Szoftver-minőségbiztosítás

Biztonság kritikus szoftverek
Hibatűrés
Szoftver-diverzitás

Biztonság, biztonságosság

- **Mentesség azoktól a feltételektől, melyek halált, sérülést, anyagi és eszköz veszteséget, környezeti kárt okoznak.**
- Az az elvárás egy rendszerrel szemben, hogy meghatározott feltételek mellett nem kerül olyan állapotba, hogy az emberi életet veszélyeztetne.
- Relatív: nincs teljesen biztonságos rendszer, csak biztonságosabb (mint egy másik)
  - **elfogadható szintű kockázat**ot jelent
    - kockázat áthelyezés
  - korlátozott feltételek mellett érvényes
  - Egy rendszer kockázatoktóli szababadságfokának mértéke.

Terminológia

- Veszély [kockázat] (hazard)
- Incidens (near miss)
- Baleset, szerencsétlenség (accident, mishap)
- Kockázat (risk)
- Hibás működés (fail operation)
- Biztonságos csökkentett működés (fail safe operation)

Biztonság kritikus rendszerek

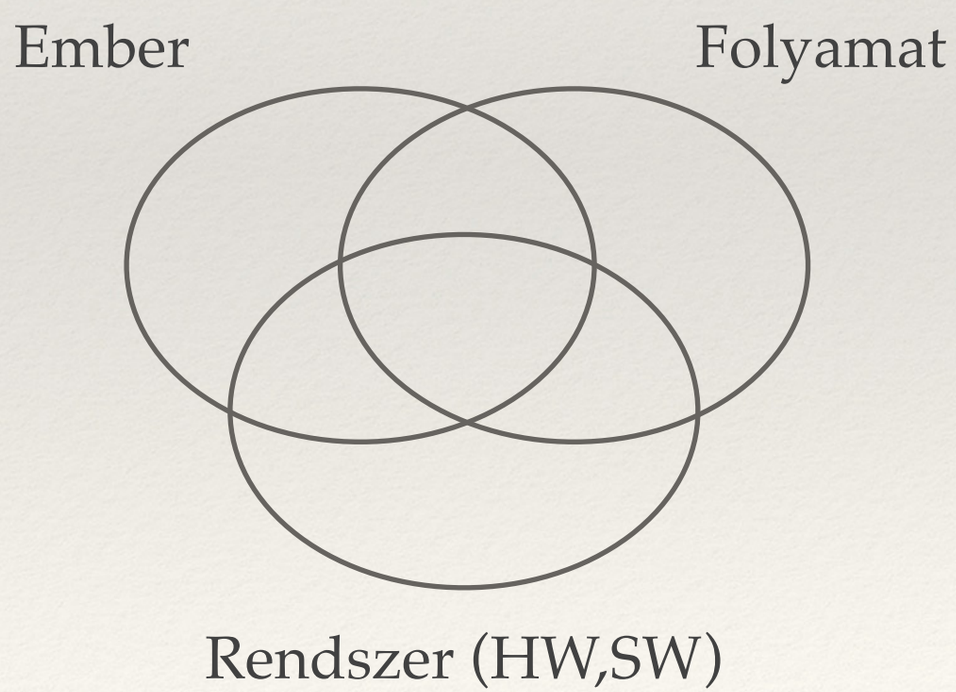
- A meghibásodás (hibás működés) következménye elfogadhatatlan.
- Hagyományos területek: egészségügyi alkalmazások, légi közlekedés, atom energia ipar, fegyverrendszerek stb.
- Kritikus infrastruktúra alkalmazások: telekommunikáció, elektromos energia termelés és elosztás, bank és pénzügyi rendszerek, stb.

A biztonság mérnöki megközelítése

- **Biztonság** ~ kockázatok (risk) kezelése
  - balesetek, szerencsétlenségek elkerülése
- **Megbízhatóság** ~ szándékolt működés (adott időben, környezetben)
  - meghibásodások elkerülése
- Biztonság ~ Megbízhatóság
  - meghibásodások -> kockázatok (hazard) [veszély]

Biztonsági kontextus

- Több összetevő:
- Hardver megbízhatóság
  - fizikai / sztochasztikus megközelítés
- Szoftver megbízhatóság
  - szisztematikus hibák
- Emberi megbízhatóság
- Folyamatok megbízhatósága



RAMS megközelítés

- **R**eliability **A**vailability **M**aintainability **S**afety
- RAMS paraméterek: kvantitatív, kvalitatív indikátorok a rendszer működésére vonatkozóan
- RAM paraméterek ellenőrzése, elemzése és felhasználása a biztonsági elemzésekben, tervezésben
  - kockázat elemzés
  - hibamódok elemzése
  - RAM modellezés
  - hiba-fa elemzés
  - biztonsági integritás szint elemzés

RAMS tevékenységek az életciklusban

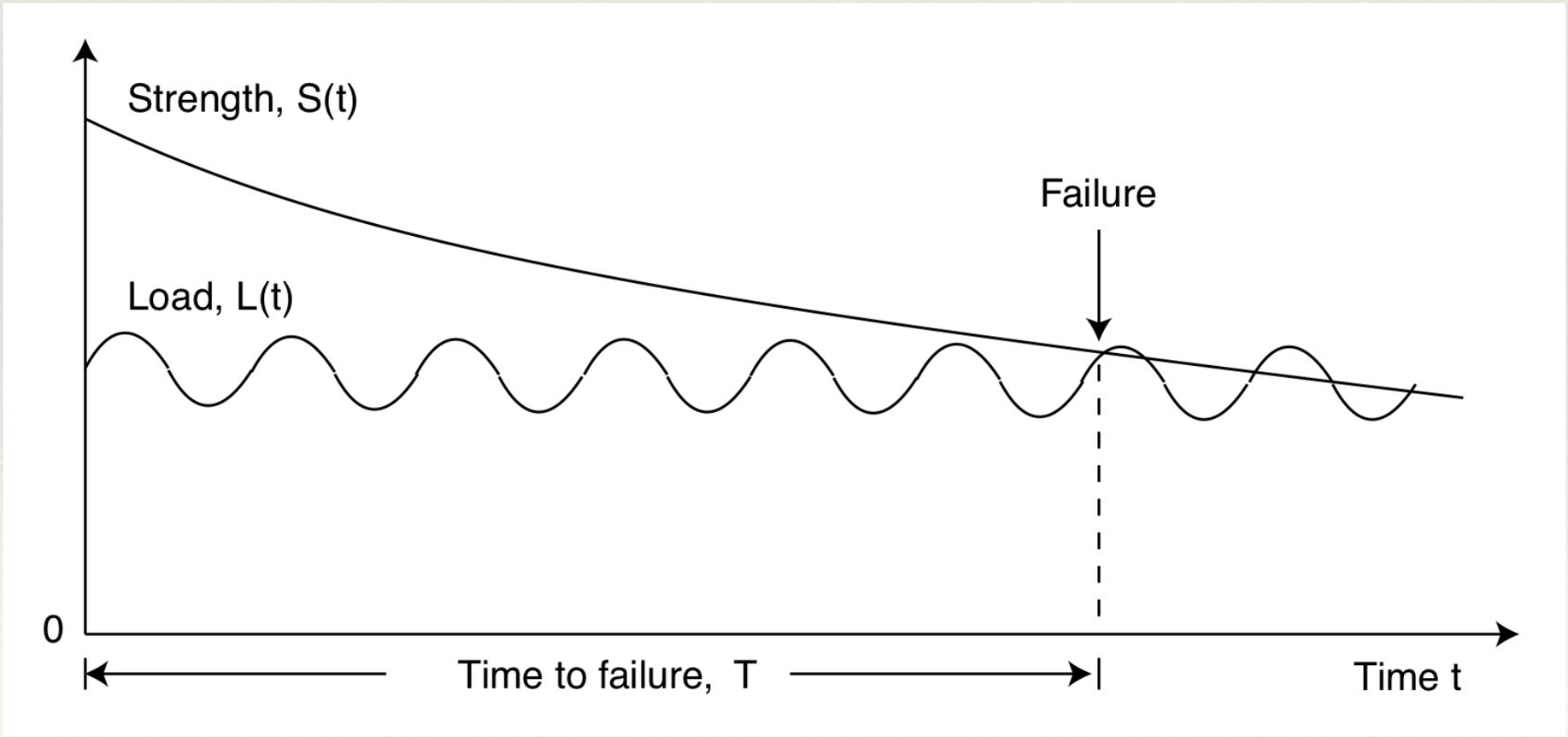
- Követelmény meghatározás
  - Misszió definíció
  - RAMS **követelmények** meghatározása
- Tervezés
  - Megbízhatóság allokálás (hol mennyi)
  - Kockázat/hazárd (veszély) azonosítás
  - Tervezési opciók értékelése
- Implementálás
  - **Terv validálás**
  - **Megbízhatósági tesztelés**
  - RAMS teljesítmény biztosítása
- Működtetés
  - Adatgyűjtés és elemzés

Megbízhatósági mértékek

- Megbízhatósági (túlélési) függvény R(t)
- Meghibásodási ráta függvény h(t)
- Hibák közötti átlagos idő (MTBF)
- Átlagos maradék élettartam (MRL)

