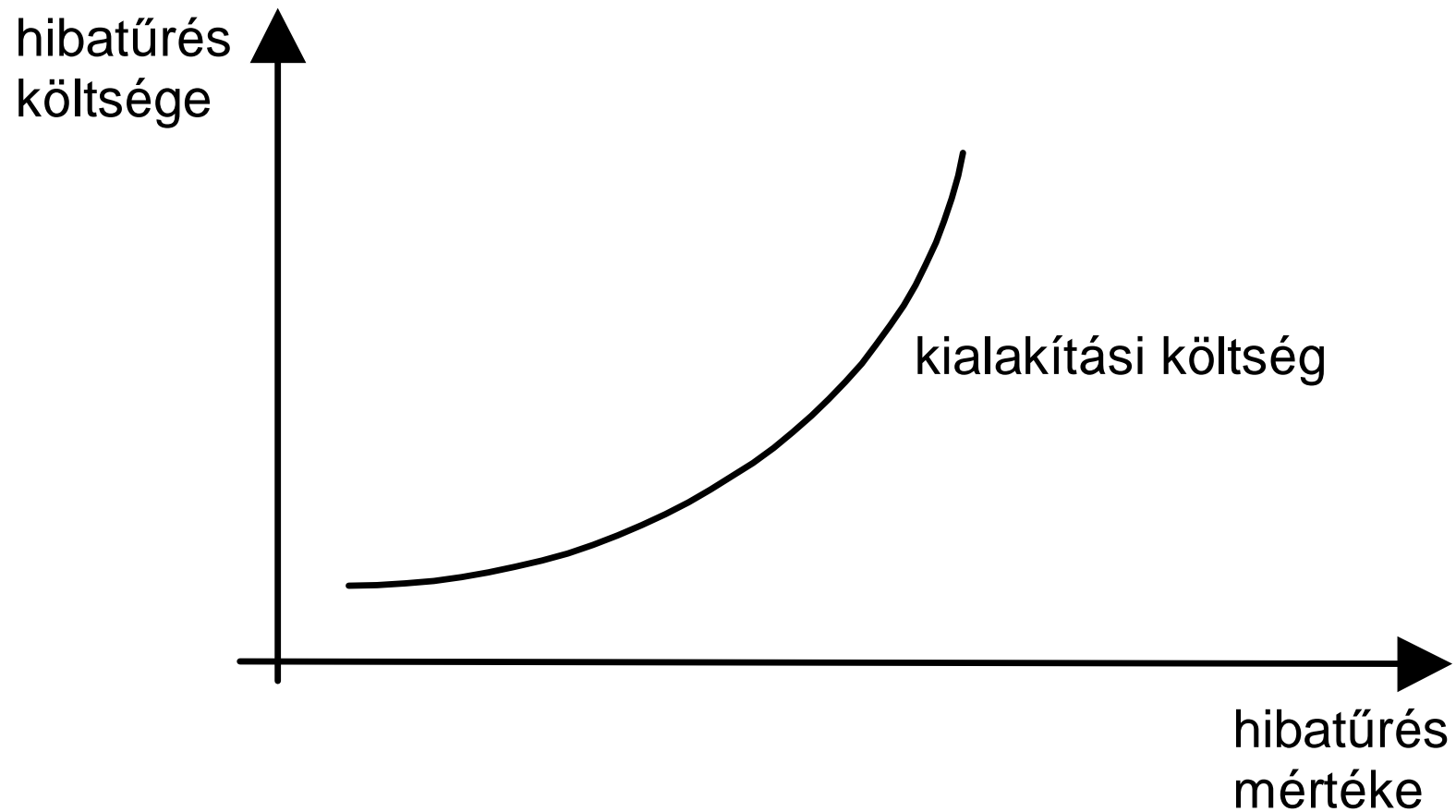
## 4. Idő redundancia

- Tiszta eset: utasítás újrapróbálás (retry)
  - alacsony hardver szinten: processzor utasítás
  - időleges hibák esetén hatásos
- Idő redundancia “velejárója” a többi típusnak  
Valós idejű rendszerek: tervezési szempont mennyire garantálható a hibakezelés ideje
  - állandósult hardver hibák: maszkolás, meleg tartalék
  - időleges hardver hibák: előrelépő helyreállítás
  - szoftver tervezési hibák: N-verziós programozás
- $a(t)$ , válaszidő SLA fontos tényezője

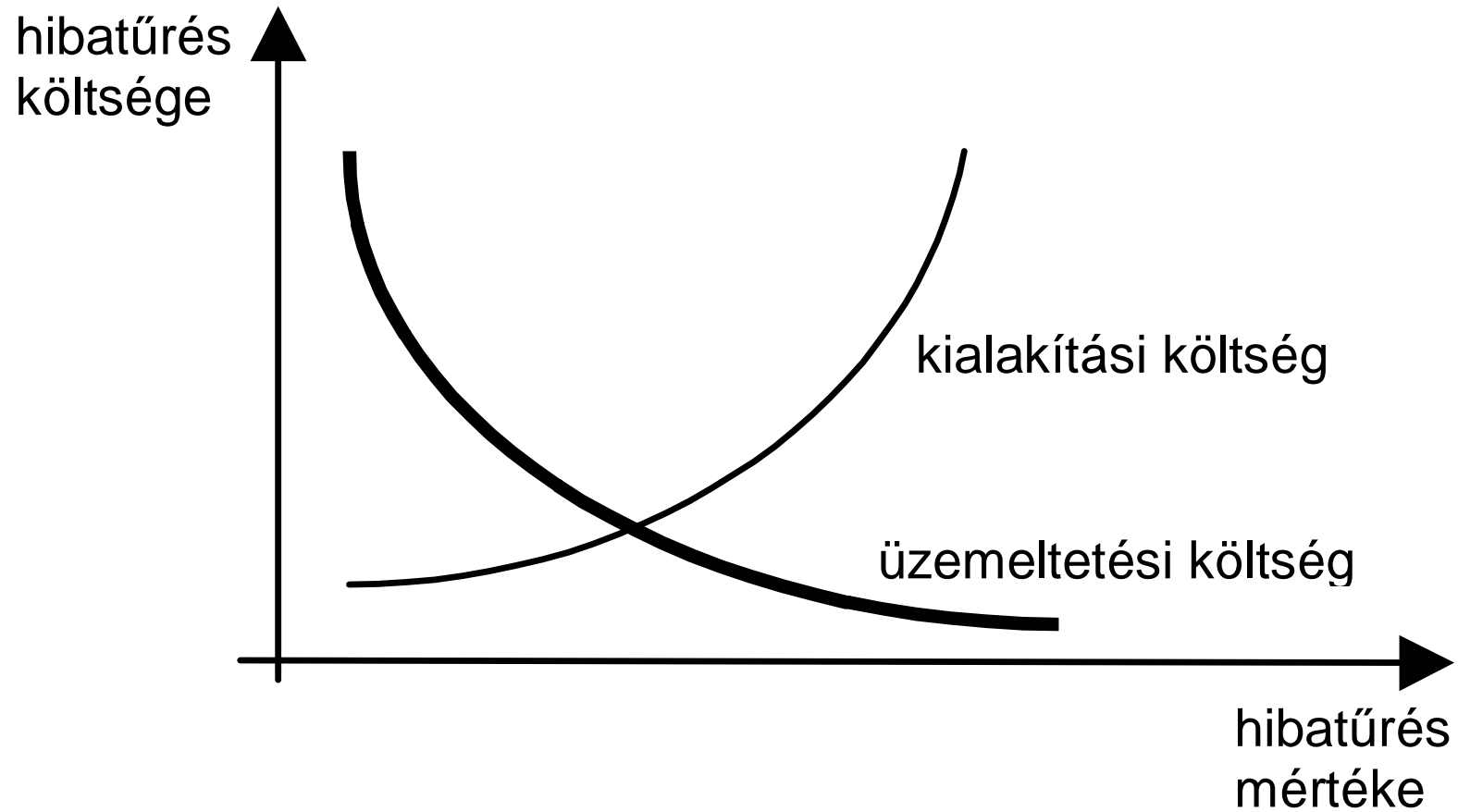
# Hibák kezelése

- **Hardver tervezési hibák** (< 1%):
  - nincs figyelembe véve (ld. jól tesztelt komponensek)
- **Hardver állandósult működési hibák** (10%):
  - **hardver redundancia** (pl. tartalék processzor)
- **Hardver időleges működési hibák** (70-80%):
  - **idő redundancia** (pl. utasítás újravégrehajtás)
  - **információ redundancia** (pl. hibajavítás)
  - **szoftver redundancia** (pl. állapotmentés és helyreállítás)
- **Szoftver tervezési hibák** (10-20%):
  - **szoftver redundancia** (pl. eltérő tervezésű modulok)

