

# A szoftver ellenőrzés szerepe

## Alapfogalmak

Majzik István  
majzik@mit.bme.hu

<http://www.inf.mit.bme.hu/>

### Tartalomjegyzék

- Motiváció
  - Milyen minőségi igények vannak a szoftverrel szemben, és mit tud ma a szoftveripar?
  - Miért olyan nagy a szoftver ellenőrzési technikák jelentősége?
- A verifikáció és validáció technikái (áttekintés)
  - Milyen tipikus technikák vannak?
- Fejlesztési életciklus modellek
  - Milyen szerepet kapnak a tipikus technikák az egyes fejlesztési folyamatokban?
- Fejlesztési szabványok szerepe
  - Hogyan valósul meg a szisztematikus ellenőrzés?

## Elvárások: Szoftverek és rendszerek hibamentessége

- Szolgáltatási szint szerződések (SLA)
  - Telekom: „Öt kilences”: 99,999% (5 perc/év kiesés)
- Biztonságkritikus rendszerek:
  - Szabvány előírások a hibák gyakoriságára
  - Biztonságintegritási szintek (SIL) szerint

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	$10^{-9} \leq \text{THR} < 10^{-8}$

Ha 15 év az élettartam, akkor ez alatt kb. 750 berendezésből 1-ben lesz hiba

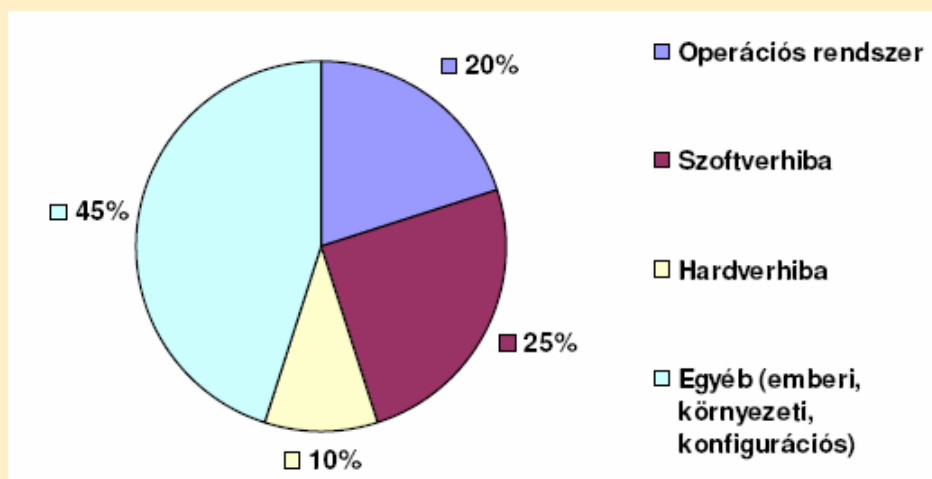
Hiba nélküli működés ~ 11.000 év?

## Kliens-szerver rendszerek meghibásodása

Az IEEE Computer felmérése kliens-szerver rendszerekben:

- Hardver hiba: 10%
- Szoftver hiba: 40% (szerver 30%, kliens 5%, hálózati 5%)
- Emberi hiba: 15%
- Környezeti hatás: 5%
- Tervezett leállítás: 30%

Egy másik elemzés:



# Szoftverek minőségi problémái

## Jelentések több szektorból (Forrester):

- „Defibtech issues a worldwide recall of two of its defibrillator products due to **faulty self-test software** that may clear a previously detected low battery condition. The issue affects approximately 42,000 units worldwide. (February 2007)”
- „Cricket Communications recalls about 285,000 of its cell phones due to a **software glitch** that causes audio problems when a caller connects to an emergency 911 call. (May 2008)”
- „Toyota recalls 160,000 cars with hybrid engines due to a failure of its engine control software. (October 2005)”
- „Nissan recalls 16,365 Murano and Infiniti EX 35 vehicles in February 2008 due to a software problem causing airbag failure.”
- „In May 2008, Chrysler recalls about 25,000 Jeep Commanders to repair transmission control software that allows the engine to stall at high speeds and about 50,600 vehicles in February 2007 to reprogram antilock brake software.”

## Nemzetközi statisztikák szoftver projektekre

- Tipikus kódméret:
  - 10 kLOC ... 1000 kLOC
- Fejlesztési idő:
  - 0,1 - 0,5 mérnökév / kLOC (nagy méretű szoftver)
  - 5-10 mérnökév / kLOC (kritikus szoftver)
- Hiba eltávolítás (ellenőrzés, tesztelés, javítás):
  - 45 - 75% ráfordítás
- Hibasűrűség változása:
  - 10 - 200 hiba / kLOC jön létre a fejlesztés során



Ellenőrzési technikák

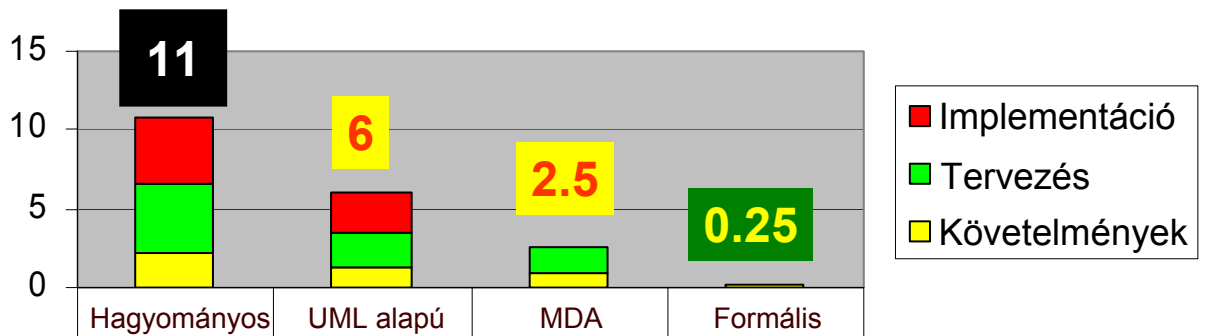
- 0,01 - 10 hiba / kLOC maradhat az üzembe helyezésig

# Egy magyarországi felmérés

- Hibák száma 1 kLOC-ra:

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

Hibák száma (hiba/ezer kódsor)



	Hagyományos	UML alapú	MDA	Formális
Implementáció	4,2	2,55	0	0
Tervezés	4,4	2,2	1,6	0,1
Követelmény	2,2	1,3	1	0,1

## Hibák az alkalmazás életről

### Fejlesztési folyamat

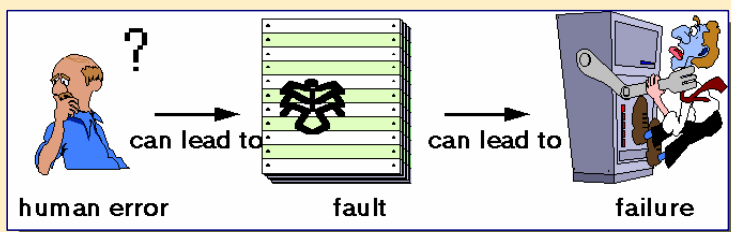
- Konceptcionális hibák
- Tervezési hibák
- Implementációs hibák

**Verifikáció és validáció a tervezés során**

### Működő termék

- Hardver hibák
- Konfigurációs hibák
- Kezelői hibák

**Hibatúrás működés közben (ld. korábban)**



## Verifikáció és validáció

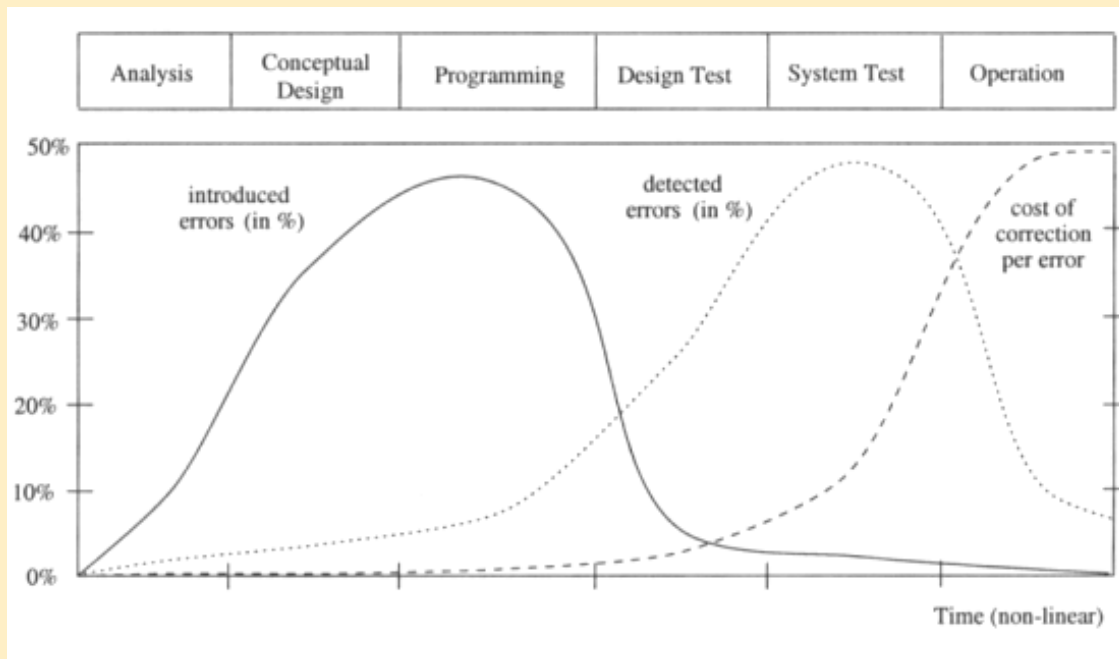
Verifikáció (igazolás)	Validáció (érvényesítés)
„Jól tervezem-e a rendszert?”	„Jó rendszert készítettem-e?”
Összhang ellenőrzése a fejlesztési fázisokban, illetve ezek között	A fejlesztés eredményének ellenőrzése
Fejlesztési lépések során használt tervek (modellek) és specifikációjuk közötti megfelelés ellenőrzése	A kész rendszer és a felhasználói elvárások közötti megfelelés ellenőrzése
Objektív folyamat; formalizálható, automatizálható	Szubjektív elvárások lehetnek; elfogadhatósági ellenőrzés
Hibamodell: Tervezési, implementációs hibák	Hibamodell: Követelmények hiányosságai is
Nincs rá szükség, ha automatikus a leképzés követelmény és implementáció között	Nincs rá szükség, ha a specifikáció tökéletes (elég egyszerű)

## Szoftver és hardver jellegzetességek

Hardver	Szoftver
Több technológiai tapasztalat és jobban automatizált tervezés	Egyedi fejlesztések, kisebb fokú automatizáltság
Sokszor használt alkatrészek	Kevesebb (C)OTS komponens, eltérő alkalmazási feltételek
Hosszabb távra tervezés (költséges javítás és csere)	Gyors életciklus, piacra kerülés (felhasználó tesztel...)
Gyártási minőségbiztosítás, automatizált ellenőrzés	Kézi ellenőrzés, kis lépések az automatikus ellenőrzés felé