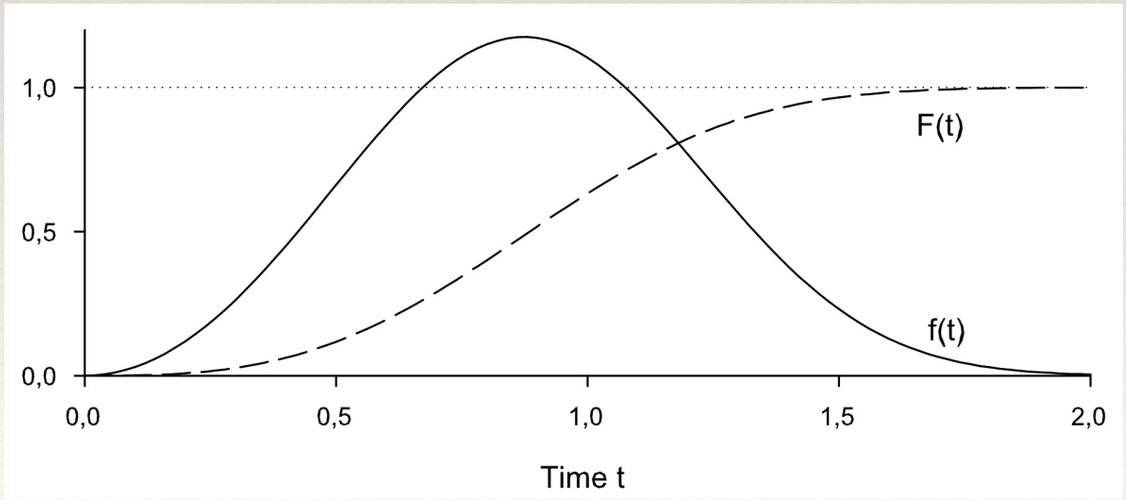
Megbízhatóság

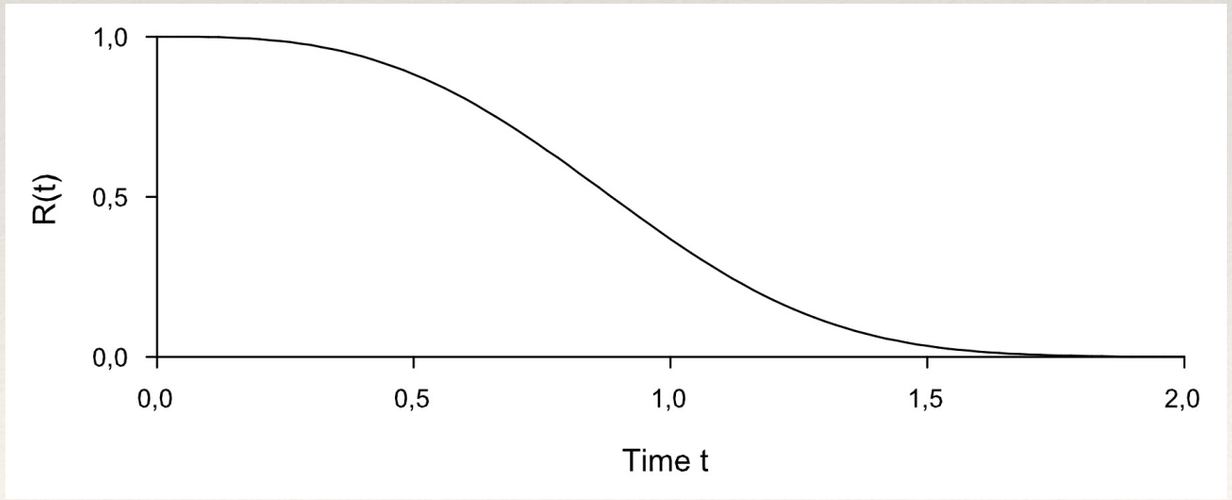
- Időfüggő
- A meghibásodás ideje T (time to failure): val. változó
  - pl. hardverekre:



Megbízhatósági függvény

- T eloszlás függvénye **F(t)=Pr(T≤t)** t>0 (annak a valószínűsége, hogy a meghibásodás a (0,t] intervallumban bekövetkezik
- Megbízhatósági függvény **R(t)=Pr(T>t)=1-F(t)** (annak a valószínűsége, hogy az egység nem fog meghibásodni a (0,t] idő intervallumban



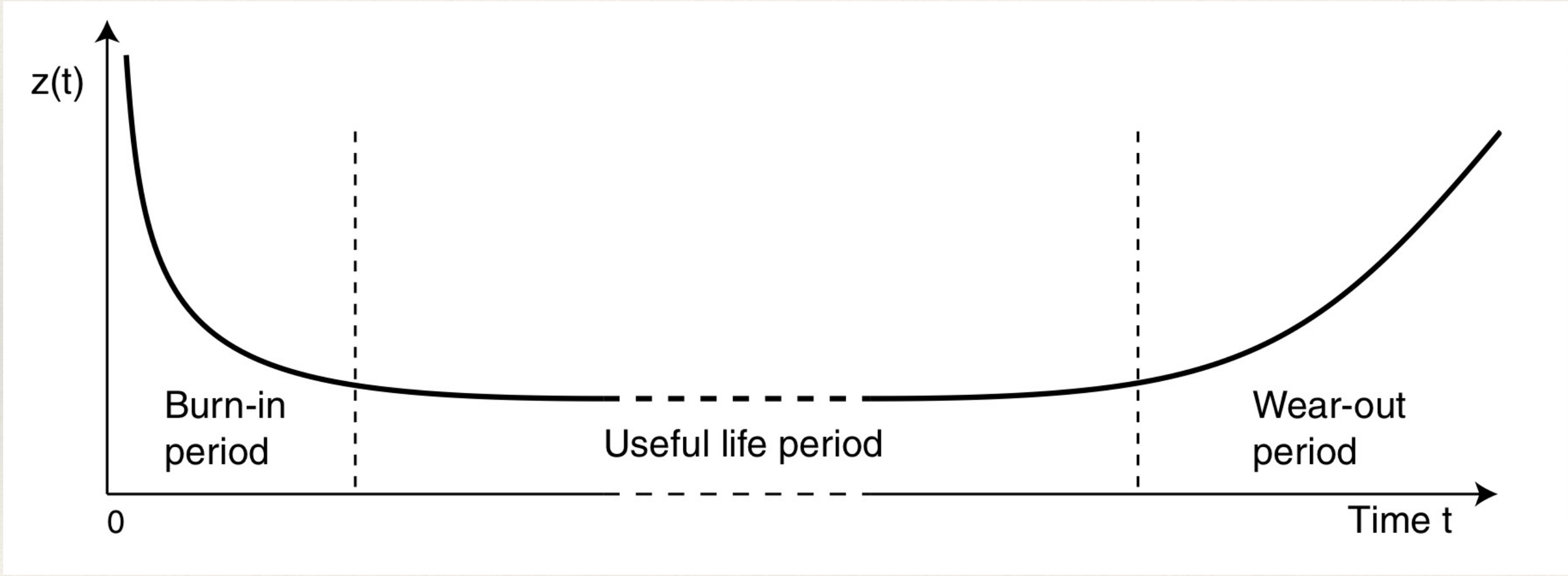


Hibaráta függvény

- Hibaráta függvény (hazard function): $h(t)=f(t)/R(t)$
  - meghibásodás valószínűsége egy rövid idő ($\Delta t \to 0$) alatt, ha t időpontban működik a rendszer
  - F(t) exp. eloszlás esetén konstans (λ)
  - F(t) Weibull eloszlás ($\alpha>1$) esetén monoton nő (elhasználódás)
  - F(t) Pareto eloszlás esetén monoton csökken (bejáratás)

Hibaráta függvény hardver esetén

- Fürdőkád görbe



Meghibásodás átlag ideje

- MTTF (**M**ean **T**ime **T**o **F**ail)

$$\mathrm{MTTF} = E(T) = \int_0^\infty t f(t)\, dt$$

- Ha MTTF<∞

$$\mathrm{MTTF} = \int_0^\infty R(t)\, dt$$

- Ha a rendszer javítható
  - MTTR (**M**ean **T**ime **T**o **R**epair) átlagos javítási idő
  - MTBF (**M**ean **T**ime **B**etween **F**ailures)
    - MTBF=MTTF+MTTR

Rendelkezésre állás

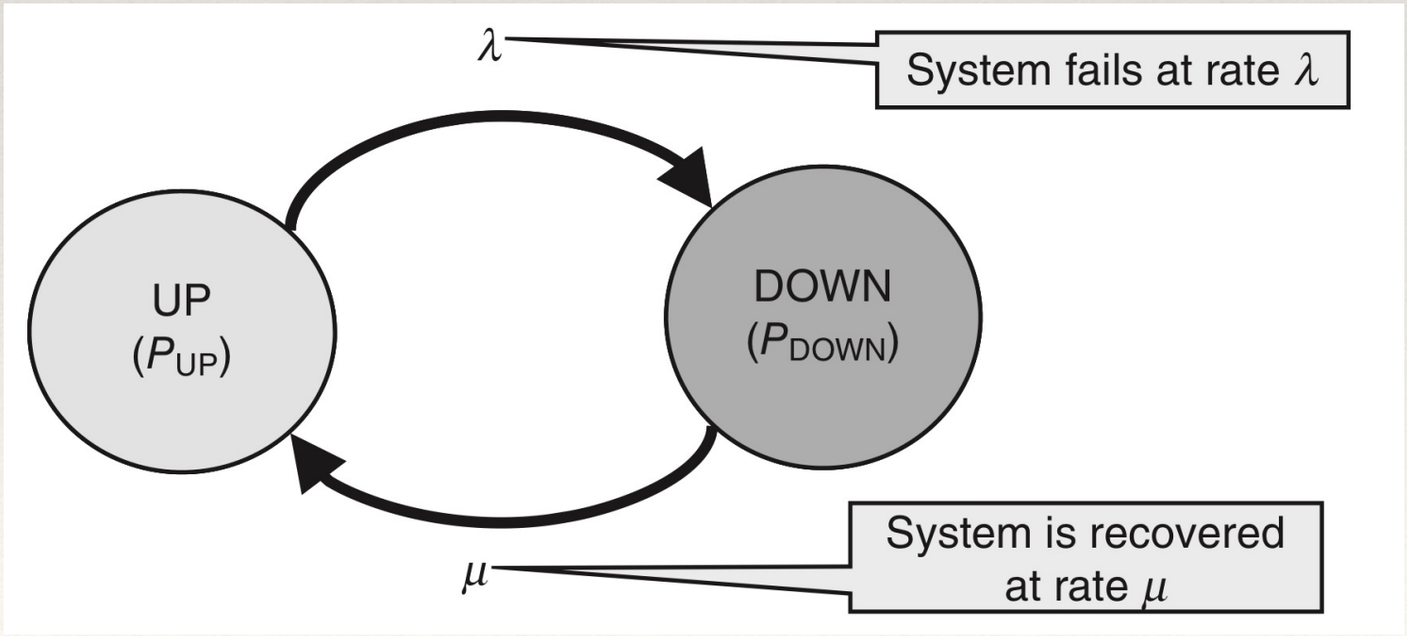
- Egy rendszer képessége, hogy olyan állapotban van, melyben a megkövetelt funkciót, szolgáltatást nyújtja
- Rendelkezésre állási függvény: A(t)=Pr(az egység működik t időpillanatban)
- Diszkrét mérték: A=MTTF/(MTTF+MTTR)