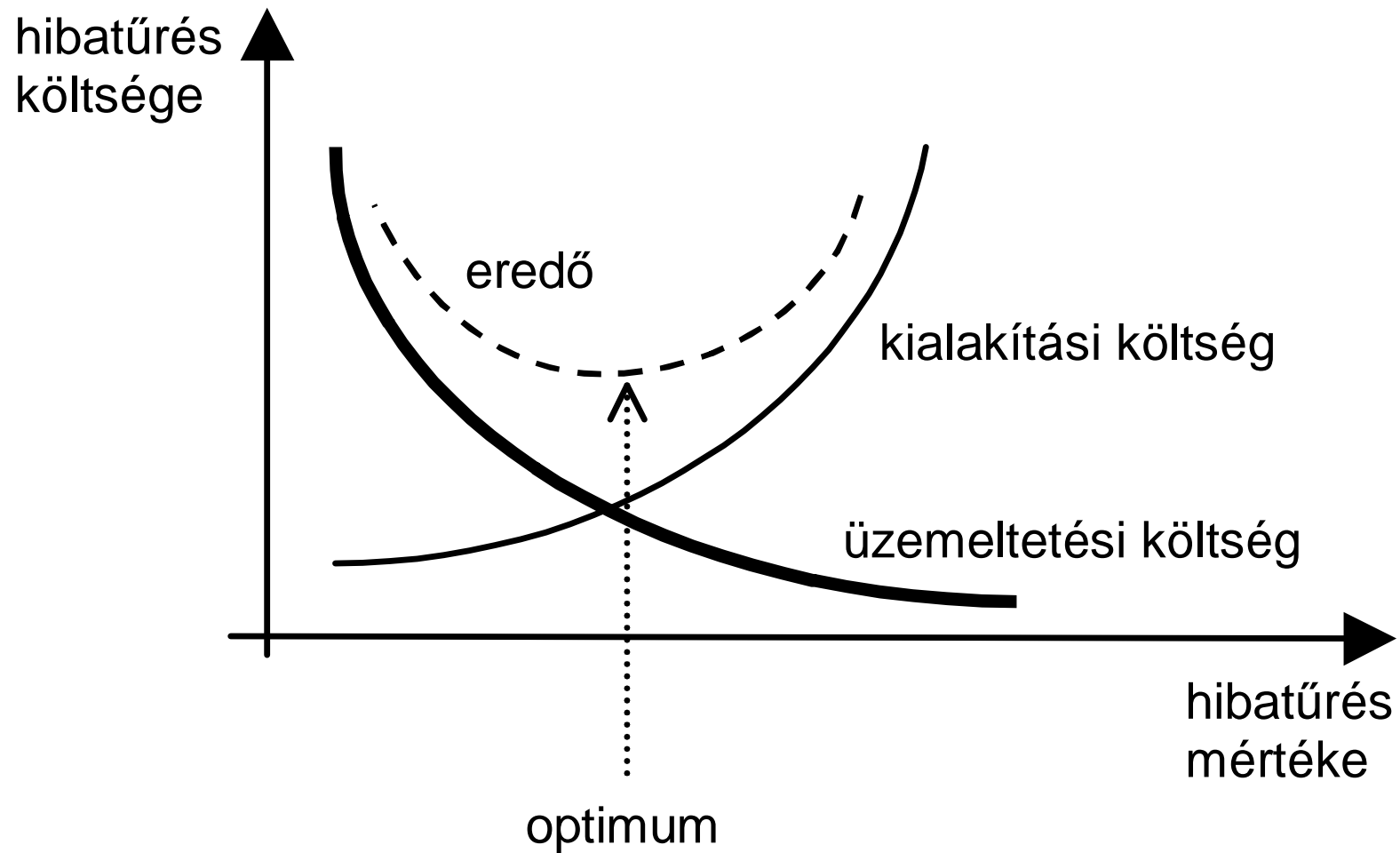
# Költségoptimalizálás



# Költségoptimalizálás



# Költségoptimalizálás



# Tartalomjegyzék

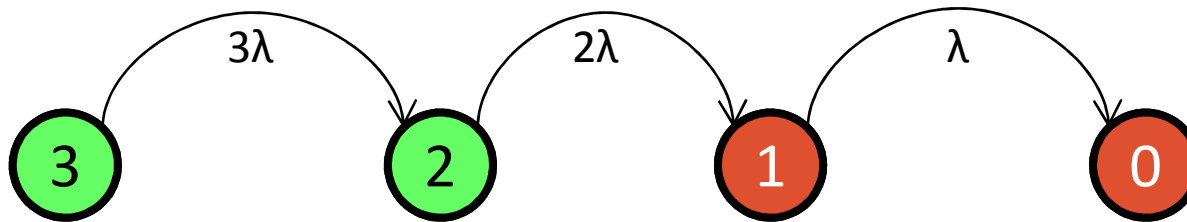
- A szolgáltatásbiztonság fogalma
- A szolgáltatásbiztonságot befolyásoló tényezők
- A szolgáltatásbiztonság eszközei
- **Szolgáltatásbiztonság analízise**

# Szolgáltatásbiztonság analízise

- Miért van szükség analízisre?
  - Ha bőséges redundanciát biztosítunk, az nem elég?
- A redundancia költséges ☹️
- Csak a jól **tervezett** redundancia éri el a célját!
  - Mennyiség
  - Hideg / meleg
  - Javítás
  - ...

# Példa

- 3 merevlemezről álló RAID-5 tömb
  - Két lemeznyi hasznos terület, plusz paritás
  - Egy lemez meghibásodását tűri
  - Első meghibásodás rátája:  $3\lambda$   
(meleg tartalék  $\rightarrow$  mindhárom lemez kieshet!)

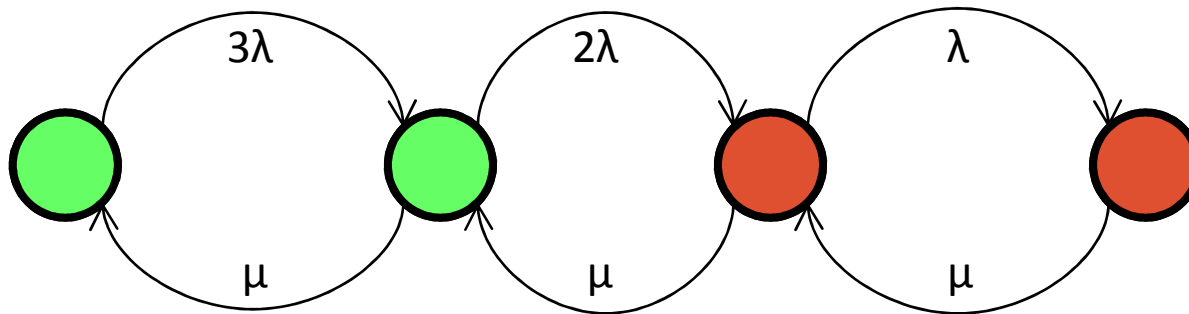


- MTTF:  $1/3\lambda + 1/2\lambda = 5/6\lambda$
- $5/6\lambda < 1/\lambda$  – egy lemez jobb!

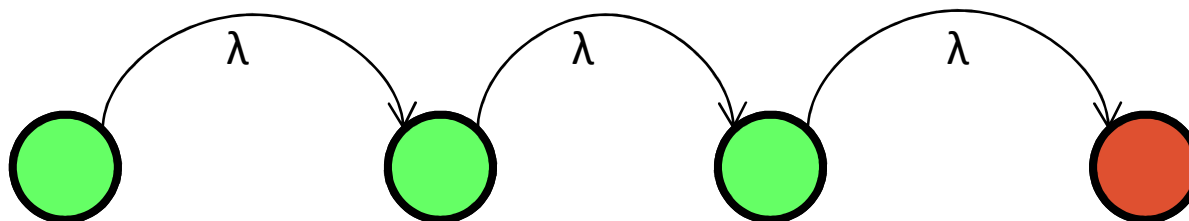


# Példa

- A redundancia rontotta a megbízhatóságot!
- Megoldás: a kiesett lemezt gyorsan cserélni kell
  - Javítási folyamat felvétele a Markov-láncba



- Másik példa: három villanykörte, hideg tartalék



# Szolgáltatásbiztonság analízise

- Feladatok:
  - Hibamódok, meghibásodások azonosítása
  - Analízis: **kvalitatív és kvantitatív**
- Módszerek
  - Ellenőrző listák
  - Táblázatok  
(pl. FMEA: Failure Mode and Effect Analysis)
  - Hibafák
  - Állapot alapú módszerek (pl. Petri hálók)
  - ...

# Ellenőrző lista

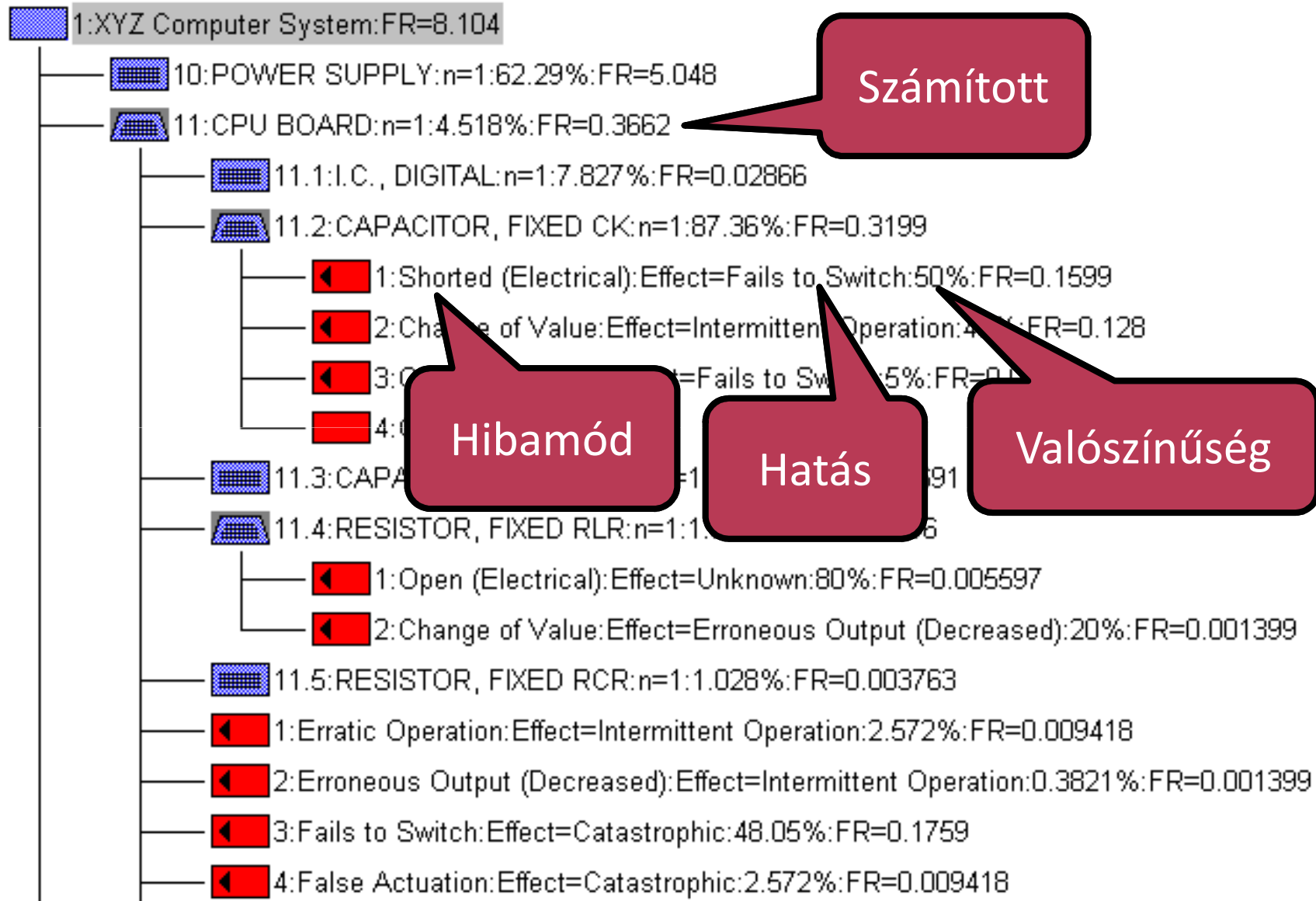
- Technika:
  - Tapasztalatok rendszerezett összegyűjtése
  - Alkalmazás mint „ ökölszabályok”
- Biztosítja:
  - Ismert hibaforrások nem maradnak ki
  - Kipróbált módszereket alkalmaz
- Hátrányok:
  - A lista nem teljes és nehezen kezelhető
  - Téves biztonságérzetet ad
  - Más környezetben az alkalmazhatóság kérdéses