Tendencia: A hardver „közeledik” a szoftver felé  
(ld. programozható hardverek, FPGA)

# Költségvonzat

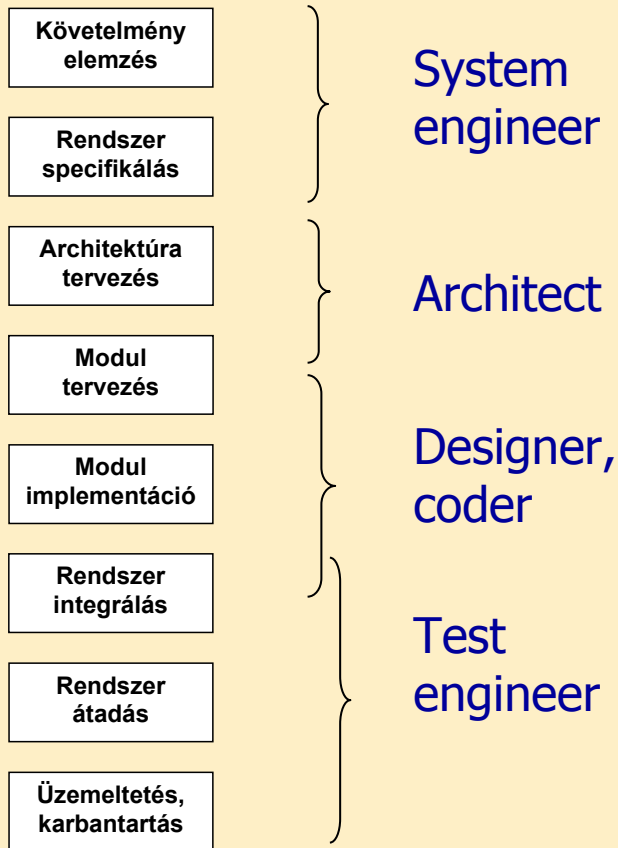


- **Korai verifikáció csökkenti a költségeket**
  - Validációs tesztelésnél korábban detektálni kellene a hibákat ...

## Tartalomjegyzék

- **Motiváció**
  - Milyen minőségi igények vannak a szoftverrel szemben, és mit tud ma a szoftveripar?
  - Miért olyan nagy a szoftver ellenőrzési technikák jelentősége?
- **A verifikáció és validáció technikái (áttekintés)**
  - Milyen tipikus technikák vannak?
- **Fejlesztési életciklus modellek**
  - Milyen szerepet kapnak a tipikus technikák az egyes fejlesztési folyamatokban?
- **Fejlesztési szabványok szerepe**
  - Hogyan valósul meg a szisztematikus ellenőrzés?

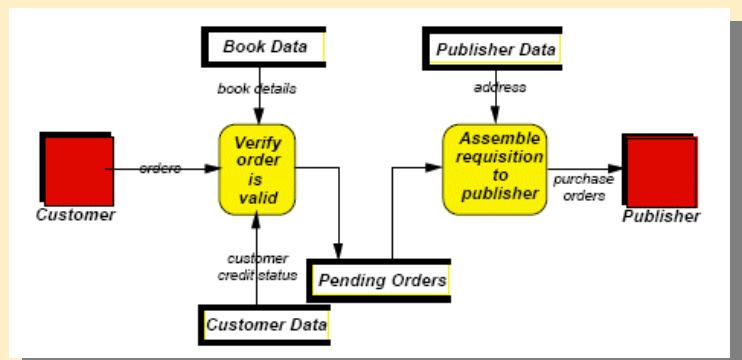
## Fejlesztési folyamatok tipikus lépései



Ütemezés, sorrendezés az életciklus modelltől függ!

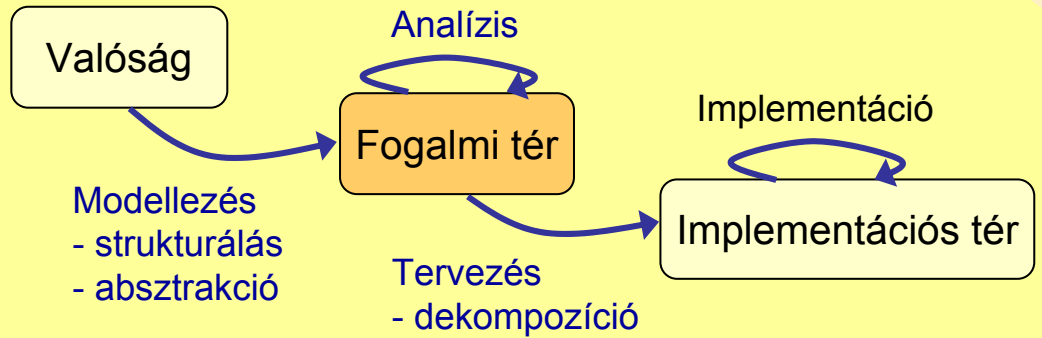
## A tipikus ellenőrzési technikák

	Feladat	V&V szempont	V&V technika
Követelmény elemzés	Funkciók, aktorok, használati esetek áttekintése	- Kockázatok - Kritikusság	- Ellenőrző listák - FMEA, FT, ET, ...
Rendszer specifikálás			
Architektúra tervezés			
Modul tervezés			
Modul implementáció			
Rendszer integrálás			
Rendszer átadás			
Üzemeltetés, karbantartás			



## A tipikus ellenőrzési technikák

- Követelmény elemzés
- Rendszer specifikálás**
- Architektúra tervezés
- Modul tervezés
- Modul implementáció
- Rendszer integrálás
- Rendszer átadás
- Üzemeltetés, karbantartás



Feladat	V&V szempont	V&V technika
Funkcionális és nem-funkcionális követelmények rögzítése	<ul style="list-style-type: none"> <li>- Teljesség</li> <li>- Ellentmondás-mentesség</li> <li>- Ellenőrizhetőség</li> <li>- Megvalósíthatóság</li> </ul>	<ul style="list-style-type: none"> <li>- Statikus analízis (kézi vagy automatikus átvizsgálás)</li> <li>- Szimuláció</li> </ul>

## A tipikus ellenőrzési technikák

- Követelmény elemzés
- Rendszer specifikálás**
- Architektúra tervezés
- Modul tervezés
- Modul implementáció
- Rendszer integrálás
- Rendszer átadás
- Üzemeltetés, karbantartás

Egy beléptető rendszer specifikációja (Event-B):

személyek:  $prs \neq 0$  (halmaz)  
 épületek:  $bld \neq 0$  (halmaz)  
 jogosultság:  $aut \in prs \leftrightarrow bld$  (bináris reláció)  
 tartózkodás:  $sit \in prs \rightarrow bld$  (teljes fv.)  
 invariáns:  $sit \subseteq aut$

Egy funkció (lehetséges történet):

```
pass = ANY p,b WHERE (p,b) ∈ aut ∧ sit(p) ≠ b
      THEN sit(p) := b END
```

Feladat	V&V szempont	V&V technika
Funkcionális és nem-funkcionális követelmények rögzítése	<ul style="list-style-type: none"> <li>- Teljesség</li> <li>- Ellentmondás-mentesség</li> <li>- Ellenőrizhetőség</li> <li>- Megvalósíthatóság</li> </ul>	<ul style="list-style-type: none"> <li>- Statikus analízis (kézi vagy automatikus átvizsgálás)</li> <li>- Szimuláció</li> </ul>



## A tipikus ellenőrzési technikák

Felülvizsgálat:

1. Ellenőrző lista összeállítása
2. Bemutató készítés a fejlesztő által
3. Kérdések összeállítása a szakértők részéről, fejlesztő válaszol
4. Megbeszélés, majd jelentés készítés

Egyenrangú átvizsgálás (peer review) típusok:

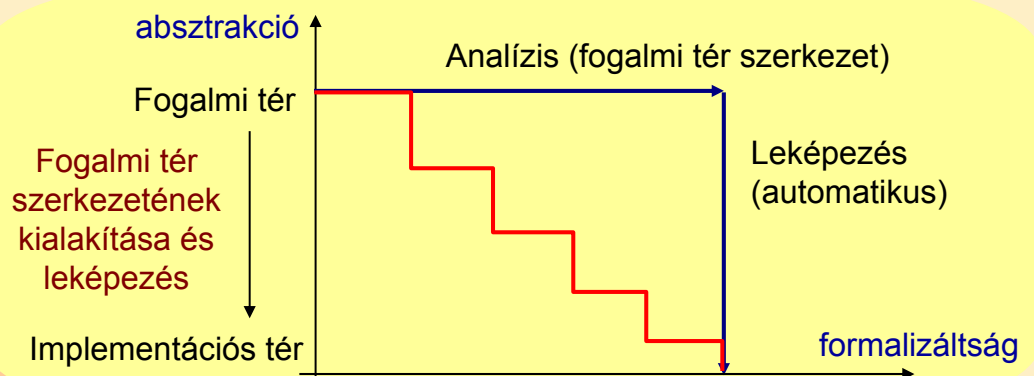
- Körbenforgó (round-robin) modulonként más-más vezetővel
- Bejárás (walkthrough): fejlesztő „vezeti” az ellenőrzőket
- Szemle (inspection): ellenőrző lista alapján

Követelmény elemzés
<b>Rendszer specifikálás</b>
Architektúra tervezés
Modul tervezés
Modul implementáció
Rendszer integrálás
Rendszer átadás
Üzemeltetés, karbantartás

Feladat	V&V szempont	technika
Funkcionális és nem-funkcionális követelmények rögzítése	<ul style="list-style-type: none"> <li>- Teljesség</li> <li>- Ellentmondás-mentesség</li> <li>- Ellenőrizhetőség</li> <li>- Megvalósíthatóság</li> </ul>	<ul style="list-style-type: none"> <li>- Statikus analízis (kézi vagy automatikus átvizsgálás)</li> <li>- Szimuláció</li> </ul>

## A tipikus ellenőrzési technikák

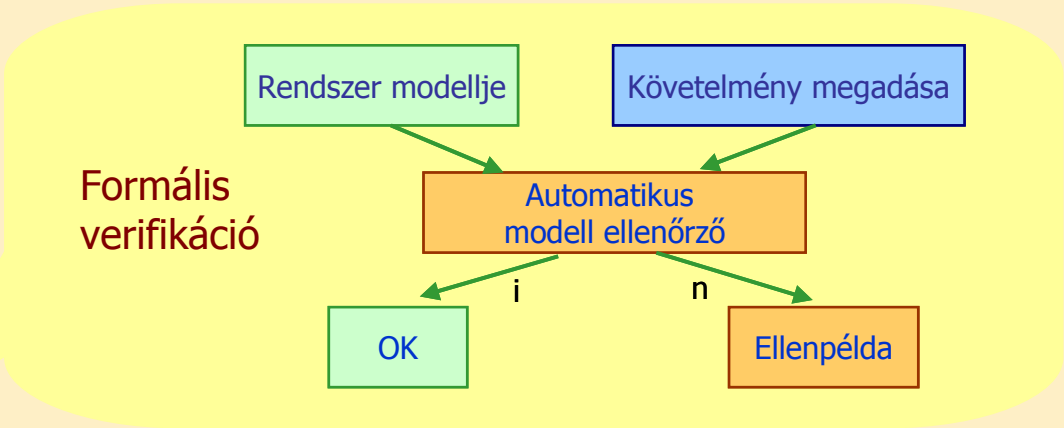
Követelmény elemzés
Rendszer specifikálás
<b>Architektúra tervezés</b>
Modul tervezés
Modul implementáció
Rendszer integrálás
Rendszer átadás
Üzemeltetés, karbantartás