Karbantarthatóság

- Annak a valószínűsége, hogy egy adott karbantartási tevékenység egy adott idő alatt (idő intervallumban) kivitelezhető.
- Annak a mértéke, hogy egy rendszer visszaállítható meghatározott állapotba (megkövetelt funkcionalitás nyújtására képes) karbantartással.

Markov láncok

- Sztochasztikus folyamat
  - diszkrét, memória nélküli (Adott jelen mellett a jövő feltételesen független a múlttól.)
  - adott valószínűségi változó eloszlása alapján változtatja az állapotát (átmenet-valószínűségek)



Markov láncok (folyt.)

- Állapottér (halmaz)
- Állapot egy adott időpontban
  - trajektória
  - Tranzíció mátrix

$$p_{ij} = \mathbb{P}(X_{t+1} = j \mid X_t = i).$$

- Valószínűségi eloszlás egy időpontban

$$\begin{aligned} X_0 &\sim \boldsymbol{\pi}^T \\ X_1 &\sim \boldsymbol{\pi}^T P \\ X_2 &\sim \boldsymbol{\pi}^T P^2 \\ &\vdots \\ X_t &\sim \boldsymbol{\pi}^T P^t. \end{aligned}$$

Markov láncok és rendelkezésre állás

- Állapotok valószínűségének összege

$$P_{UP}+P_{DOWN}=1$$

- Hosszútávon állandósult állapotban

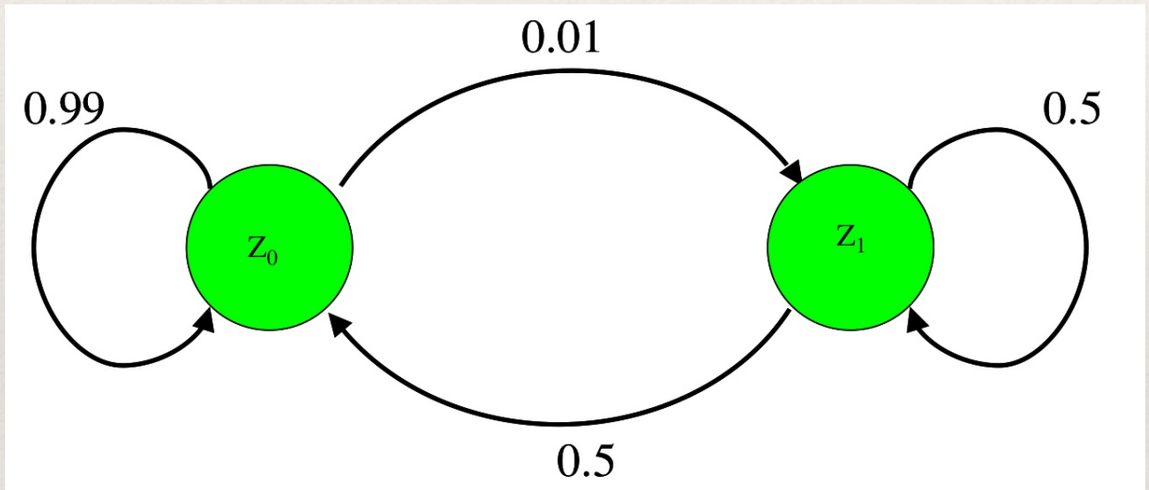
$$RateIn_{State}-RateOut_{State}=0$$

$$\left.\begin{aligned} P_{\mathrm{UP}} + P_{\mathrm{DOWN}} &= 1 \\ P_{\mathrm{UP}} \times \lambda - P_{\mathrm{DOWN}} \times \mu &= 0. \end{aligned}\right\}$$

$$P_{\mathrm{UP}} = \frac{\mu}{\lambda + \mu}.$$

Példa

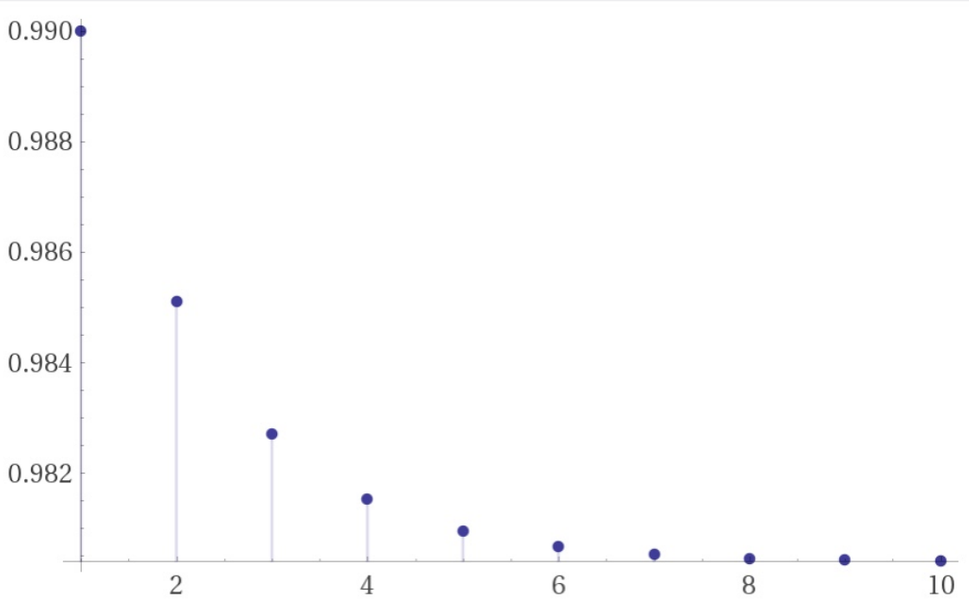
Tekintsünk egy on-demand működésű szervert. Átlagban minden 100 kérésből 1-nél nem megfelelő eredményt szolgáltat. Egy ilyen meghibásodás után, 50% valószínűséggel megjavul és újra jól működik a következő kérés hatására. Mennyi a rendszer rendelkezésre állása?



A=0.5/(0.5+0.01)=0.980392

discrete plot	$\left(\begin{pmatrix}1\\0\end{pmatrix}^{\mathsf{T}}\cdot\begin{pmatrix}0.99 & 0.01\\0.5 & 0.5\end{pmatrix}^{n}\right)[\![1,\,1]\!]$	$n = 1$ to 10

Plot



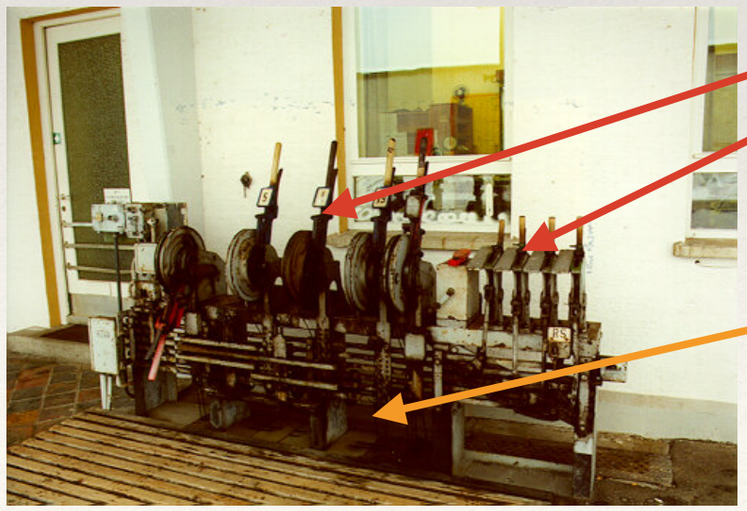
Megbízhatóság (Dependability)

