

Szoftver modul/unit tesztelés

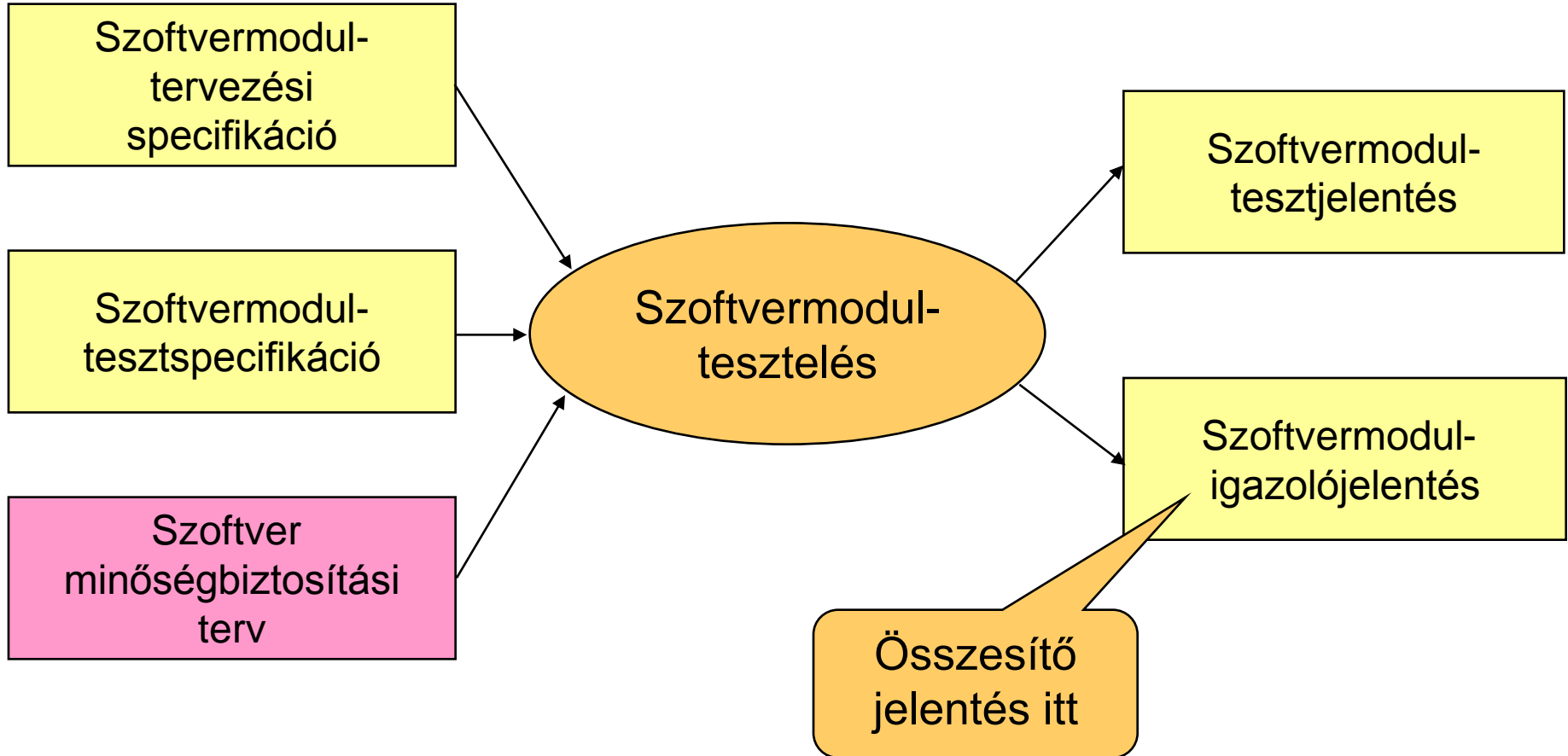
Majzik István, Micskei Zoltán

Budapesti Műszaki és Gazdaságtudományi Egyetem

Méréstechnika és Információs Rendszerek Tanszék

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

Szoftvermodul tesztelés



Példa: Szabványok előírásai (EN 50128)

- Funkcionális és fekete doboz tesztelés (HR)
 - Határérték elemzés (HR)
 - Ekvivalencia osztályok és bemeneti adatfelosztás (HR)
 - Ok-okozati diagramok
 - Folyamatszimuláció
 - Prototípus készítés / animáció
- Teljesítménytesztelés
 - Lavina / stressz tesztelés (SIL 1 R -> SIL 3 HR)
 - Válaszidő- és memória-kikötések tesztelése (HR)
 - Teljesítmény követelmények tesztelése (HR)
- Interfész-tesztelés (HR)
- Adatrögzítés és elemzés (HR)
- Strukturált alapú tesztelés (SIL1 R -> SIL3 HR)

Szoftvermodul tesztelési célok

Tesztelés:

- A program olyan futtatása, hogy a hibák kiderüljenek

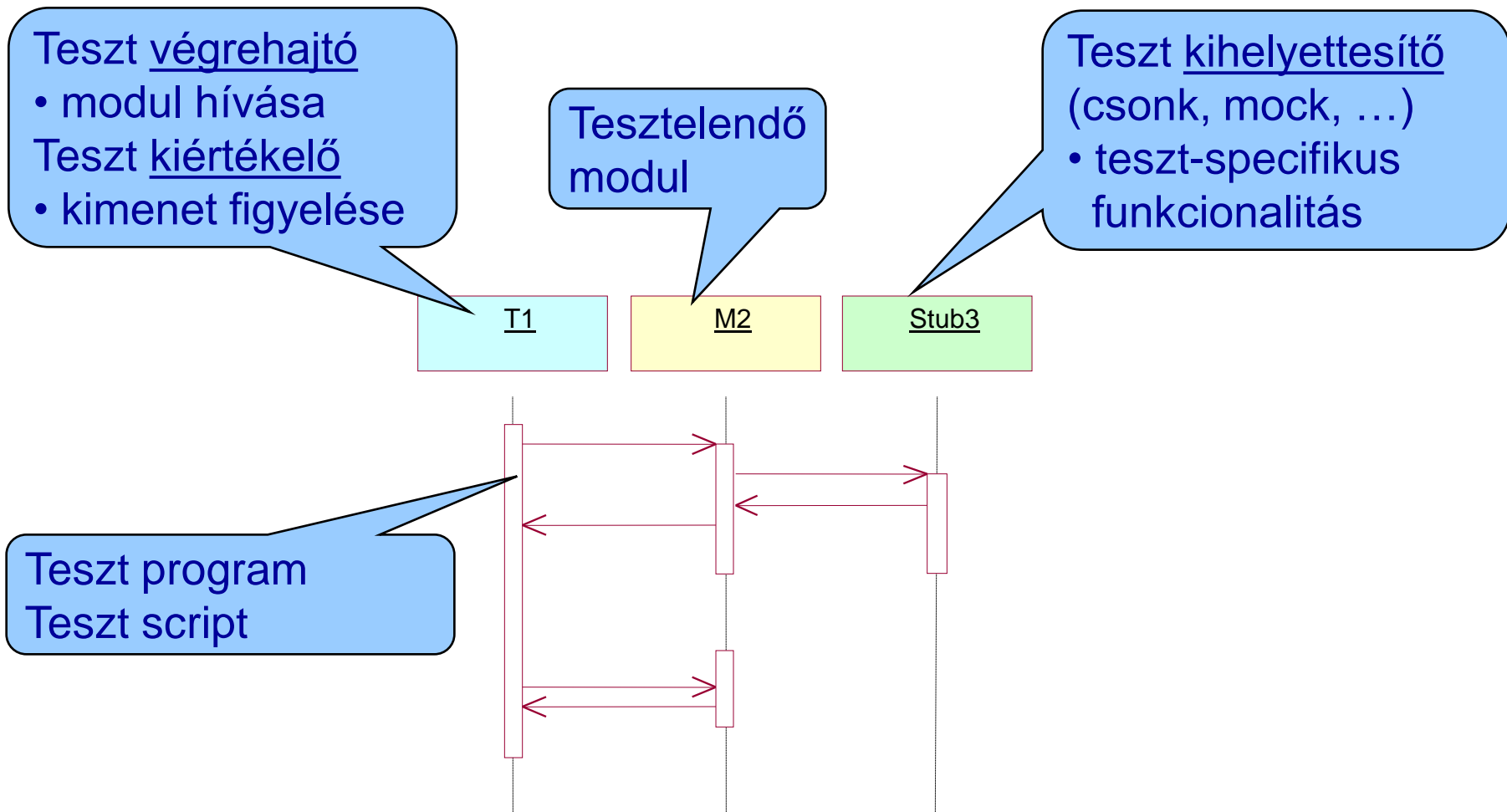
Kimerítő tesztelés:

- Minden lehetséges futást kipróbál
- Gyakorlatban nem kivitelezhető

Mottók:

- Dijkstra: A tesztelés csak hibák jelenlétét, és nem hibamentességet tud kimutatni.
- Hoare: A tesztelés egy induktív bizonyítás része: Ha a program jól működik *adott teszt adat(ok)ra*, akkor várhatóan *hasonló adatokra* is jól működik.

