A szoftverellenőrzés szerepe

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?

Elvárások: Szoftverek és rendszerek hibamentessége

- Szolgáltatási szint szerződések (SLA)
  - Rendelkezésre állás (telekom): 99,999% (5 perc/év kiesés)
- Biztonságkritikus rendszerek:
  - Elviselhető hibagyakoriság biztonsági funkcióra (THR)
  - Biztonságintegritási szintek (SIL)

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

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

Hiba nélküli működés ~ 11.000 év?

Hibák az alkalmazás életciklusban

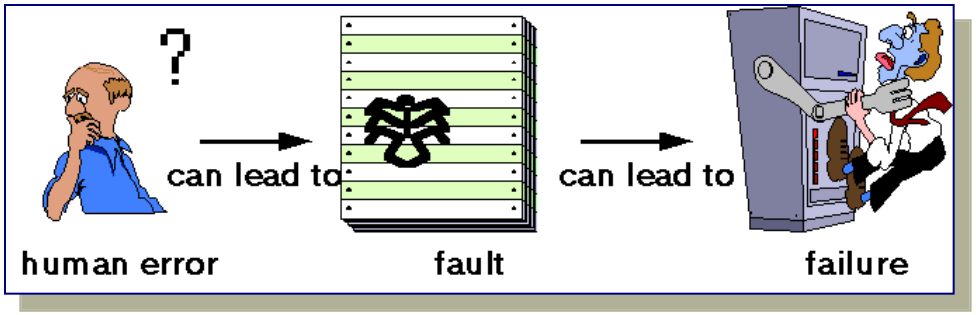
Fejlesztési folyamat — **Működő termék**

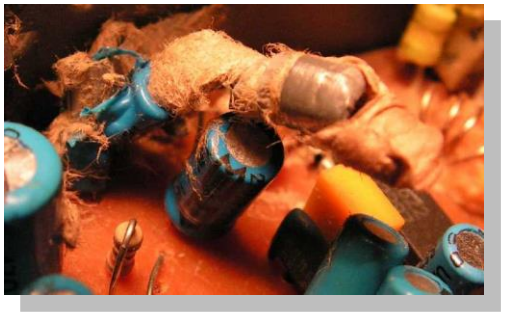
- Specifikációs hibák
- Tervezési hibák
- Implementációs hibák

Verifikáció és validáció a tervezés során

- Hardver hibák
- Konfigurációs hibák
- Kezelői hibák

Hibatűrés működés közben (pl. redundancia)



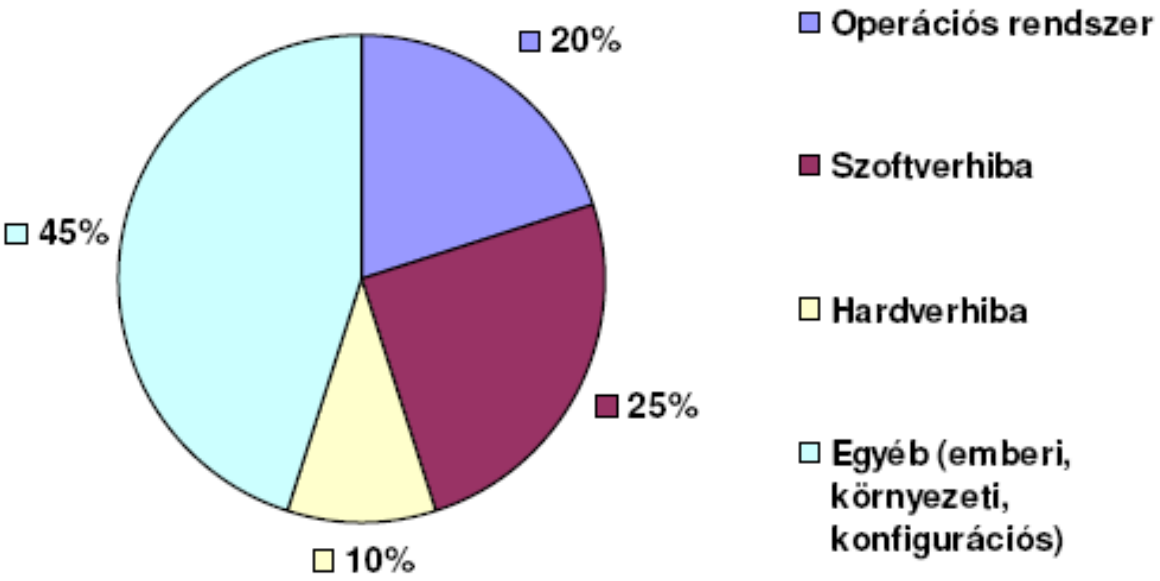


Kliens-szerver rendszerek meghibásodása

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 leállás: 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 ráfordítás:
  - Nagyméretű átlagos szoftver: 0,1 - 0,5 mérnökév / kLOC
  - Biztonságkritikus szoftver: 5-10 mérnökév / kLOC
- Hiba eltávolítás (ellenőrzés, tesztelés, javítás):
  - Az összes ráfordítás 45 - 75%-a!
- Hibasűrűség változása:
  - 10 - 200 hiba / kLOC jön létre a fejlesztés során

Ellenőrzési technikák

  - 0,1 - 10 hiba / kLOC maradhat az üzembe helyezésig

Milyen szoftver hibagyakoriság a tipikus?

DB Mobility Networks Logistics

How many „Bugs“ do we have to expect?

- **Typical production type SW has 1 … 10 bugs per 1.000 lines of code (LOC).**
- **Very mature, long-term, well proven software: 0,5 bugs per 1.000 LOC**
- **Highest software quality ever reported :**
  - ***Less than 1 bug per 10.000 LOC***
  - **At cost of more than 1.000 US$ per LoC *(1977)***
  - ***US Space Shuttle with 3 m LOC* costing 3b US$ (out of 12b$ total R&D)**

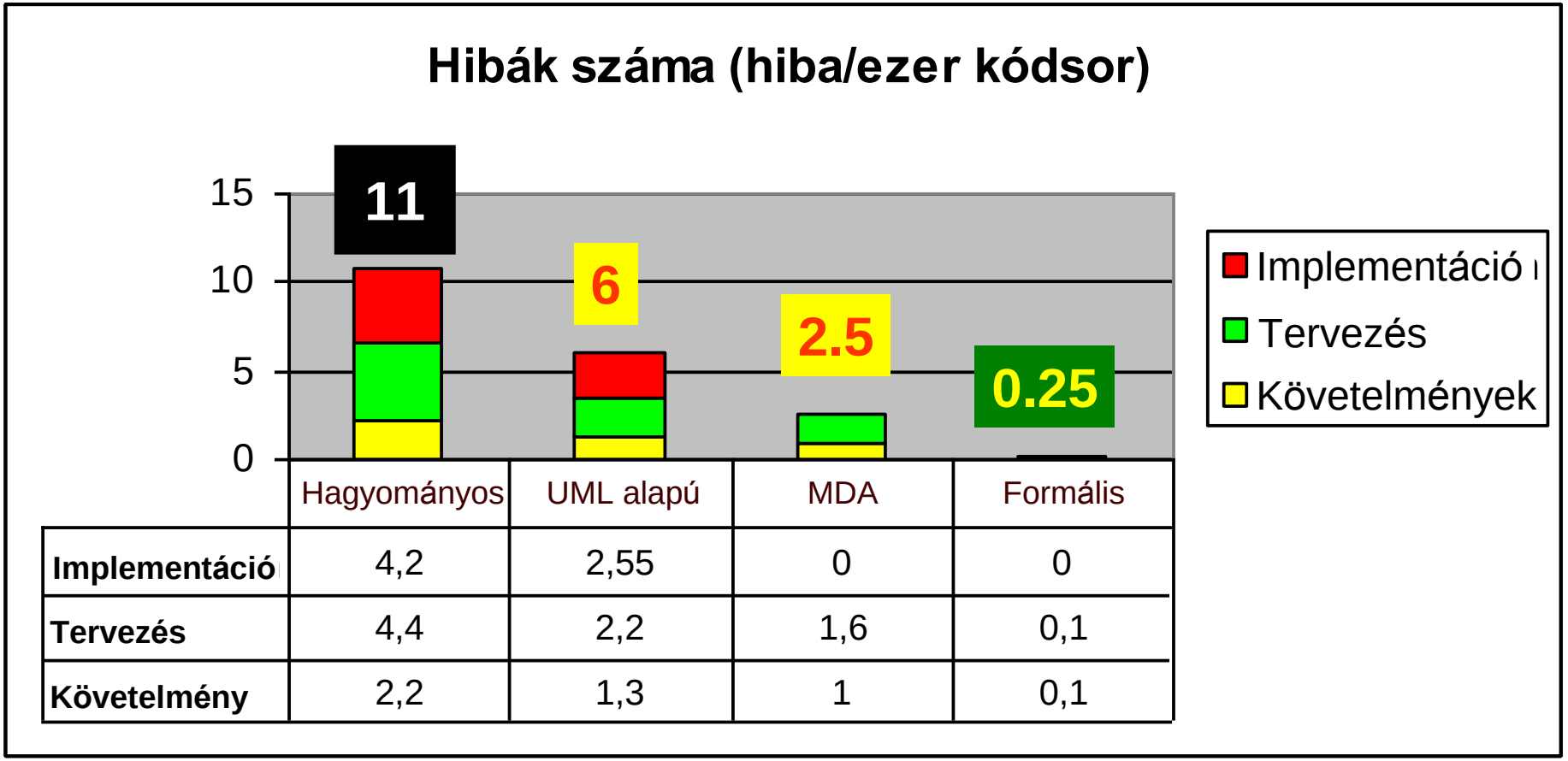
➔ **Cost level not typical for the railway sector (< 100€/LoC)**