Feladat	V&V szempont	V&V technika
<ul style="list-style-type: none"> <li>- Specifikáció modul szintű bontása</li> <li>- Hardver-szoftver együttes tervezés</li> <li>- Kommunikáció tervezés</li> </ul>	<ul style="list-style-type: none"> <li>- Funkciók fedése</li> <li>- Interfész illeszkedés</li> <li>- Kockázat elemzés</li> <li>- Ütemezés</li> </ul>	<ul style="list-style-type: none"> <li>- Statikus analízis</li> <li>- Szimuláció</li> <li>- Teljesítmény, megbízhatóság, biztonság analízise</li> </ul>

# A tipikus ellenőrzési technikák

- Követelmény elemzés
- Rendszer specifikálás
- Architektúra tervezés
- Modul tervezés**
- Modul implementáció
- Rendszer integrálás
- Rendszer átadás
- Üzemeltetés, karbantartás

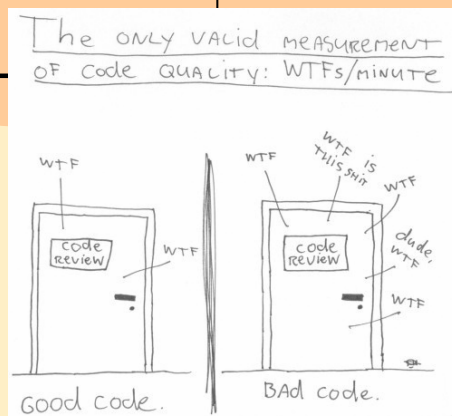


Feladat	V&V szempont	V&V technika
Részletes belső működés tervezése (algoritmusok, adatstruktúrák)	- Kritikus belső algoritmusok, protokollok helyessége	- Statikus analízis - Szimuláció - Formális verifikáció - Gyors prototípus

# A tipikus ellenőrzési technikák

- Követelmény elemzés
- Rendszer specifikálás
- Architektúra tervezés
- Modul tervezés
- Modul implementáció**
- Rendszer integrálás
- Rendszer átadás
- Üzemeltetés, karbantartás

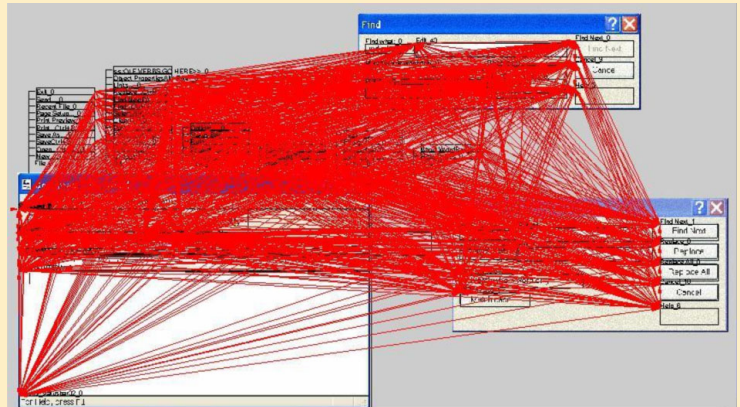
Feladat	V&V szempont	V&V technika
Szoftver implementáció	- Biztonságos - Ellenőrizhető - Karbantartható kód	- Kódolási előírások (szabványok) ellenőrzése: kód átvizsgálás
Modul működés igazolása	- Modul terveknek való megfelelés	- Statikus analízis - Tesztelés - Regressziós tesztelés



## A tipikus ellenőrzési technikák

- Követelmény elemzés
- Rendszer specifikálás
- Architektúra tervezés
- Modul tervezés
- Modul implementáció
- Rendszer integrálás**
- Rendszer átadás
- Üzemeltetés, karbantartás

Feladat	V&V szempont	V&V technika
Modulok illesztése, hardver-szoftver összeállítás	- Együttes működés megfelelősége	- Integrációs tesztelés (tipikusan inkrementális)

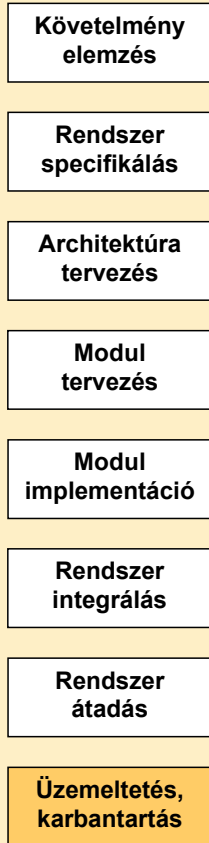


## A tipikus ellenőrzési technikák

- Követelmény elemzés
- Rendszer specifikálás
- Architektúra tervezés
- Modul tervezés
- Modul implementáció
- Rendszer integrálás
- Rendszer átadás**
- Üzemeltetés, karbantartás

Feladat	V&V szempont	V&V technika
Specifikáció teljesítésének bemutatása	- Rendszer-specifikációnak való megfelelés	- Rendszertesztelés - Mérések, monitorozás
Elvárások teljesítése	- Követelményeknek és elvárásoknak való megfelelés	- Validációs tesztelés - Próbaüzem alatti ellenőrzés

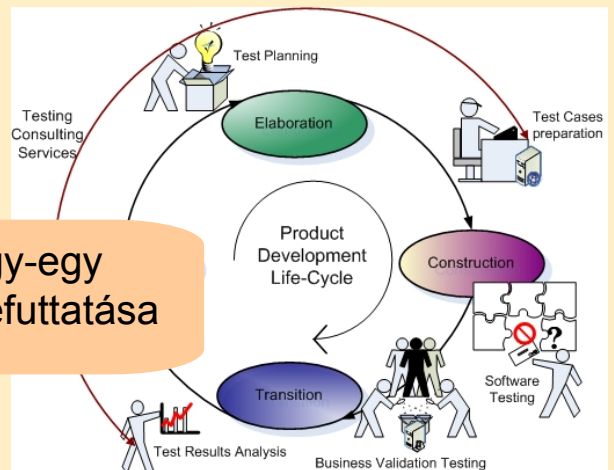
# A tipikus ellenőrzési technikák



Teendők az üzemeltetés és karbantartás során:

- Hibanaplózás és hibaanalízis (hiba előrejelzés)
- Módosítások verifikációja és validációja

Módosításokra egy-egy „mini életciklus” lefuttatása



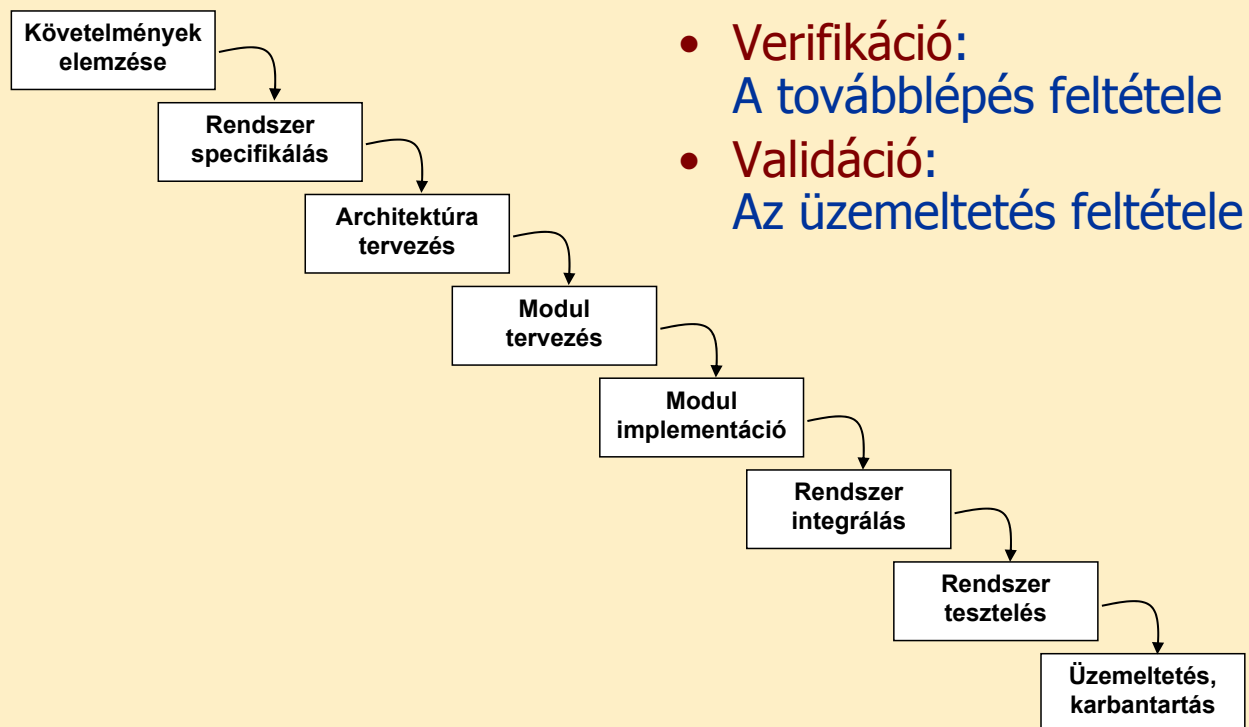
## Tartalomjegyzék

- Motiváció
  - Milyen minőségi igények vannak a szoftverrel szemben, és mit tud ma a szoftveripar?
  - Miért olyan nagy a szoftver ellenőrzési technikák jelentősége?
- A verifikáció és validáció technikái (áttekintés)
  - Milyen tipikus technikák vannak?
- Fejlesztési életciklus modellek
  - Milyen szerepet kapnak a tipikus technikák az egyes fejlesztési folyamatokban?
- Fejlesztési szabványok szerepe
  - Hogyan valósul meg a szisztematikus ellenőrzés?