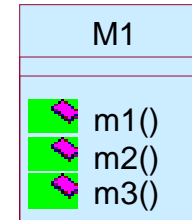
Tesztelési környezet: Egy modul tesztelése



Teszttervezés módszerei

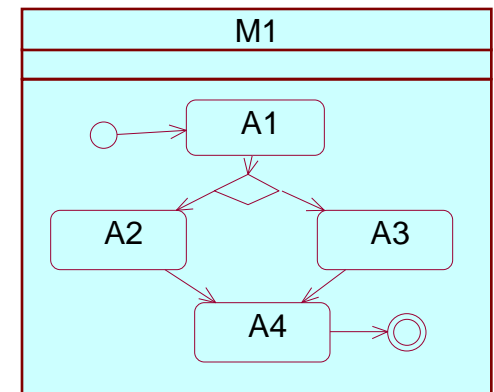
I. Specifikáció alapú tesztelés

- A rendszer mint „fekete doboz” adott
- Csak a külső viselkedés (funkció) ismert, a belső felépítés (pl. forráskód) nem
- Tesztelés alapja: **specifikált funkciók** léte; extra funkciók hiánya



II. Struktúra alapú tesztelés

- A rendszer mint „üvegdoboz” adott
- A belső struktúra is ismert
- Tesztelés alapja a belső működés: programgráf bejárása



I. Specifikáció alapú tesztelési módszerek

Cél:

- A funkcionális specifikációra építve,
- reprezentatív adatok keresése az egyes funkciók teszteléséhez.

Módszerek:

1. Ekvivalencia particionálás
2. Határérték-analízis
3. Ok-hatás analízis / Döntési táblák
4. Kombinatorikus módszerek
5. Véges automata alapú módszerek
6. Használati eset tesztelés

1. Ekvivalencia particionálás

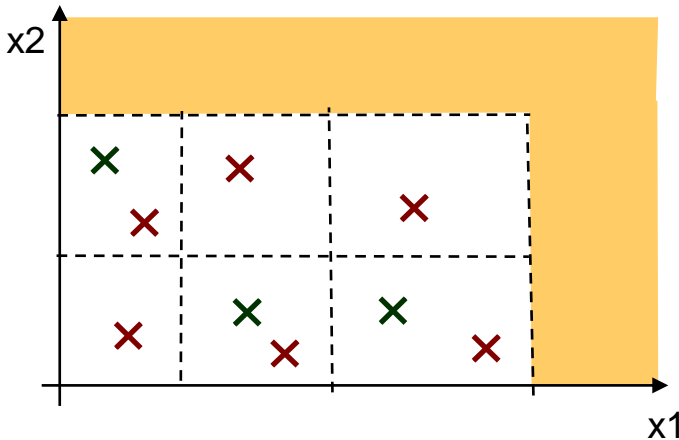
- **Bemenet és kimenet ekvivalencia osztályai:**
 - Adatok, amelyek várhatóan ugyanazt a hibát fedik le (ugyanazt a programrészt járják be)
 - Cél: **Egy-egy** ekvivalencia osztályból **egy-egy** teszt adat (az adott bemenethez illetve kimenet alapján); a többi adat esetén a helyesség „induktívan következik”
- **Bemenet/kimenet szerepét ismerni kell**
 - A tesztelő tudásán múlik a módszer hatékonysága
- **Meghatározás heurisztikus folyamat:**
 - **Egyező** funkcionalitást biztosító adatok, hasonló eredmények
 - **Érvényes és érvénytelen** adatok

Példa: NextDate program ekvivalencia osztályok

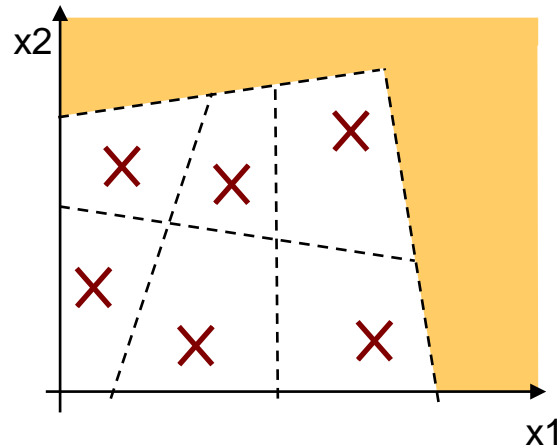
Bemenet	Érvényes	Érvénytelen
Hónap	V1: 30 napos hónap V2: 31 napos hónap V3: február	I1: ≥ 13 I2: ≤ 0 I3: nem szám I4: üres
Nap	V4: 1-30 V5: 1-31 V6: 1-28 V7: 1-29	I5: ≥ 32 I6: ≤ 0 I7: nem szám I8: üres
Év	V8: 1582-9999 V9: nem szökőév V10: szökőév V11: század nem szökőév V12: század szökőév	I9: ≤ 1581 I10: ≥ 9999 I11: nem szám I12: üres
Speciális	V13: 1752.09.03-1752.09.13.	I13: 1582.10.5-1582.10.14.

Példa: Gyenge és erős ekvivalencia osztályok

- Tesztek meghatározása több bemenet esetén:
 - **Érvényes** ekvivalencia osztályok:
egy teszt minél több osztályt fedjen le
 - **Érvénytelen** ekvivalencia osztályok:
először minden érvénytelen osztályhoz külön teszt legyen
(egymás hatását ne oltsák ki), majd több osztály kombinációja
- **Erős és gyenge ekvivalencia osztályok:**



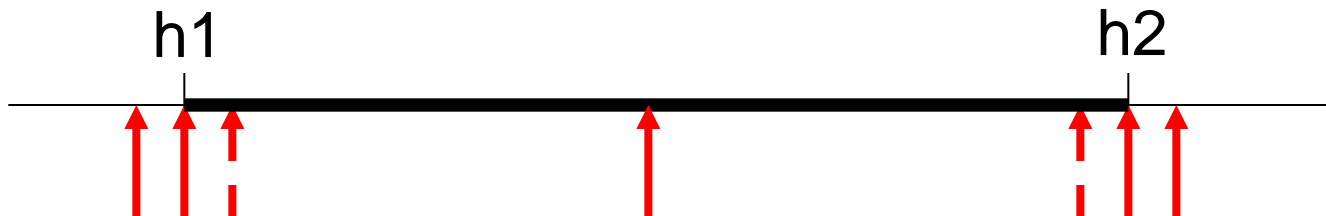
- Gyenge normál
- Erős normál



- Dimenzióként nem független: **Erős** osztályok

2. Határérték-analízis

- Az adattartományok határait vizsgálja
 - Egy-egy ekvivalencia osztály **hatáira** koncentrálnak
 - **Bemeneti és kimeneti** tartományokra is
 - Alsó és felső határokra
- Tipikus megtalált hibák
 - Hibás relációs operátorok
 - Hibák a ciklusok be- és kilépési feltételeinél
 - Hibák az adatstruktúrák méreténél
- Tipikus adatok:
 - Egy határérték 3 tesztet jelent, egy tartomány 5-7 tesztet jelent



3. Ok-hatás analízis

A bemenetek és kimenetek kapcsolatának vizsgálata, ha az egyszerűen leírható

- **Ok:** egy-egy bemeneti ekvivalencia osztály
- **Hatás:** egy-egy kimeneti ekvivalencia osztály
- Ezekből logikai változókat képzünk

Boole-gráf: Okok és hatások összekapcsolása

- ÉS, VAGY kapcsolatok
- Meg nem engedett kombinációk

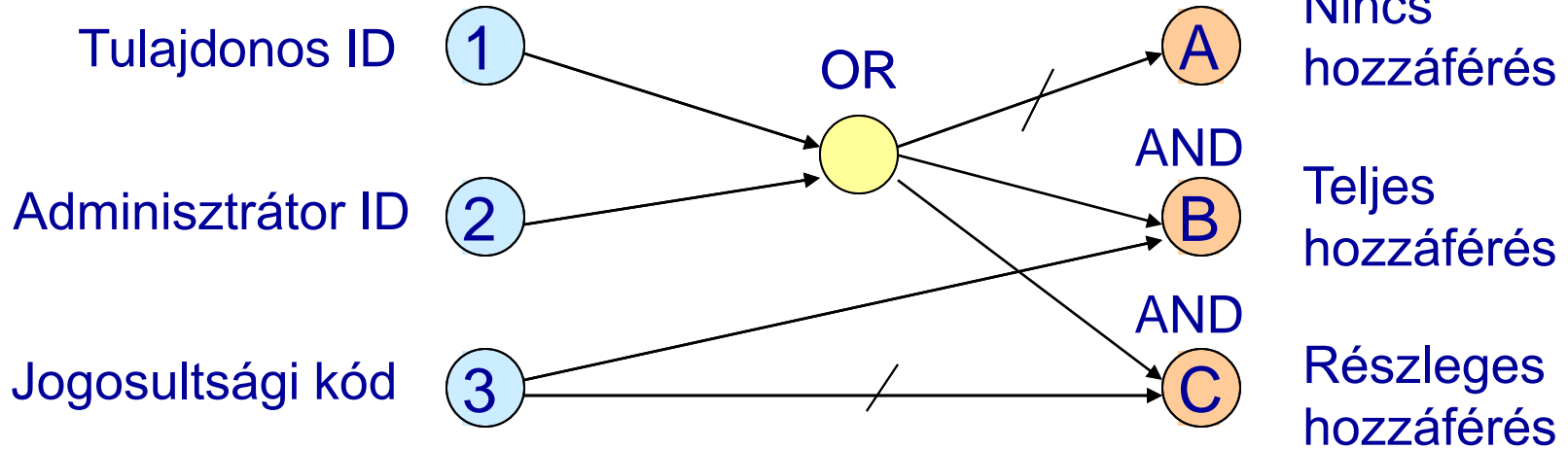
Tesztelési cél: A gráf szisztematikus végigjárása