- **Typical ETCS OBU kernel software size is about 100.000 LOC or more**
  - **That means: 100 … 1.000 undisclosed defects per ETCS OBU**
  - **Disclosure time of defects can vary between a few days …. thousands of years**

Forrás: K-R. Hase, Deutsche Bahn AG: „Open Proof in Railway Safety Software”, FORMS/FORMAT Conference, December 2-3, 2010, Braunschweig, Germany

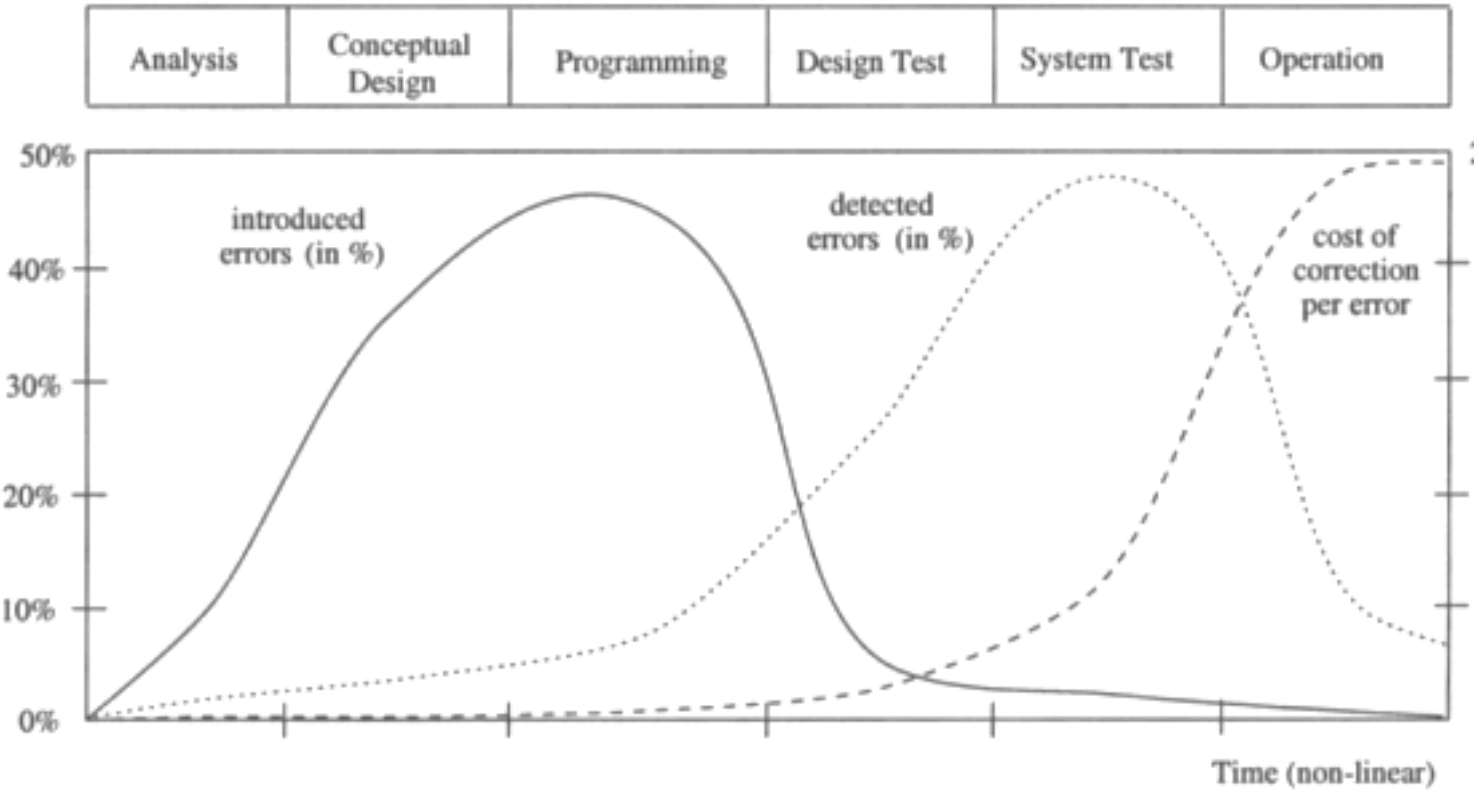
Egy (magyarországi) mérés eredményei

- 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



	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

Költségvonzat

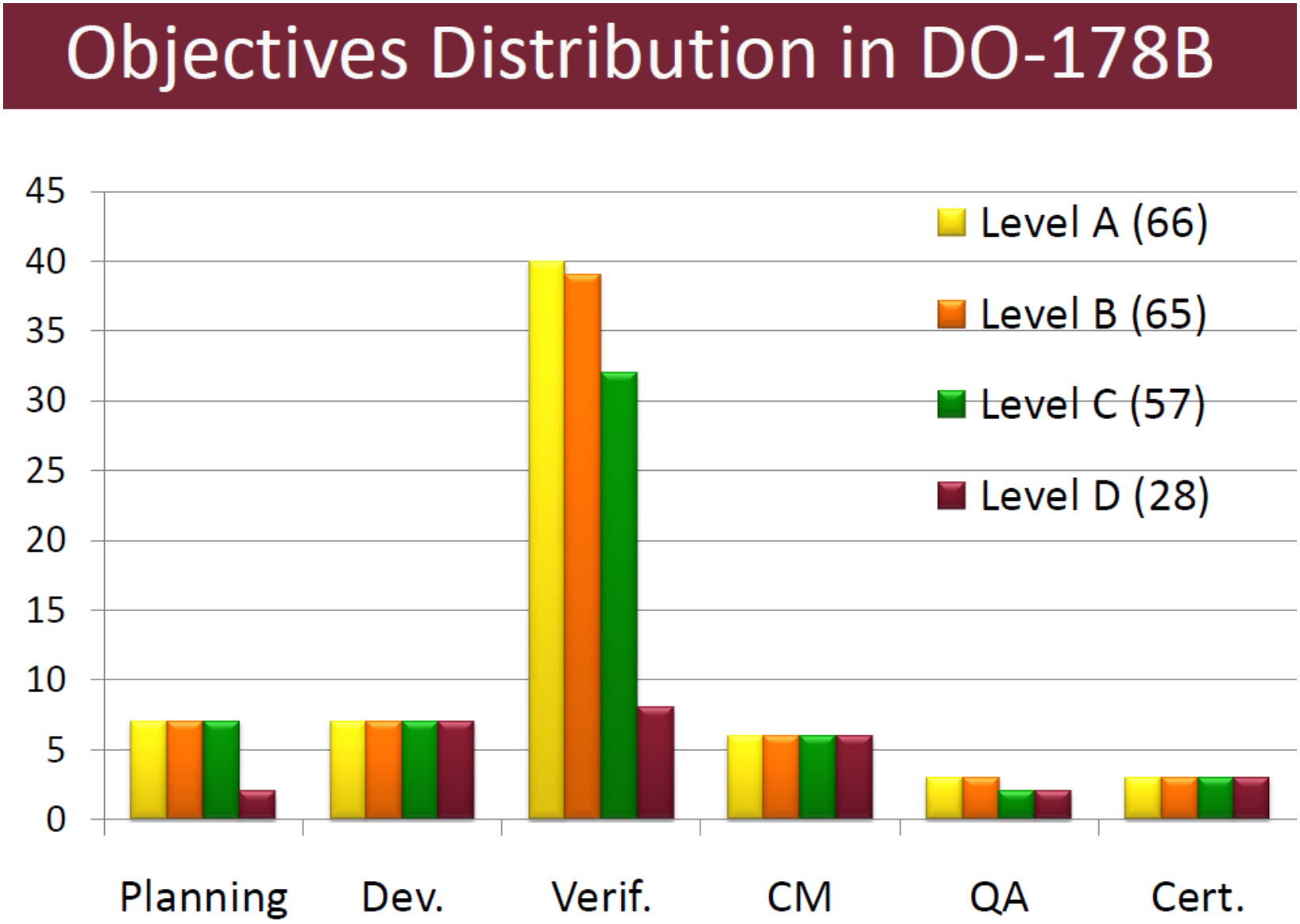


- Korai verifikáció csökkenti a költségeket
  - Validációs tesztelésnél korábban detektálni kellene a hibákat

