Telepítés

A rendszert elkészülte után a megrendelőnél üzembe kell helyezni

- A környezettel kapcsolatos feltételezések esetleg tévesek voltak;
- Az új rendszerrel szemben ellenállást tapasztalhatunk a befogadó oldalon;
- A rendszernek egy ideig esetleg együtt kell létezni más rendszerekkel;
- Fizikai problémák is felléphetnek a telepítés során (pl. kábelezési gondok);
- Az operátorok betanításról gondoskodni kell.

A rendszer evolúciója

Nagy rendszerek hosszú élettartamúak. Lépést kell tartani a változó követelményekkel.

Az evolúció költséges!

- A változásokat technikai és üzleti szempontból is elemezni kell;
- Az alrendszerek egymásra hatása miatt nem várt problémák adódhatnak;
- Ritkán ismertek az eredeti tervezési megfontolások;
- A rendszer struktúrája sérül a folyamatos változtatások során.

Azon régebbi rendszer, amelynek fenntartása elengedhetetlen: **legacy rendszer**

Rendszerek leépítése

A rendszer működésből való kivonása annak hasznos élettartama után.

Szükséges lehet veszélyes, vagy környezetszennyező anyagok ártalmatlanítása.

- Már a rendszertervezés során ezt tervezni kell!

Szükség lehet adatkonverzióra más rendszerekben való felhasználás céljából.

Összefoglalás

Az ember-gép rendszerek tartalmaznak számítógépes hardvert, szoftvert, valamint emberi kezelőket. Célja valamilyen üzleti cél elérésének segítése.

A globális eredő tulajdonságok a rendszer egészének működését jellemzik, nem pedig valamelyik részegységét.

A rendszertervezés folyamata: specifikáció, tervezés, fejlesztés, integráció és tesztelés. A rendszerintegráció és tesztelés különösen kritikus.

Verifikáció és validáció

(tesztelés „tudományos néven”)

Bevezetés

Verifikáció:

„Jó minőségű terméket fejlesztünk?”
(*Jól fejlesztünk*?)

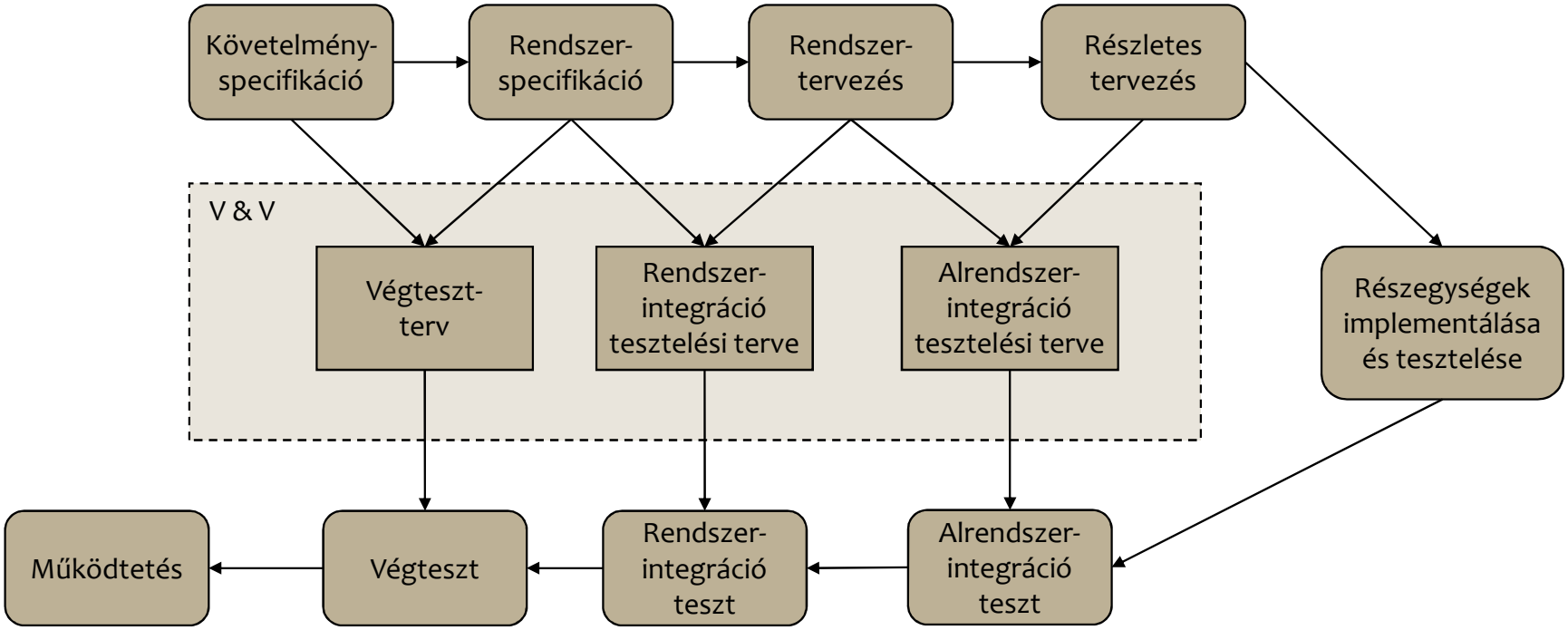
A szoftver teljesítse a specifikációt.