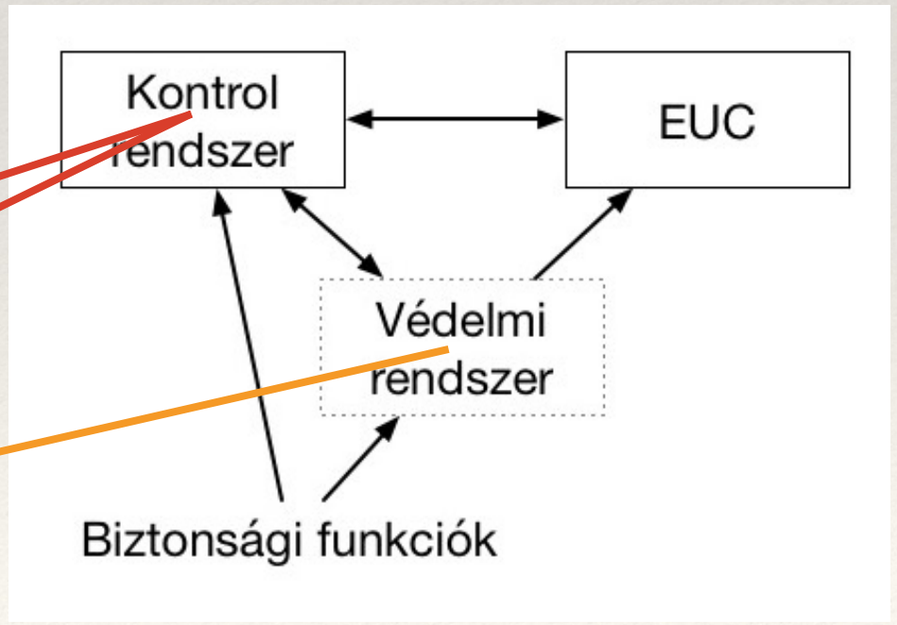
- Gyűjtő fogalom; összefoglalja a rendelkezésre állási, a megbízhatósági és karbantarthatósági jellemzőket.
- Annak a mértéke, hogy egy rendszer működtethető és a megkövetelt funkciókat nyújtja tetszőleges időpontban a meghatározott profillal.

Biztonság kritikus rendszerek

- Biztonsági műszerezett rendszer (**S**afety **I**nstrumented **S**ystem): olyan rendszer, mely eléri és fenntartja a biztonságos állapotot biztonsági funkciók implementálásával egy adott berendezésre (**E**quipment **U**nder **C**ontrol) nézve.

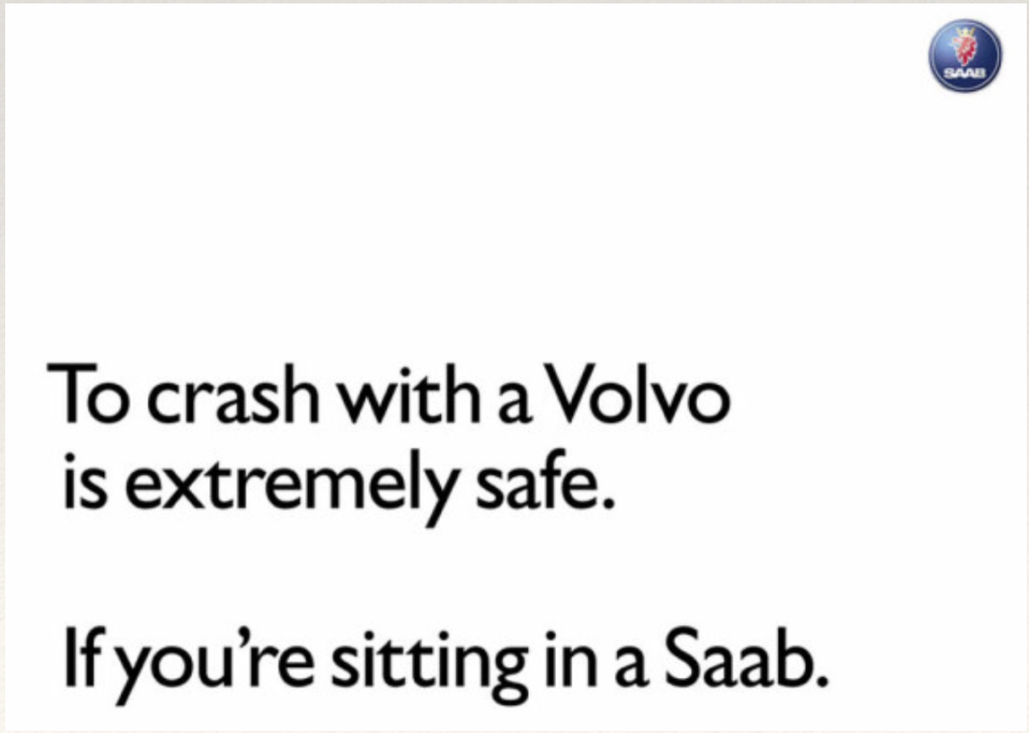
- EUC + EUC control system





Funkcionális biztonsági követelmények

- Biztonsági funkció követelmények: A SIS által megvalósított biztonsági követelmények
- Biztonság integritási követelmények: A biztonsági funkciók kielégítő szintű nyújtásának valószínűségére vonatkozó követelmények. Ezek a követelmények kockázat elemzéssel határozhatók meg.



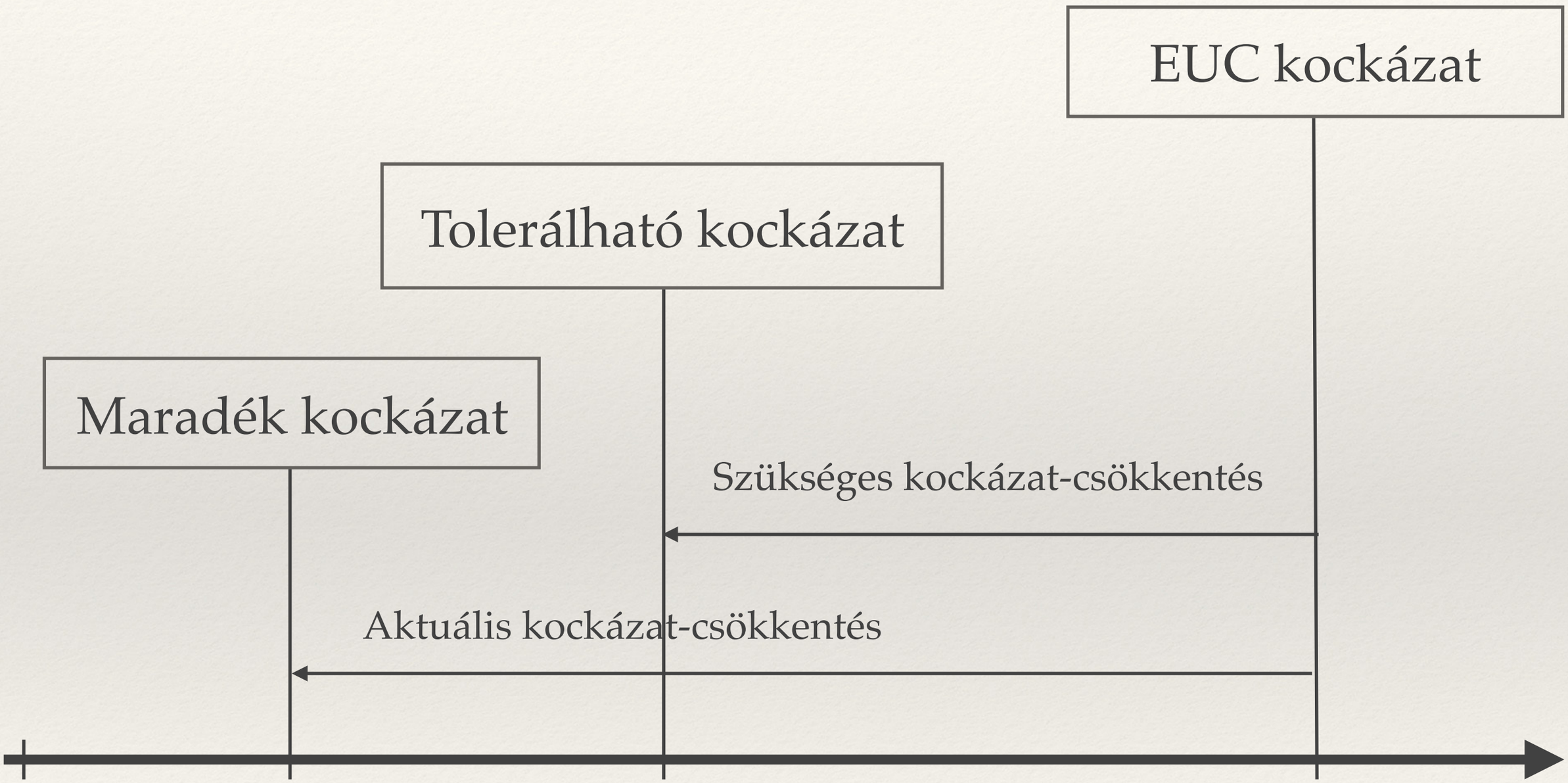
Biztonságos állapot

- A kontrolált berendezés olyan állapota, melyben a biztonsági követelmények teljesülnek.
- A kontrolált berendezés egy potenciálisan veszélyes állapotból a végső biztonságos állapotba való átmenet közben felvehet átmeneti biztonságos állapotokat. A biztonságos állapot folytonos kontrolt követelhet.