# Hibamód és hatás analízis (FMEA)

- Meghibásodás és hatásaik felsorolása

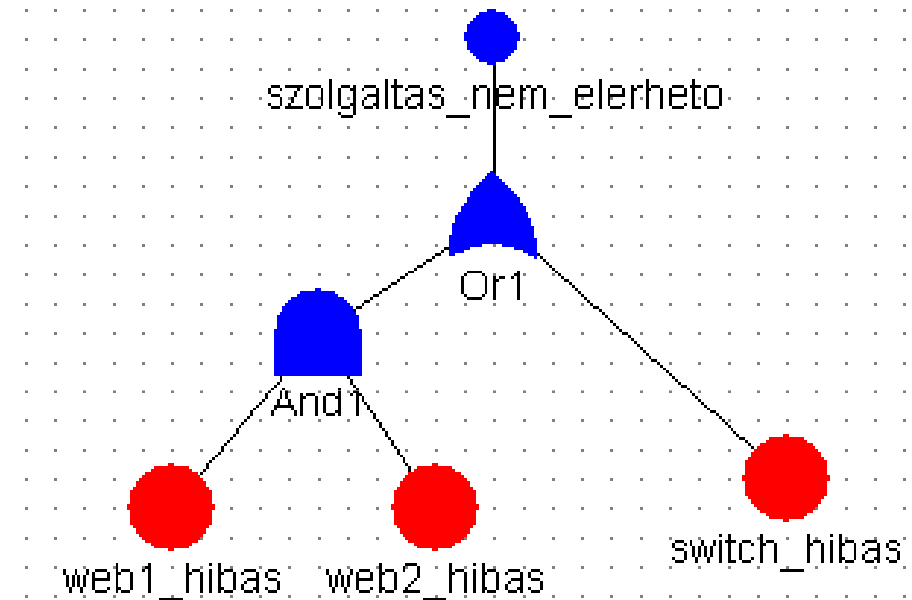
Komponens	Hibamód	Valószínűség	Hatás
Webszerver	HW hiba	10%	Szolg. kiesés, alkatrész csere
	SW frissítés	90%	Időleges kiesés
SQL szerver	Diszk megtelik	20%	Csak statikus tartalom érhető el
...			

# Példa: Vezérlő elektronika



# Hibafa (Fault tree)

- Hogyan állhat elő a gyökérben lévő hibajelenség?



- Elemek (részlet)

- AND kapu
- OR kapu
- Téglalap: alrendszer
- Kör: alapszintű meghibásodások

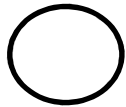
# Hibafa – analízis

- Kvalitatív:
  - egyszeres hibapont (SPOF) azonosítása
  - kritikus esemény: több úton is hibajelenséget okoz
- Kvantitatív:
  - alapszintű eseményekhez valószínűség rendelése (nehéz: honnan lesznek jó adataink?)
  - gyökérelem jellemzőjének (pl. megbízhatóság) számolása
  - Problémák: nem független események...

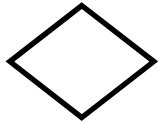
# Hibafa grafikus elemkészlet



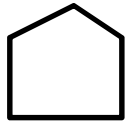
legfelső szintű vagy közbenső esemény



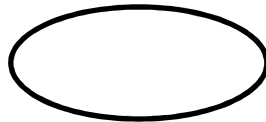
elsődleges (alapszintű) esemény



tovább nem vizsgált esemény



normál esemény (nem hiba vagy veszély)