V&V: 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)

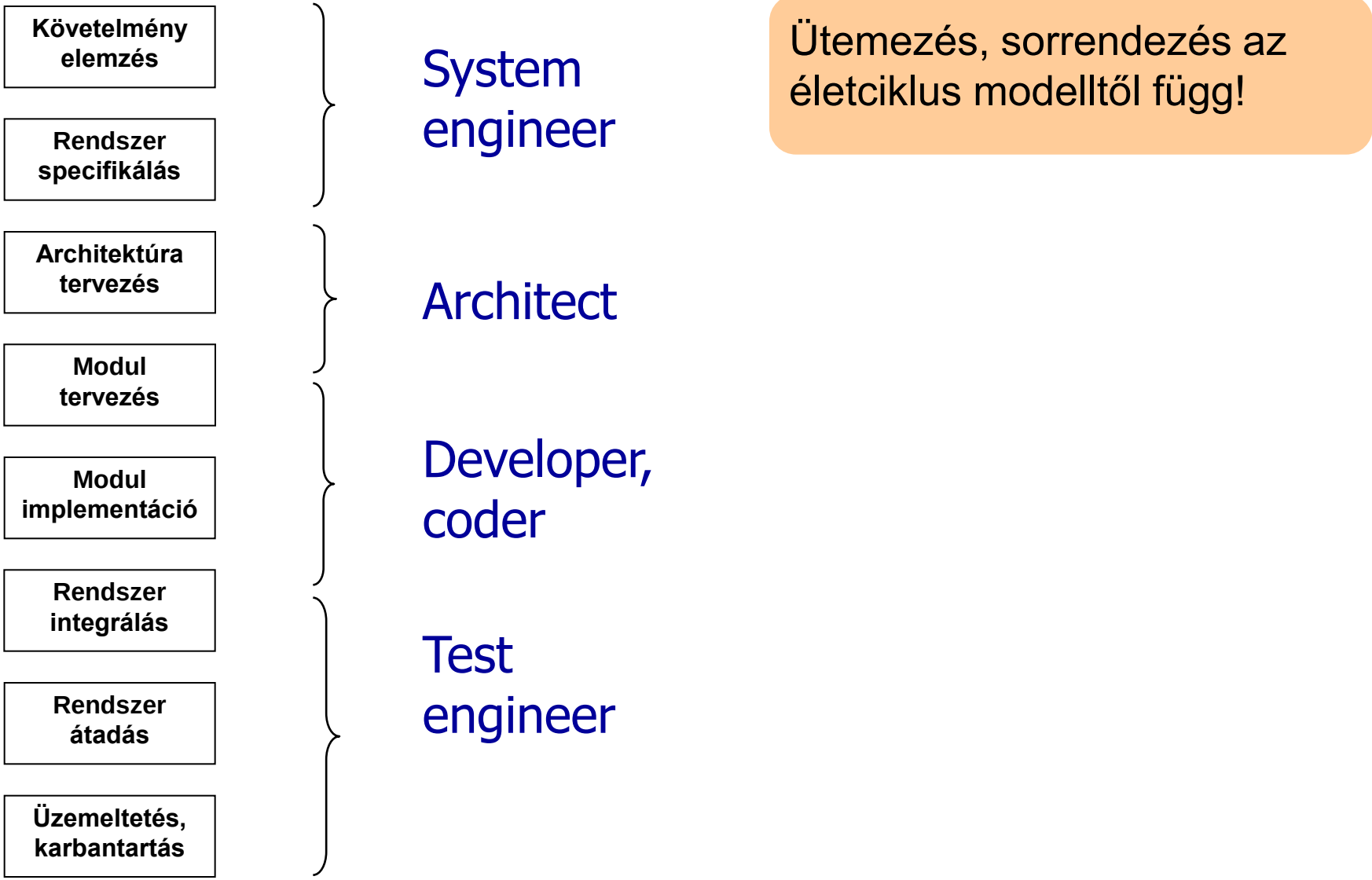
Példa: Repülőgép fedélzeti szoftverek fejlesztése



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 folyamatok tipikus lépései

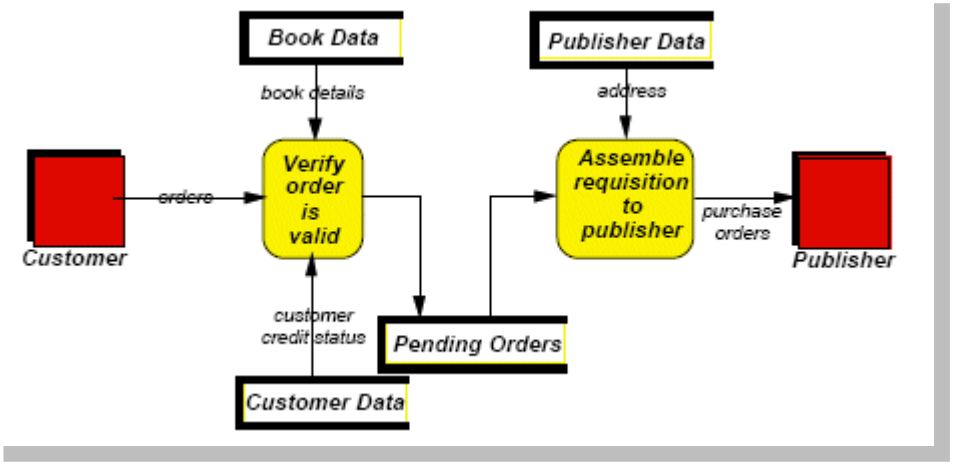


Ütemezés, sorrendezés az életciklus modelltől függ!

Követelmény elemzés

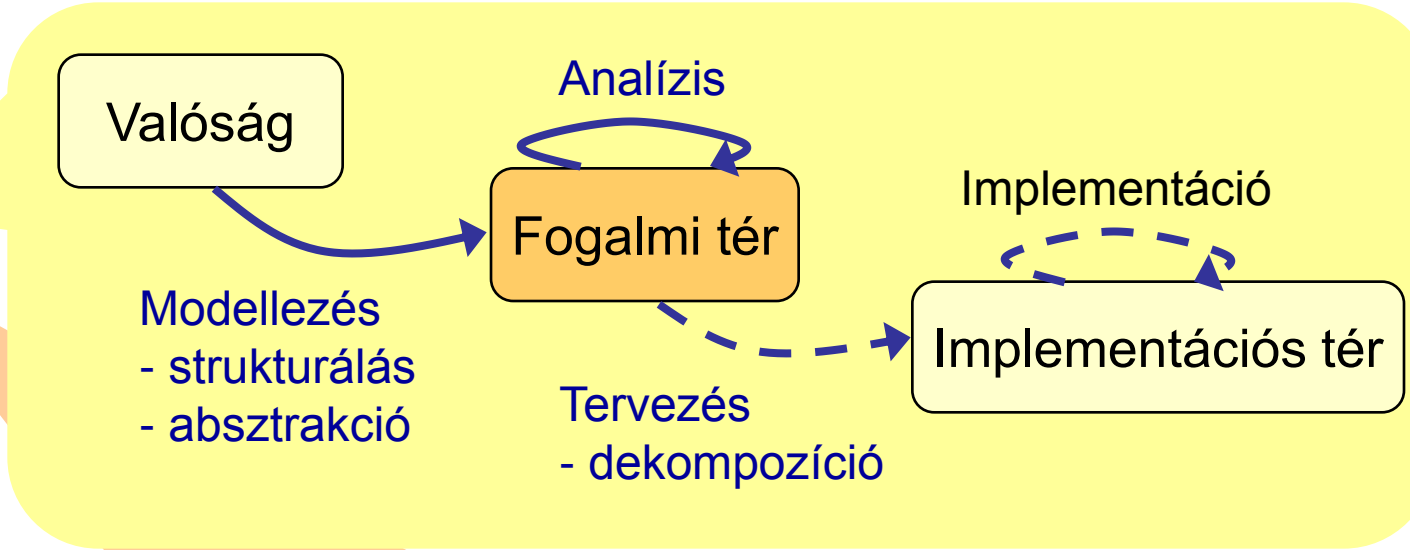
- **Követelmény elemzés**
- **Rendszer specifikálás**
- **Architektúra tervezés**
- **Modul tervezés**
- **Modul implementáció**
- **Rendszer integrálás**
- **Rendszer átadás**
- **Üzemeltetés, karbantartás**