Validáció:

„A megfelelő terméket fejlesztjük?”
(*Jót fejlesztünk*?)

A szoftver azt csinálja, amit a felhasználó tényleg akar.

A rendszerfejlesztés V-modellje



A V & V eljárás

Az egész életciklusra jellemző

- A V & V-t a szoftver fejlesztés minden lépésénél alkalmazni kell.

Két fő feladat:

- A rendszerbeli hibák felfedezése.
- Annak felmérése, hogy a rendszer hasznos-e és használható-e a felhasználási környezetben.

A V & V célja

A verifikáció és validáció a szoftver iránti **bizalmi alap**ot teremt:

A szoftver el tudja látni a feladatát

NEM azt jelenti, hogy teljesen hibamentes.

Azt jelenti, hogy elég jó ahhoz, hogy ellássa feladatát.

(A feladat típusa határozza meg, milyen mértékű bizalom kell.)

A bizalmi szint

A rendszer céljától, a felhasználók elvárásaitól, valamint a piaci viszonyoktól függ:

- a szoftver célja
  - A bizalmi szint függ attól, hogy mennyire kritikus a szoftver a szervezet számára.
- felhasználó elvárások
  - A felhasználóknak bizonyos szoftverekkel szemben nagyon alacsony elvárásaik vannak.
- piaci környezet
  - A gyors piacra dobás fontosabb lehet, mint a hibák megtalálása.

A V & V tervezése

A tesztelési és vizsgálati eljárások sikere érdekében körültekintő tervezésre van szükség.

A tervezést már a fejlesztés korai fázisában el kell kezdeni.

A terv határozza meg a statikus vizsgálat és a tesztelés helyes egyensúlyát.

A V & V tervezése a tesztelési eljárás irányelveit fogalmazza meg, nem kell a termék tesztelését itt leírni.

A V&V terv

A tesztelő eljárás.

A tesztelési eljárás főbb fázisainak leírása.

Követelmények követhetősége.

A teszteket úgy kell megtervezni, hogy minden követelményt külön lehessen tesztelni.

Tesztelt elemek.

A fejlesztési eljárás azon elemeit specifikáljuk, amelyeket tesztelni kell.

A tesztelés menetrendje.

A tesztelés menetrendje a szükséges erőforrások foglalásával. Természetesen szorosan kapcsolódik a teljes projekt ütemezéséhez.

A tesztek rögzítésének eljárása.

A teszteket nemcsak futtatni kell, hanem az eredményeket szisztematikusan rögzíteni is. A tesztelési eljárásnak felülvizsgálhatónak kell lenni.

Hardver és szoftver szükségletek.

A szükséges szoftver eszközök listája a becsült hardver használattal együtt.

Kényszerek.

A tesztelési eljárást befolyásoló kényszerek, pl. munkaerő hiány.

A V&V két alapvető típusa

Szoftvervizsgálatok – statikus V&V

- Problémák feltárása a rendszer statikus reprezentációjának analízise segítségével.