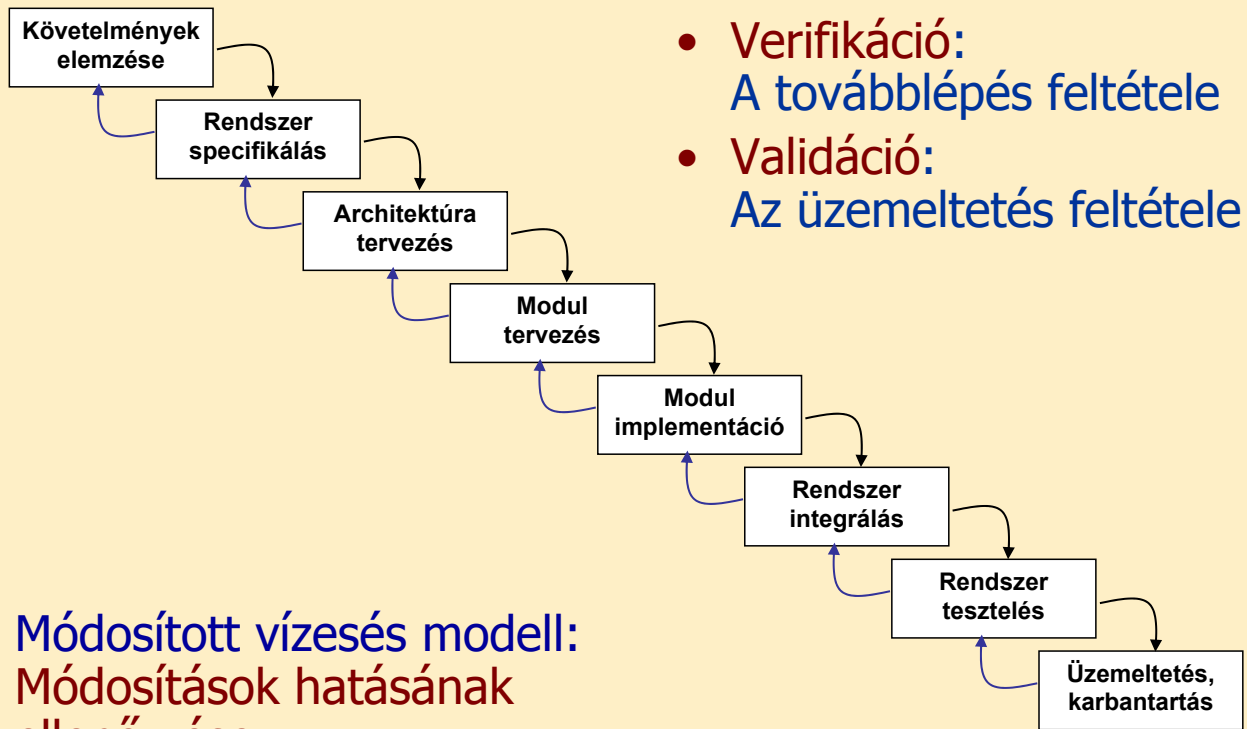
# Szoftverfejlesztési (életciklus) modellek

- Miért van szükség életciklus modellekre?
  - Komplexitás kezelése
    - Fázisokra osztás, mérföldkövek rögzítése
    - Elosztott fejlesztés és integráció alapja
  - Változások kezelése
    - Követelmény módosulás, hibajavítás hatásainak kezelése
    - Új eszközök és technológiák bevezetése
- Generikus szoftverfejlesztési folyamat modellek:
  - Szekvenciális fejlesztés: Vízésés és V-modell
  - Evolúciós fejlesztés: Gyors alkalmazásfejlesztés
  - Iteratív fejlesztés: Spirál modell
  - Modell alapú (formális) fejlesztés: 4G fejlesztési modell
  - Iteratív-inkrementális fejlesztés: Unified Process

## 1. Vízésés modell



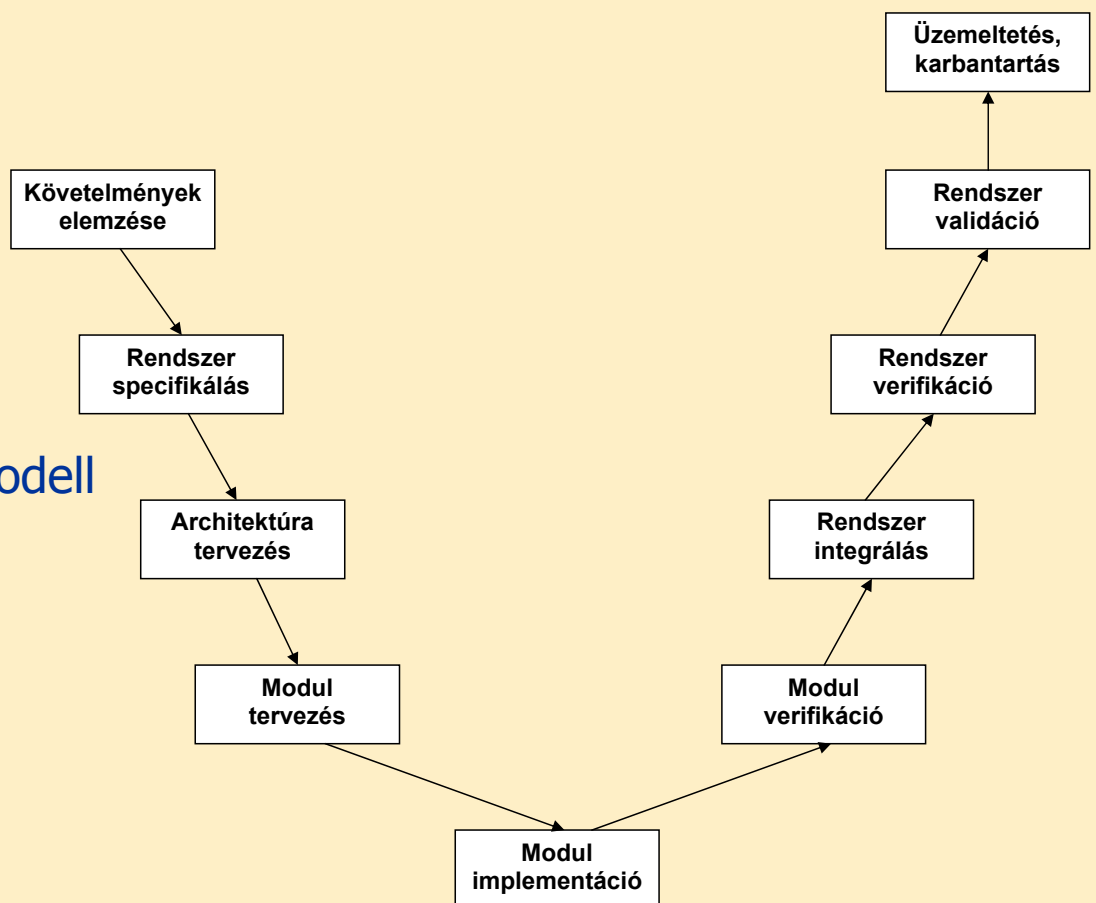
# 1. Vízesés modell



- **Verifikáció:**  
A továbblépés feltétele
- **Validáció:**  
Az üzemeltetés feltétele

- **Módosított vízesés modell:**  
Módosítások hatásának ellenőrzése  
(pl. regressziós tesztelés)