Feladat	V&V szempont	V&V technika
Funkciók, aktorok, használati esetek felvétele	- Kockázatok - Kritikusság	- Ellenőrző listák - Hibamód és -hatás analízis



Rendszer specifikálás

- 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	- Teljesség - Ellentmondás-mentesség - Ellenőrizhetőség - Megvalósíthatóság	- Statikus analízis (kézi vagy automatikus átvizsgálás) - Szimuláció

Rendszer specifikálás

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

(Szakértői) 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

Feladat	V&V szempont	technika
Funkcionális és nem-funkcionális követelmények rögzítése	- Teljesség - Ellentmondás-mentesség - Ellenőrizhetőség - Megvalósíthatóság	- Statikus analízis (kézi vagy automatikus átvizsgálás) - Szimuláció

Rendszer specifikálás

- 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:	$sit \subseteq aut$	

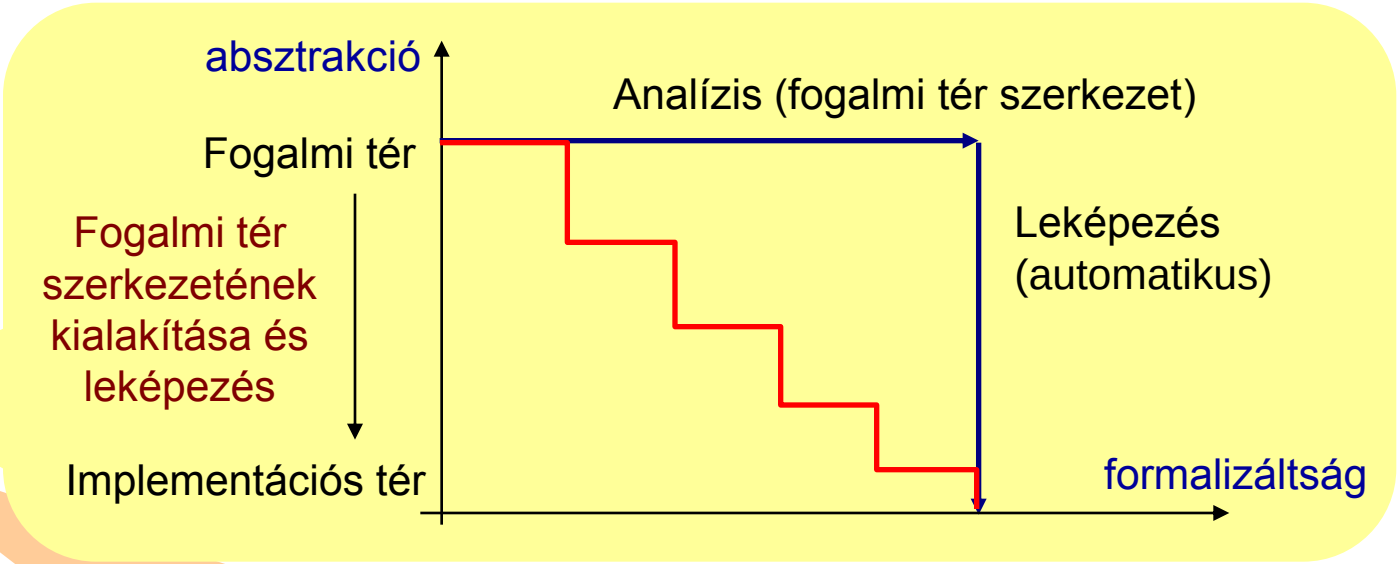
Egy funkció (lehetséges történés):

```
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	- Teljesség - Ellentmondás-mentesség - Ellenőrizhetőség - Megvalósíthatóság	- Statikus analízis (kézi vagy automatikus átvizsgálás) - Szimuláció