feltétel egy összetett esemény  
bekövetkezéséhez



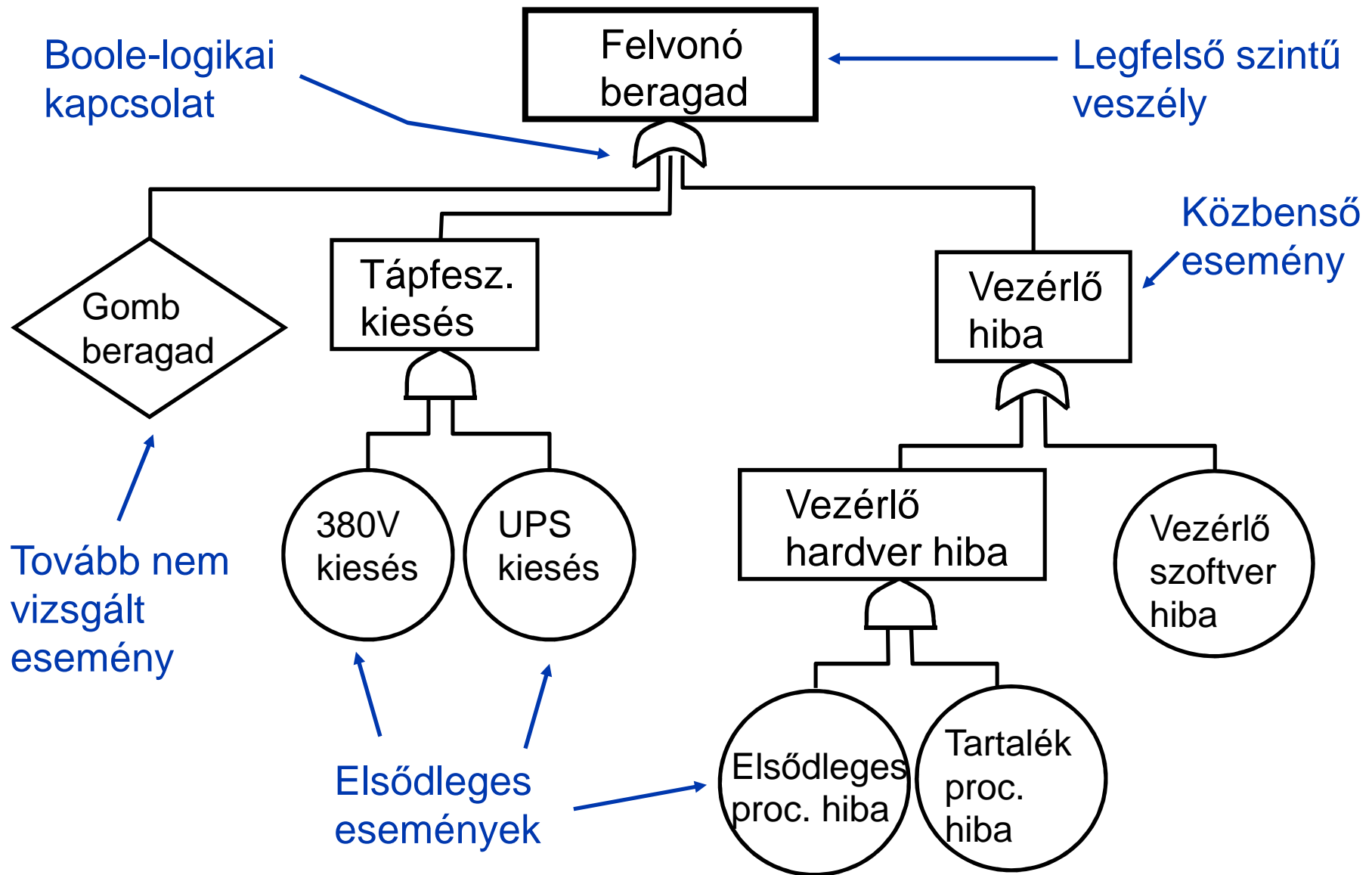
ÉS kapu



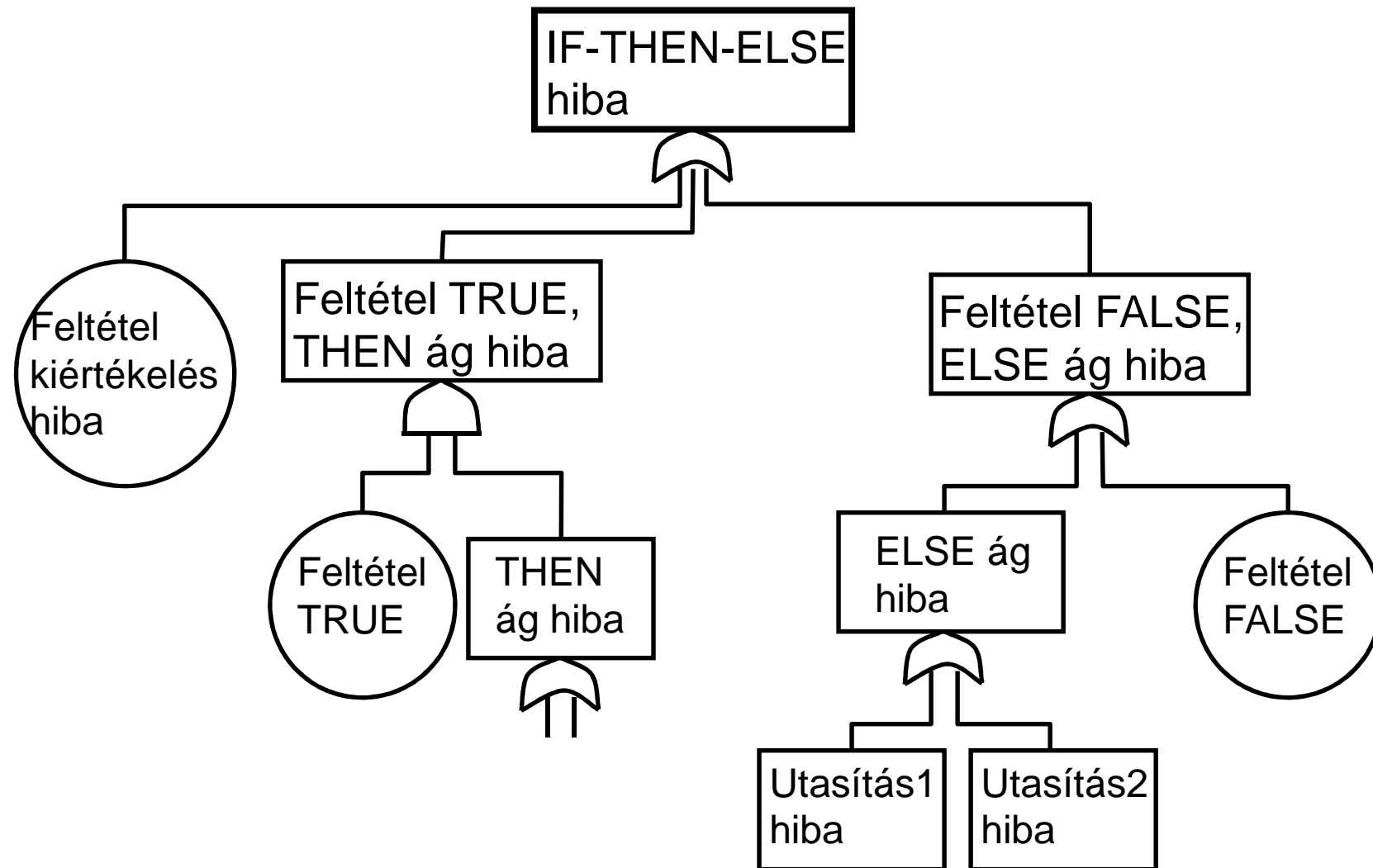
VAGY kapu



# Hibafa példa: Felvonó



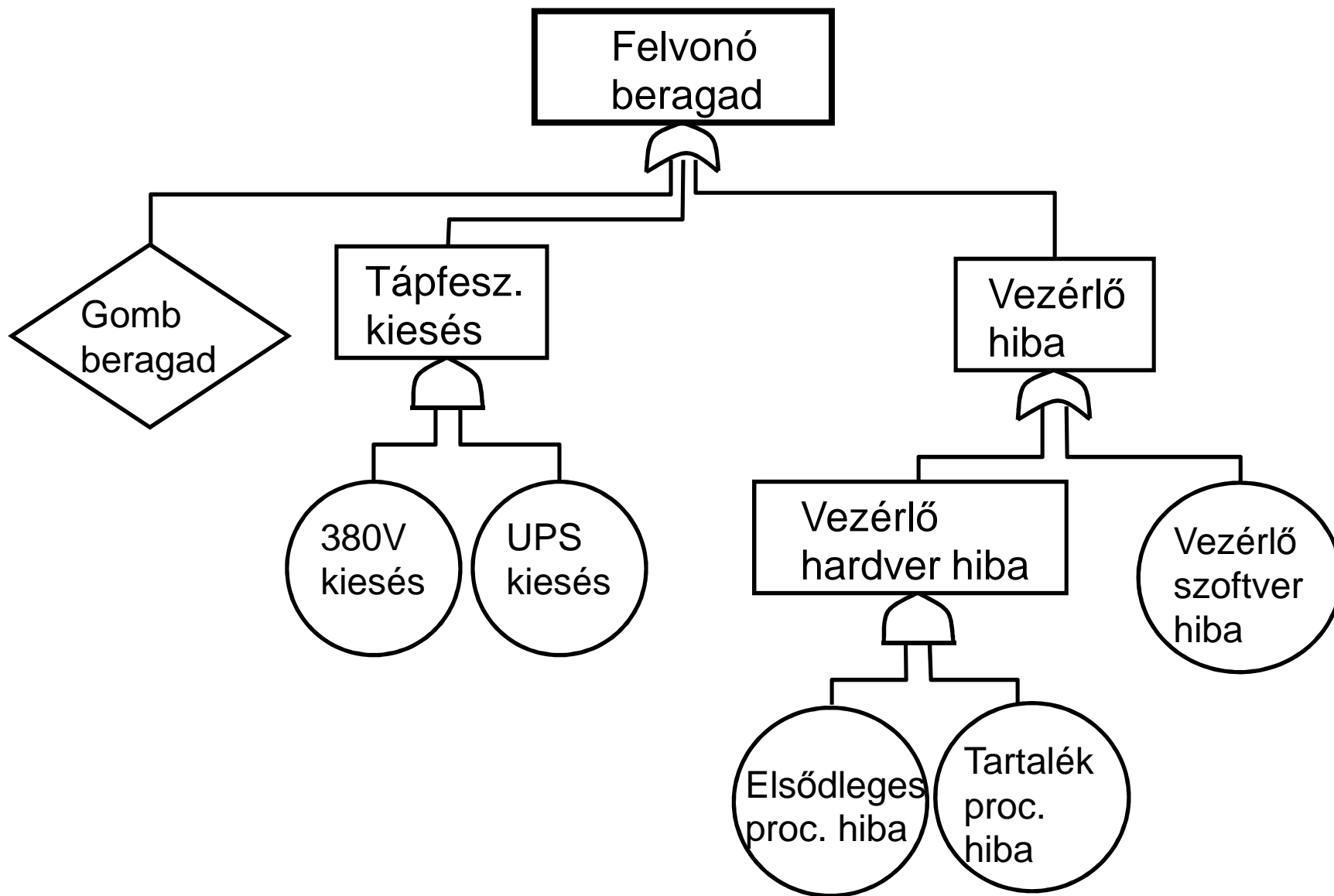
# Hibafa példa: Szoftver minta



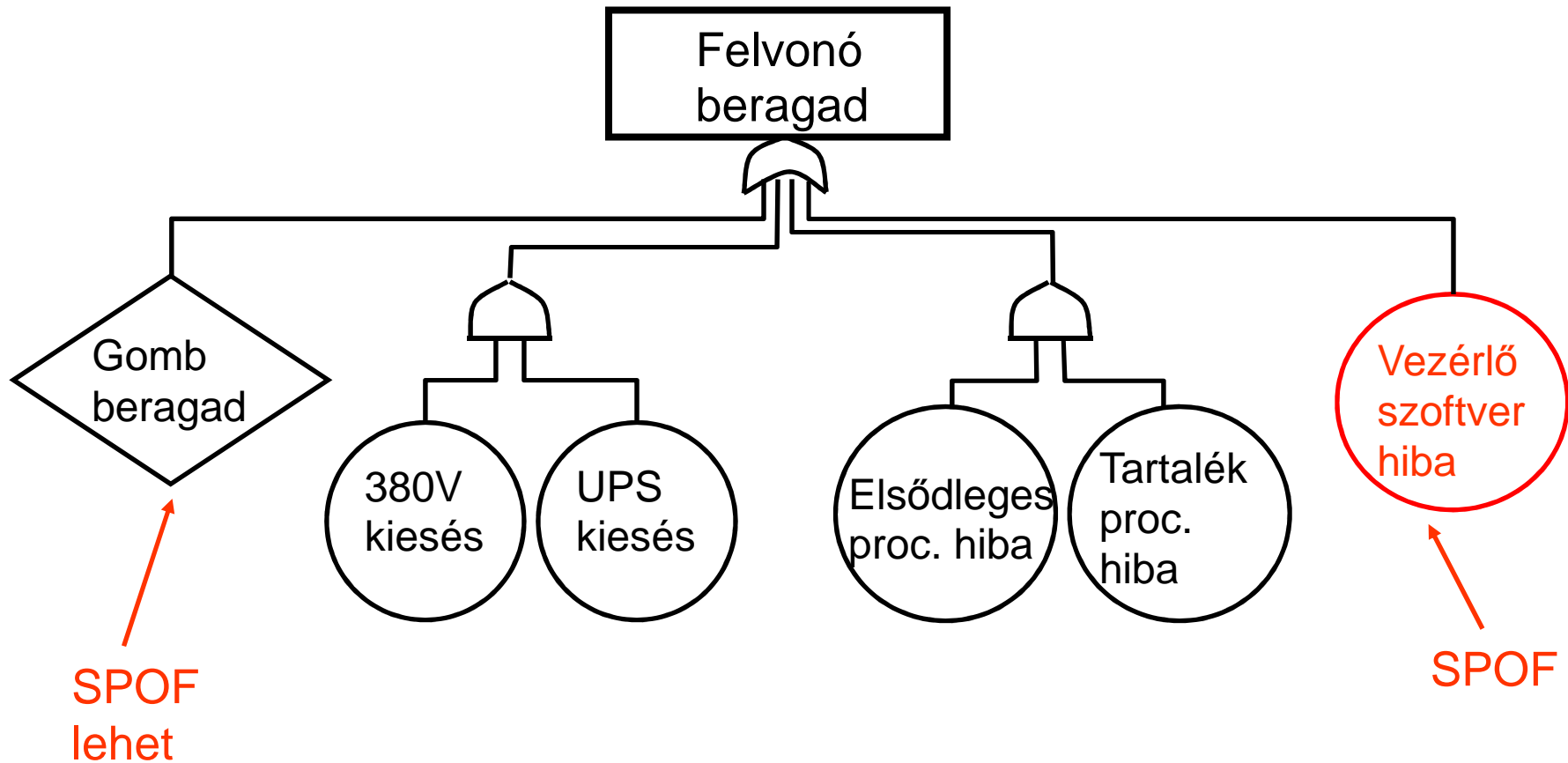
# Minőségi (kvalitatív) analízis

- Hibafa **redukció**: Közbenső események és pseudo-események feloldása  
→ diszjunktív normál forma (OR a legtetején)
- **Vágat**:  
AND kapuval összefogott elsődleges események
- **Minimális vágathalmaz**: Nem redukálható
  - Nincs olyan, aminek részhalmaza is megtalálható
- **Azonosítható**:
  - **egyszeres hibapont** (SPOF)
  - kritikus esemény (több vágatban is szerepel)