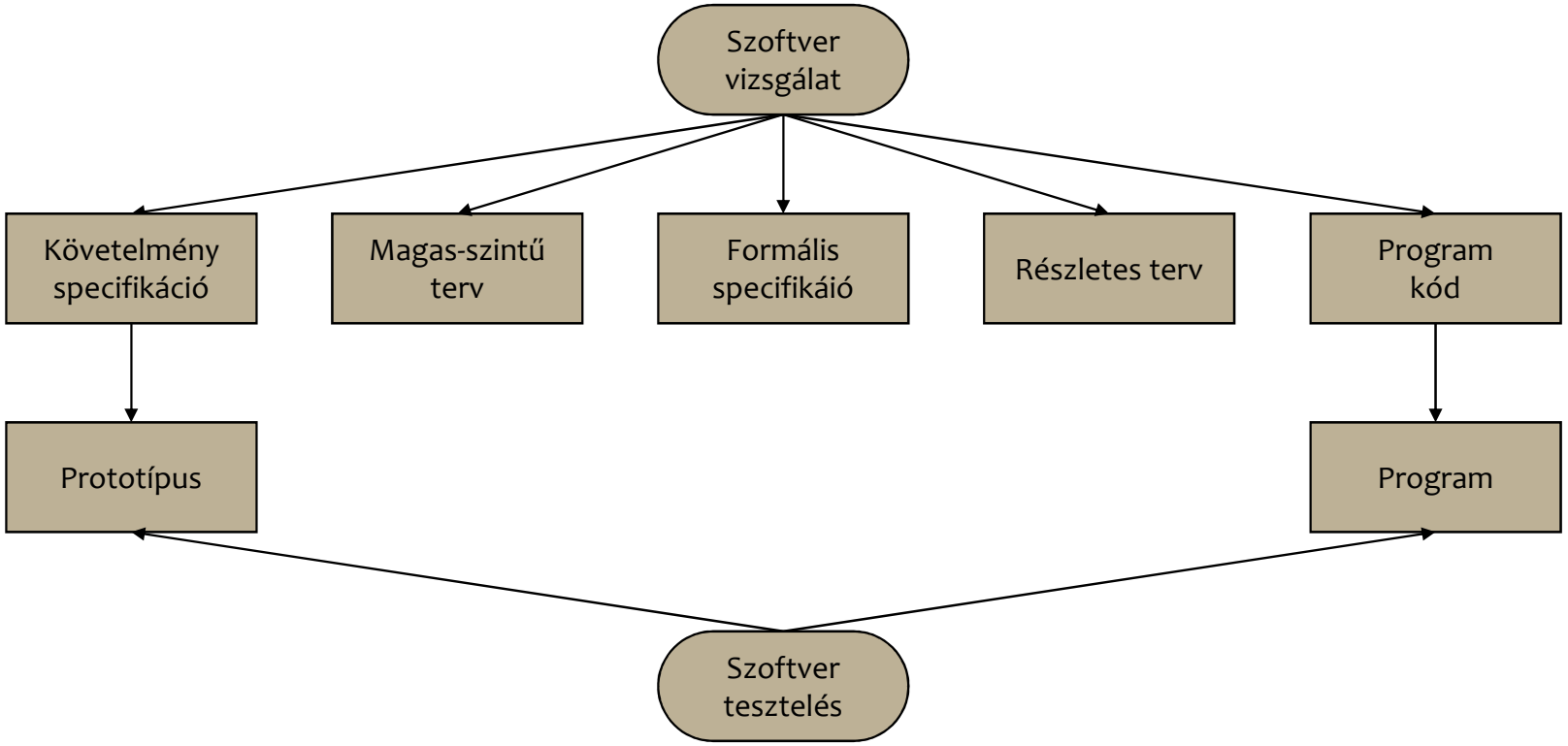
(Kiegészíthető eszköz-alapú dokumentum- és forráskód-analízissel.)

Szoftvertesztelés – dinamikus V&V

- Kísérletezés és a termék viselkedésének megfigyelése.

(A rendszert teszt-adatokkal futtatva működés közben figyeljük a viselkedését.)

Statikus és dinamikus V&V



Összefoglalás

A verifikáció és validáció nem ugyanazt jelenti.

A verifikáció a specifikáció teljesítését mutatja, a validáció pedig azt, hogy a program kielégíti a felhasználó igényeit.

A tesztelési eljárást tesztelési tervek segítik, ezek elkészítése szükséges.

A statikus verifikációs technikák hibadetektálás céljából vizsgálják és analizálják a programot.

A hibák felderítésének nagyon hasznos módja a szoftvervizsgálat.

A vizsgálat során hibakeresés céljából a programkódot egy kis létszámú csoport szisztematikusan átvizsgálja.

Szoftvervizsgálatok

(statikus V&V)

Bevezetés

Emberek (gépek) vizsgálják a forrás valamilyen reprezentációját anomáliák és hibák után kutatva.