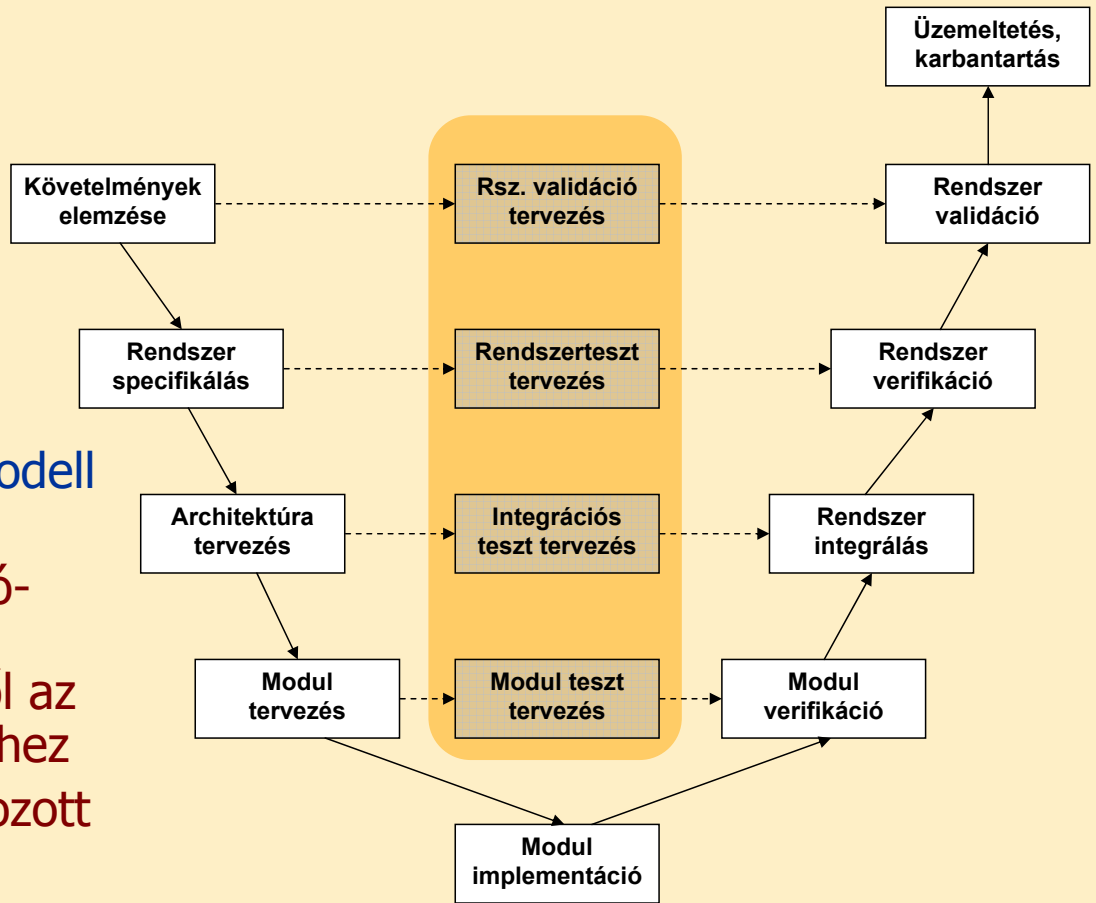
# 2. V-modell



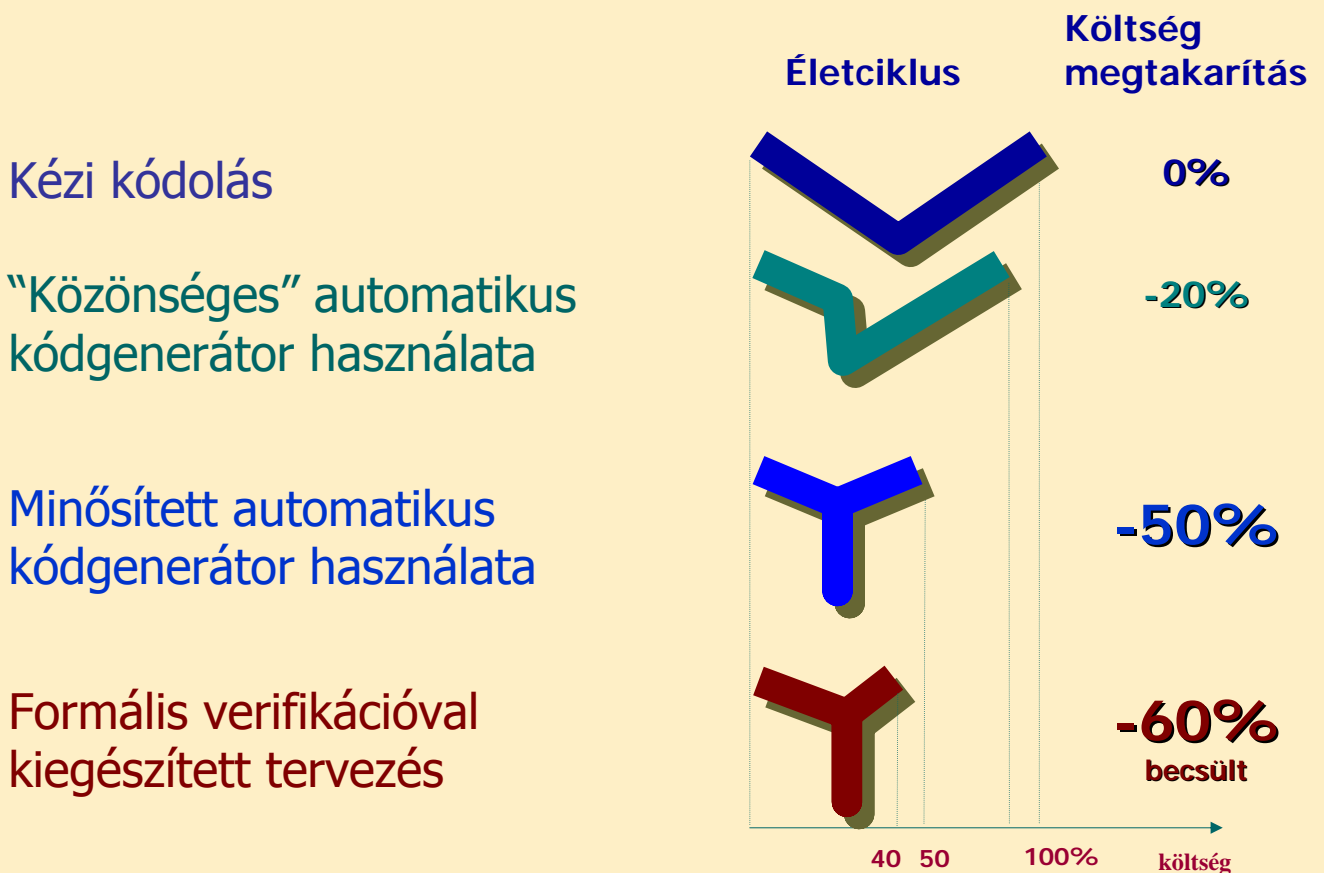
- **Vízesés modell alapú**

## 2. V-modell

- Vízesés modell alapú
- Információ-áramlás a tervezéstől az ellenőrzéshez
- Meghatározott V&V



## Modell alapú fejlesztés: V-től az Y modellig



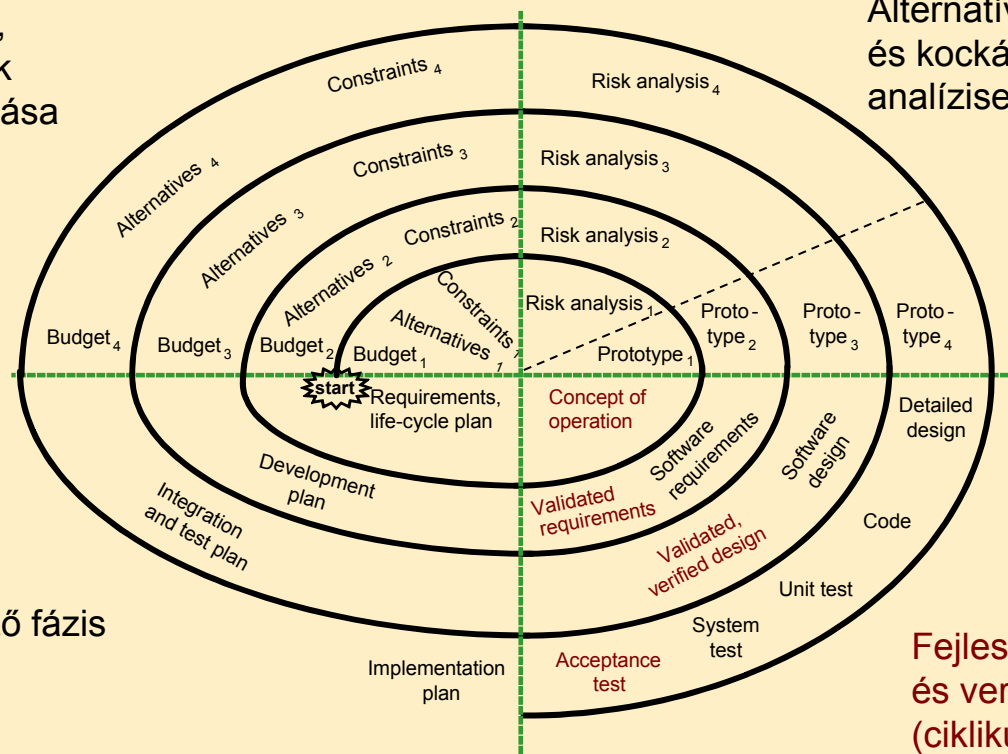
### 3. Evolúciós alkalmazásfejlesztés (RAD)

- Kezdeti implementáció gyors fejlesztése, majd **több verzió** keresztül finomítás
  - Feltáró fejlesztés: Felhasználóval egyeztetve
    - Ismert követelmények alapján kiindulva az első verzió
  - Gyors prototípus készítése (kritikus funkciókra)
    - Validációs fázisig, prototípus bemutatás, átdolgozás
  - Hiányosan specifikált rendszerek esetén is alkalmas
- V&V jellegzetességek:
  - Prototípus tesztelés jelentősége nagy
  - Integrációs ellenőrzések (tesztek) szerepe nő
    - Újabb funkciók tesztelése
  - Regressziós tesztelés módosítások után

### 4. Spirál modell

Célok, alternatívák, korlátozások meghatározása

Alternatívák és kockázatok analízise



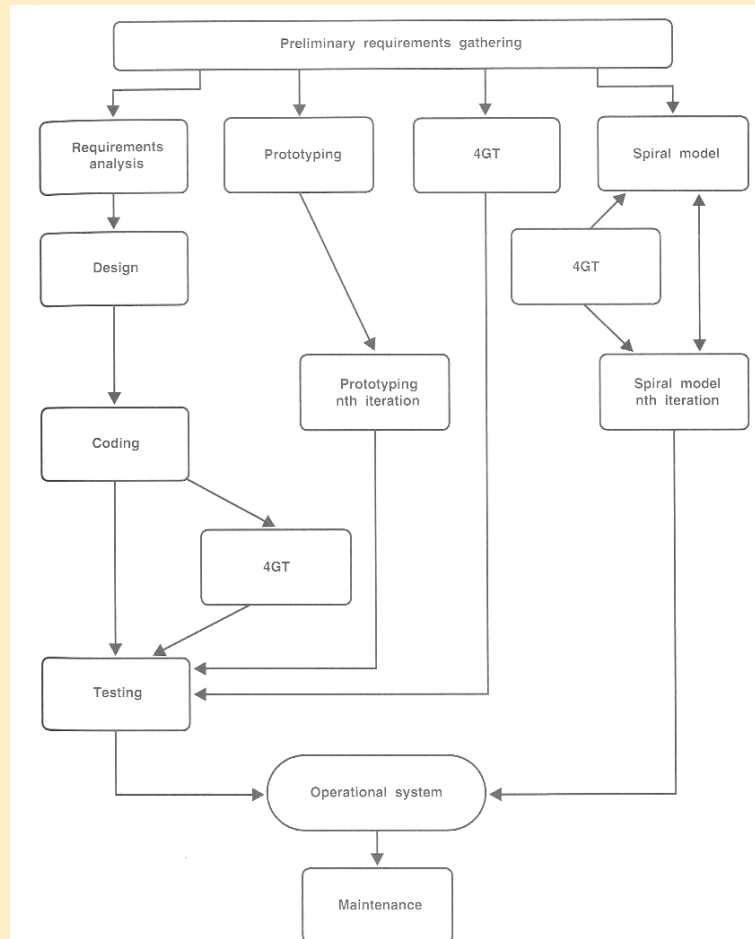
A következő fázis tervezése

Fejlesztés és verifikáció (ciklikusan)



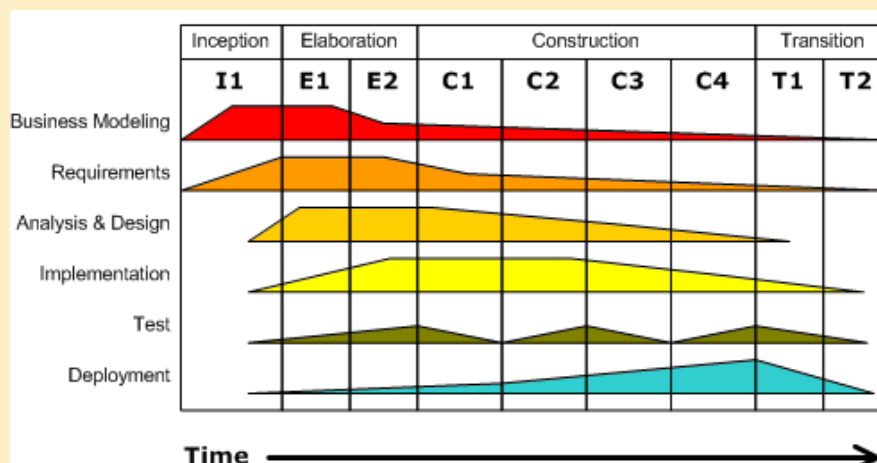
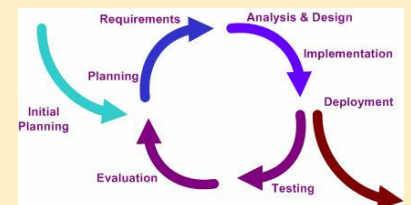
## 5. „Negyedik generációs” modell

- Integrálás:
  - Jól megfogható követelmények: Hagyományos fejlesztés
  - Rosszul specifikált követelmények: Gyors prototípus fejlesztés
  - Formalizálható követelmények: Modell alapú fejlesztés (CASE eszközök), helyességmegőrző
  - Iteratív is lehet
- Modell alapú verifikáció



## 6. Unified Process

- Inkrementális és iteratív
  - Fázisok (időben kötött) iterációkra osztva
  - Mindegyik iteráció egy teljes (mini) fejlesztési ciklus
  - Az ellenőrzés fókuszusa az egyes fázisokban eltérő
    - Integrációs és regressziós tesztelés jelentős szerepet kap