# Hibafa példa: Felvonó



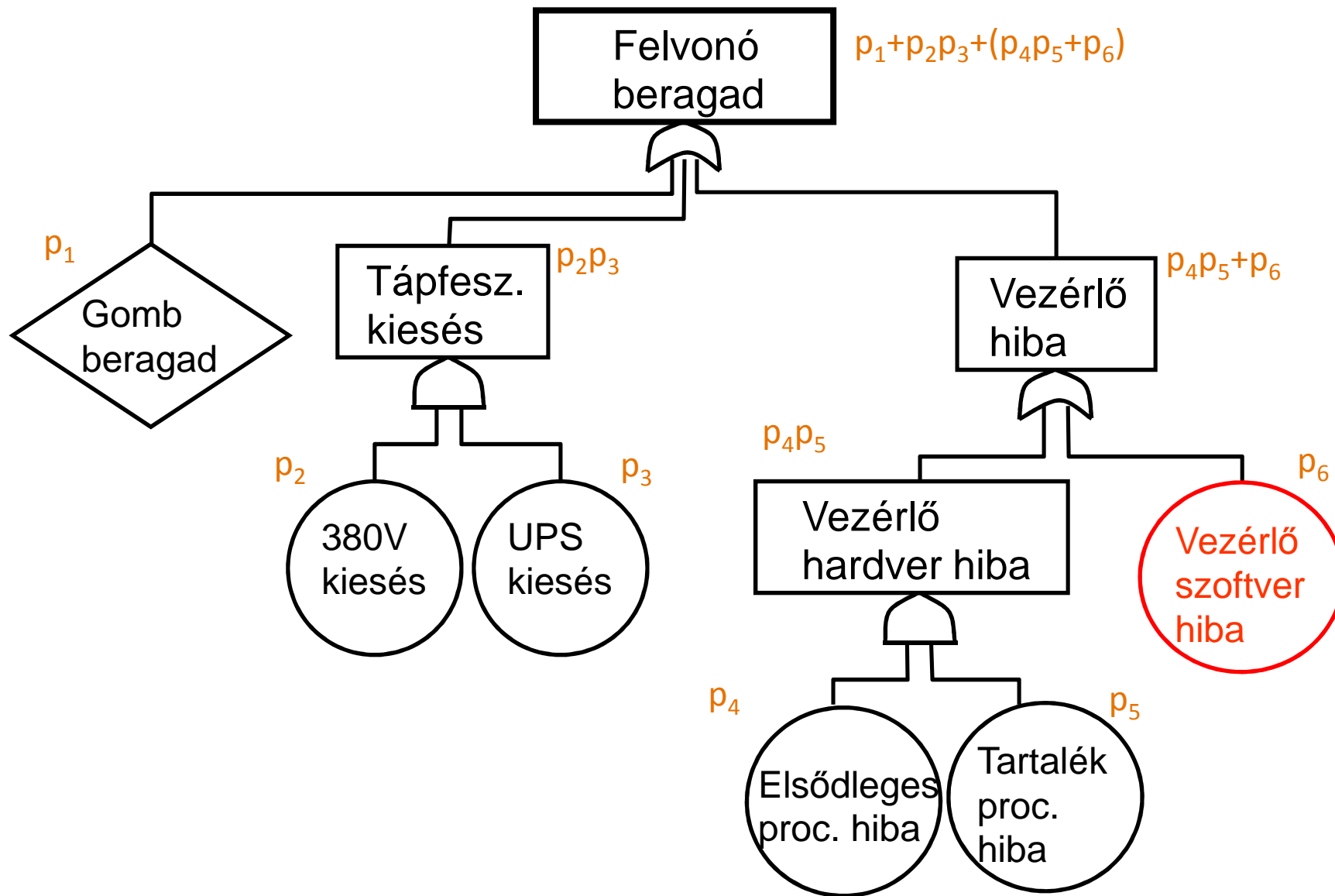
# Redukált hibafa példa: Felvonó



# Mennyiségi (kvantitatív) analízis

- Alapszintű eseményekhez rendelt **valószínűségek**
  - komponens-adat, tapasztalat, becslés
- Rendszerszintű veszély valószínűség számítása
  - AND kapu: **szorzat** (ha **független** események)  
pontos:  $P\{A \text{ és } B\} = P\{A\}P\{B|A\}$
  - OR kapu: **összegzés** (**felső becslés**)  
pontos:  $P\{A \text{ vagy } B\} = P\{A\}+P\{B\}-P\{A \text{ és } B\} \leq P\{A\}+P\{B\}$
- Problémák:
  - korreláló hibák
  - időbeli (hiba)szekvenciák kezelése

# Hibafa példa: Felvonó



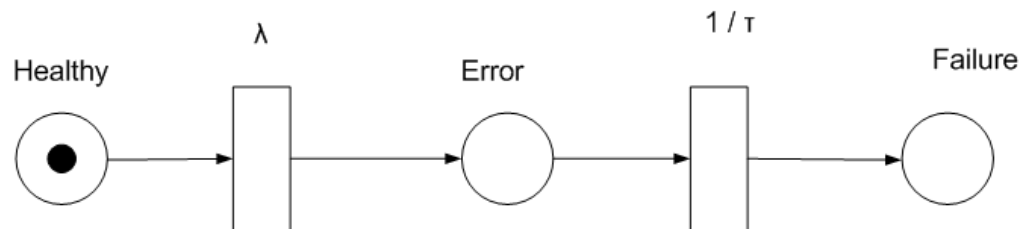
# Meghibásodási adatok

- Analízis alapja: meghibásodási valószínűségek
- Honnan lesznek jó adatok:
  - Becslés
  - Saját monitorozó rendszer
  - Külső tanulmányok, számok (hihetőség, pontosság?)
- Példák:
  - Cisco switch MTBF ~ 200000 óra (=22,8 év)
  - IBM S/390 mainframe MTTF 45 év
  - Windows XP MTTF 608 óra
  - webserverver MTTF ~ 16 nap...



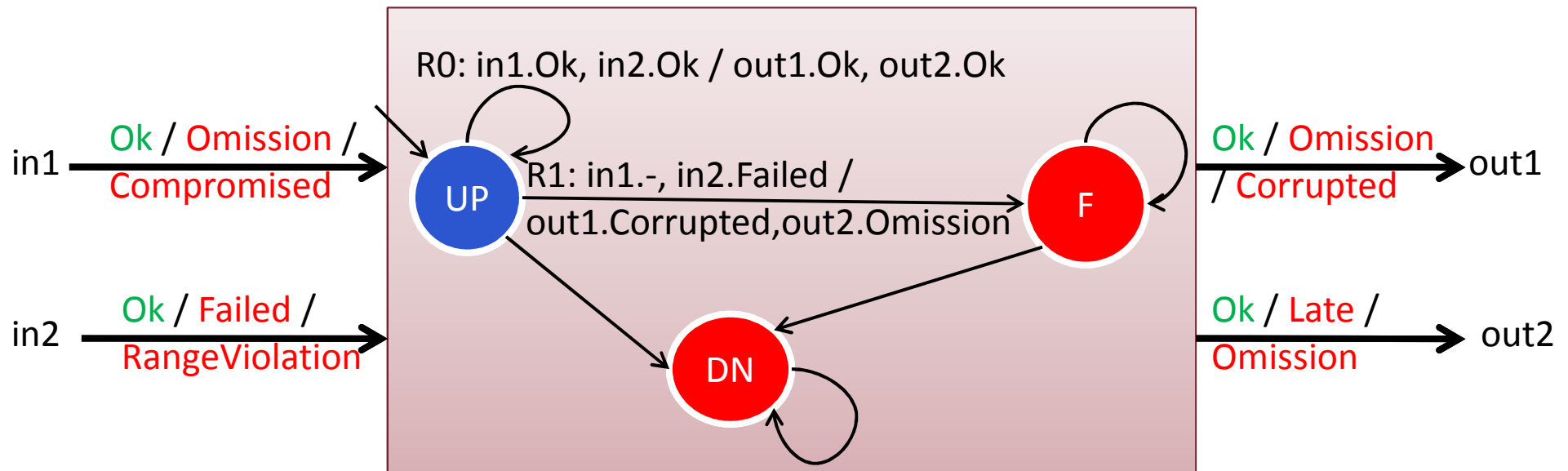
# Állapot alapú módszerek

- Hibák kvalitatív leírása: diszkrét viselkedésmodell
  - Állapotgép, adatfolyamháló, folyamat, Petri-háló...
- Kvantitatív: időzítés az állapotátmenetekhez
  - Determinisztikus
  - Valószínűségi eloszlás alapján:  
folytonos idejű, markovi sztochasztika



# Hibamodellezés adatfolyamhálóval

- Kvalitatív hibamodell → adatfolyamháló
  - Komponens → adatfolyam csomópont
  - Belső hibamódok → csomópont állapotai
  - Komponens kapcsolatok → csatornák
  - Kommunikációs hibamódok → csatorna tokenek



# Hibamodellezés adatfolyamhálójával

- Hibaterjedés
  - Hibás komponens állapot → hibás üzenet
  - Hibás üzenet → hibás komponens állapot
- Kvalitatív analízis
  - Előre: egy adott hibának mik a következményei?
  - Hátra: egy hibajelenségnek mi lehet az oka?
- Egy (nem teljes) megoldási módszer:
  - Constraint Satisfaction Problem (CSP)  
Kényszerkielégítési feladat

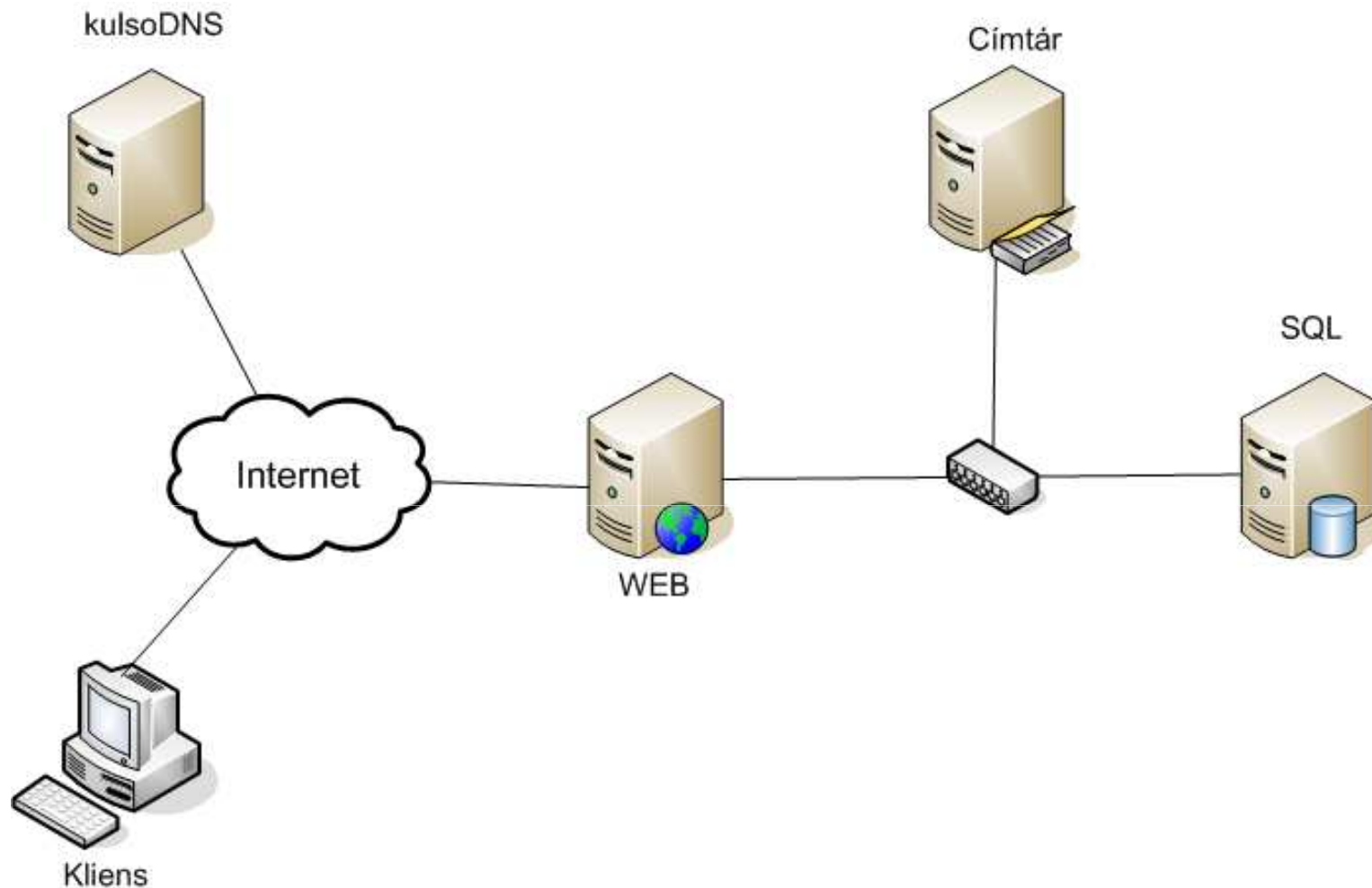
# Feladat

- Táborúzőn szeretnénk tükörtojást sütni.
  - Egészen pontosan két tojásból szeretnénk készíteni.
  - Sajnos nem biztos, hogy valóban van még két tyúktojásunk, így az is elképzelhető, hogy kénytelenek vagyunk kisebb adaggal beérni (vagy semmivel).
  - Ráadásul, ha csak az egyik tojás is záp, már az is elrontja az egész ételt.
  - Azzal is számolni kell, hogy a tűz esetleg túl gyenge; ilyenkor, ha egyetlen tojás nem is, de két tojás kellemetlenül lassan sül ki, mi pedig éhesek vagyunk.