A vizsgálathoz nem kell a rendszert futtatni, így implementáció előtt is megtehető.

A rendszer bármely reprezentációja vizsgálható: követelmények, terv, konfigurációs adatok, teszt adatok, stb.

Bevezetés

A reprezentáció felülvizsgálatának formális módszere

Célja kizárólag a hibák jelenlétének jelzése, (nem pedig javítása).

A hibák lehetnek (pl.)

- logikai hibák;
- anomáliák a kódban, amik hibás állapotot jelezhetnek (pl. nem inicializált változó);
- egyes szabványok nem teljesítése.

A szoftvervizsgálat típusai

Dokumentum átvilágítás

- Követelmények
  - Felhasználói
  - Rendszer
- Tervek
  - Rendszer, alrendszer, modul
  - Teszt

Program (forráskód) átvilágítás

Automatizált forráskód elemzés

Formális verifikáció

A vizsgálat előfeltételei

Precíz, teljes reprezentáció.

A csoport tagjainak ismerni kell a szervezet működési szabályait.

Szintaktikailag helyes kód, vagy valamilyen más rendszer-reprezentáció.

Egy hiba-ellenőrző listát kell készíteni.

A menedzsmentnek el kell fogadni, hogy a szoftver vizsgálat növeli a költségeket.

A menedzsment ne használja a szoftver vizsgálatot a dolgozók értékelésére (pl. ki hibázott).

A vizsgálat hatékonysága

Egyetlen vizsgálat több hibát is feltárhat. A tesztelés során egy hiba elfedhet más hibákat, így ott többszöri végrehajtás kell.

Az újrafelhasználás és a programozói tapasztalat miatt a felülvizsgálók valószínűleg találkoztak már a gyakran előforduló hibákkal.

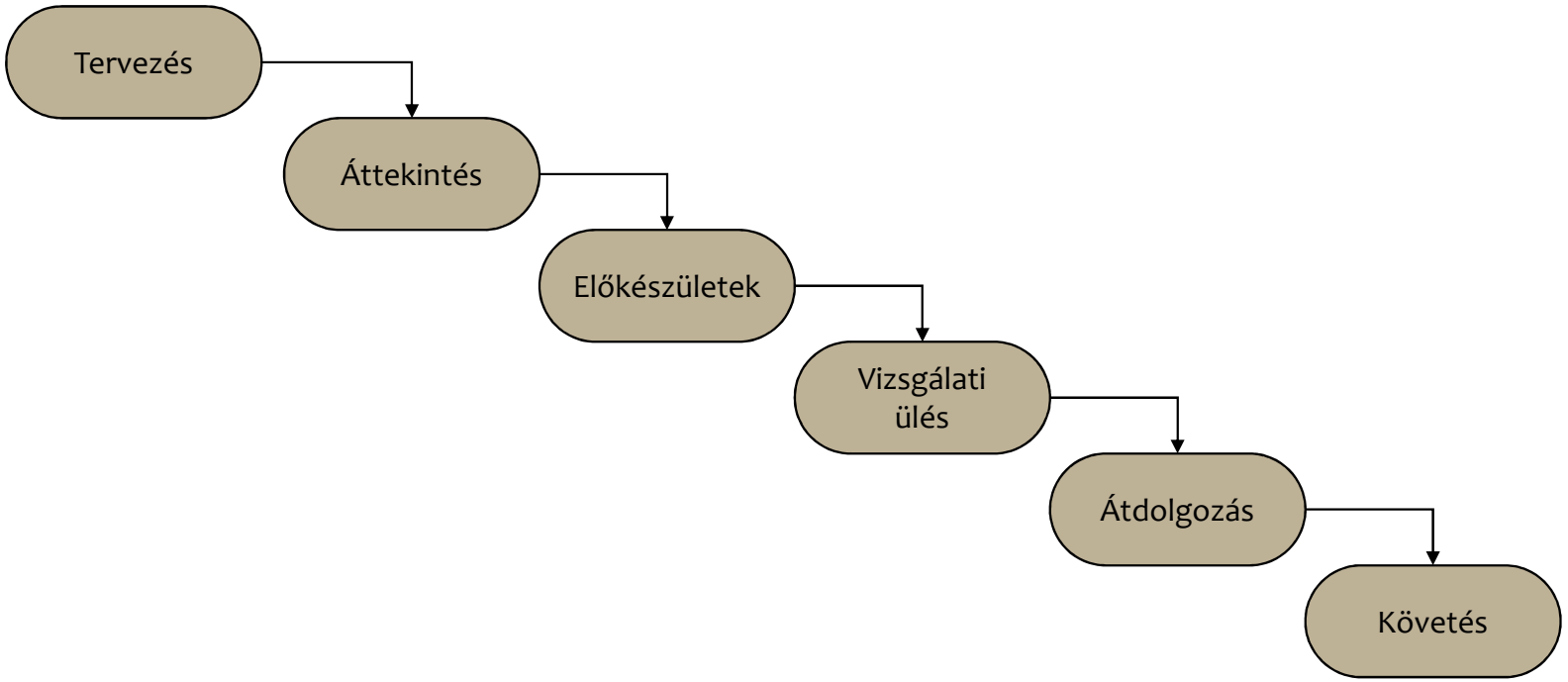
Szoftver vizsgálatok és szoftver tesztelés