- Logikai hálózat igazságtáblázatának lefedése
- Egy oszlop egy tesztnek felel meg

Példa: Ok-hatás leírása

Bemenetek:

Kimenetek:



	T1	T2	T3	...	
Bemenetek	1	0	1	0	
	2	1	0	0	
	3	1	1	1	
Kimenetek	A	0	0	1	
	B	1	1	0	
	C	0	0	0	

4. Kombinatorikus módszerek

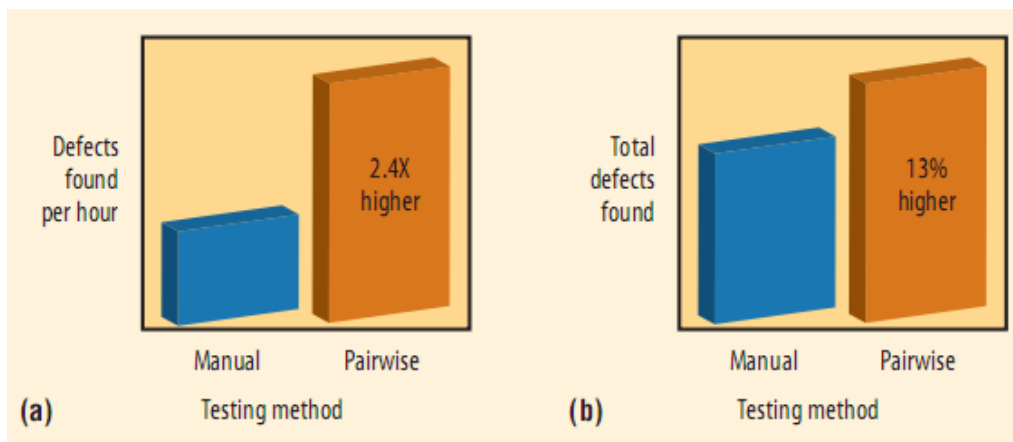
Cél: Paraméterek kombinációjának tesztelése

- Paraméterek kombinációja okozza a legtöbb hibát
- Ritka kombinációk veszélyesek lehetnek
- Ad hoc, „best guess” tesztelés
 - Intuíció, követelmények, tipikus hibák alapján
- Minden választás (each choice)
 - Minden lehetőség szerepeljen egyszer (alapkészlet)
- N-szeres tesztelés (n-wise testing)
 - Tetszőlegesen választott n darab paraméter **minden lehetséges kombinációjának** lefedése a tesztelési cél
 - Speciális eset ($n=2$): **Páronkénti tesztelés**
 - Eszközök: Pl. <http://www.pairwise.org>

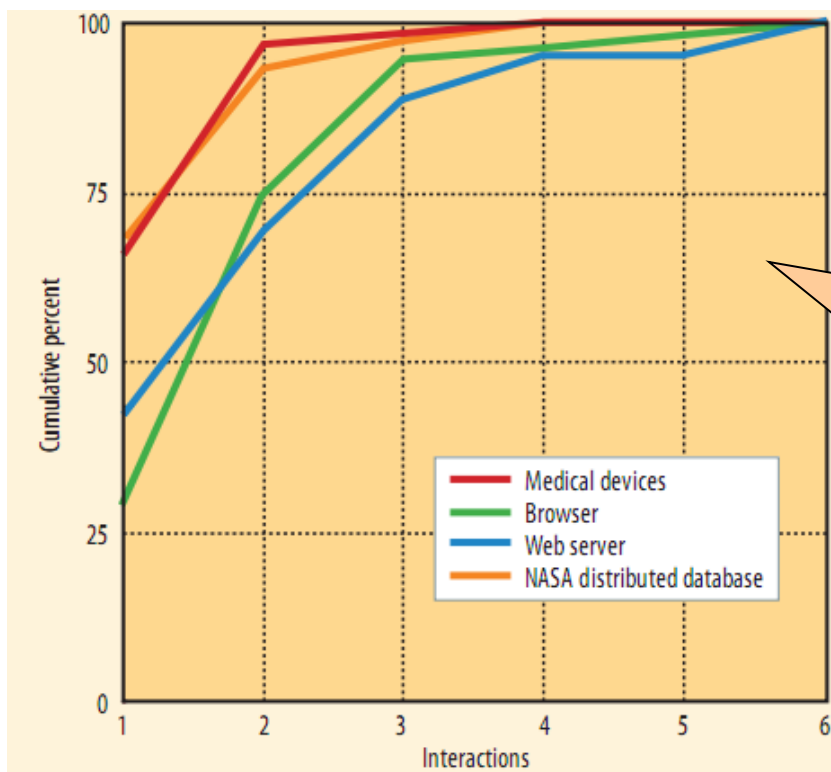
Példa: Pair-wise tesztelés

- Adottak a következő konfigurációs lehetőségek:
 - OS: Windows, Linux
 - CPU: Intel, AMD,
 - Protocol: IPv4, IPv6
- Minden választás: Hány kombináció van?
- Páronkénti tesztelés: Milyen tesztkészlet elég?
- Lehetséges megoldás:
 - 1: Windows, Intel, IPv4
 - 2: Windows, AMD, IPv6
 - 3: Linux, Intel, IPv6
 - 4: Linux, AMD, IPv4

Példa: N-wise testing hatékonysága



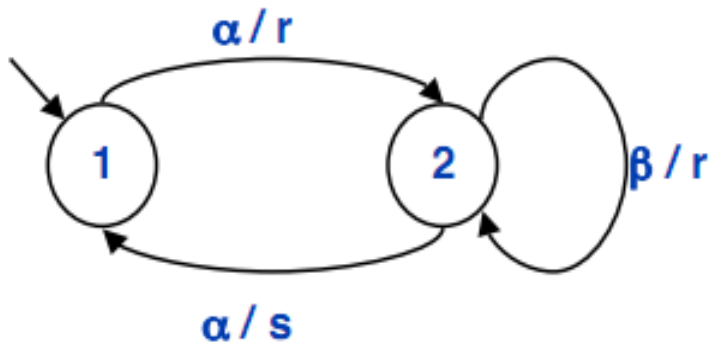
Ad-hoc és páronként szisztematikus tesztelés összehasonlítása (10 projektre)



A hibák jelentős része 2 paraméter kapcsolatán múlik (de alkalmazástól függően lehet még elég sok hiba, ami 3 vagy több paraméter speciális kombinációja esetén deríthető ki)

5. Véges automata alapú tesztelés

- A **specifikáció** egy véges automatával adott
- Tipikus tesztelési célok:
 - Minden állapot, minden átmenet, nem megengedett átmenetek tesztelése, stb.



- **Problémák:**
 - Milyen állapotban van a rendszer?
 - Végállapot / kezdőállapot
- **Módszerek**
 - Automatikus tesztgenerálás (ld. később)

6. Használati eset tesztelés

- Tesztek származtathatók a **használati esetekből**
- Tesztesetek:
 - Fő ág („happy path”, „mainstream”)
 - Ellenőrzés: utófeltételek vizsgálata
 - Alternatív lefutások: mindegyikhez külön teszteset
 - Előfeltételek (nem)teljesülése
- Tipikusan integrációs és elfogadási tesztek része

Módszerek együttes alkalmazása

Alap módszerek tipikus sorrendje:

1. Ekvivalencia particionálás
2. Határérték-analízis
3. Ok-hatás analízis, vagy kombinatorikus, vagy véges automata alapú

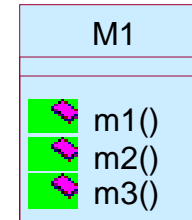
Kiegészítés: Véletlen tesztek

- Véletlen teszt adatok generálása
- Kis számítási teljesítményt igényel, gyors
- Hibafedése nem garantálható
- Teszt eredmény kiértékelése a nehezebb:
 - Válasz számítása, szimulálása
 - Csak „elfogadhatósági vizsgálat” (durva hibák kiszűrése)

Teszttervezés módszerei

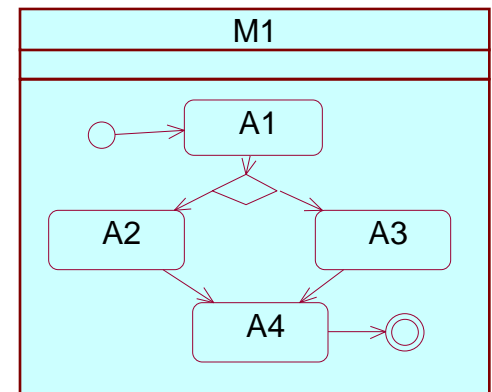
I. Specifikáció alapú tesztelés

- A rendszer mint „fekete doboz” adott
- Csak a külső viselkedés (funkció) ismert, a belső felépítés (pl. forráskód) nem
- Tesztelés alapja: specifikált funkciók léte; extra funkciók hiánya