# Feladat

- A „tábortűz” komponensnek milyen bemenetei és kimenetei azonosíthatóak a fenti modellben? Mik lesznek a komponens és a csatornák lehetséges hibamódjai?
- Milyen hibaterjedés lehetséges ebben a rendszerben?
- Készítse el a "Nem eszünk semmit" rendszerszintű hibajelenséghez tartozó hibafát!

# Példa: szolgáltatásbiztonság analízise

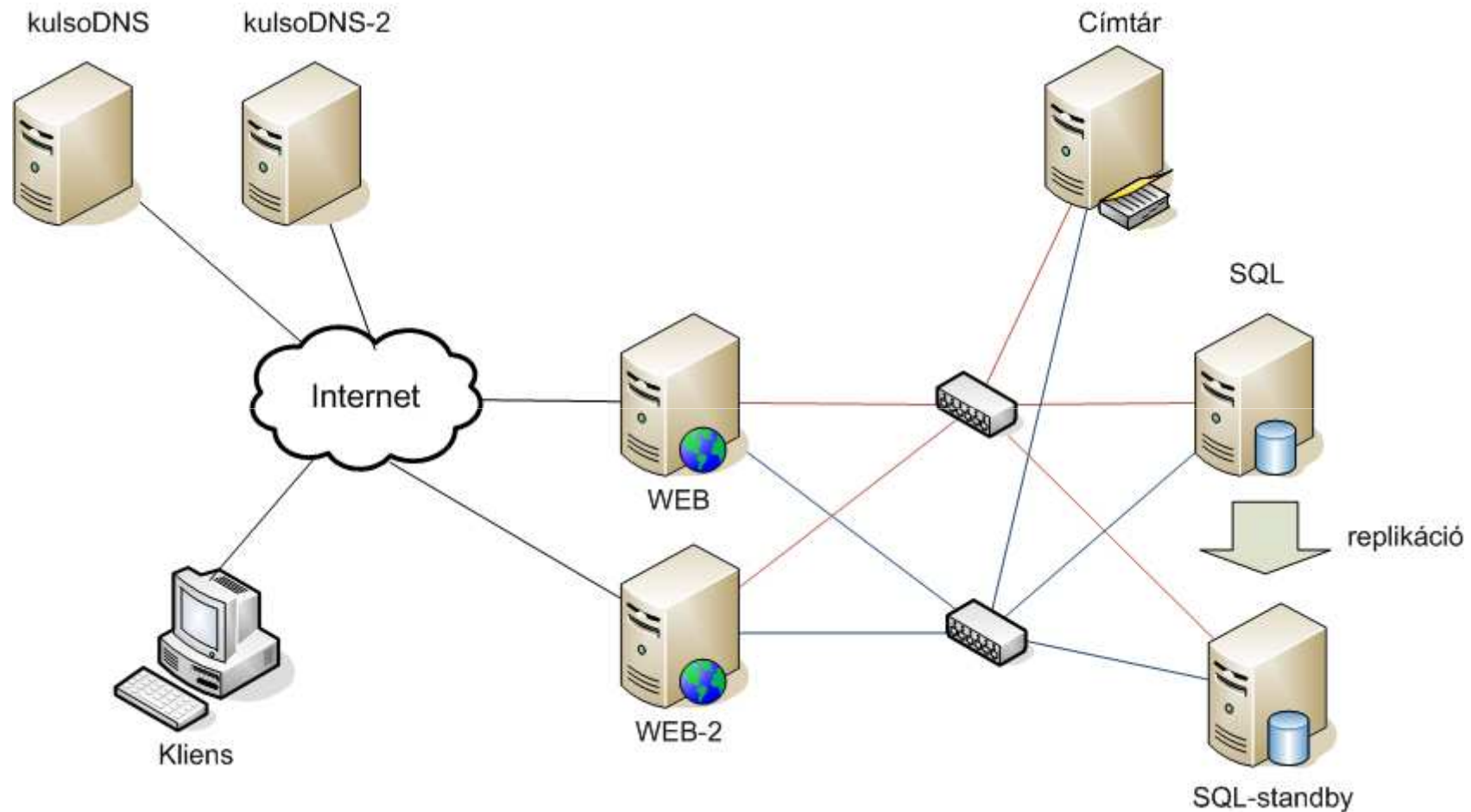


**Feladat:** Milyen meghibásodások esetén nem lesz elérhető a szolgáltatás (webáruház)?

# Feladat: Meghibásodások azonosítása

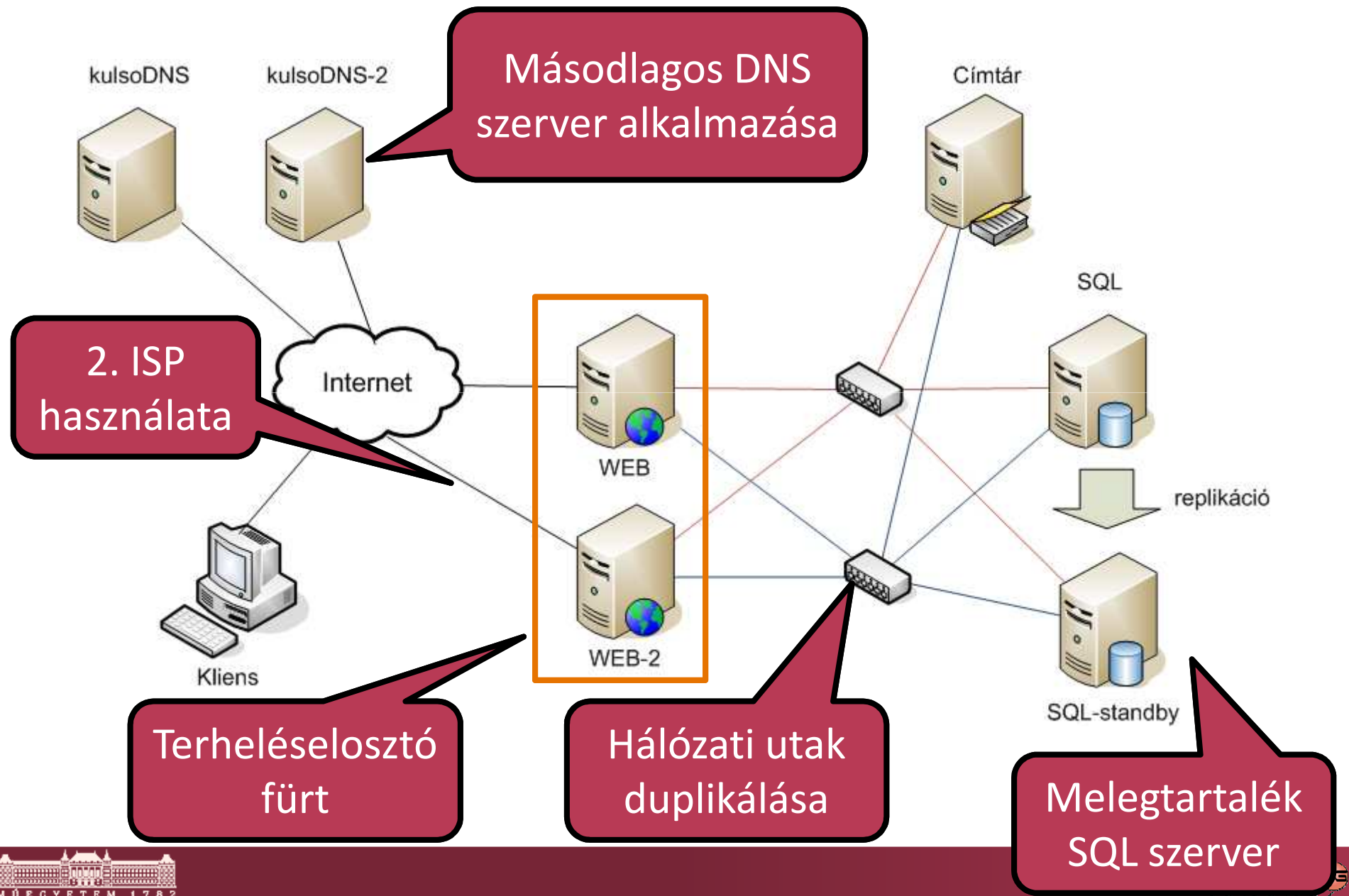
- Milyen meghibásodás esetén nem lesz elérhető a szolgáltatás (webáruház)?
- Áramkimaradás, HW hiba, hálózati elem/kábel hiba, szerver szolgáltatások hibája, alkalmazás hiba, frissítés telepítése, túlterhelés, támadás, félrekonfigurálás, verzió inkompatibilitás, vírus...