Architektúra tervezés

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

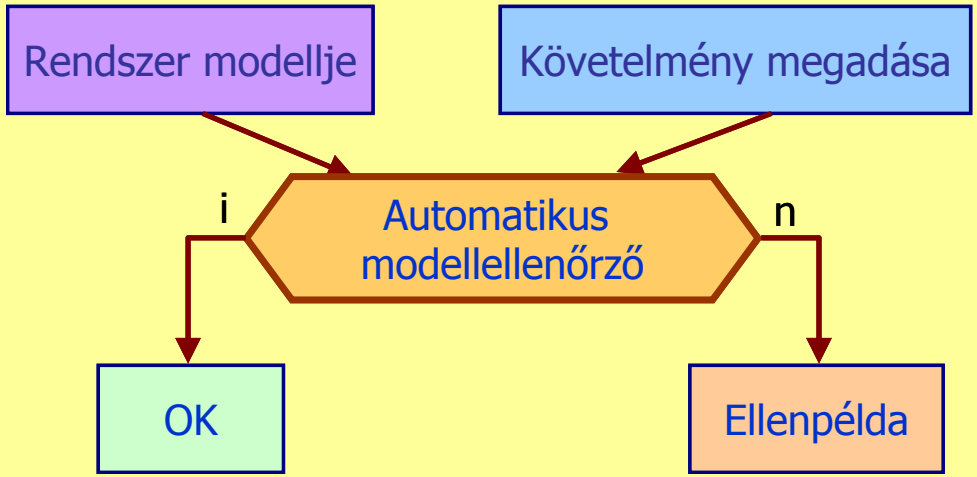


Feladat	V&V szempont	V&V technika
- Specifikáció modul szintű bontása - Hardver-szoftver együttes tervezés - Kommunikáció tervezés	- Funkciók fedése - Interfész illeszkedés - Extra-funkcionális tulajdonságok megfelelősége	- Statikus analízis - Szimuláció - Teljesítmény, megbízhatóság, biztonság analízise

Modul tervezés (részletes tervezés)

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

Formális verifikáció

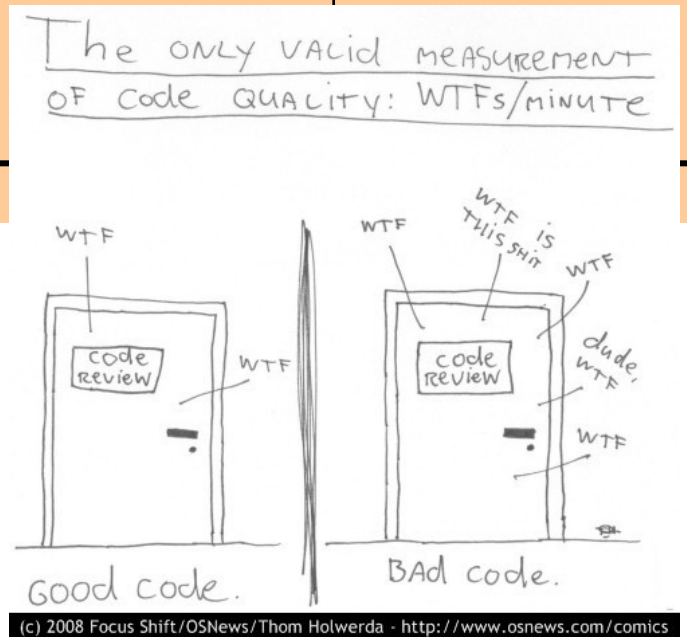


Feladat	V&V szempont	V&V technika
Részletes belső működés tervezése (algoritmusok, adatstruktúrák)	- Kritikus belső algoritmusok, protokollok helyessége	- Statikus analízis - Szimuláció - Formális verifikáció - Gyors prototípus

Modul implementáció

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

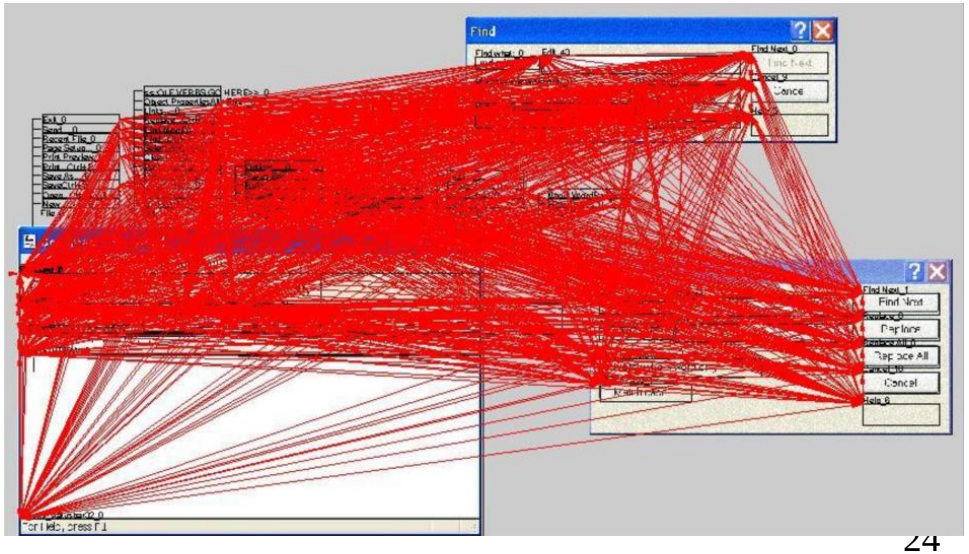
Feladat	V&V szempont	V&V technika
Szoftver implementáció	- Biztonságos - Ellenőrizhető - Karbantartható forráskód	- Kódolási előírások (szabályok) ellenőrzése: statikus kód átvizsgálás