Védelmi rétegek

- A védelmi rétegek szabályozottan redukálják a kockázatot **megelőzéssel** és **veszélycsökkentéssel**.
- A védelmi rétegek jellemzői:
  - specifikusság (a veszélyes események bizonyos körére)
  - függetlenség (rétegek egymástól, közös hibaforrások kizárása)
  - megbízhatóság (véletlen és szisztematikus hibák)
  - auditálhatóság, validálhatóság

Biztonsági mechanizmusok célja

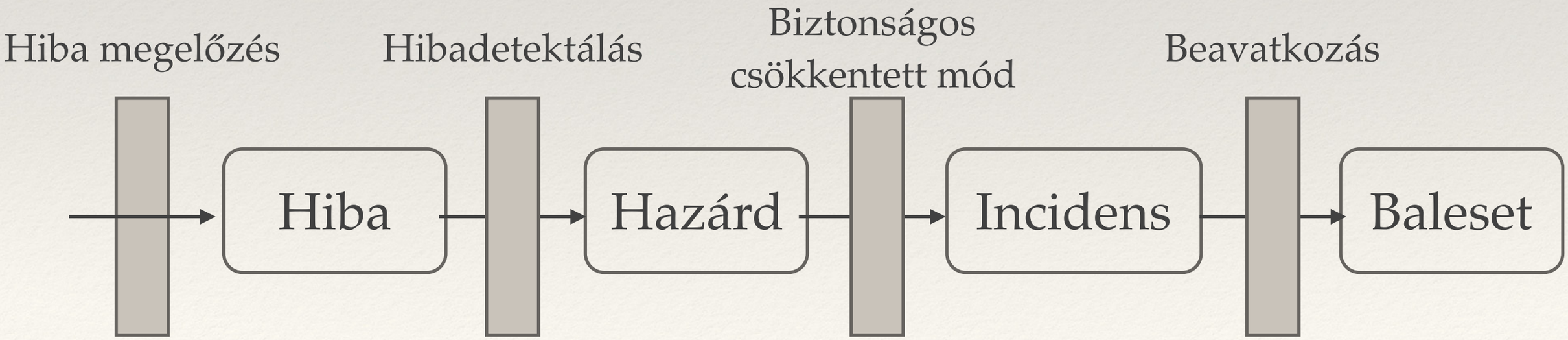


Biztonsági mechanizmus stratégiák

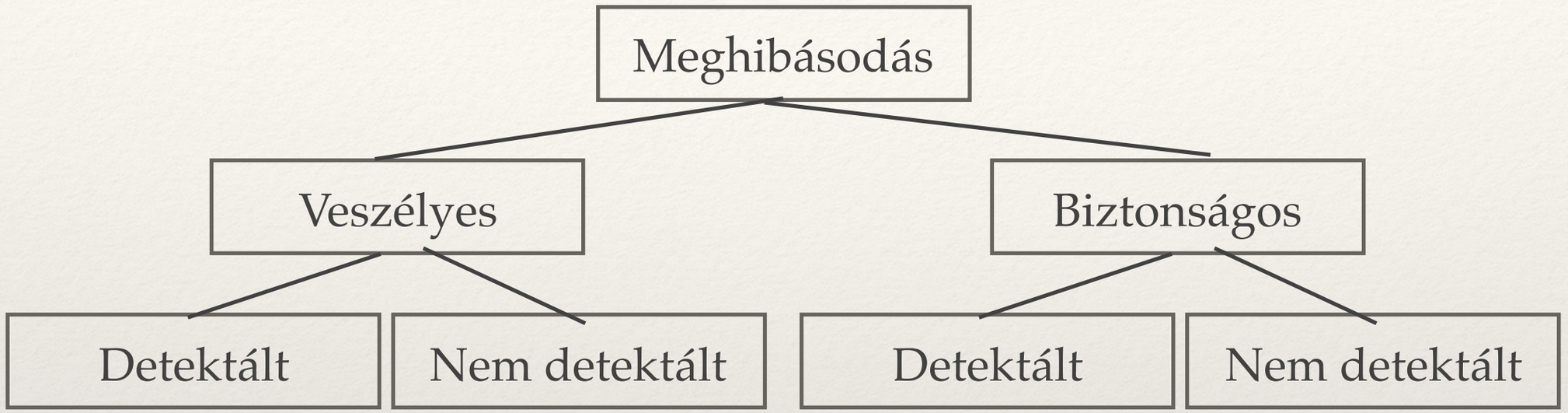
- Safe-life
  - Hiba kizárás, felújítás/csere mielőtt meghibásodás történne.
- Fault-tolerant
  - Hibatűrés/hiba elfedés
- Fail-safe
  - Hibabiztos megvalósítás (hibás állapot biztonságos)

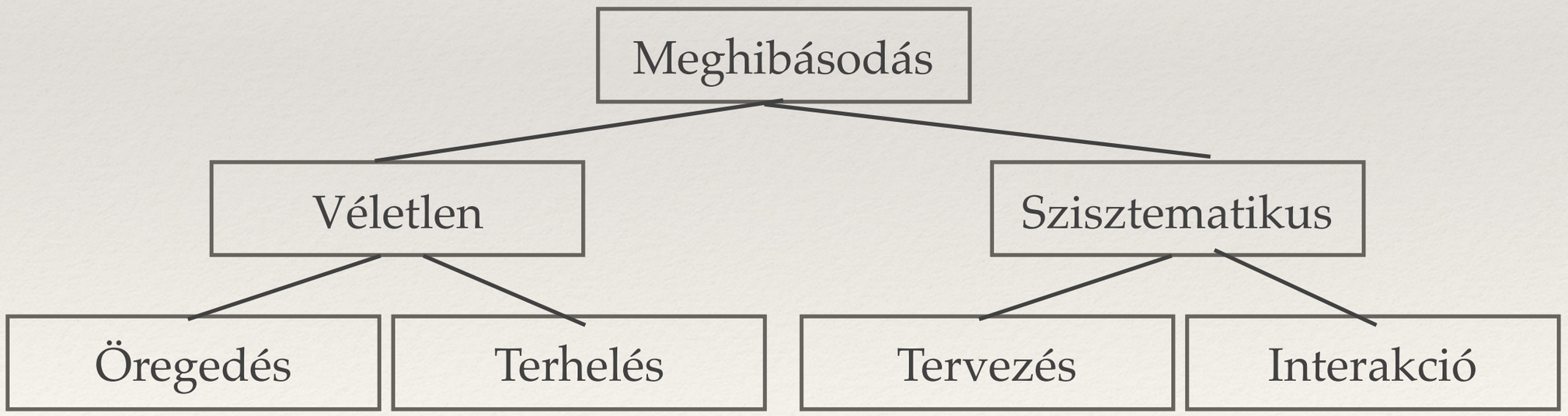
Veszély-csökkentés

- Hibázások kizárása
- Hibák detektálása és elhatárolása
- Veszély csökkentése biztonságos csökkentett működéssel
- Operátori (humán) beavatkozások



Meghibásodások kategorizálása





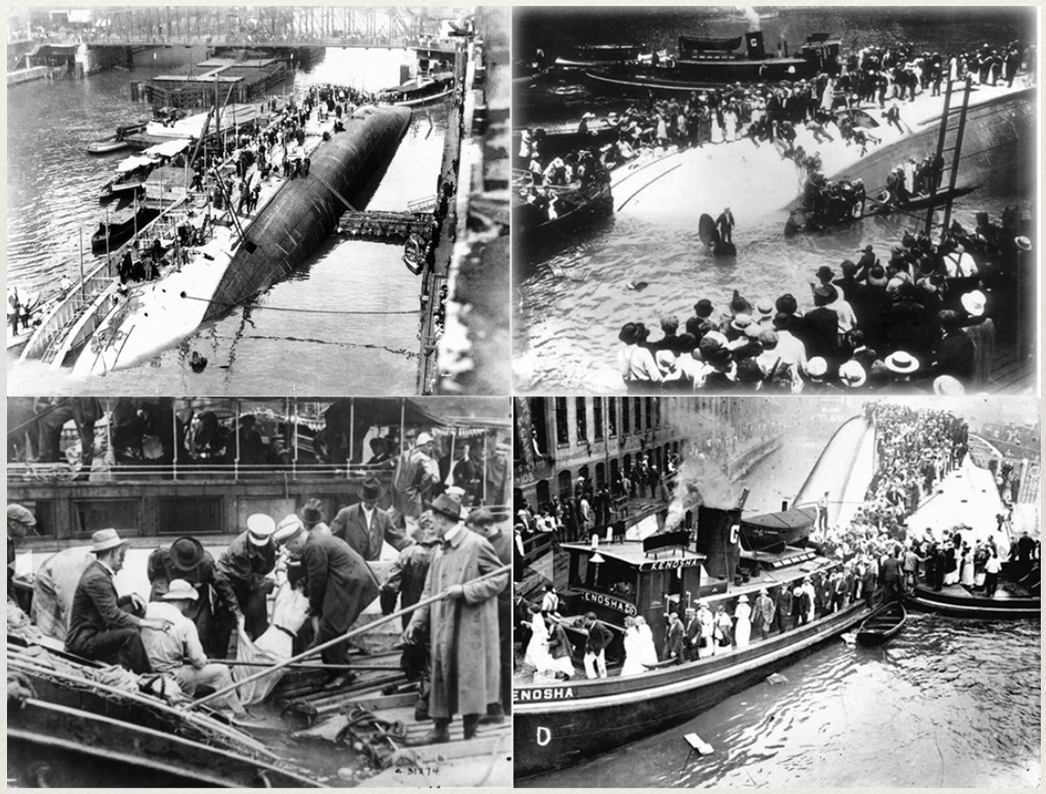
Kockázat priorizálás

- A kockázat:
  - Kockázat=Valószínűség*Következmény_súlyossága
  - várható költség

		Valószínűség				
		Nagyon magas	Magas	Közepes	Alacsony	Nagyon alacsony
Következmény súlyossága	Nagyon magas	Nagyon magas	Nagyon magas	Nagyon magas	Magas	Magas
	Magas	Nagyon magas	Magas	Magas	Közepes	Közepes
	Közepes	Magas	Magas	Közepes	Közepes	Alacsony
	Alacsony	Magas	Közepes	Közepes	Alacsony	Nagyon alacsony
	Nagyon alacsony	Közepes	Alacsony	Alacsony	Nagyon alacsony	Nagyon alacsony