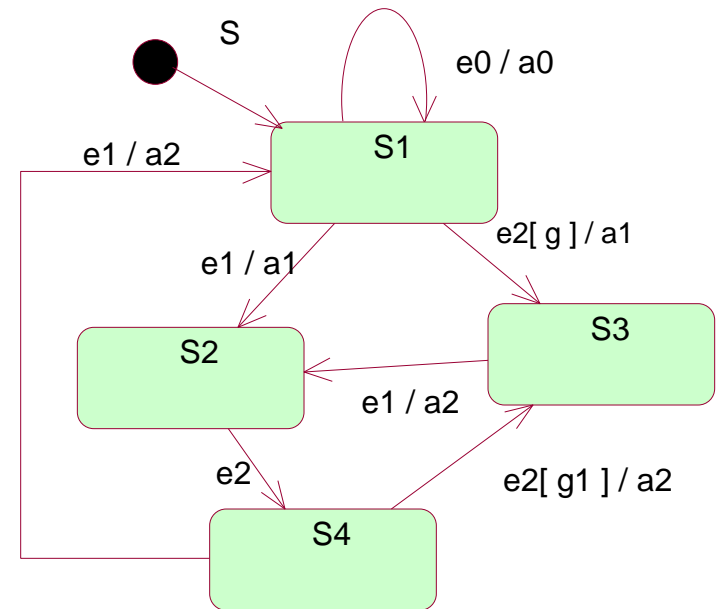
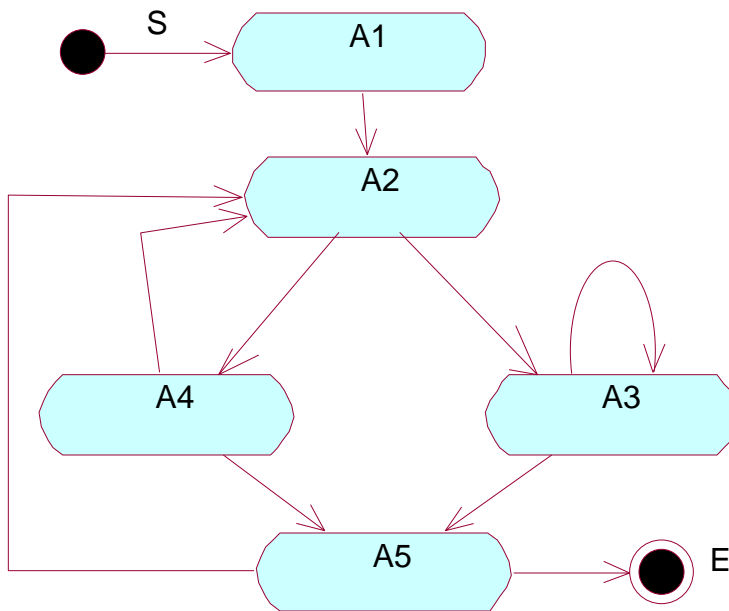
II. Struktúra alapú tesztelés

- A rendszer mint „üvegdoboz” adott
- A belső struktúra is ismert
- Tesztelés alapja a **belső működés**: programgráf bejárása



A belső struktúra

- Jól kezelhető struktúra:
 - **Modell alapján:** pl. aktivitás diagram, állapottérkép diagram



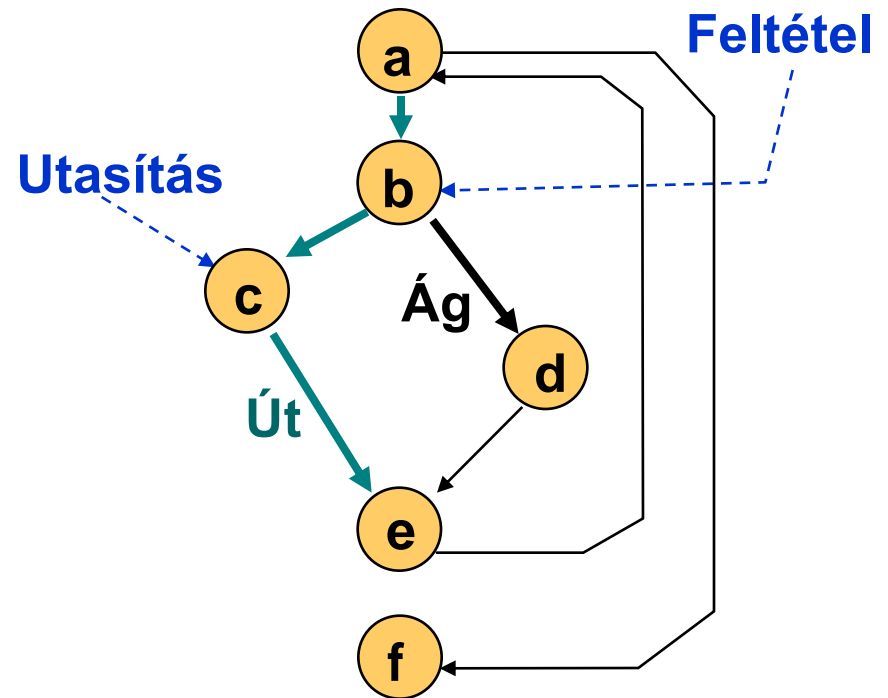
A belső struktúra

- Jól kezelhető struktúra:
 - Modell alapján: pl. aktivitás diagram, állapottérkép diagram
 - **Forráskód alapján:** vezérlési gráf (programgráf)

Forráskód:

```
a: for (i=0; i<MAX; i++) {  
b:   if (i==a) {  
c:     n=n-i;  
      } else {  
d:       m=n-i;  
      }  
e:   printf(“%d\n”,n);  
      }  
f:   printf(“Ready.”)
```

Vezérlési gráf:



Tesztminőségi mértékszámok

A tesztelés hatékonyságának számszerű jellemzése:

A tesztelhető elemek mekkora részét teszteltük, pl.

- | | |
|----------------------|------------------------|
| 1. Utasítások | → Utasítás fedettség |
| 2. Döntési ágak | → Döntési ág fedettség |
| 3. Feltételek | → Feltétel fedettség |
| 4. Végrehajtási utak | → Út fedettség |

Ez nem a hibafedés!

Szabványok előírása lehet (DO-178B, MSZ EN 50128,...)

- 100% utasítás fedettség általában alapkövetelmény

Mértékszámok (kritériumok) áttekintése

- Vezérlési folyamat alapú kritériumok
 - Utasítás lefedettség
 - Döntési ág lefedettség
 - Feltétel lefedettségek
 - Útvonal lefedettség
- Adatfolyam alapú kritériumok
 - Definiálás – használat fedettségek
 - Definíciómentes útvonalak fedettsége
- Módszerek kombinációja

Alapfogalmak

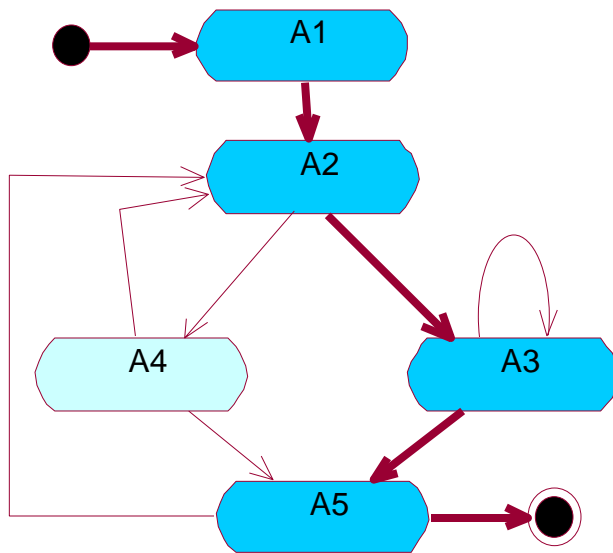
- Utasítás (statement)
- Blokk (block)
 - Utasítások egybefüggő sorozata, amik között nincs elágazás vagy függvényhívás
- Feltétel (condition)
 - Egyszerű vizsgálat, amiben nincs logikai (Boole) operátor
- Döntés (decision)
 - Egy ág bejárásának eldöntéséhez tartozó, nulla vagy több logikai operátorral összekötött feltételből álló kifejezés
- Út (path)
 - Utasítások sorozata, tipikusan a modul be és kilépési pontja között

1. Utasítás lefedettség (Statement coverage)

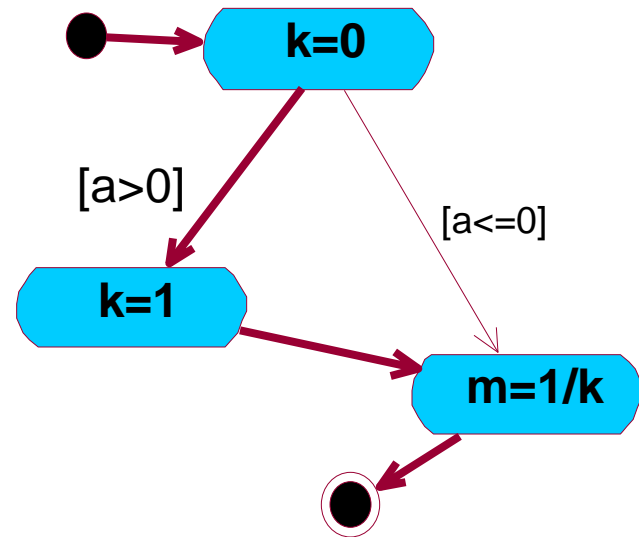
Definíció:

$$\frac{\text{Tesztelés során végrehajtott utasítások száma}}{\text{Összes utasítás száma}}$$

Utasítások kihagyási feltételeit nem veszi figyelembe!