Modul működés igazolása	- Modul terveknek való megfelelés	- Statikus analízis - Tesztelés - Regressziós tesztelés



Rendszer integrálás

- 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)



Rendszer átadás

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

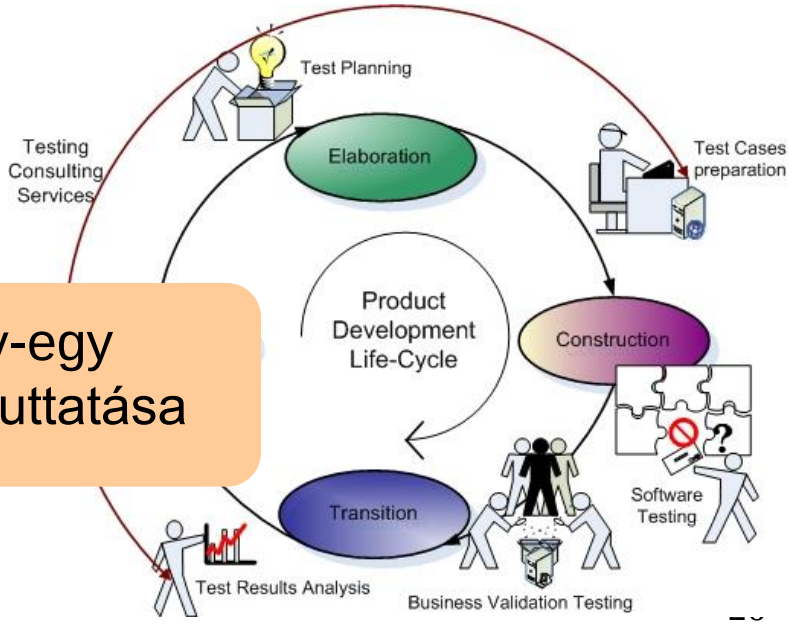
Feladat	V&V szempont	V&V technika
Specifikáció teljesítésének igazolása	- Rendszer-specifikációnak való megfelelés	- Rendszertesztelés - Mérések, monitorozás
Felhasználói elvárások teljesítése	- Követelményeknek és elvárásoknak való megfelelés	- Validációs tesztelés - Próbaüzem alatti ellenőrzés

Üzemeltetés és karbantartás

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

Teendők az üzemeltetés és karbantartás során:
- Hibanaplózás és hibaanalízis (hiba előrejelzéshez is)
- Módosítások verifikációja és validációja

Módosításokra egy-egy „mini életciklus” lefuttatása



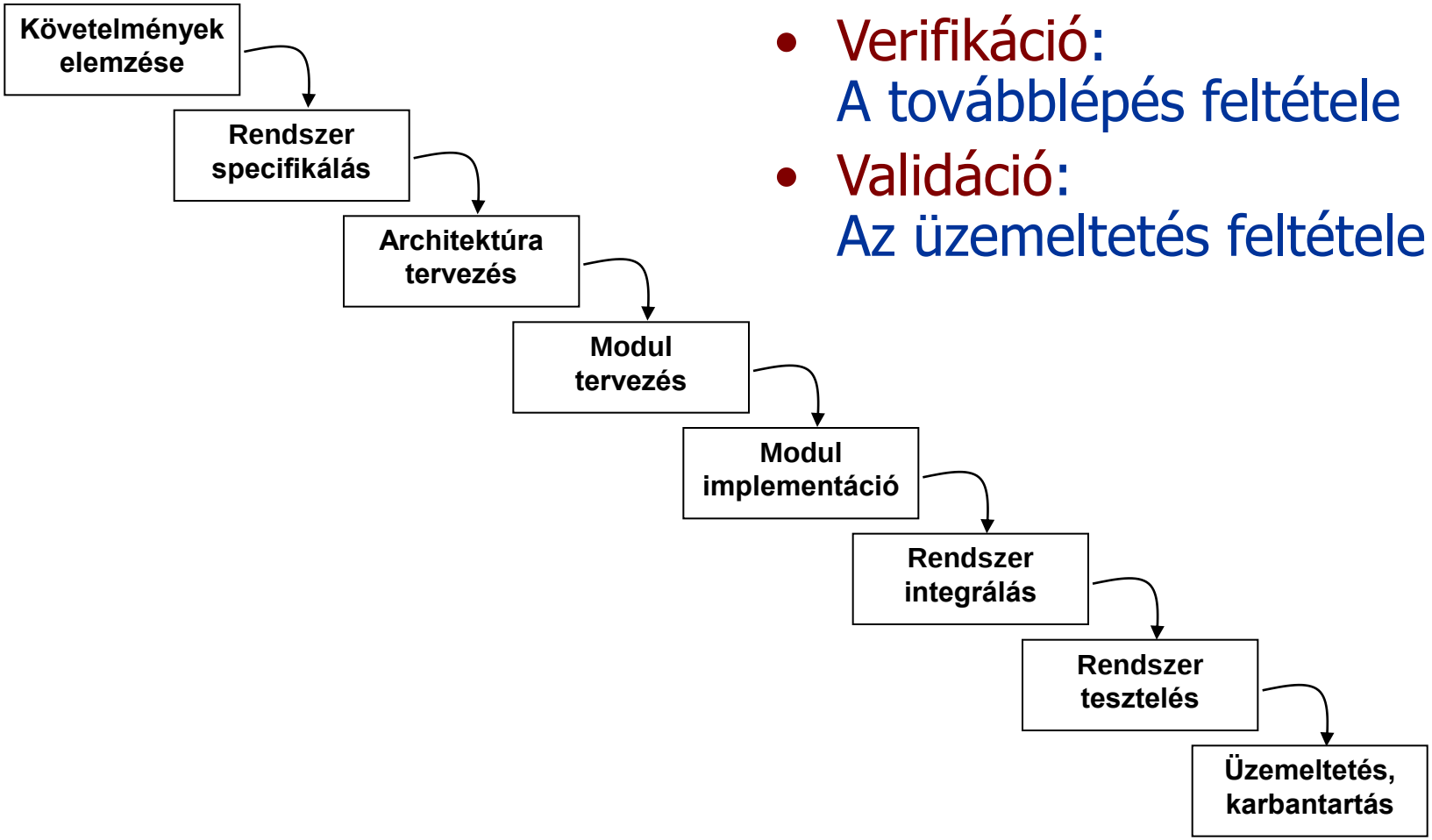
Tartalomjegyzék

- 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?

Szoftverfejlesztési (életciklus) modellek

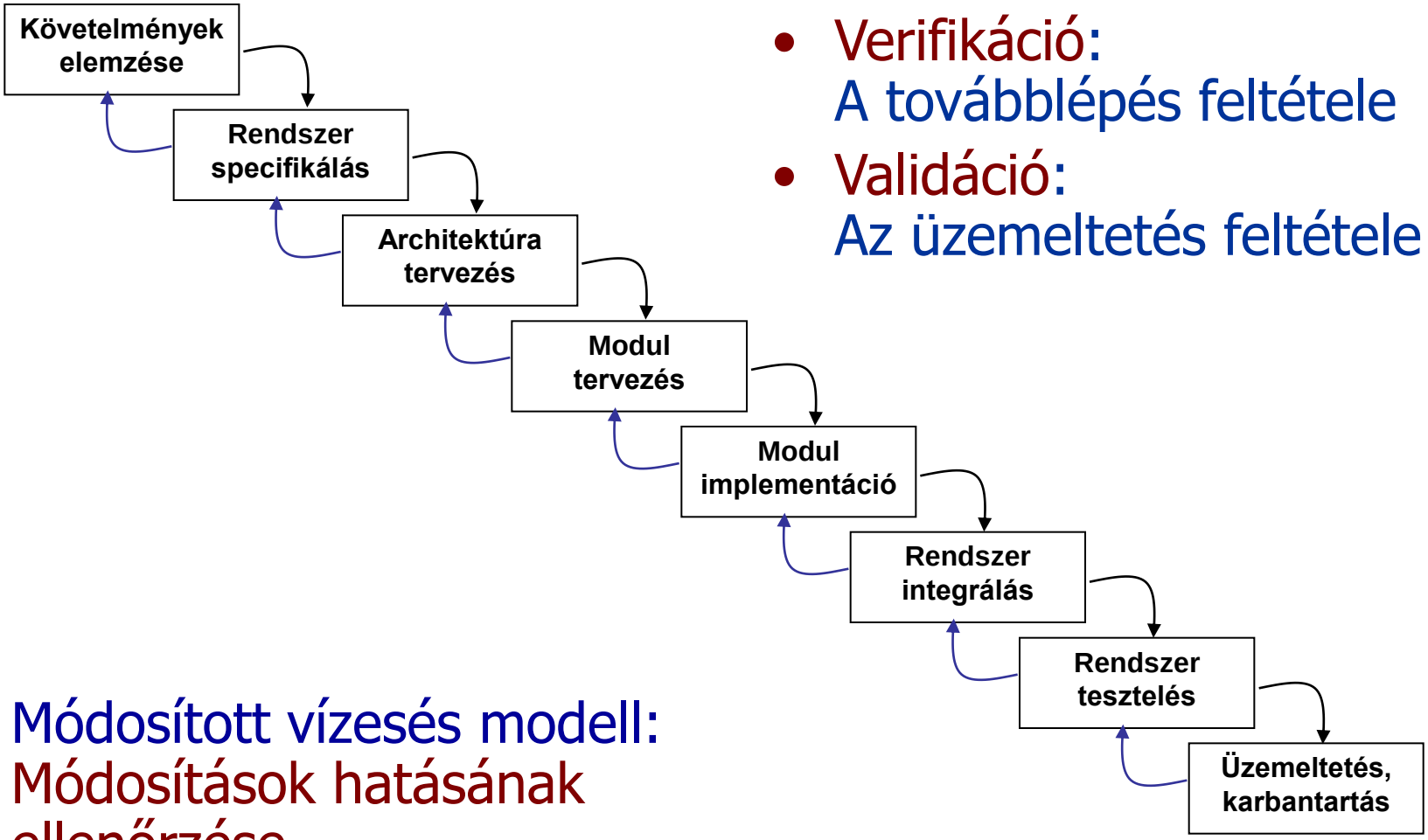
- Miért van szükség életciklus modellre?