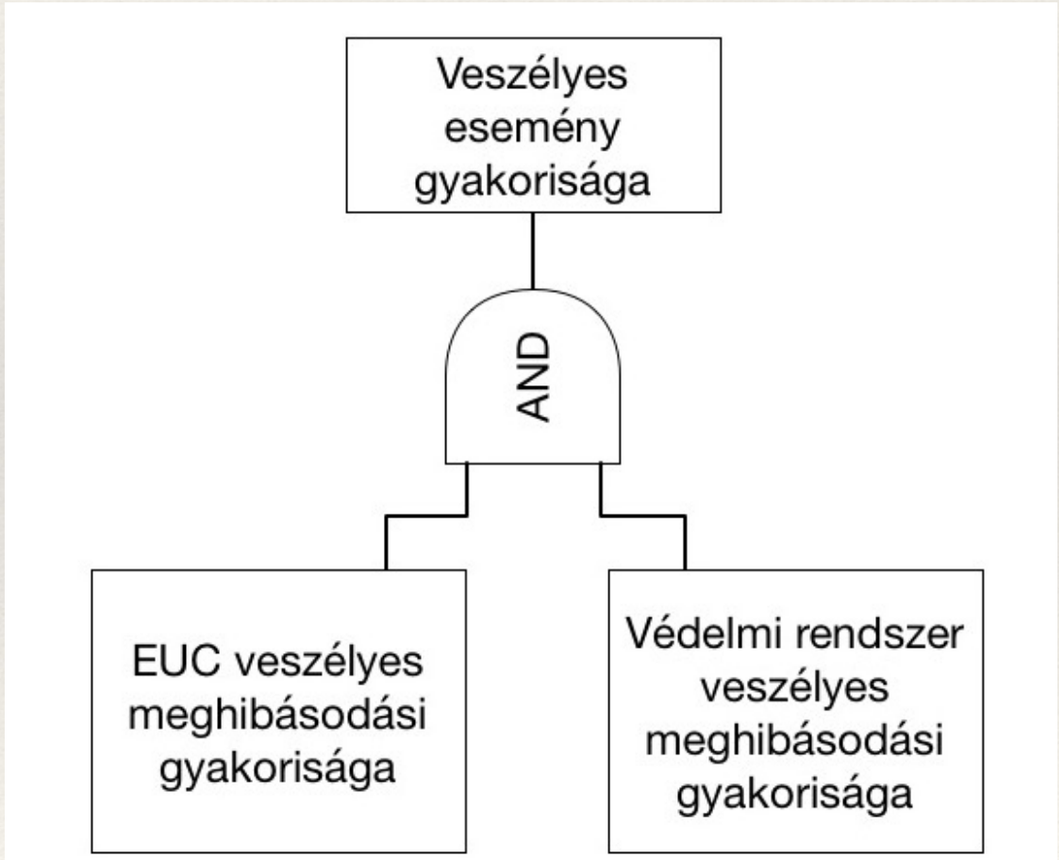
Védelmi rendszer vs. komplexitás

- Eastland jelenség
  - A hozzáadott kiegészítő védelmi rendszer nem szándékoltan (mellékhatás) csökkenti a rendszer biztonságát (overall dependability)
- Eastland katasztrófa
  - 1915.07.24 Chicago River, több mint 800 halott
  - eredeti tervezési hibák (instabil), + 1915 Seamen's act (több mentőcsónak) => még rosszabb stabilitás



Biztonság integritási szintek

- Biztonság integritás: annak a valószínűsége, hogy egy biztonsági rendszer a megfelelő szinten nyújtja biztonsági funkcióit adott feltételek mellett, adott időtartam alatt
- Diszkrét szintek (0,[1-4])
- veszélyes meghibásodások valószínűségének **célszintje**
  - nem biztonságos meghibásodások valószínűsége
  - szakaszos (on-demand) és folytonos működésre
  - EUC + szeparált védelmi rendszer



Biztonság integritási szintek (folyt.)

- Biztonsági követelmények definiálása
  - (1) Funkcionális követelmény
  - (2) Biztonság integritási szintek (**SIL**)
    - Meghibásodási ráták (tolerálható) megadása
    - Kockázat elemzés (kvantitatív, kvalitatív) -> SIL
      - megbízhatósági mérték (~? biztonság)

Biztonság integritási szintek (folyt.)

- Biztonság: folytonos skálán a teljesen biztonságostól a biztos katasztrófáig (kockázat)
- Szoftverek használata: szisztematikus és véletlen hibák aránya eltolódott
- Klasszikusan: megbízhatóság -> biztonság, meghatározás hibaráták alapján
- Szoftver: minden meghibásodás szisztematikus, **nem lehetséges a biztonság meghatározása véletlen meghibásodások elemzésével**
- Szoftverek komplexitása: **lehetetlen a hibák nem-létének bizonyítása**

Szoftver SIL

- Szoftverek hibái szisztematikus hibák => önmagában nem lehet valószínűségi alapon kezelni => biztonság integritási megközelítés
- A rendszer (HW+SW) viselkedésének (folyamatok függetlensége, kritikus adatok védelme, időzítési viselkedés) elemzése
- Szisztematikus hibából fakadó meghibásodások elkerülése -> megfelelő szoftverfejlesztési megoldások

Szoftver SIL (folyt.)

- Nehézségek
  - SIL leképezés alrendszer- és architektúra-követelményekre
  - Fejlesztési eljárások tudják-e garantálni az elvárt meghibásodási rátát
- Szoftverhibák -> meghibásodások
  - nem megfelelő hardver hibakezelés
  - szoftver összeomlások (bármikor)
  - nem biztonságos válasz a bemenetekre
- Bizonyítani kell, hogy az adott fejlesztési módszer biztosítja a veszélyes szoftver meghibásodásokra meghatározott célszámot

Szoftver SIL (folyt.)

- SIL: szabványokban diszkrét szintekre szabályozva (0-4)
- SIL 0: biztonság releváns, de nem éri el a SIL1 szintet
- Eszközökre (tools) vonatkozó szabályozás:
  - T1 osztály
    - Nem generál kimenet, mely közvetlenül befolyásolja a végrehajtható kódot/ futás során használt adatokat.
  - T2 osztály
    - A végrehajtható kód verifikálását támogatja (hiba feltárást befolyásolja).
  - T3 osztály
    - Olyan kimenetet generál mely része a végrehajtható kódnak/használt adatoknak

Hibamódok

- Elhagyás (omission)
- Felesleges (nem várt) tevékenység (commission)
- Korai tevékenység
- Késői tevékenység
- Értékhiba

Hibatűrés

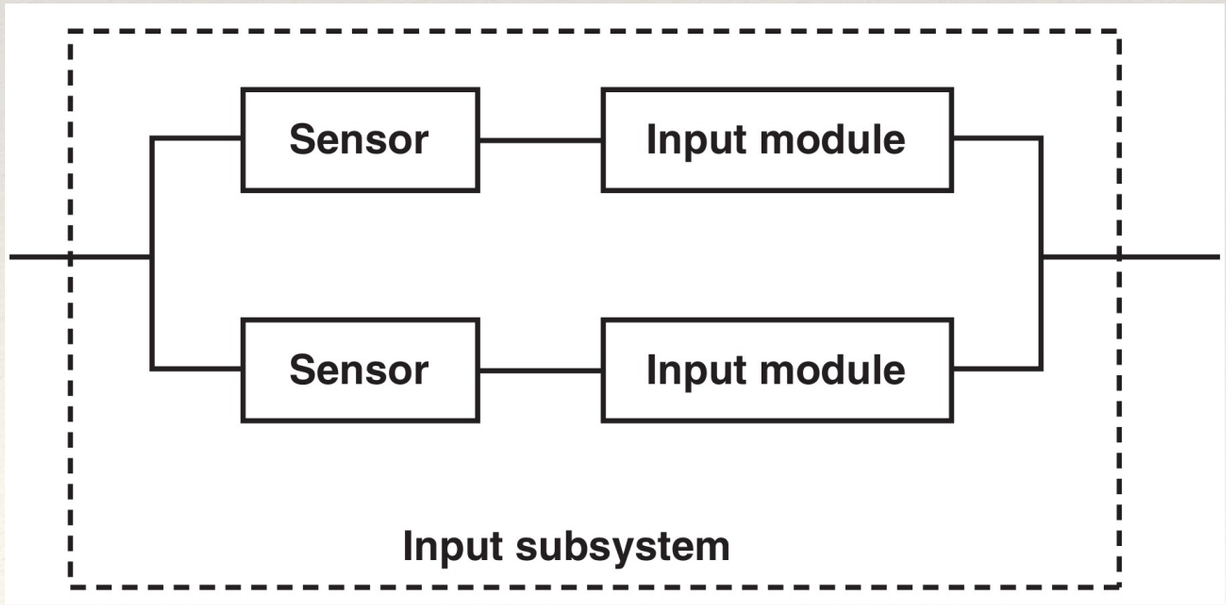
- Egy rendszer azon tulajdonsága, hogy hibák (meghibásodások) esetén is képes helyesen tovább működni
- Csökkentett (biztonságos) működés (fail safe operation)
- Fokozatos leépülés (graceful degradation)
- Robusztusság
  - Nincs egypontos meghibásodás (SPOF)
  - Hiba detektálás
  - Hibák izolációja (domino elv kiküszöbölése)
- Tartalék elemek
  - Replikáció
  - Redundancia
  - Diverzitás