Utasítás fedettség: 80%



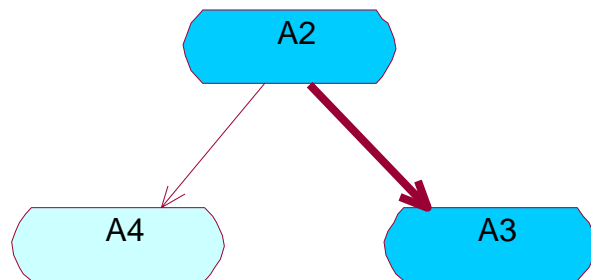
Utasítás fedettség: 100%

2. Döntés lefedettség (Decision coverage)

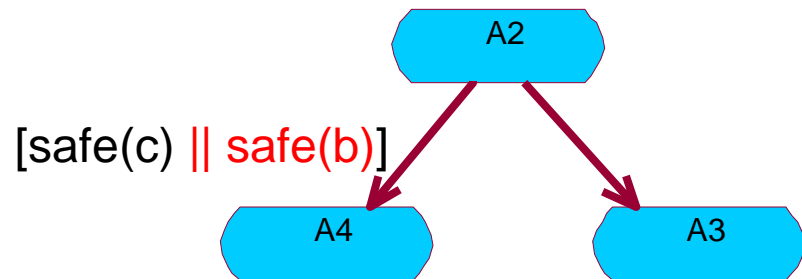
Definíció:

$$\frac{\text{Tesztelés során végrehajtott döntési ágak száma}}{\text{Összes lehetséges döntési ág száma}}$$

Nem vesz figyelembe minden feltétel-kombinációt!



Döntési ág fedettség: 50%



Döntési ág fedettség: 100%

3. Feltétel lefedettség (Condition coverage)

Generikus fedettségi mérték feltétel fedettségekhez:

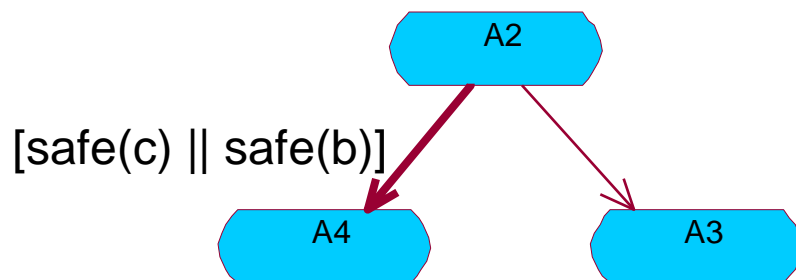
$$\frac{\text{Feltételek **tesztelt** kombinációinak száma}}{\text{Feltételek **megcélzott** kombinációinak száma}}$$

Megcélzott kombinációk feltétel lefedettség esetén:

- Minden feltétel legyen **igaz és hamis beállítású is** a tesztelés során
 - Nem feltétlenül eredményez 100% döntés lefedettséget!

Példa 100%-os feltétel lefedettséghez:

1. `safe(c) = true, safe(b) = false`
2. `safe(c) = false, safe(b) = true`



Másféle definíció:

- Minden feltétel legyen **igaz és hamis kiértékelésű** is
 - Ez nem ugyanaz mint a fenti, a lusta kiértékelés miatt

4. Feltétel/döntés lefedettség

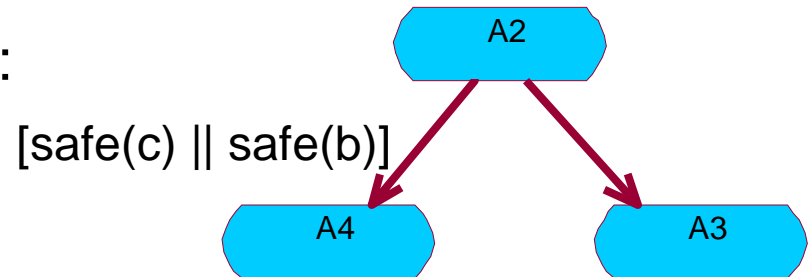
Condition/Decision Coverage (C/DC)

Definíció (megcélzott kombinációk):

- Minden feltétel felveszi az összes lehetséges kimenetét legalább egyszer, és
- minden döntés felveszi az összes lehetséges kimenetét egyszer

Példa 100%-os C/DC lefedettséghez:

1. `safe(c) = true, safe(b) = true`
2. `safe(c) = false, safe(b) = false`



Nem vizsgálja meg, hogy a feltételnek tényleg van-e hatása a döntés eredményére!

5. Módosított feltétel/döntés lefedettség

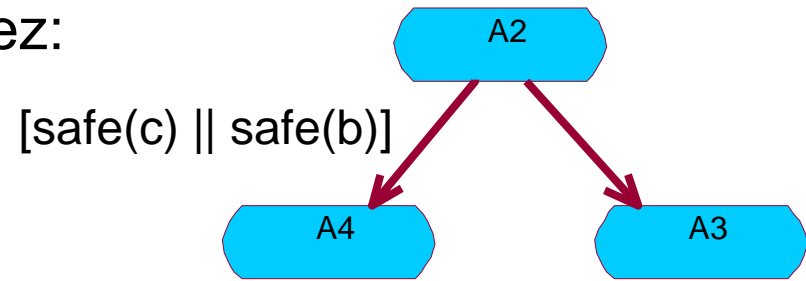
Modified Condition/Decision Coverage (MC/DC)

Definíció (megcélzott kombinációk):

- Minden feltétel felveszi az összes lehetséges kimenetét legalább egyszer, és
- minden döntés felveszi az összes lehetséges kimenetét egyszer, és
- minden feltétel a többtől függetlenül befolyásolja a hozzá tartozó döntés kimenetelét

Példa 100%-os MC/DC lefedettséghez:

1. $\text{safe}(c) = \text{true}$, $\text{safe}(b) = \text{false}$
2. $\text{safe}(c) = \text{false}$, $\text{safe}(b) = \text{true}$
3. $\text{safe}(c) = \text{false}$, $\text{safe}(b) = \text{false}$



6. Minden feltétel kombináció lefedettsége

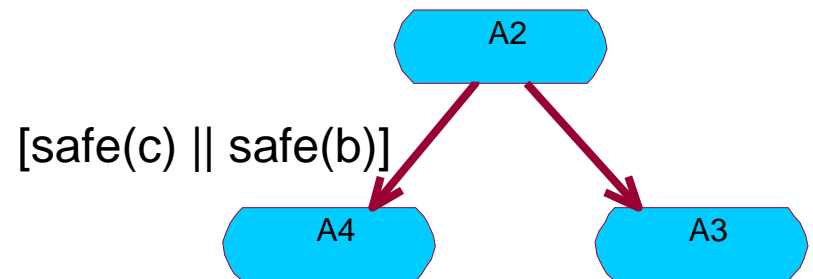
Multiple Condition Coverage

Definíció (megcélzott kombinációk):

- A feltételek kimeneteinek minden lehetséges kombinációja bekövetkezzen a tesztelés során
 - Általában a feltételek számával exponenciálisan növekedő számú teszt szükséges
 - Kevesebb, ha a lusta kiértékelést is figyelembe vesszük

Példa 100%-os feltétel kombináció lefedettséghez:

1. $\text{safe}(c) = \text{true}$, $\text{safe}(b) = \text{false}$
2. $\text{safe}(c) = \text{false}$, $\text{safe}(b) = \text{true}$
3. $\text{safe}(c) = \text{false}$, $\text{safe}(b) = \text{false}$
4. $\text{safe}(c) = \text{true}$, $\text{safe}(b) = \text{true}$



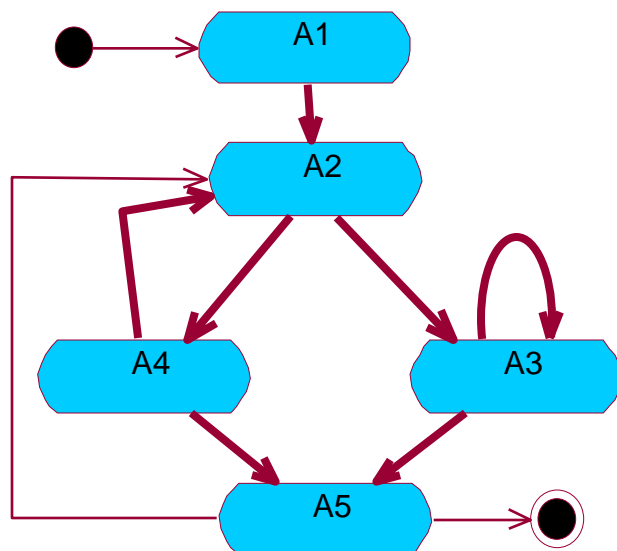
7. Alap út lefedettség (Basis path coverage)