  - Komplexitás kezelése
    - Fázisokra osztás, mérföldkövek rögzítése
    - Elosztott fejlesztés és integráció alapja
  - Változások kezelése
    - Követelmény módosulás, hibajavítás hatásainak kezelése
    - Új eszközök és technológiák bevezetése
- Generikus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

1. Vízesés modell



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

1. Vízesés modell

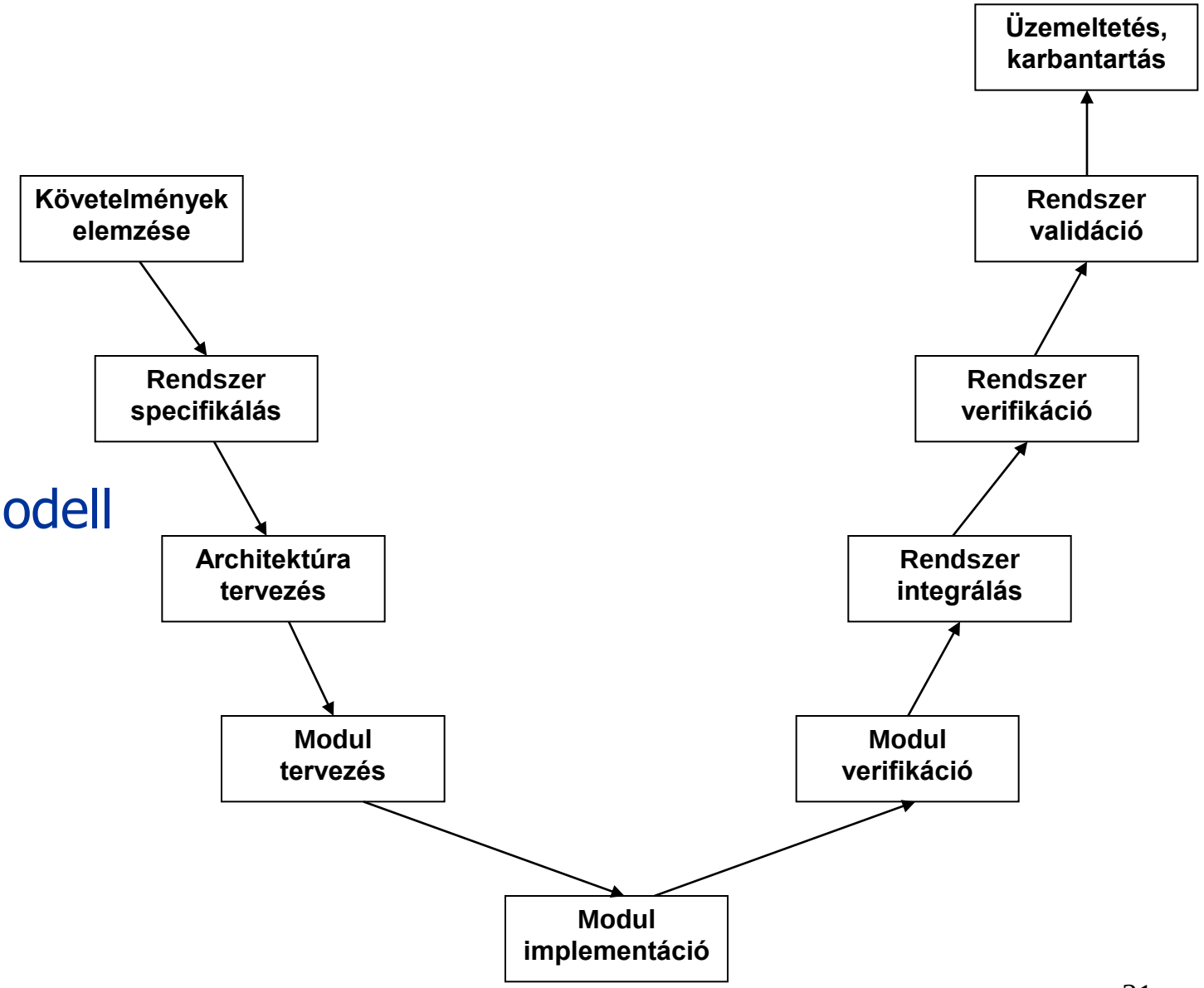


- Verifikáció:
  A továbblépés feltétele
- Validáció:
  Az üzemeltetés feltétele

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

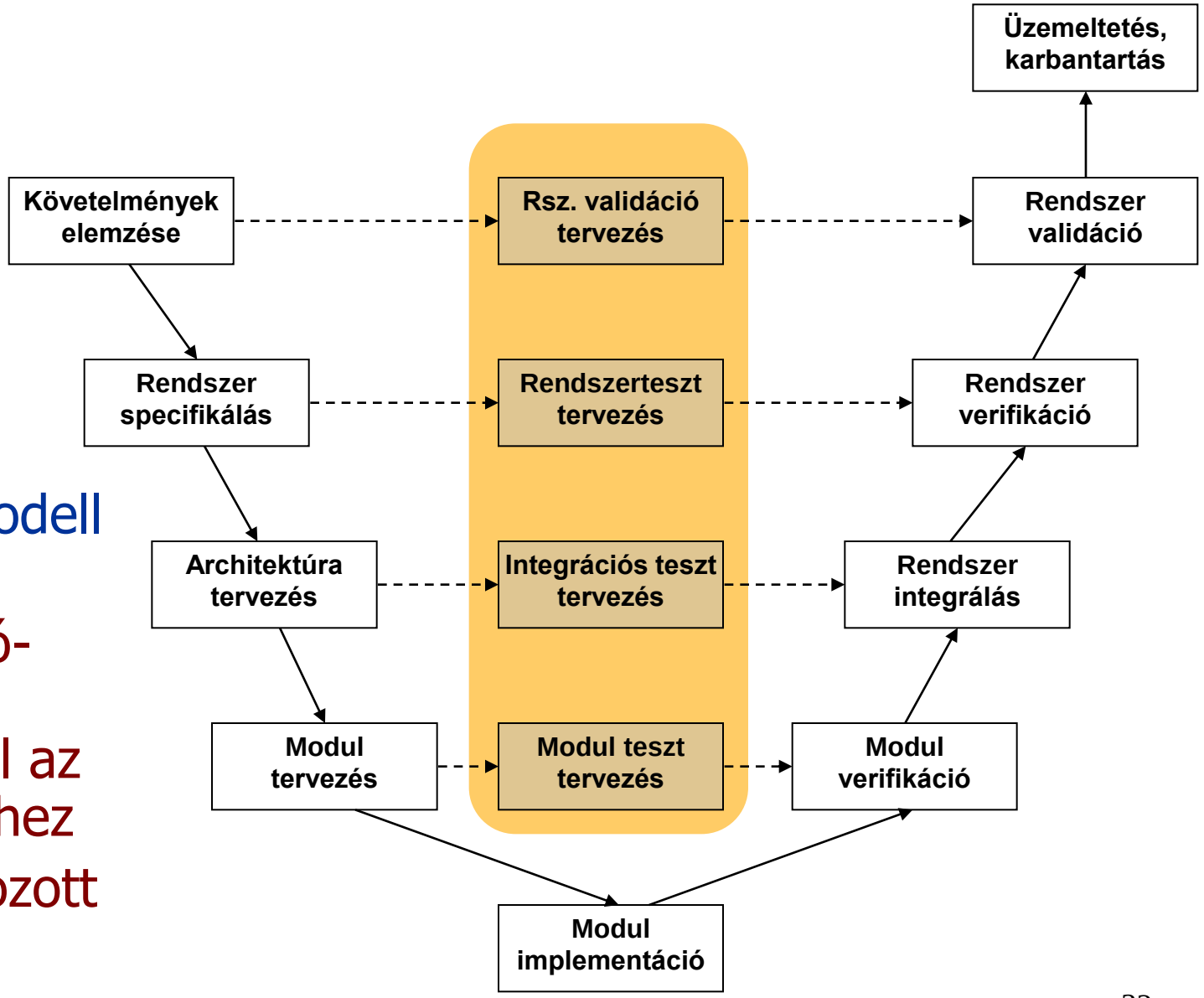
2. V-modell

- Vízesés modell alapú



2. V-modell

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



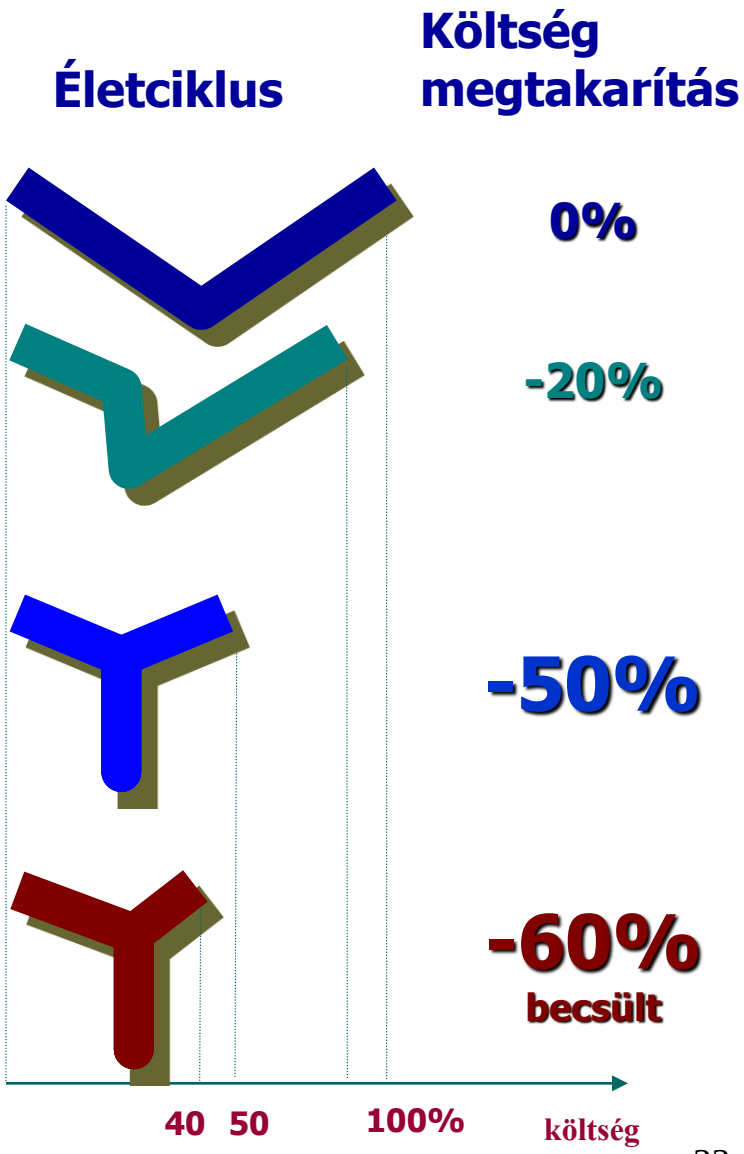
Modell alapú fejlesztés: V-től az Y modellig

Kézi kódolás

"Közönséges" automatikus kódgenerátor használata

Minősített automatikus kódgenerátor használata

Formális verifikációval kiegészített tervezés



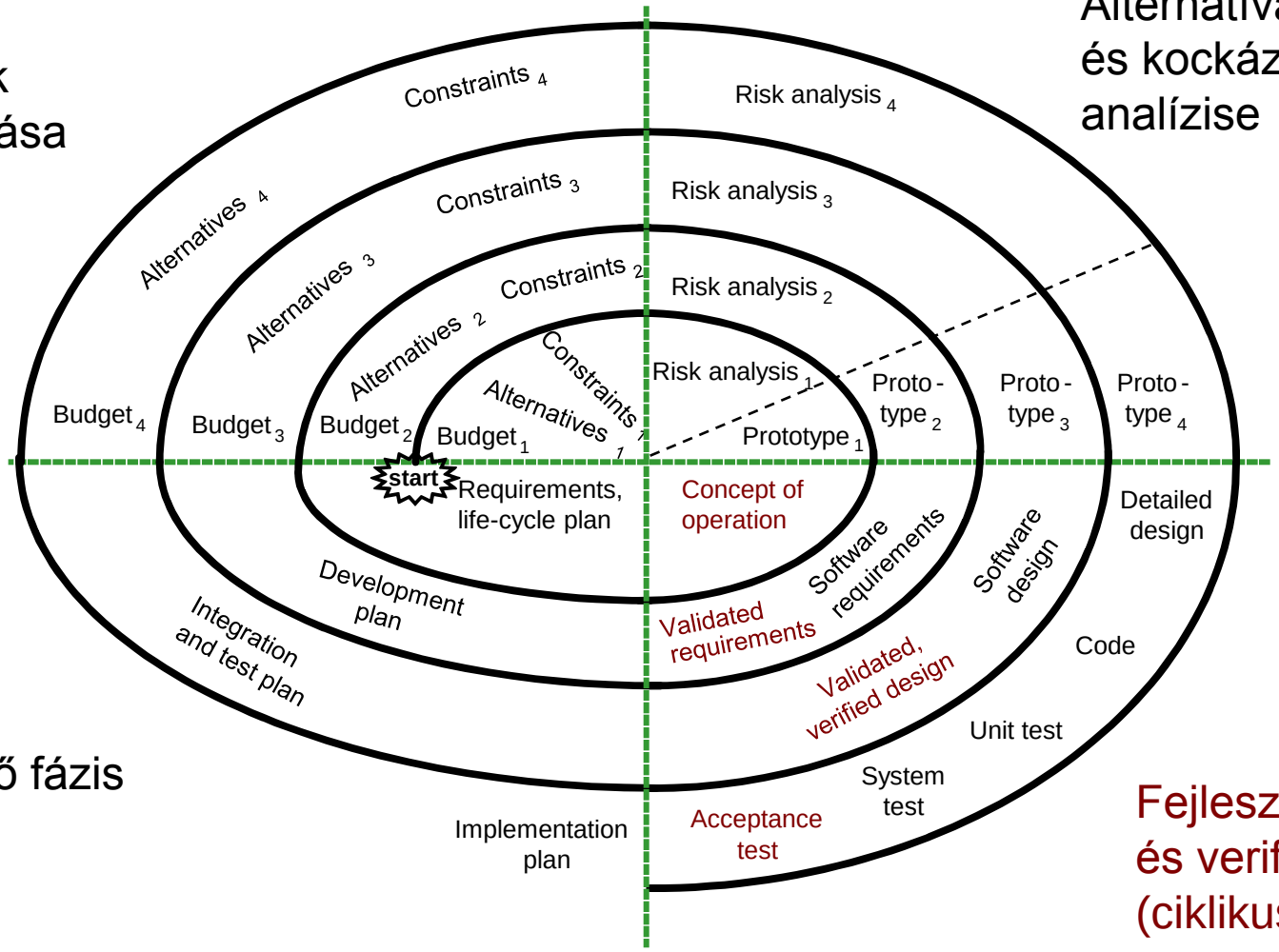
3. Evolúciós alkalmazásfejlesztés (RAD)