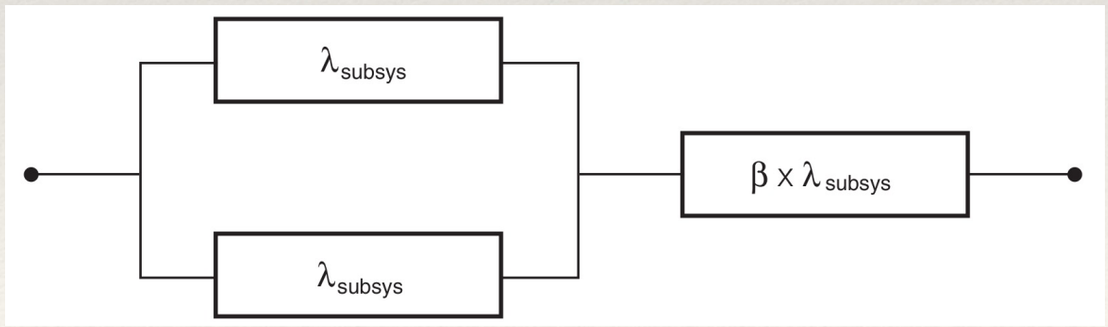
Redundáns struktúrák

- Hardver redundancia
  - párhuzamos duplikálás, tartalék elemes működés, TMR
- Szoftver redundancia
  - szoftver diverzitás
- Információs redundancia
  - "paritás" információ
- Idő redundancia
  - ismételt számítások

MooN rendszerek

- Olyan biztonsági rendszer, melyben N független csatorna van úgy összekötve, hogy a biztonsági funkció ellátásához M csatorna helyes működése elégséges.
- Közös hibaok: a csatornák nem teljes szeparációjából adódó hibalehetőség



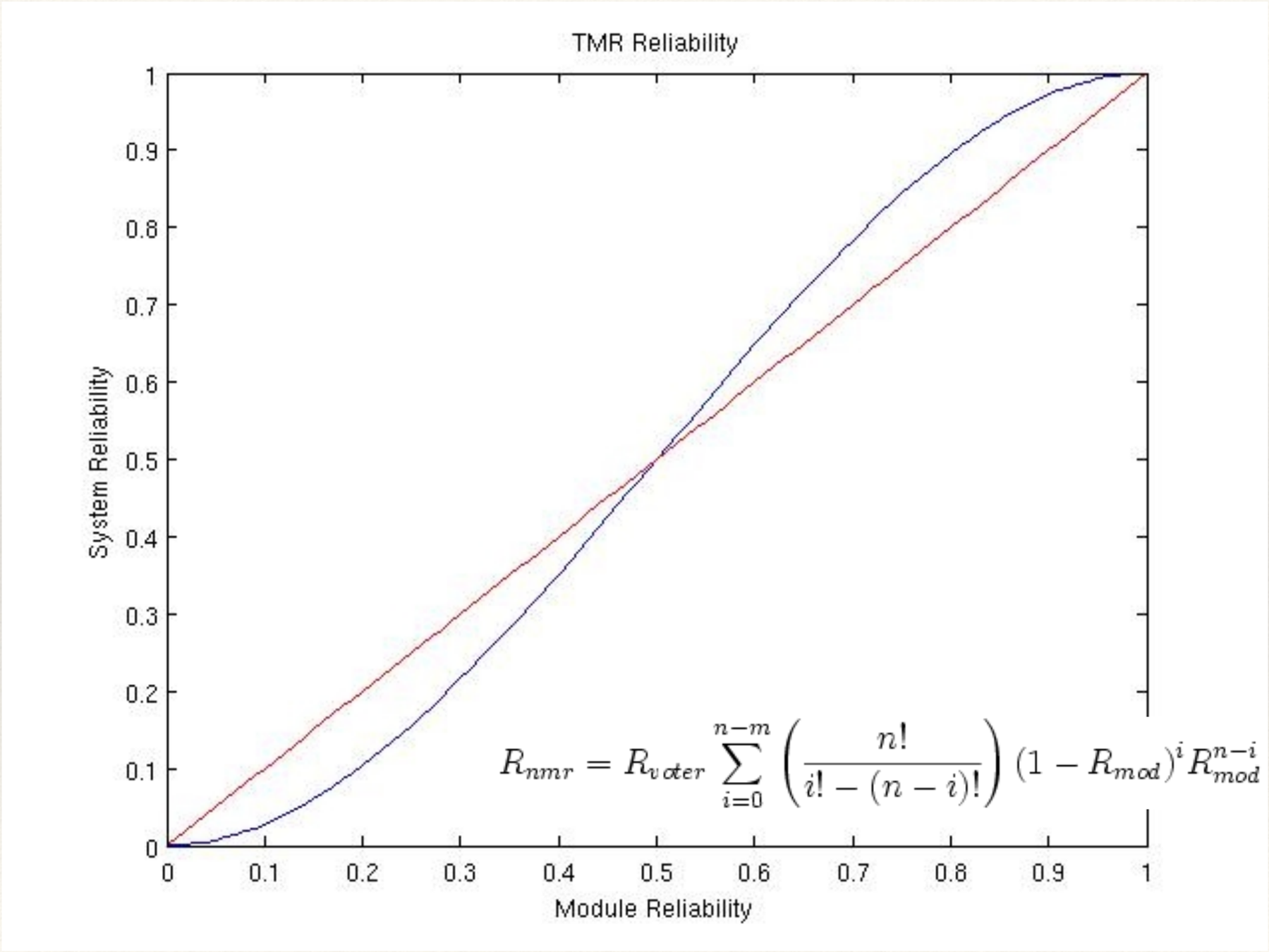


Redundancia és hibatűrés

- Hardver hibatűrés (HFT)
  - hány hibás modult tolerál a rendszer
- (ÉS) szavazás (voting)
  - hány modulnak kell működnie
- Redundancia
  - meghibásodás után hány út (kombináció) marad

Arch.	Voting	HFT	Red.
1oo1	1	0	0
1oo2	1	1	1
2oo2	2	0	0
2oo3	2	1	1
2oo4	2	2	3/1

TMR megbízhatósága

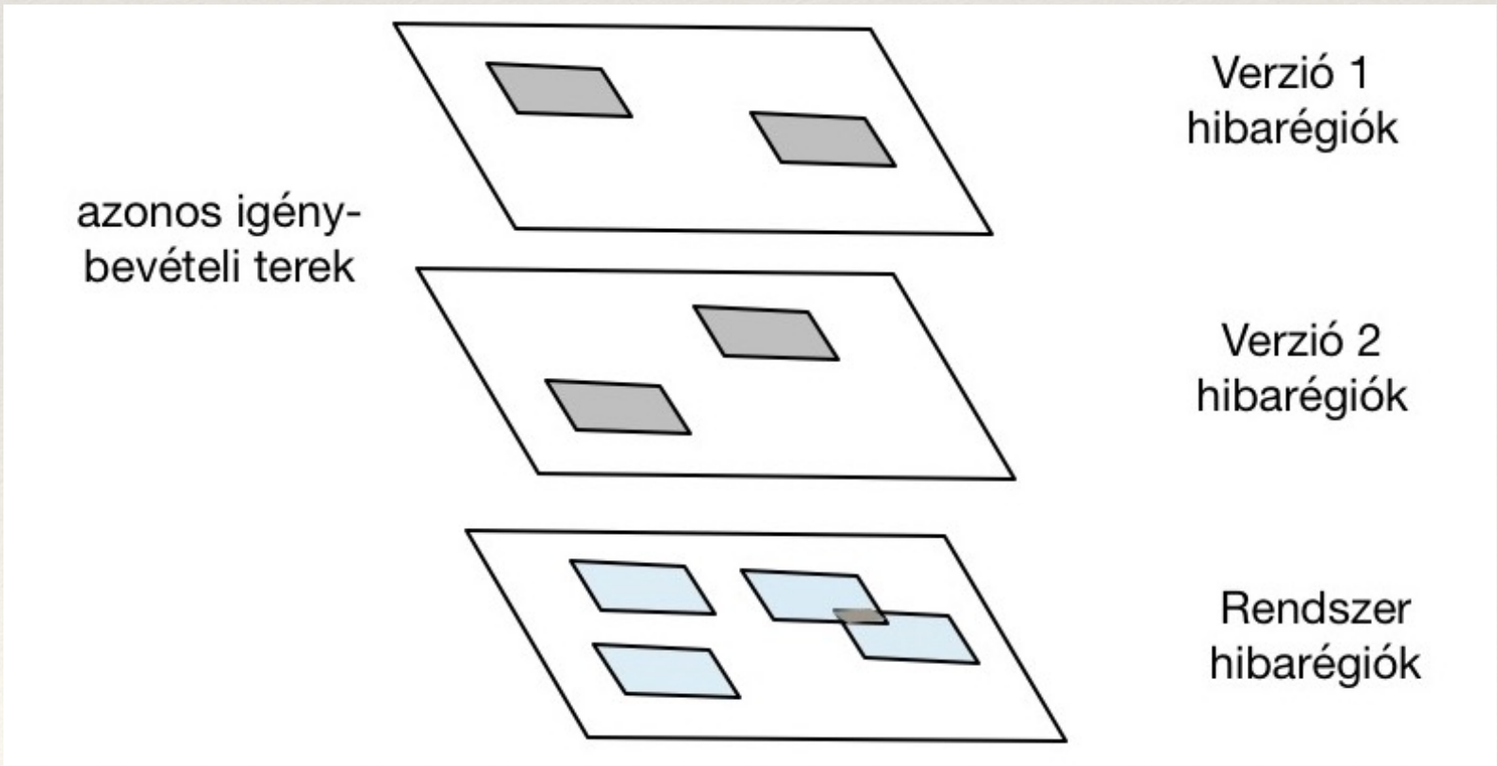


Szoftver diverzitás

- Egyfajta redundancia
  - Funkcionálisan ekvivalens, függetlenül fejlesztett verziók
- Előnyök megbízható szoftverek előállításánál
  - Maradék tervezési szoftverhibák detektálása és maszkolása futás közben
    - Voter komponens: korrekt kimenet az egyes verziók hibái esetén is
  - Véletlen hardver hibák detektálása
  - Szisztematikus hardver hibák detektálása
- Stratégiák: **D**iversity **S**eeking **D**ecision