# Agilis szoftverfejlesztés

- Extreme Programming
  - Rövid iterációk, működő kódra koncentrálni, rendszeres (napi) integrációval és szoros státusz követéssel (fejlesztők, megrendelők)
  - „Test first programming”:
    - Funkcionális tesztek „story card” alapon
    - Minden változás (új funkció) esetén tesztelés
- Test Driven Development
  - Inkrementális, lépések egy-egy új funkcióhoz:
    1. Teszt írása az új funkcióhoz (futtatása sikertelen)
    2. Kódolás (a teszt sikeres futtatásához)
    3. Kód átdolgozása, tisztítása (refactoring) újratestteléssel
  - Automatikus unit tesztelésre épít

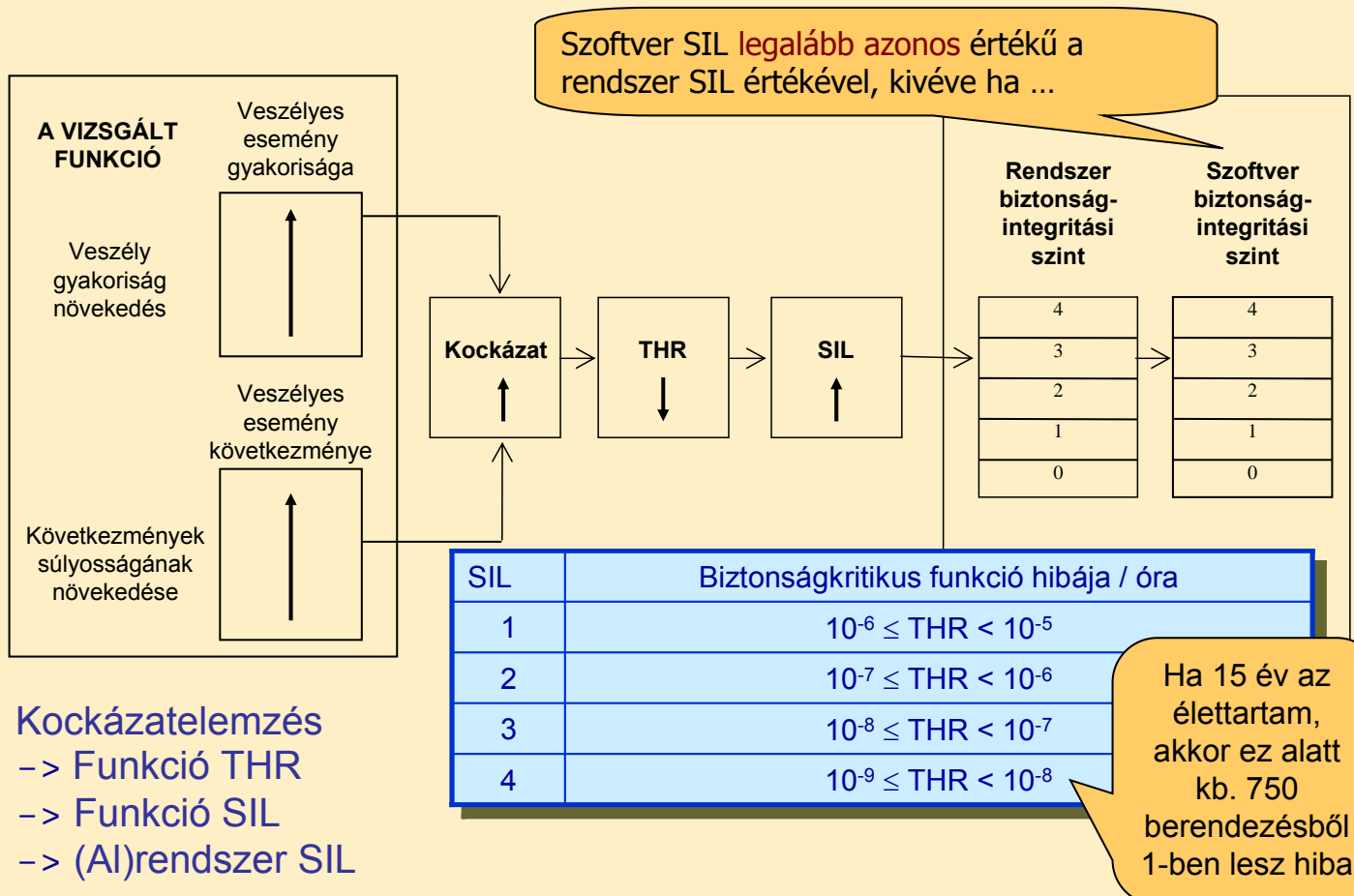
## Tartalomjegyzék

- Motiváció
  - Milyen minőségi igények vannak a szoftverrel szemben, és mit tud ma a szoftveripar?
  - Miért olyan nagy a szoftver ellenőrzési technikák jelentősége?
- A verifikáció és validáció technikái (áttekintés)
  - Milyen tipikus technikák vannak?
- Fejlesztési életciklus modellek
  - Milyen szerepet kapnak a tipikus technikák az egyes fejlesztési folyamatokban?
- Fejlesztési szabványok szerepe
  - Hogyan valósul meg a szisztematikus ellenőrzés?

# Jellegzetes példa: Biztonságkritikus rendszerek

- Széles körben használt szabványok
  - IEC 61508: Functional safety in electrical / electronic / programmable electronic safety-related systems
  - EN 50128: Vasúti irányítástechnika szoftverek
  - ISO 26262: Biztonsági funkciók gépjárművekben
  - DO 178B: Repülőgép fedélzeti rendszerek
- Specialitás: Biztonságintegritási szint (SIL)
  - Meghatározása: Kockázatelemzés alapján
  - „Elviselhető veszélygyakoriság” – THR
    - Folytonos: Veszélyt okozó hibajelenség gyakorisága
    - Nem folytonos: Veszélyt okozó hibajelenség valószínűsége
  - THR kategóriák:
    - Biztonságintegritási szintek:  
Safety Integrity Level: SIL 1, 2, 3, 4 (növekvő)

## SIL meghatározás alapelve



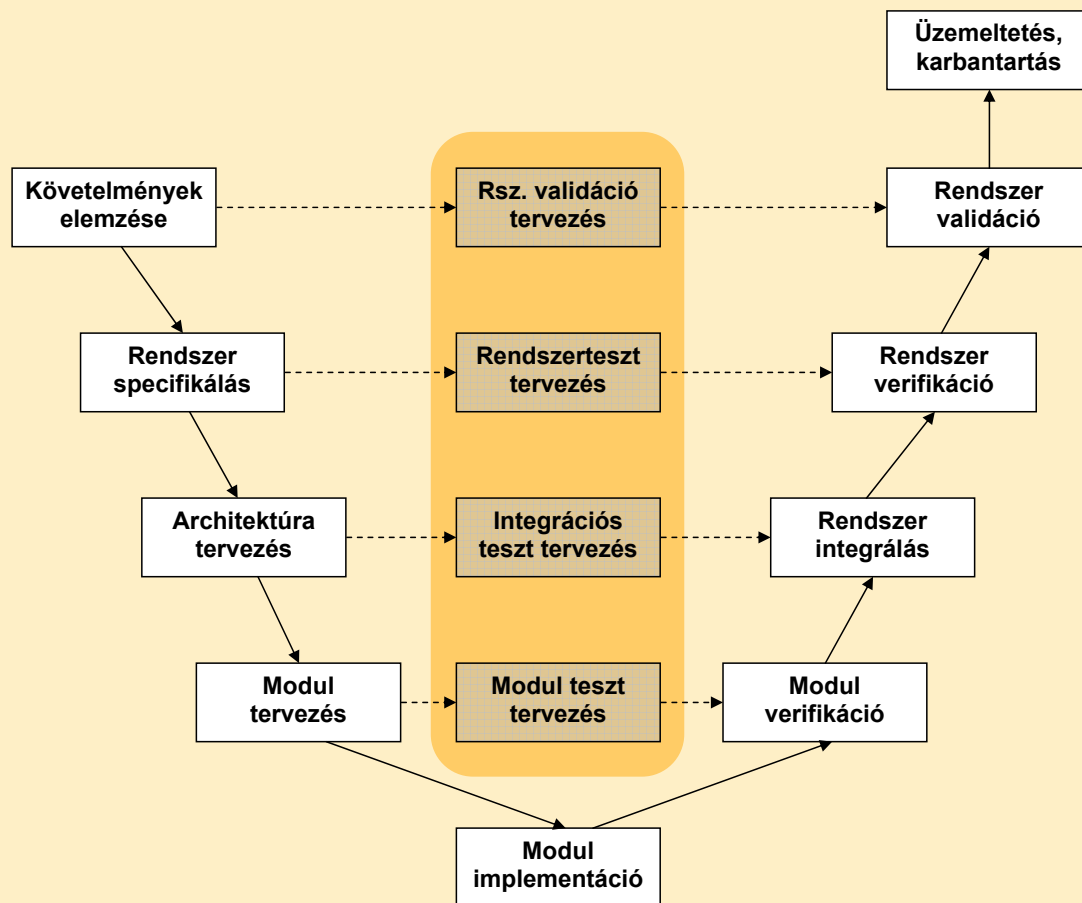
# A SIL követelmények betartása

- **Véletlen meghibásodásokra:**
  - A SIL tartományok betartása számításokkal ellenőrizhető
    - Kvantitatív analízis, megbízhatósági modellezés
- **Szisztematikus meghibásodásokra (pl. szoftver):**
  - Teljes hibamentesség nem garantálható komplex szoftverre
    - Cél: Bennmaradó hibák számának csökkentése
  - Számításokkal nem ellenőrizhető valószínűség!
  - **Módszer- és eszközkészlet előírt a fejlesztés során: Komplex „megoldás-csomag” az egyes SIL szintekhez**
    1. Fejlesztési folyamat (életciklus modell)
    2. Előírt technikák és intézkedések (megoldás-csomag)
    3. Előírt dokumentáció
    4. Szervezeti rend (felelősségek)

## 1. A fejlesztési folyamat

- **Jellegzetességek:**
  - Biztonságigazolás: „Biztonsági ügy” elkészítése (safety case)
  - Értékelés majd tanúsítás (certification)
  - Hatósági felügyelet
- **Általában jól definiált fázisokat tartalmaz**
  - Jól meghatározott specifikáció, ismert környezet
  - Definiált fejlesztési lépések (pl. V-modell)
- **Szigorú feltételekhez kötött előrelépés:**  
**Hangsúlyos a fejlesztési lépések ellenőrzése**
  - Hibák **kockázata** nagy (felelősség)
  - Üzembehelyezés utáni javítás **költsége** nagy

# V-modell: Jól meghatározott ellenőrzések



## 2. Módszerek és intézkedések megadása

Tesztelés

MÓDSZER / INTÉZKEDÉS	Említés helye	SW-SILO	SW-SIL1	SW-SIL2	SW-SIL3	SW-SIL4
1. Valószínűségi tesztelés	B47	-	R	R	HR	HR
2. Teljesítéstesztelés	D6	-	HR	HR	M	M
3. Funkcionális és "fekete doboz" tesztelés	D3	HR	HR	HR	M	M
4. Modellezés	D5	-	R	R	R	R
Követelmény:						
1. Az 1, 2, 3 és 4 szoftver-biztonságintegritási szintek esetén a 2. és 3. módszer kombinációja jóváhagyottnak tekintendő.						