A vizsgálatok és a tesztelés egymást kiegészítő verifikációs technikák.

A vizsgálat ellenőrzi, hogy a specifikációnak megfelel-e, de azt nem, hogy a valós felhasználói igényeket kielégíti-e.

A vizsgálatok nem tudják ellenőrizni a nem-funkcionális jellemzőket, pl. teljesítmény, használhatóság, stb.

A vizsgálat folyamata



A vizsgálat folyamata

A rendszer ismertetése a felülvizsgáló csoport számára.

A csoport tagjai megkapják a kódot és az egyéb kapcsolódó dokumentumokat.

A vizsgálat megtörténik, a felfedezett hibákat feljegyzik.

A felfedezett hibák javítása érdekében a szükséges módosítások elvégzése.

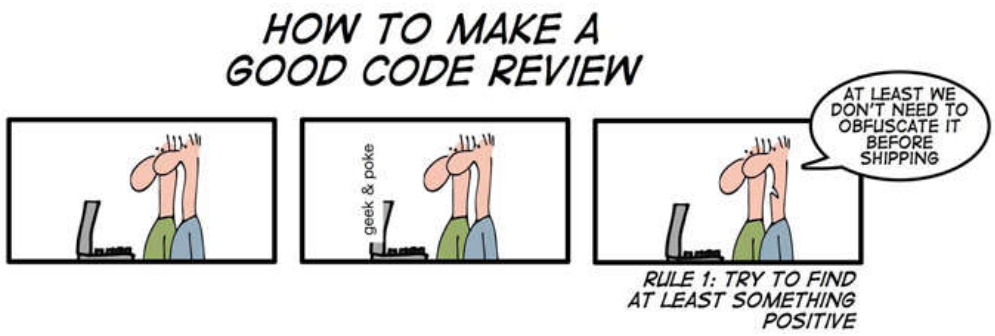
Szükség esetén újabb vizsgálat.

Szerepek a vizsgálat során

Szerző vagy tulajdonos	A programozó vagy tervező, aki a program vagy egyéb dokumentum létrehozásáért felelős. Az ő felelőssége a vizsgálat során feltárt hibák javítása is.
Vizsgáló	Hibák, hiányosságok, inkonzisztenciák keresése a programokban és dokumentációban. Esetleg a vizsgáló bizottság feladatkörén kívül álló kérdéseket is felvethet.
Felolvasó	A vizsgálati ülésen bemutatja a kódot vagy dokumentumot.
Írnok	A vizsgálati ülés eredményeit rögzíti.
Elnök vagy moderátor	Menedzseli és segíti a vizsgálat menetét. Az eredményeket a fő moderátornak jelenti.
Fő moderátor	A vizsgálati folyamat javításáért, a hiba-ellenőrző lista frissítéséért, eszközökért, stb. felelős.

A vizsgálat típusai

Code review – egy vagy több (tapasztaltabb) programozó átnézi a kódot



Páros programozás – egyik programozó fejleszt, a másik „csak” figyel

Refactoring – már tesztelt, működő kód szerkezetének javítása

A vizsgálat típusai (folyt.)

Technical review

- teljesítmény optimalizálásra vagy nehezen megtalálható hibák felfedésére
- külső szakértői átvizsgálás / minősítés valamely tulajdonságra (pl. MEI, TÜV)

Inspekció

- szoftver teljes átvizsgálása (nem csak valamely tulajdonságra vagy funkcióra)
- külső szakértő
- hosszú idő (több hónap is lehet)

Ellenőrző listák

A gyakori hibákat tartalmazó ellenőrző lista használandó a vizsgálat levezetésére.