- Kezdeti verzió gyors fejlesztése, majd több verzión 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 beillesztése és tesztelése
  - Regressziós tesztelés módosítások után

4. Spirál modell

Célok, alternatívák, korlátozások meghatározása

Alternatívák és kockázatok analízise

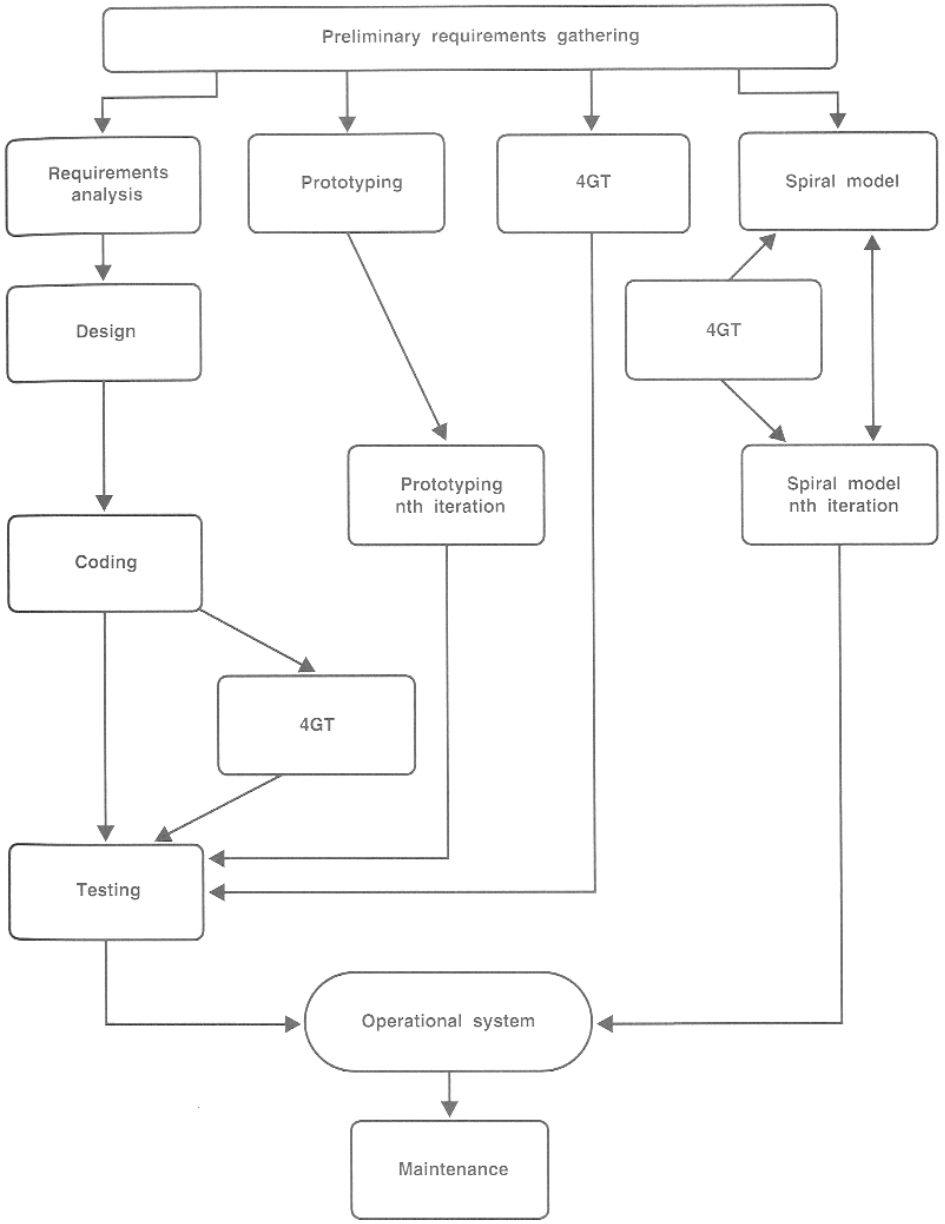


A következő fázis tervezése

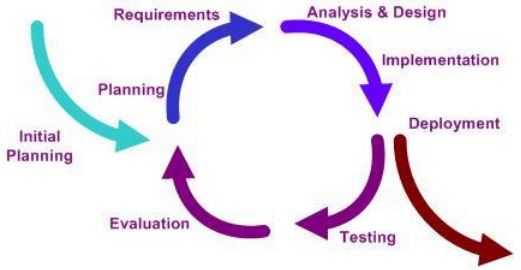
Fejlesztés és verifikáció (ciklikusan)

5. „Negyedik generációs” modell

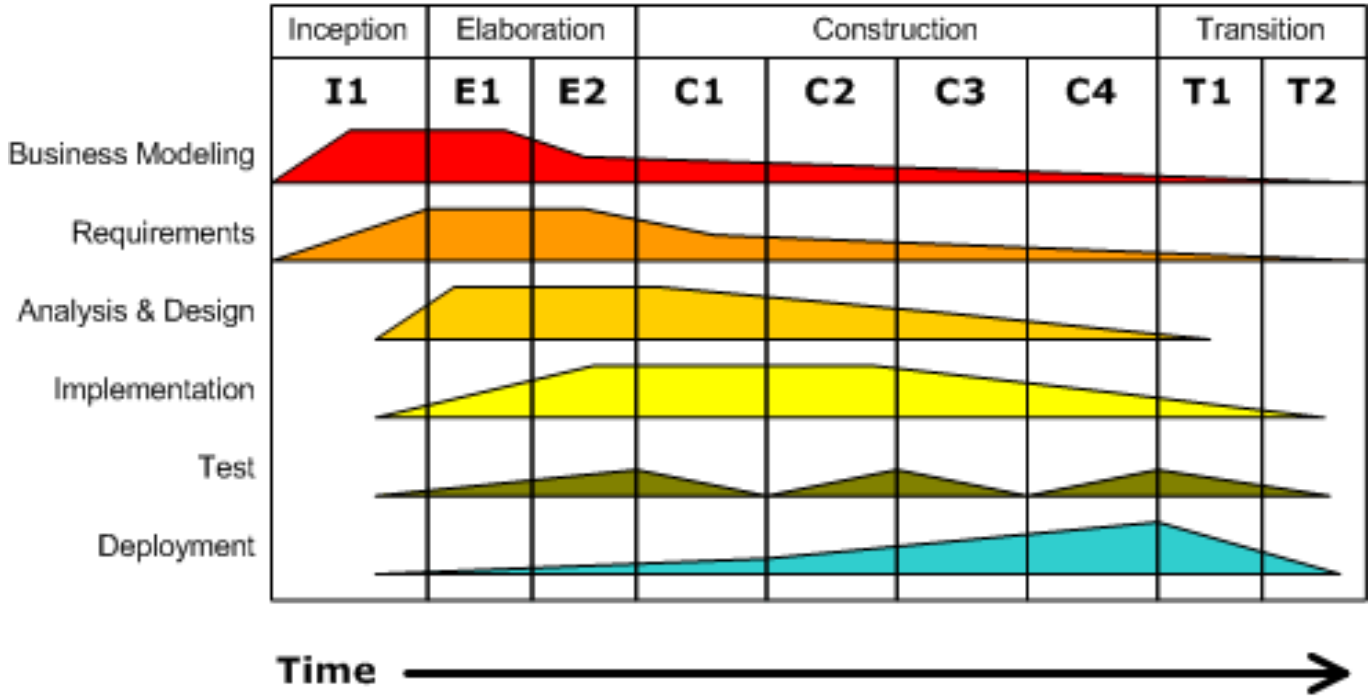
- Integrálás:
  - Jól megfogható követelmények: Hagyományos fejlesztés
  - Rosszul specifikált követelmények: Gyors prototípus fejlesztés
  - Formalizálható követelmények: Modell alapú fejlesztés (CASE eszközök), helyességmegőrző lépésekkel
  - Iteratív is lehet
- Modell alapú verifikáció



6. Unified Process



- Inkrementális és iteratív
  - Fázisok (időben kötött) iterációkra osztva
  - Az ellenőrzés fókusza az egyes fázisokban eltérő
    - Integrációs és regressziós tesztelés jelentős szerepet kap



Agilis szoftverfejlesztés

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

Áttekintés

- 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?