Definíció:

Tesztelés során bejárt független utak száma
Összes független út száma

100% út lefedettség együtt jár:

- 100% utasítás lefedettség, 100% döntés lefedettség
- feltétel lefedettség nem garantált



Út lefedettség: 80%

Utasítás fedettség: 100%

Tesztelés az út fedettség alapján 1/2

- **Cél: Független utak bejárása**
 - Független utak a tesztelés szempontjából:
Van olyan utasítás vagy elágazás,
ami a másokban nincs meg
- **A független utak maximális száma:**
 - **CK**, ciklomatikus komplexitás
 - Szabályos vezérlési gráf alapján meghatározható
(vezérlési gráf összekötött, 1 kimenet és 1 bemenet):
 $CK(G)=E-N+2$, ahol
 - E**: élek száma
 - N**: csomópontok száma a **G** vezérlési gráfban
- **A független utak halmaza nem egyedi**

Tesztelés az út fedettség alapján 1/2

- **Cél: Független utak**

- Független utak
- Van olyan út ami a másikba nem

- **A független utak**

- **CK**, cikloma
- Szabályos vezé

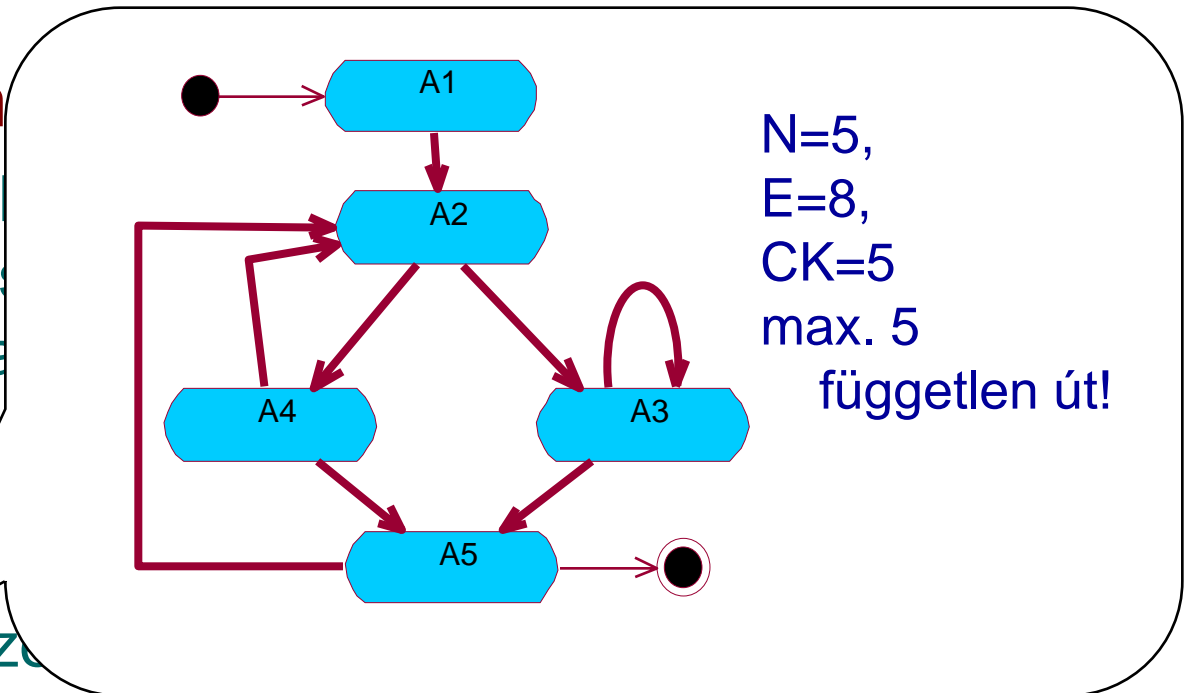
(vezérlési gráf összekötött, 1 kimenet és 1 bemenet):

$CK(G) = E - N + 2$, ahol

E: élek száma

N: csomópontok száma a **G** vezérlési gráfban

- **A független utak halmaza nem egyedi**



N=5,
E=8,
CK=5
max. 5
független út!

Tesztelés az út fedettség alapján 2/2

- Algoritmus:
 - Max. CK számú független út kiválasztása
 - Bemenetek generálása egy-egy út bejárásához
- Problémák:
 - Nem minden út bejárható (ld. feltételek).
 - Generálható-e az úthoz bemeneti szekvencia?
 - Lehetséges-e a belső változók beállítása?
 - Ciklusok: Korlátozni (minimalizálni) kell a bejárást!
- Teljesen automatikus módszerek nem léteznek
 - Szimbolikus végrehajtás: SMT megoldóval
 - Korlátok: Ciklusok, aritmetika, külső könyvtárak, ...

Kiegészítő fedettségi mértékek

- Loop
 - Ciklusok 0 (ha értelmezhető), 1, illetve többszöri végrehajtása
- Race
 - Több szál futása egyszerre egy-egy kódrészleten
- Relational operator
 - Határértékek használata összehasonlító operátorok esetén
- Weak mutation
 - Operátor vagy operandus hibákra tesztek készítése
- Table
 - Ugrási táblák (állapotgép megvalósítások) teljes tesztelése
- Linear code sequence and jump
 - Lineáris szekvenciák fedése a forráskódban (lehetnek benne vezérlési utasítások, de lineárisan bejárva)
- Object code branch
 - Gépi kódú feltételes ugrások fedése (fordító függő)

Példa: Vezérlési folyamat alapú kritériumok

```
Product getProduct(String name, Category cat) {  
    if (name == null || !cat.isValid)  
        throw new IllegalArgumentException();  
  
    Product p = ProductCache.getItem(name);  
  
    if (p == null) {  
        p = DAL.getProduct(name, cat);  
    }  
  
    return p;  
}
```