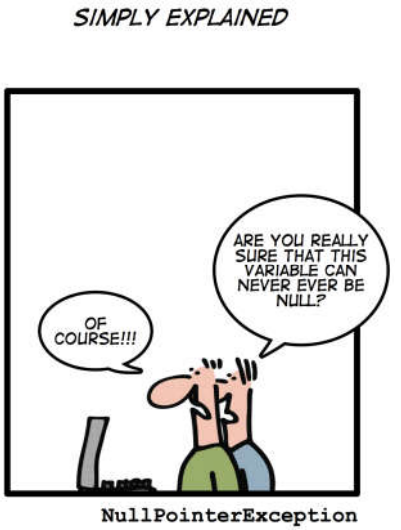
A hiba-ellenőrző listák programnyelv-specifikusak és az adott programnyelv karakterisztikus hibáit tartalmazzák.

Általában minél gyengébb a típus-ellenőrzés, annál hosszabb az ellenőrző lista.

Példák: inicializálás, konstansok elnevezése, kilépés hurokból, tömbhatár túllépés, stb.

Ellenőrző lista 1

Adathibák	Minden változó inicializálva van, mielőtt használnánk?
	Az összes konstansnak van neve?
	A tömbök felső indexe a tömb méretével egyenlő vagy ennél eggyel kisebb kell legyen?
	Karakter-tömbök használata esetén a delimiter egyértelműen definiálva van?
	Előfordulhat-e buffer overflow?
Vezérlési hibák	Minden feltételes utasításra: helyes a feltétel?
	Minden ciklus biztosan befejeződik?
	Az utasítás-blokkokat helyesen zárójeleztük?
	Case utasításnál minden lehetőséget kimerítettünk?
	Ha minden case utasítás után break kell, akkor ezek jelen vannak?
I/O hibák	Minden bemenő változót használunk?
	Minden kimeneti változónak adunk értéket visszatérés előtt?
	Váratlan bementi adatok okozhatnak-e hibát?



Ellenőrző lista 2

Interfész hibák	Minden függvény- és metódus-hívásnak megfelelő számú paramétere van?
	A paraméter-típusok megfelelők?
	A paraméterek sorrendje megfelelő?
	Ha több komponens osztott memóriát használ, akkor ugyanolyan struktúrájú memória-modellt használnak?
Tárolás-menedzsment hibák	Láncolt szerkezetek módosítása esetén minden mutató megfelelően módosítva van?
	Dinamikus memóriahasználat esetén helyes-e az allokáció?
	A nem használt memória explicit módon fel van-e szabadítva?
Kivétel-kezelési hibák	Minden lehetséges hibalehetőség figyelembe lett véve?

A vizsgálat sebessége

500 utasítás/óra az áttekintés során.

125 forrás utasítás/óra az egyéni előkészületek alatt.

90-125 utasítás/óra vizsgálható az ülésen.

A vizsgálat drága!

Pl.: 500 sor megvizsgálása kb. 40 ember óra igényű, ami kb. 500 eFt.

Így is olcsóbb lehet, mint a komponens tesztelés.

Szoftvertesztelés

(dinamikus V&V)

Szoftvertesztelés

A hibák jelenlétét és NEM hiányát jelezheti.

Az egyetlen V & V technika nem-funkcionális követelmények ellenőrzésére, hiszen a szoftvert végre kell hajtani ahhoz, hogy lássuk, miként viselkedik.

Statikus ellenőrzéssel együtt célszerű használni, hogy teljes V&V lefedettséget kapjunk.

Tesztelési tanácsok

Csak a kimerítő teszteléssel deríthető ki, hogy a program hibamentes. De a kimerítő tesztelés lehetetlen.