- **Előírások:**

**M** Kötelező

**HR** Nyomatékosan ajánlott (elhagyása indoklást igényel)

**R** Ajánlott (kombinációból kihagyható)

**-** Nincs javaslat vagy ellenérv

**NR** Ellenjavallt (használata indoklást igényel)

- **Módszerkombinációk választhatók**

## Módszerek és intézkedések megadása (EN 50128)

- Software design and implementation:

TECHNIQUE/MEASURE	Ref	SWS ILO	SWS IL1	SWS IL2	SWS IL3	SWS IL4
14. Functional/ Black-box Testing	D.3	HR	HR	HR	M	M
15. Performance Testing	D.6	-	HR	HR	HR	HR
16. Interface Testing	B.37	HR	HR	HR	HR	HR

- Functional/black box testing (D3):

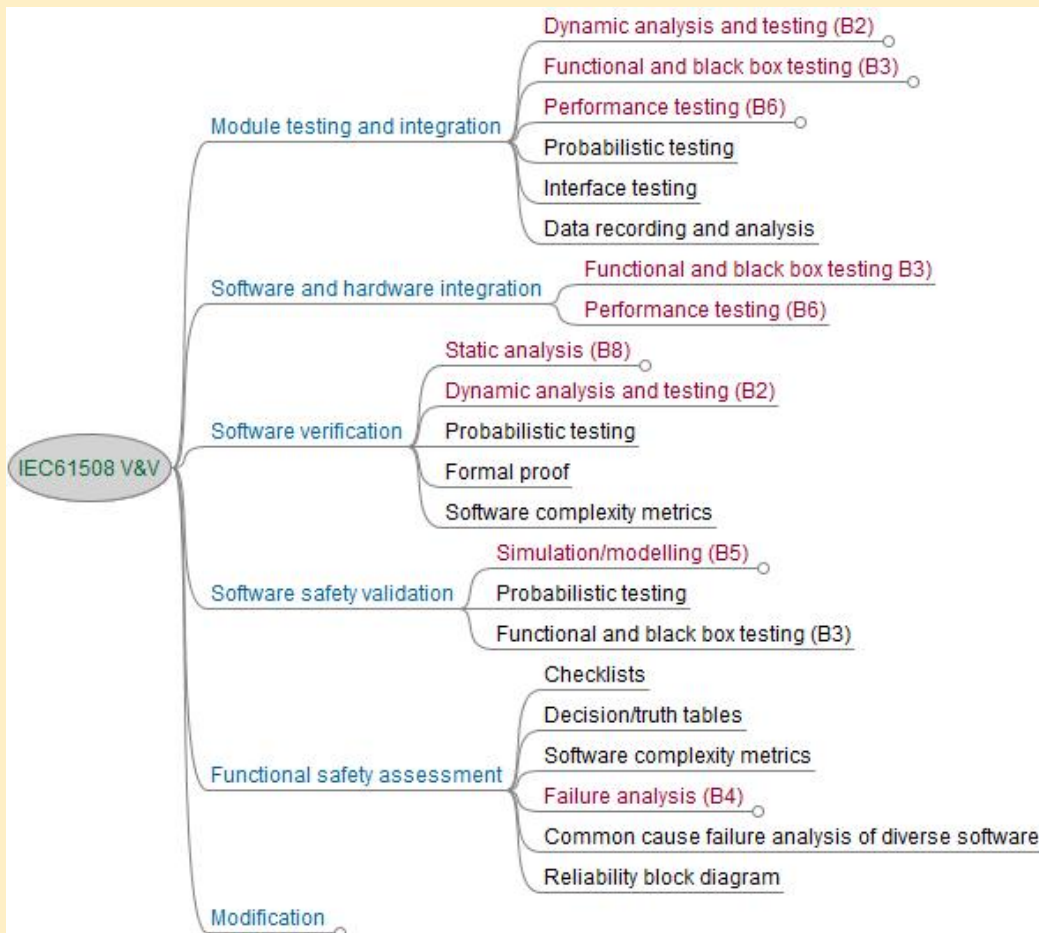
1. Test Case Execution from Cause Consequence Diagrams	B.6	-	-	-	R	R
2. Prototyping/Animation	B.49	-	-	-	R	R
3. Boundary Value Analysis	B.4	R	HR	HR	HR	HR
4. Equivalence Classes and Input Partition Testing	B.19	R	HR	HR	HR	HR
5. Process Simulation	B.48	R	R	R	R	R

## Módszerek és intézkedések megadása (EN 50128)

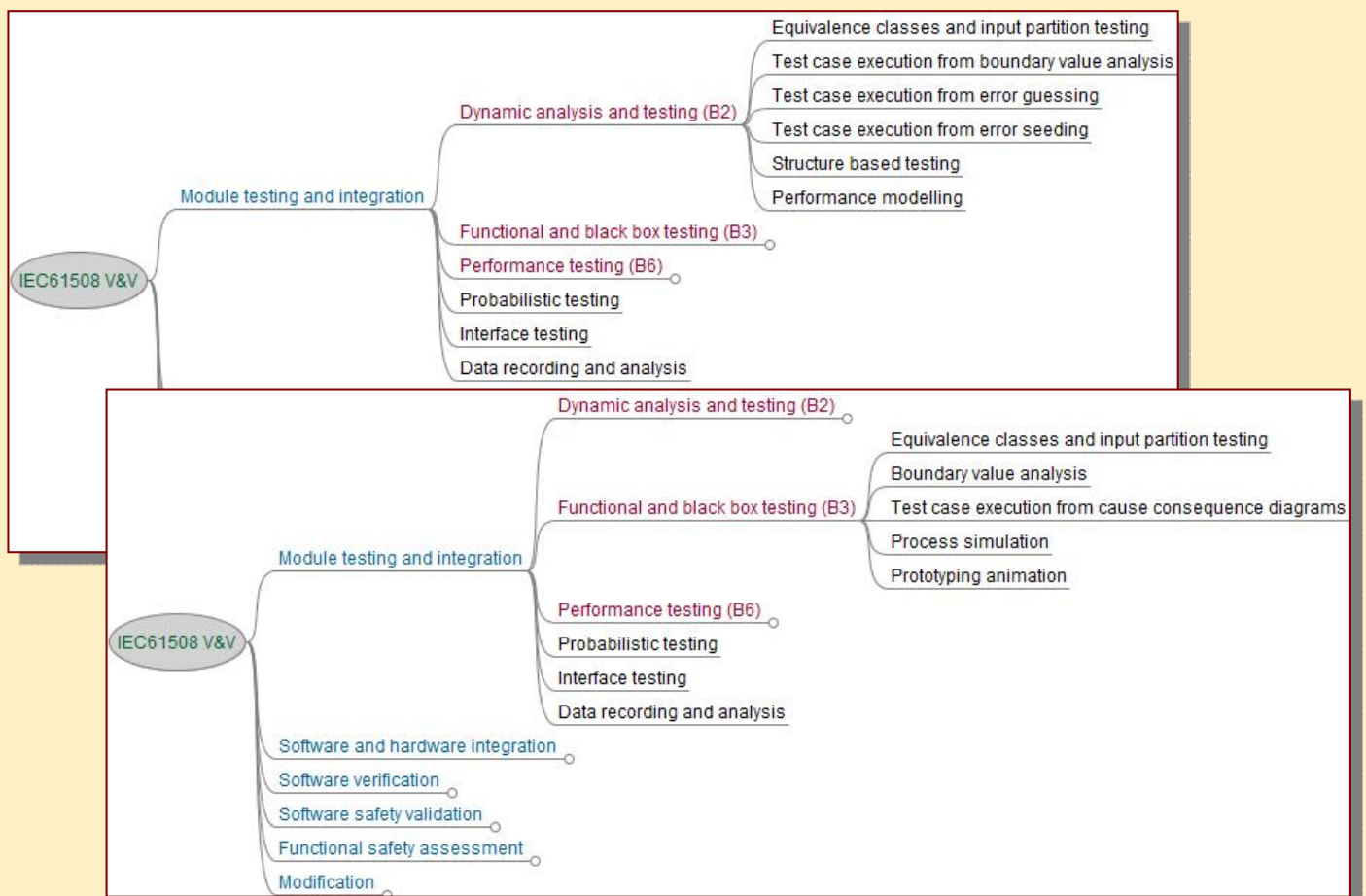
- Performance testing (D6):

TECHNIQUE/MEASURE	Ref	SWS ILO	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. Avalanche/Stress Testing	B.3	-	R	R	HR	HR
2. Response Timing and Memory Constraints	B.52	-	HR	HR	HR	HR
3. Performance Requirements	B.46	-	HR	HR	HR	HR