# Példa: hibatűrés beépítése

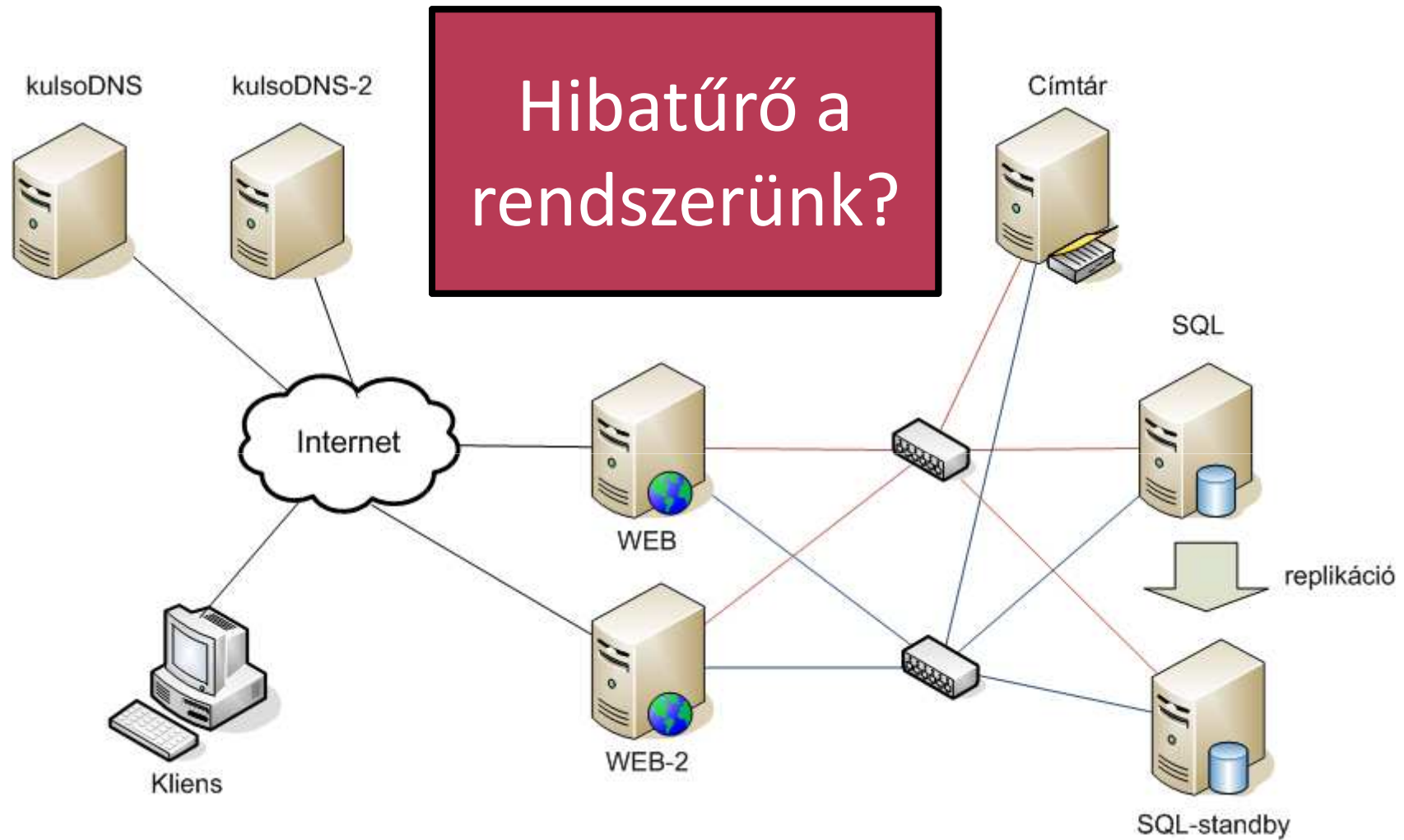




# Példa: hibatűrés beépítése

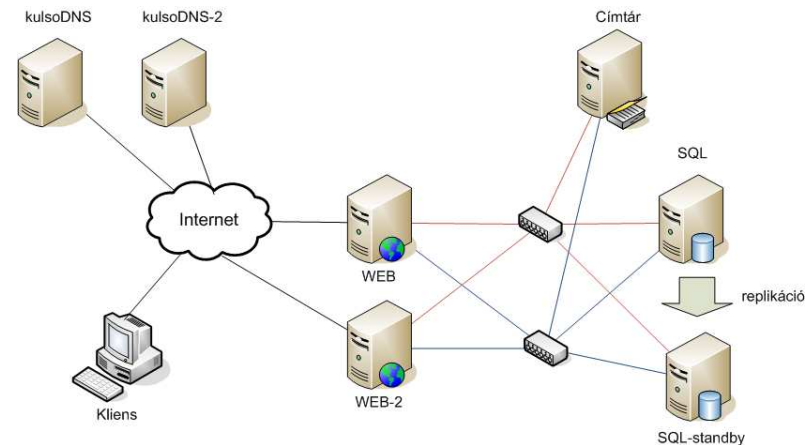


# Példa: hibatűrés beépítése



# Példa: hibatűrés beépítése

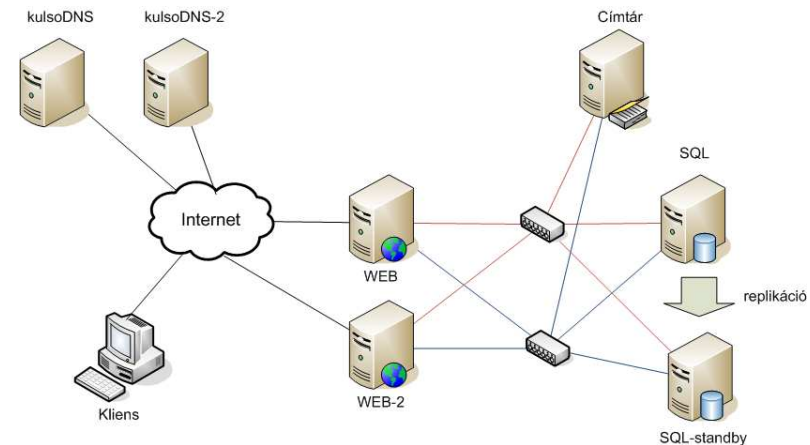
Hibatűrő a rendszerünk?



- Attól függ:
  - Bizonyos SPOF-ek ellen védekeztünk
- DE
  - sok kiesési lehetőség maradt még
  - Adatok törlése, teljes serverterem elpusztulása, adminisztrátori hibák, OS hotfix miatti újraindítás...

# Példa: hibatűrés beépítése

Hibatűrő a rendszerünk?

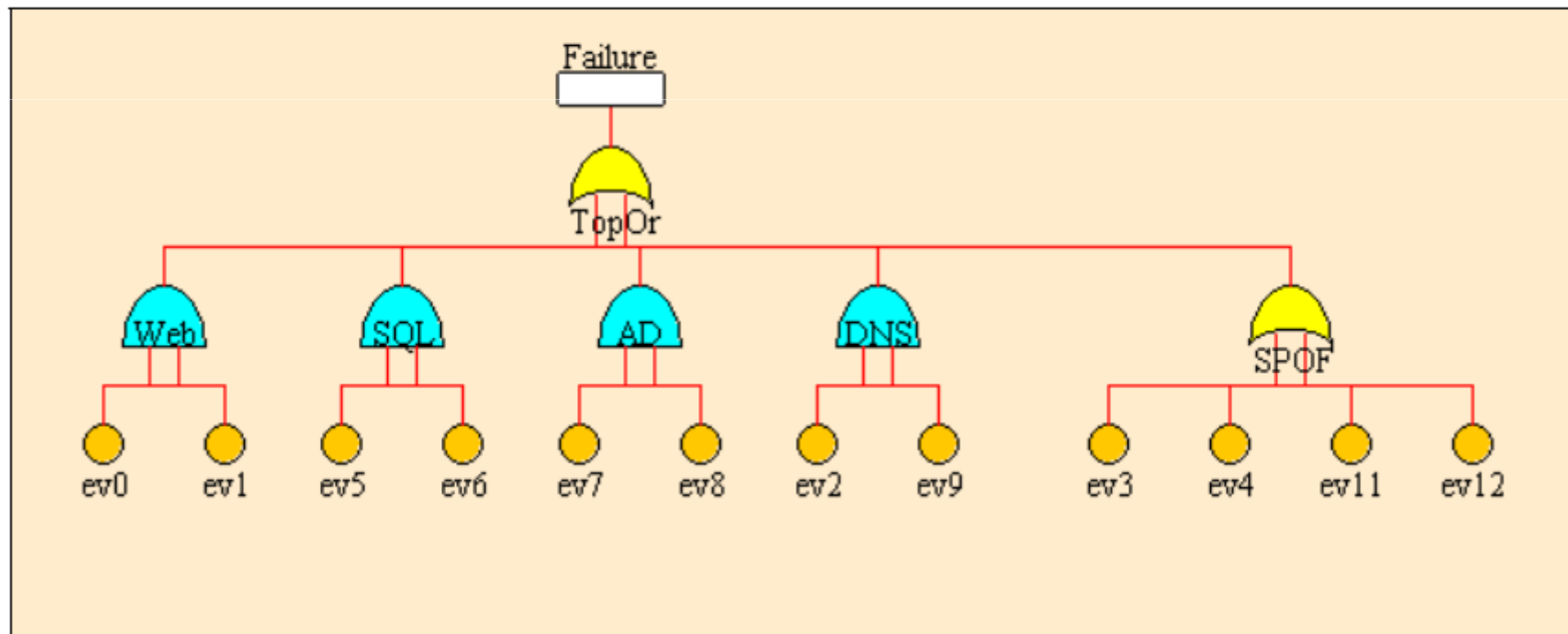


- At...
- Tanulság: mindig tudjuk, hogy
- • mi ellen akarunk védekezni,
- DE
- • milyen módszerek vannak arra,
- • megéri-e védekezni

adminisztrátori hibák, OS hotfix miatti újraindítás...