Cél: Utasítás fedettség, döntési ág fedettség, C/DC fedettség

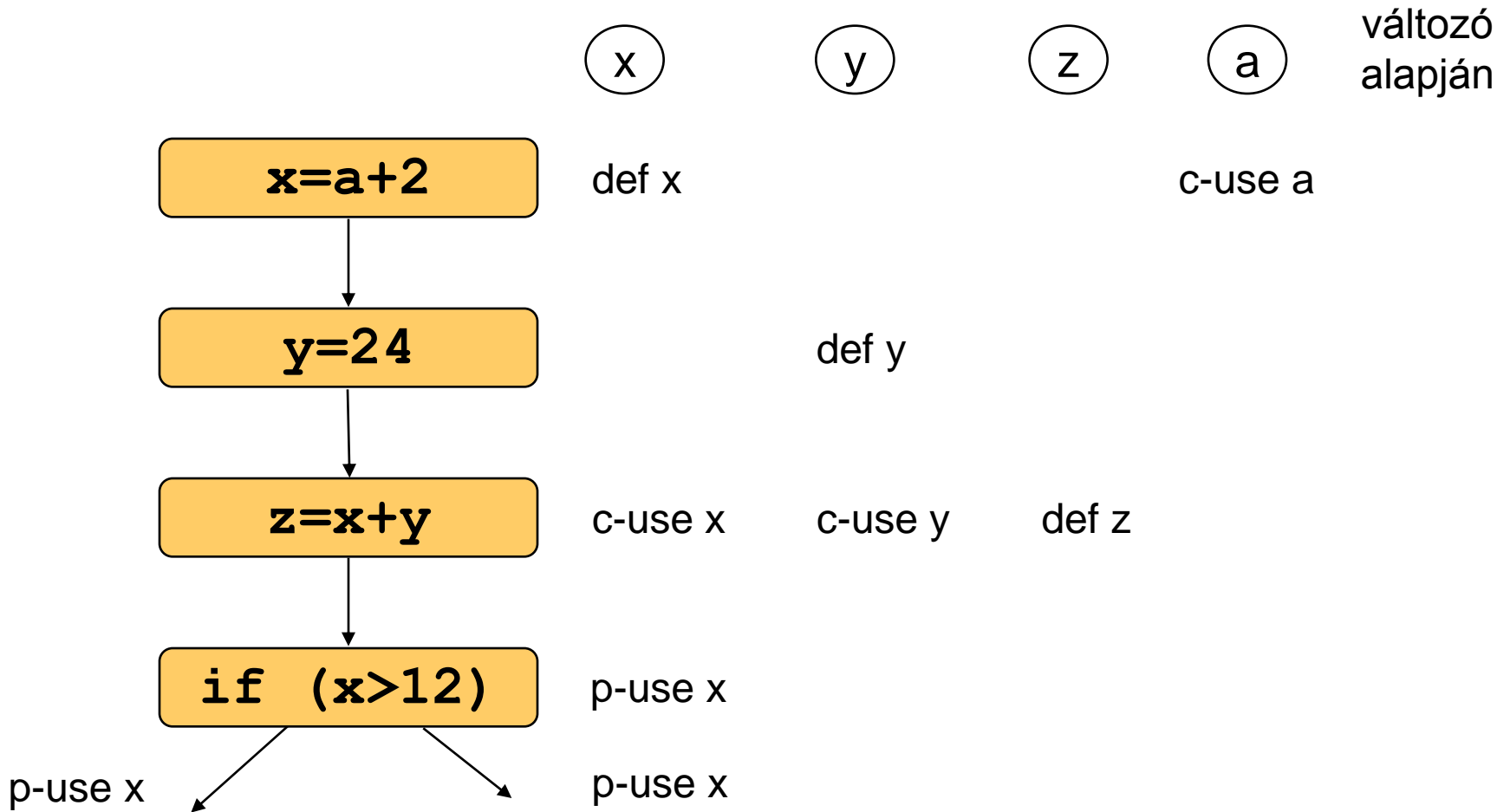
Mértékszámok (kritériumok) áttekintése

- Vezérlési folyamat alapú kritériumok
 - Utasítás lefedettség
 - Döntési ág lefedettség
 - Feltétel lefedettségek
 - Útvonal lefedettség
- Adatfolyam alapú kritériumok
 - Definiálás – használat fedettségek
 - Definíciómentes útvonalak fedettsége
- Módszerek kombinációja

Adatfolyam alapú tesztelés

- Cél: Változók értékadásának és az értékek felhasználásának vizsgálata a tesztelés során
 - Rossz értéket adtam? Rosszul használtam fel? Inicializálatlan?
- A programgráf címkézése:
 - `def v`: `v` változó értékadása
 - `c-use v`: `v` változó felhasználása számításban
 - `p-use v`: `v` változó felhasználása elágazási feltételben
- Útvonalak:
 - `def-clear v` útvonal: nincs benne `def v` címkéjű utasítás
 - `def-use v` (vagy rövidebben `d-u v`) útvonal:
 - `def v` címkéjű utasítással indul
 - `p-use v` vagy `c-use v` utasítással zárul
 - Közben `def-clear v` útvonal van
 - Nincs belső hurok (legfeljebb a teljes `d-u v` útvonal képez hurkot)

Példa a címkézésre

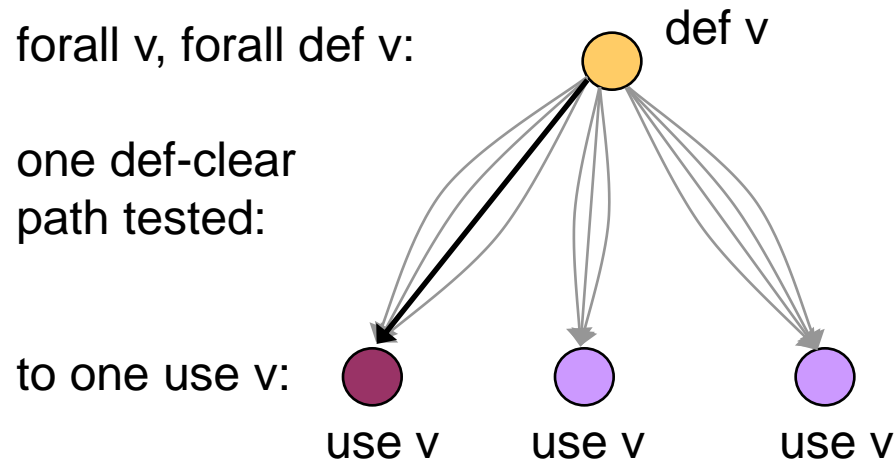


All-defs fedettségi kritérium

- All-defs:

Minden v változóra, minden **def v** utasításra:

Egy use v utasításig legalább **egy def-clear v** útvonal tesztelése
(itt **use v** lehet akár **p-use v**, akár **c-use v**)



All-p-uses, all-c-uses kritériumok

- All-p-uses:

Minden v változóra, minden **def v** utasításra:

Minden p-use v utasításig legalább **egy def-clear v** útvonal tesztelése

- All-c-uses:

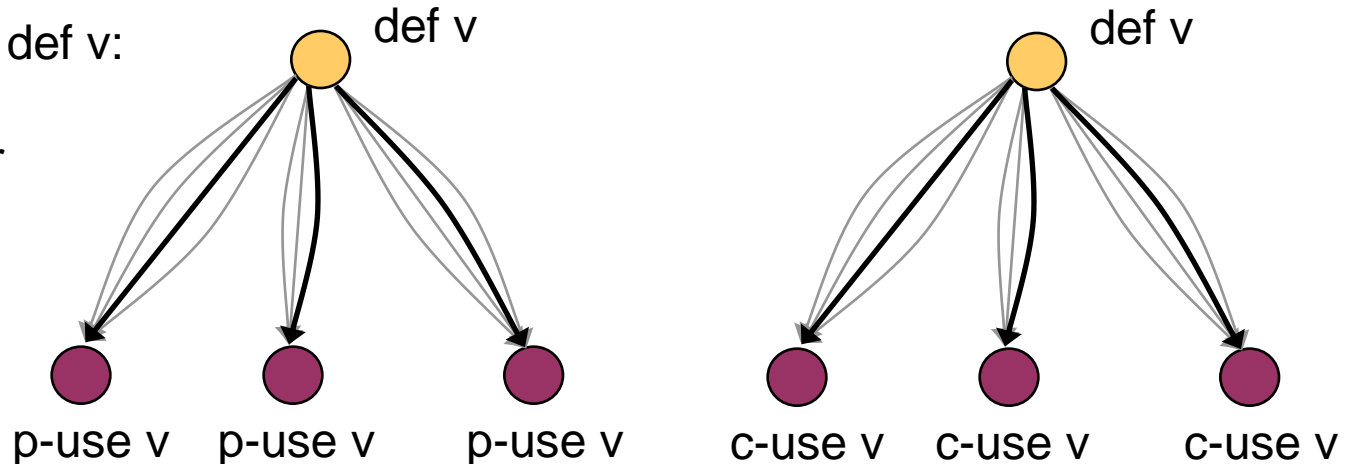
Minden v változóra, minden **def v** utasításra:

Minden c-use v utasításig legalább **egy def-clear v** útvonal tesztelése

forall v , forall def v :

one def-clear
path tested:

to all use v :



További variációk és all-uses

- **All-p-uses/some-c-uses:**

Minden **v** változóra, minden **def v** utasításra:

Minden **p-use v** utasításig és (ha **p-use v** nincs)

legalább egy **c-use v** utasításig

legalább egy **def-clear v** útvonal tesztelése

- **All-c-uses/some-p-uses:**

Minden **v** változóra, minden **def v** utasításra:

Minden **c-use v** utasításig és (ha **c-use v** nincs)

legalább egy **p-use v** utasításig

legalább egy **def-clear v** útvonal tesztelése

- **All-uses:**

Minden **v** változóra, minden **def v** utasításra:

Minden **use v** utasításig legalább egy **def-clear v** útvonal tesztelése

- Az eddigi kritériumokat magába foglalja

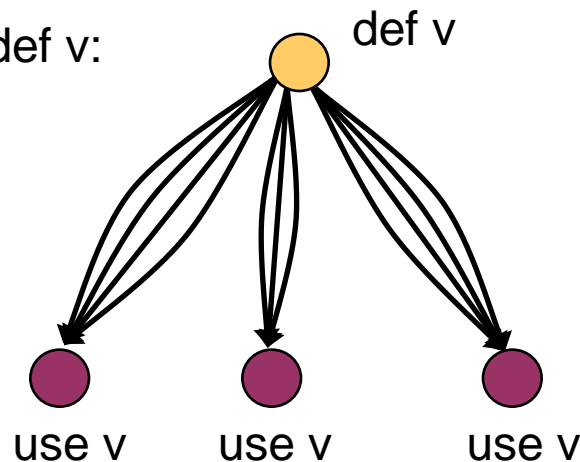
All-paths és all-du-paths

- **All-paths:**
 - Minden v változóra, minden **def v** utasításra:
Minden **use v** utasításig **minden** bejárható **def-clear v** útvonal tesztelése
 - Hurkok esetén problémás a definíció teljesítése (bejárások száma)
- **All-du-paths:**
 - Minden v változóra, minden **def v** utasításra:
Minden **use v** utasításig **minden d-u v** útvonal tesztelése

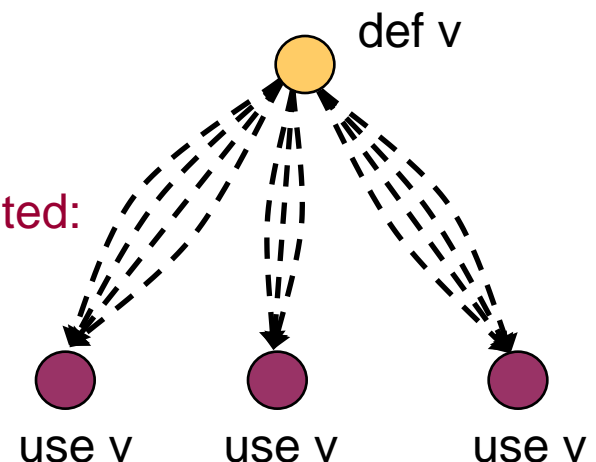
forall v , forall def v :

all def-clear
path tested:

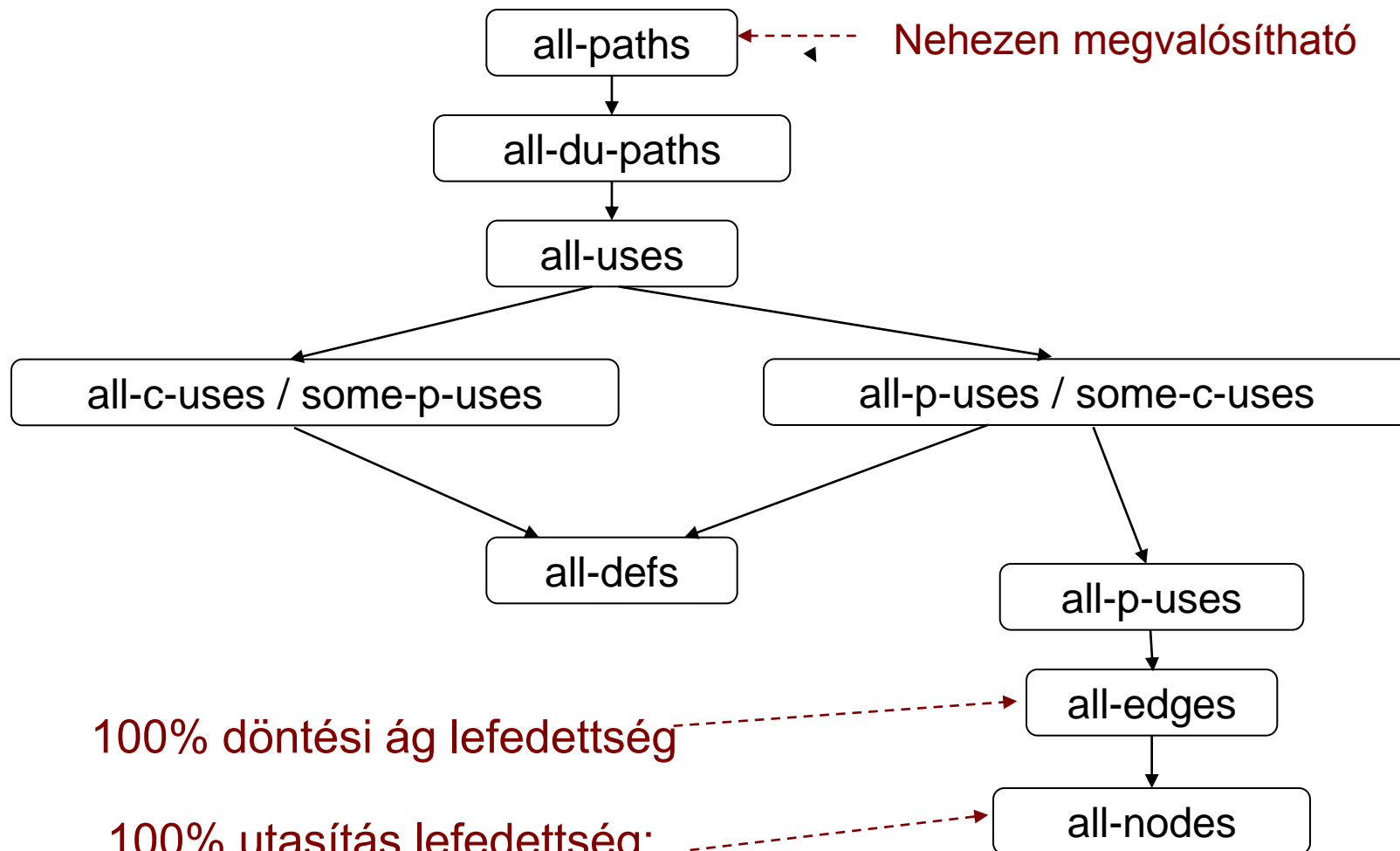
to all use v :



all d-u
path tested:



Adatfolyam alapú kritériumok hierarchiája



Mértékszámok (kritériumok) áttekintése

- Vezérlési folyamat alapú kritériumok
 - Utasítás lefedettség
 - Döntési ág lefedettség
 - Feltétel lefedettségek
 - Útvonal lefedettség
- Adatfolyam alapú kritériumok
 - Definiálás – használat fedettségek
 - Definíciómentes útvonalak fedettsége
- Módszerek kombinációja

Teszttervezési módszerek összefoglalása

- **Specifikáció és struktúra alapú módszerek**
 - Sokféle módszer illetve technika
 - Mindegyik alkalmazásához gyakorlat kell
- **A gyakorlatban általában csak az egyszerűbb módszereket használják**
 - Biztonságkritikus rendszerek: Vannak előírt módszerek (pl. DO178B szabvány: MC/DC szerinti tesztelés)
- **Technikák kombinációja hatásos általában:**
 - Példa (MS tanulmány):
 - specifikáció alapú: 83%-os kódfedés
 - + exploratory tesztelés: 86%-os kódfedés
 - + strukturális tesztelés: 91%-os kódfedés

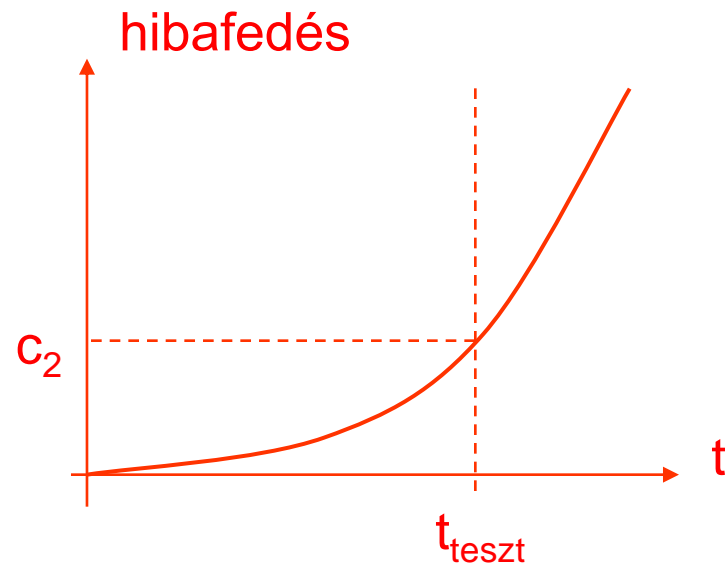
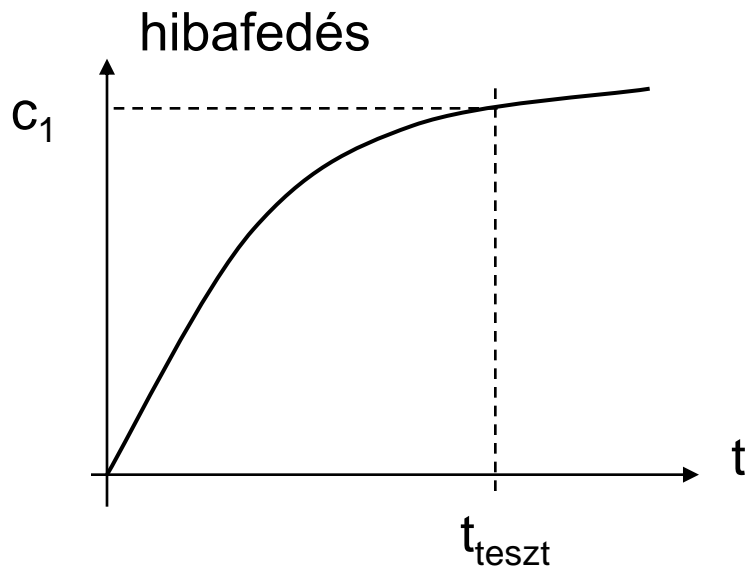
Tesztek végrehajtása

Milyen sorrendben hajtsuk végre a tesztek:

Ha várhatóan kevés a hiba:

Először a **hatékony** (nagyobb hibafedésű) tesztek!

- Hosszabb utak tesztje
- Összetett feltételű elágazások tesztje



Mire jók a teszt fedettségi mértékek?

- Mire jók?
 - Megtalálhatók azok a programrészek, ahol hiányos a tesztelés
 - Ez alapján bővíthető a teszt készlet
 - Azonosíthatók a redundáns tesztek (azonos részeket fednek le)
 - Adatfüggésre is figyelni kell (más adattal más hibát tesztelhet)
 - **Indirekt** mértéke a kód minőségének a sikeres tesztekhez tartozó fedettség
 - Inkább mértéke a tesztkészlet teljességének
 - A tesztelés befejezése a mértékekhez köthető
- Mire nem jók?
 - Az implementációból kihagyott (nem megvalósított) követelmények tesztelése
 - Kódrészletek kiragadásával történő tesztelés eredményessége kérdéses (amikor a kódrészlet elveszíti környezetét)