# V&V módszerek (IEC 61508)

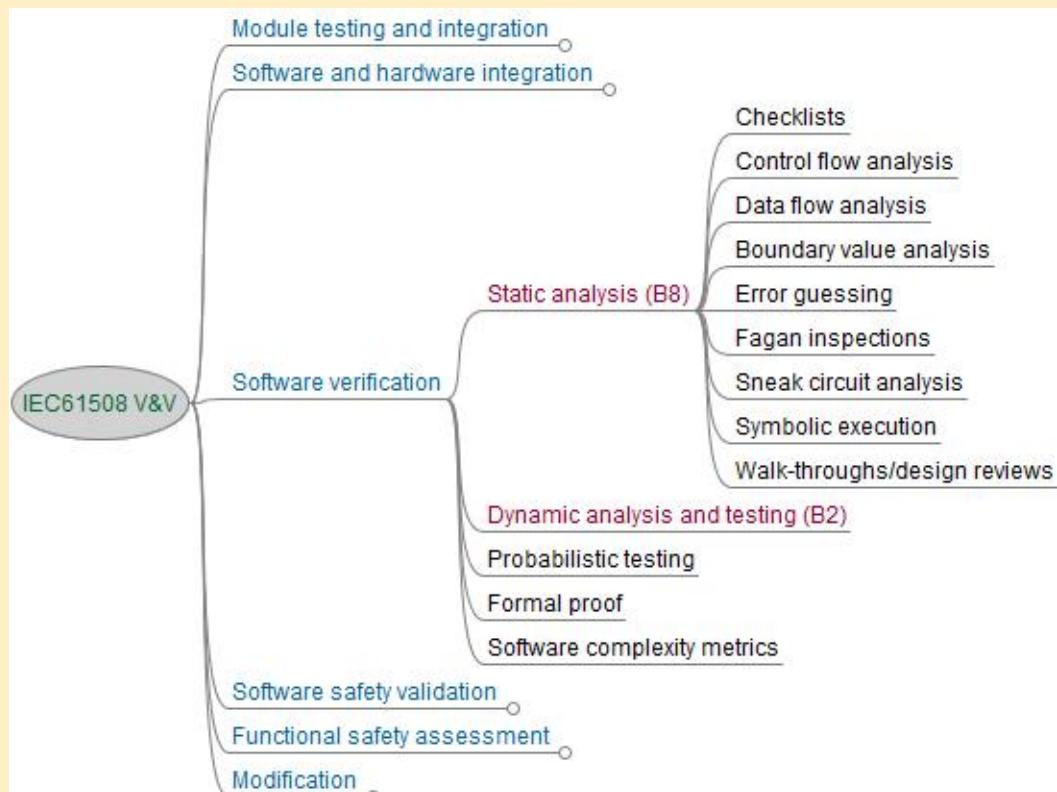


# V&V módszerek (IEC 61508) – Funkcionális tesztelés





# V&V módszerek (IEC61508) – Statikus analízis



## 3. A dokumentáció követelményei

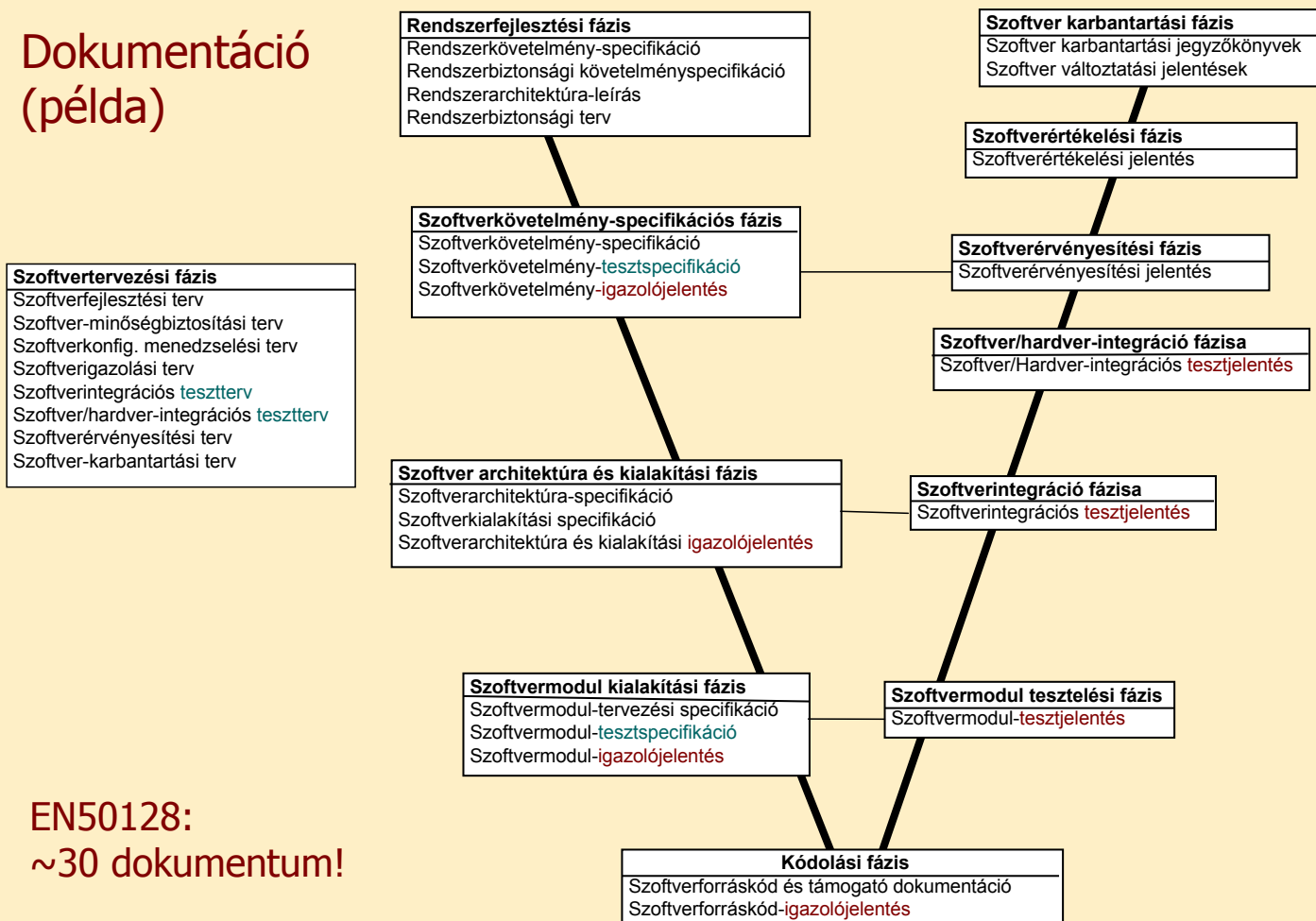
- Dokumentáció típusa
  - Átfogó
    - Pl. fejlesztési terv, verifikációs terv
  - Életciklus fázishoz kötődő
    - Pl. modul teszt jelentés, modul verifiációs jelentés
- Dokumentum **keresztreferencia táblázat**
  - Melyik életciklus fázishoz milyen dokumentáció
  - Melyik dokumentum melyik másakra épül
- Dokumentumok **követhetősége szükséges**
  - Ugyanazon terminológia, rövidítések, elnevezések
- Dokumentumok **összevonhatók**
  - Eredmény nem vesztet el
  - **Független szereplők** dokumentumai nem vonhatók össze



# Dokumentum kereszt-referencia táblázat

cím	S R S	S A	S D D	S V e r	S / H	S V a l	A s s	Q	M A	DOKUMENTUMOK
<b>FÁZISOK</b> (* = más fázisokkal párhuzamosan)										
SW KÖVETELMÉNYEK	■	◆	◆	◆	◆	◆	◆			Sw Követelményspecifikáció
										Alkalmazási Követelmények Specifikációja
	■			◆	◆	◆	◆			Sw Követelmény Teszt Specifikáció
				■						Sw Követelmények Verifikációs Jelentése
SW KONSTRUKCIÓ KIALAKÍTÁS		■	◆	◆	◆	◆				Sw Architektúra Specifikáció
			■	◆	◆	◆				Sw Konstruktó specifikáció
				■						Sw Architektúra és Konstruktó Verifikációs Jelentés
SW MODUL KONSTRUKCIÓ KIALAKÍTÁS			■	◆	◆	◆				Sw Modul Konstruktó Specifikáció
			■	◆	◆	◆				Sw Modul Teszt Specifikáció
				■						Sw Modul Verifikációs Jelentés
KÓDOLÁS			■	◆	◆	◆				Sw Forráskód
				■		◆	◆			Sw Forráskód Verifikációs Jelentés
MODUL TESZTELÉS			■	◆						Sw Modul Teszt Jelentés
SW INTERGRÁCIÓ				■						Sw Integráció Teszt Jelentés
										Adatstest Jelentés
SW/HW INTEGRÁCIÓ					■					Sw/Hw Integráció Teszt Jelentés
VALIDÁCIÓ (*)						■				Sw Validációs Jelentés

## Dokumentáció (példa)

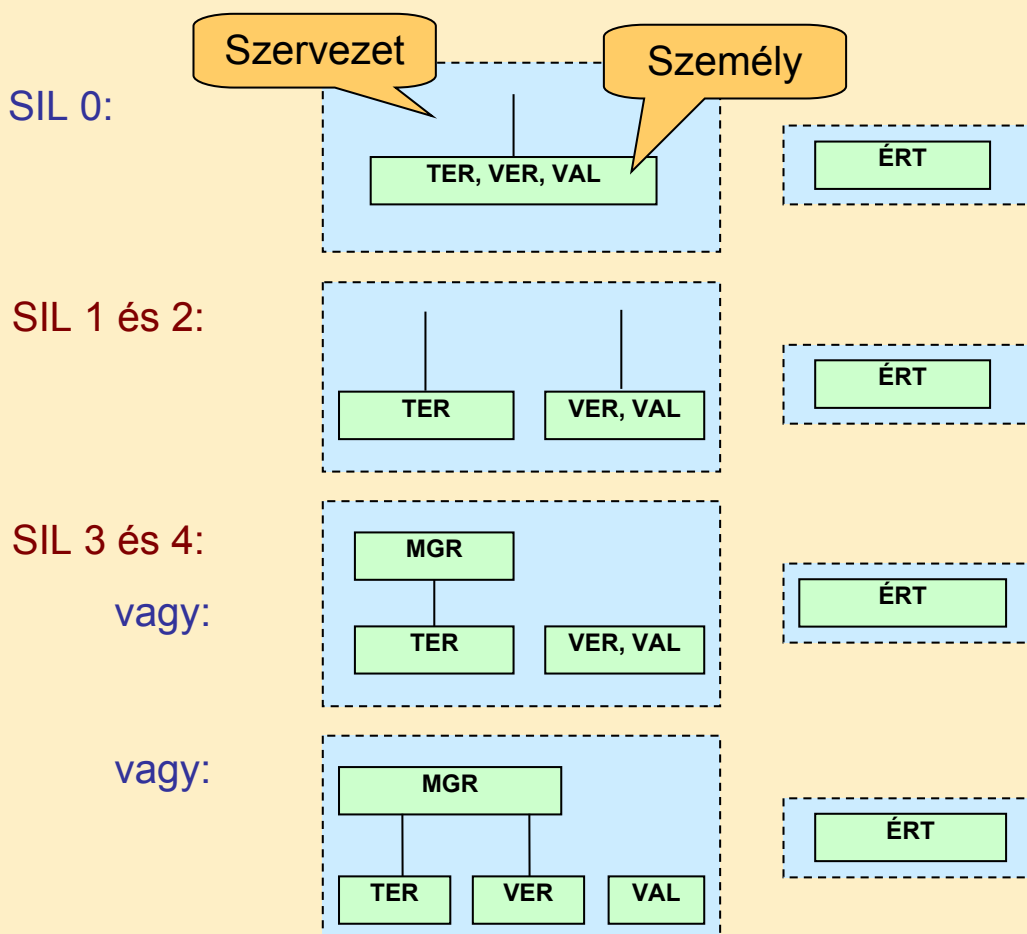


EN50128:  
~30 dokumentum!

## 4. Szervezeti rend

- **Minőségi ill. biztonsági szervezet létrehozása – a biztonságmenedzselés bizonyítása**
  - ISO 9001 vonatkozó részeinek alkalmazása
  - Konfigurációmenedzselés
- **Képzettség (alkalmasság) igazolása**
- **Szereplők:**
  - Tervező (elemző, tervező, kódoló, unit tesztelő) TER
  - Verifikátor (igazoló) VER
  - Validátor (érvényesítő) VAL
  - Értékelő (független felülvizsgáló) ÉRT
  - Projekt menedzser MGR
  - Minőségbiztosítási felelős MIN

### Minimális függetlenség követelményei



# Miről volt szó?

- **Motiváció**
  - Milyen minőségi igények vannak a szoftverrel szemben, és mit tud ma a szoftveripar?
  - Miért olyan nagy a szoftver ellenőrzési technikák jelentősége?
- **A verifikáció és validáció technikái (áttekintés)**
  - Milyen tipikus technikák vannak?
- **Fejlesztési életciklus modellek**
  - Milyen szerepet kapnak a tipikus technikák az egyes fejlesztési folyamatokban?
- **Fejlesztési szabványok szerepe**
  - Hogyan valósul meg a szisztematikus ellenőrzés?