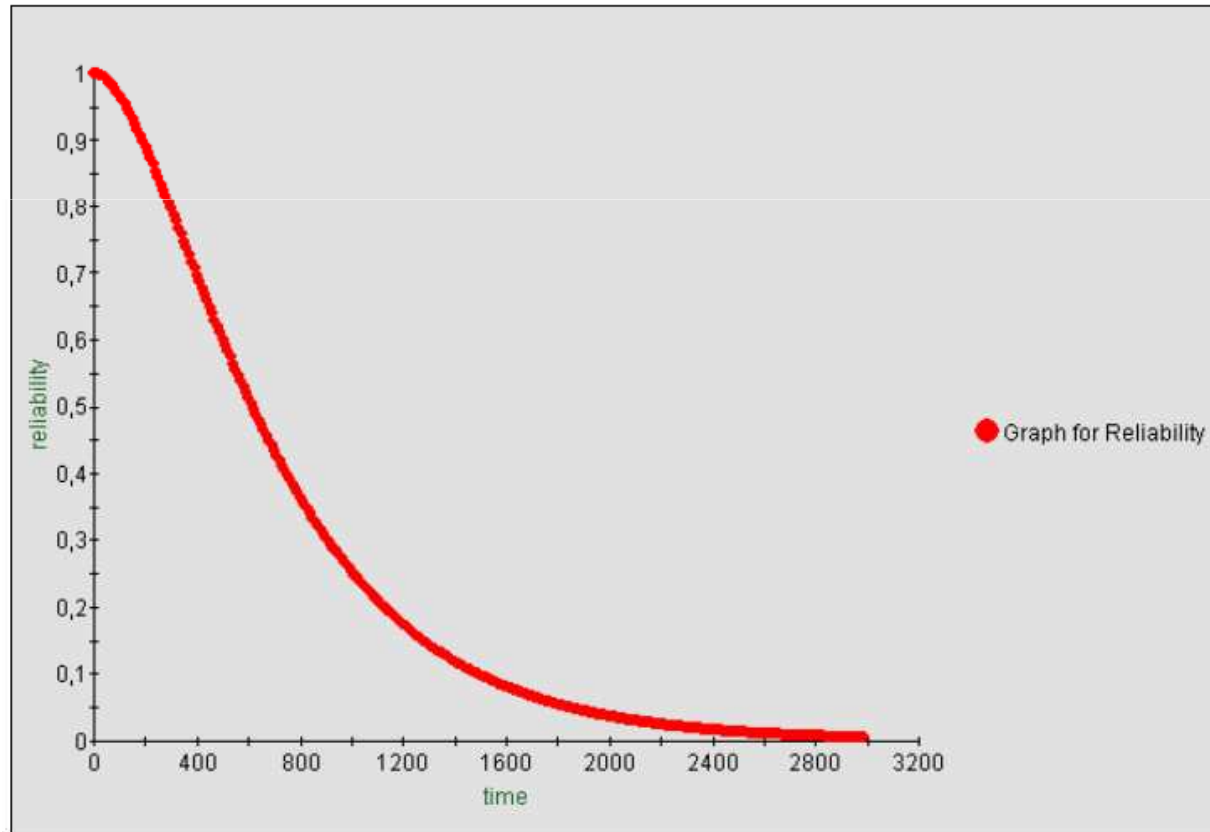
# Analízis: hibafa

- SHARPE eszköz
- Hibafa rajzolása



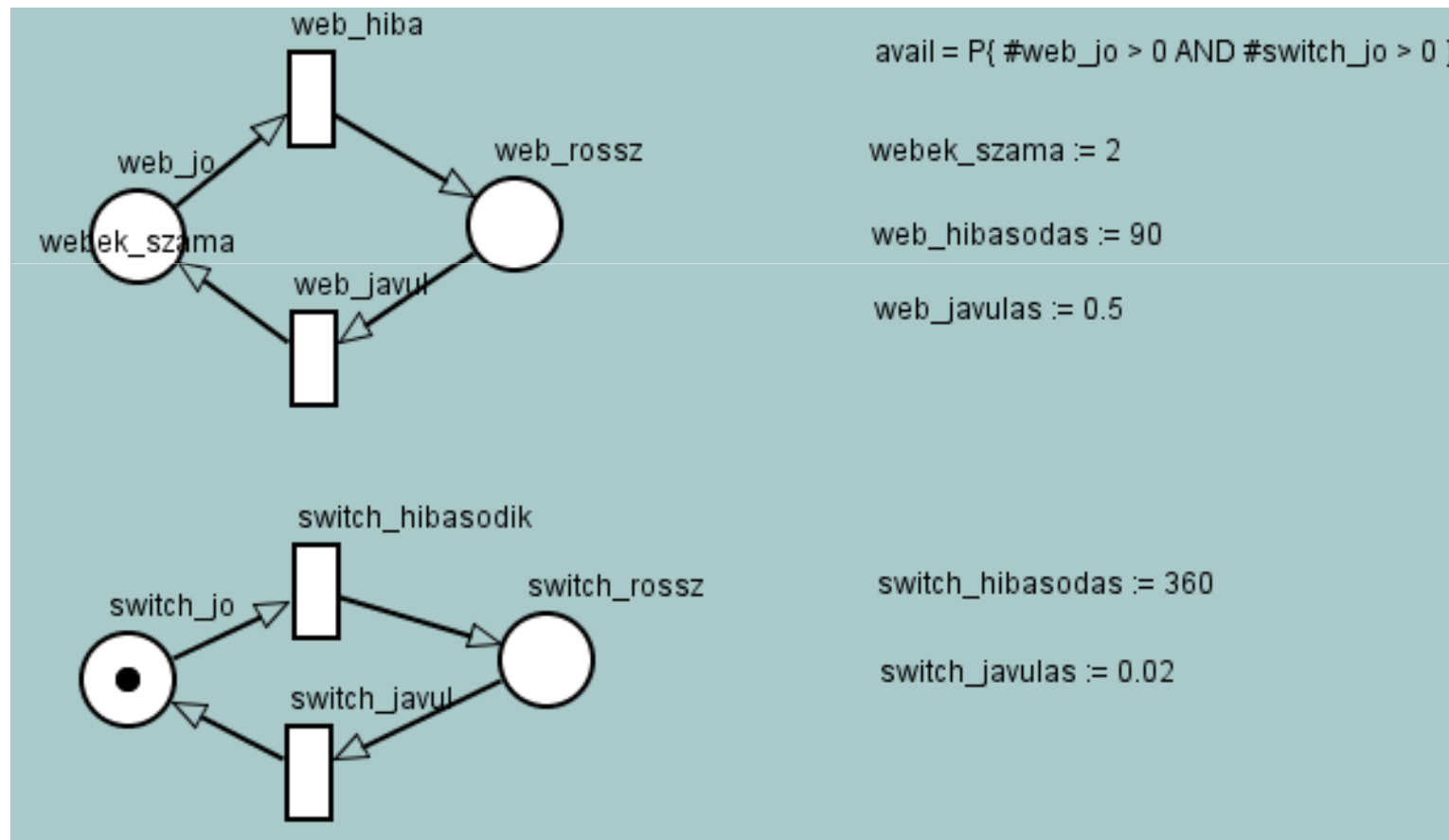
# Analízis: hibafa

- Alapszintű meghibásodásokhoz bekövetkezési valószínűség rendelése
- Megbízhatóság számolása:



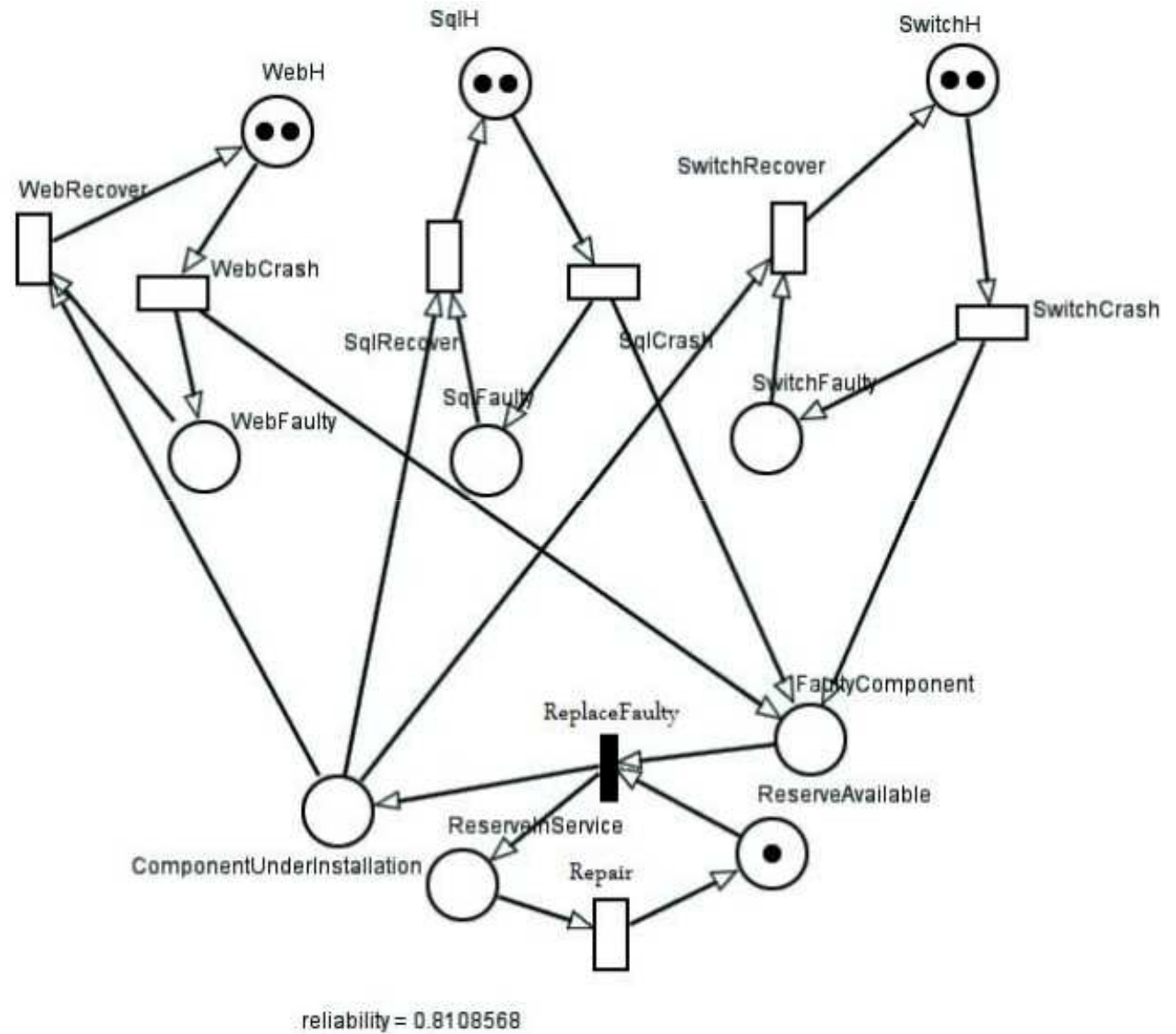
# Analízis: Petri-háló

- [TimeNET](#) eszköz
- Alap blokkok és paraméterek



# Analízis: Petri-háló

A teljes modell:





# Analízis: érzékenység és költségvizsgálat

- Érzékenység: melyik paraméter változása befolyásol a legjobban:

ÁTMENET	50% KÉSLELTETÉS	RENDELKEZÉSRE ÁLLÁS	200% KÉSLELTETÉS	RENDELKEZÉSRE ÁLLÁS
WebRecover	0,5	0,8091	2	0,8022
SqlRecover	1	0,8195	4	0,7846
SwitchRecover	0,25	0,8073	1	0,8133
Repair	2	0,9598	8	0,3955

- Költségoptimalizálás:

	Költség	Rendelkezésre állás	Kiesés	Kiesés költsége	Nyereség
Alapmodell	0	0,904	34,931	3 493 050,000	
Tartalék SQL	500 000	0,913	31,792	3 179 150,000	-186 100,000
<b>Tartalék web</b>	<b>500 000</b>	<b>0,921</b>	<b>28,945</b>	<b>2 894 450,000</b>	<b>98 600,000</b>
Tartalék mindkettőből	1 000 000	0,930	25,733	2 573 250,000	-80 200,000
Web szerver	1 000 000	0,914	31,463	3 146 300,000	-653 250,000
SQL szerver	2 000 000	0,914	31,536	3 153 600,000	-1 660 550,000
Web szerver + tartalék	2 000 000	0,933	24,565	2 456 450,000	-963 400,000
SQL szerver + tartalék	3 000 000	0,936	23,506	2 350 600,000	-1 857 550,000

# Összefoglalás

- Szolgáltatásbiztonság
  - Jellemzők, hatáslánc, eszközök
- Hibatűrés
  - Redundancia megjelenése
- Analízis:
  - Mérnöki és matematikai módszerek
  - Hibamódok azonosítása
  - Megfelelő védekezési módszer kiválasztása