A tesztelési vezérelvek definiálják a tesztek kiválasztásának módját. Pl.:

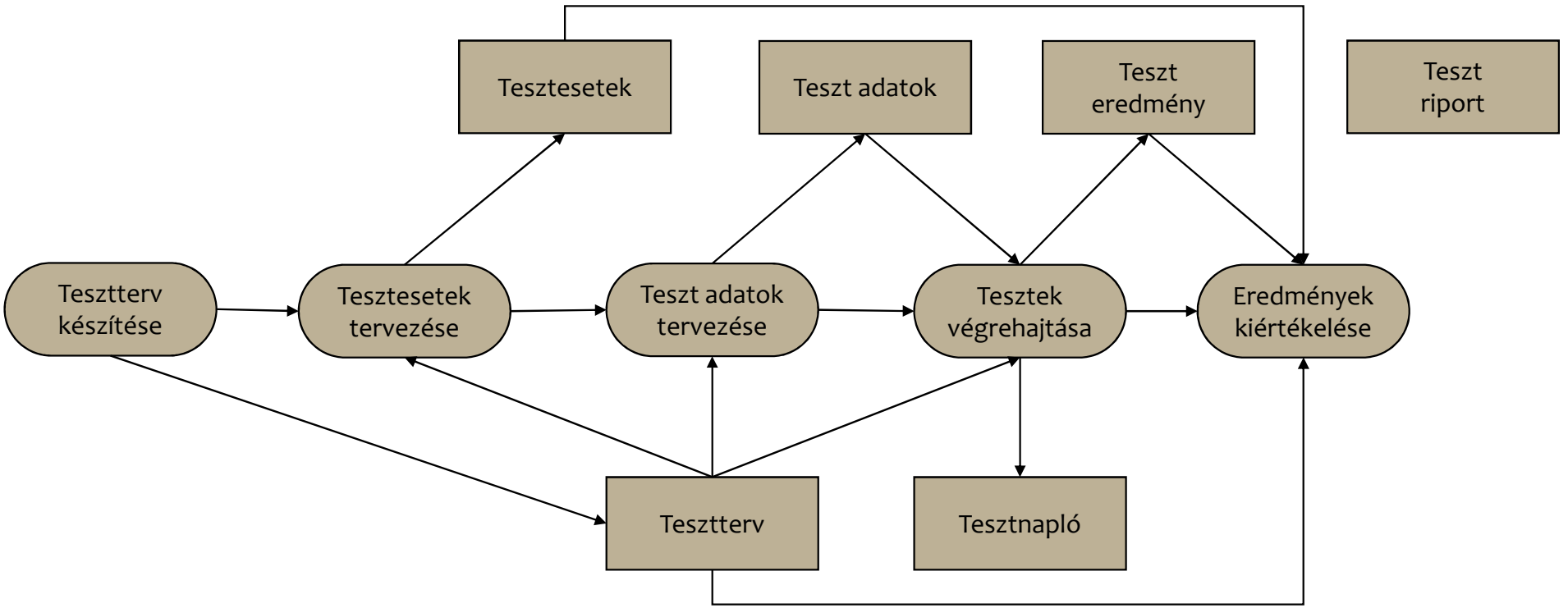
- A menükön keresztül elérhető valamennyi funkciót le kell tesztelni;
- Az azonos menün keresztül elérhető funkciók kombinációit tesztelni kell;
- Ahol felhasználói bevitel van, minden funkciót ellenőrizni kell helyes és helytelen adatokkal.

Tesztelési tanácsok (folyt.)

Ötletek a tesztelőknek: hogyan érdemes olyan teszteket választani, amelyek kimutatják a rendszer hibáit.

- Olyan bemenetek választása, amelyek a rendszert hibaüzenetek (az összes!) generálására kényszerítik;
- Olyan bemenetek tervezése, ami puffer túlcsorduláshoz vezethet;
- Ugyanazon bemenet vagy bemeneti sorozat többszöri ismétlése;
- Érvénytelen kimenetek kikényszerítése;
- Túl nagy vagy túl kicsi számítási eredmények kikényszerítése.

A szoftvertesztelés folyamata



A szoftvertesztelés szintjei

Komponensteszt (fejlesztő)

- A teljes rendszer egy-egy komponensét teszteli önmagában

Integrációs teszt (fejlesztő)

- Komponensek közötti együttműködést teszteli

Rendszerteszt (független csoport)

- A teljes rendszert, azaz minden komponenst együtt tesztel

Átvételi teszt (független csoport)

- Felhasználói teszt, a már kész rendszeren

A szoftvertesztelés technikái

Fekete dobozos (black-box)

- specifikáció alapú tesztelés
- a tesztelő csak a specifikációt ismeri, a forráskódot nem

Fehér dobozos (white boksz)

- kód (struktúra) alapú tesztelés
- a tesztelő a specifikációt és a forráskódot is ismeri