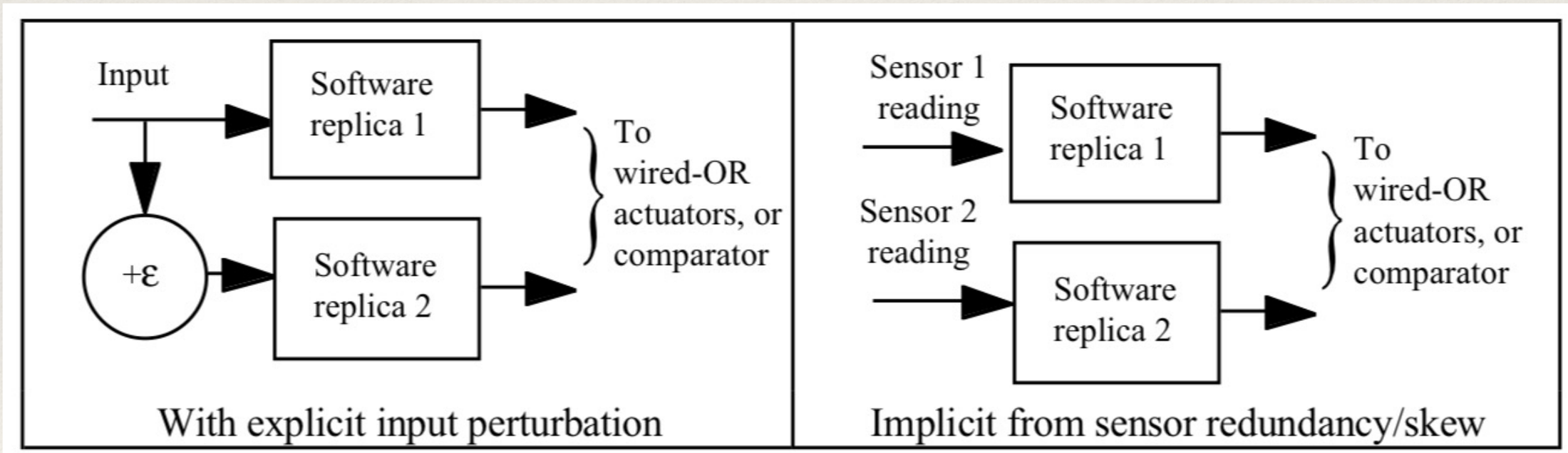
Szoftver diverzitás (folyt.)

- Cél: **hibák esetén eltérő viselkedés** (a verziók között) -> detektálhatóság
- Diverzitás koncepció:
  - számítás igénybevétele (demand)
  - viselkedés ~ hibázási régiók (hibák típusa, helye)



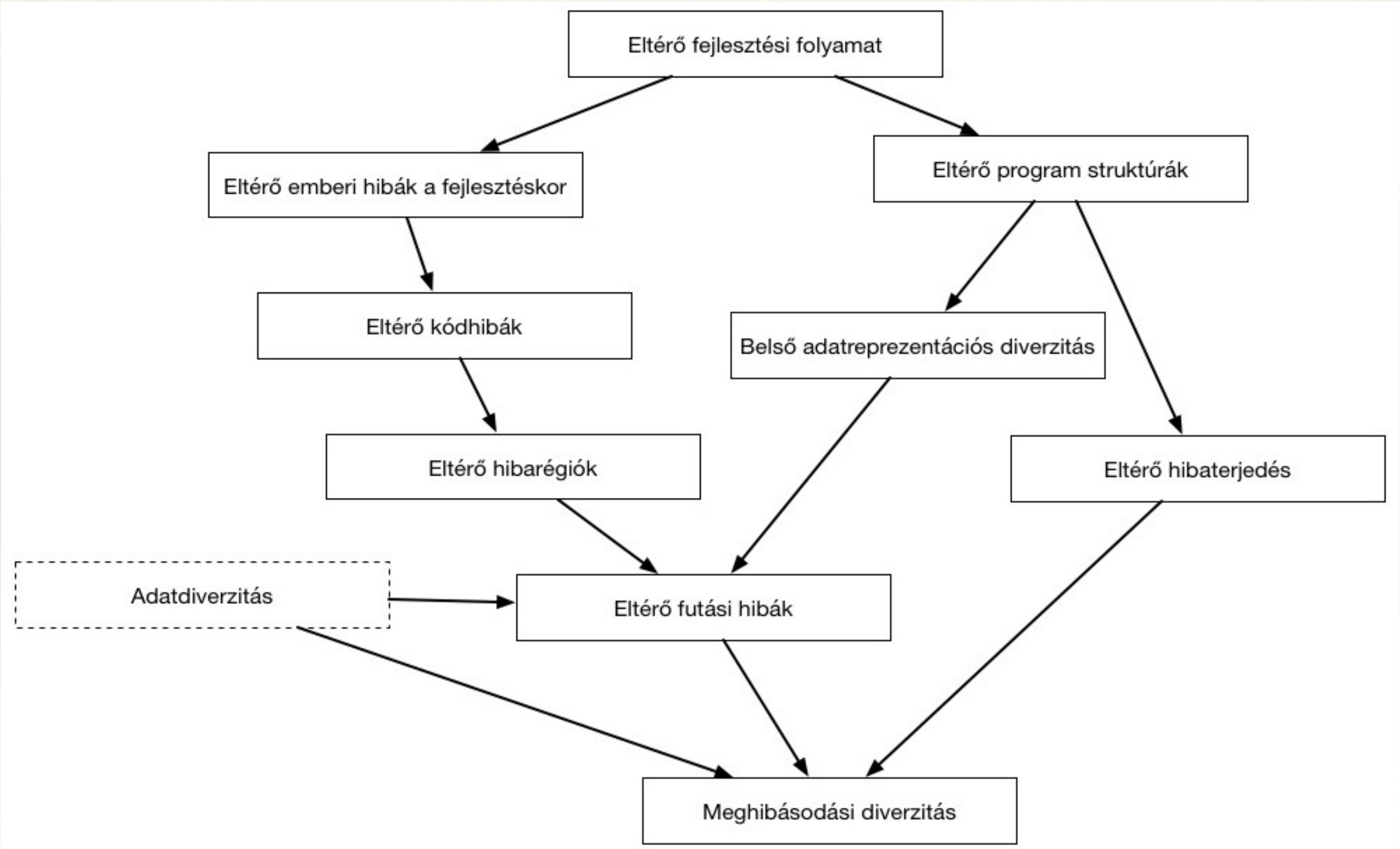
Diverzitási stratégiák (DSDs)

- Adat diverzitás
  - akár azonos szoftver verziók esetén is



Diverzitási stratégiák (DSDs)

- Funkcionális diverzitás (eltérő verziók)



Diverzitási stratégiák (DSDs)

- Emberi hibázások a fejlesztéskor
  - véletlenszerű zavaroknak tekinthetők a fejlesztési tevékenység közben
  - a tevékenység kényszerei befolyásolják
    - az alkalmazási terület nehézségei indokolják (specifikációs problémák>kódolási problémák)
    - megoldandó probléma + fejlesztési módszerek, folyamat, csapat
- Diverzitási döntések
  - bizonyos életciklus tevékenységek során
  - specifikus alrendszerekhez

Diverzitási stratégiák (DSDs)

- Szeparált fejlesztő csapatok
- Eltérő programozási nyelvek
- Eltérő követelmény spec.
- Eltérő fejlesztési módszerek
- Eltérő V&V módszerek
- Eltérő fejlesztő eszközök
- Eltérő futási környezetek
- Eltérő időzítések alkalmazása
- Eltérő hardver
- Eltérő algoritmusok használata
- Eltérő adatreprezentáció

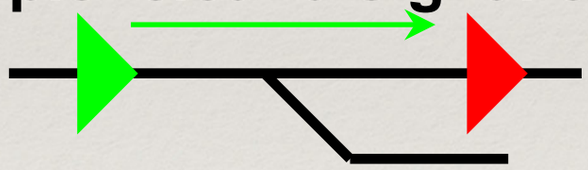
Diverzitási megoldások

- Két csatornás szoftver
  - Biztonsági csatorna (safety bag)
  - Triple Modular Redundancy (2oo3)
  - N-verziós programozás
  - Javító blokkok módszere

SW diverzitás pl. ELEKTRA CC

- **CCA and CCB are not built to do the same.**
- **Any possible action of the CC is classified whether it directs the system to a more safe state (**action-safe**) or a potentially unsafe state (**restraint-safe**).**
  - CCA that has the full functionality to perform any actions.
  - CCB checks all actions of CCA that are defined restraint-safe. If CCB disagrees, the action is not executed.
  - CCB will never execute an restraint-safe action by itself. It always waits for a CCA initiation.
  - CCA and CCB perform any action-safe operations by themselves. No interaction is done here.

Example: clear a signal of a train route (simplified)



CCA

```
IF vp_route_set ANDIF
   vp_switch.locked ANDIF
   NOT vp_track.occupied
    ANDIF
   NOT vp_signal.disturbed
THEN
  p_set_signal_free();
FI;
```

CCB

```
SetSignalFree: RULE;
  (RouteCommandFromCCA)
  (Switch locked)
  NOT (Track occupied)
  NOT (Signal disturbed)
  ==>
  p_clear_signal();
END SetSignalFree;
```