Robusztusság tesztelés

Majzik István és Micskei Zoltán
Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék
http://www.inf.mit.bme.hu/

Tartalomjegyzék

Definíciók

Robusztusság (IEEE Std 610.12.1990):

- „The degree to which a system operates correctly in the presence of
  - exceptional inputs or
  - stressful environmental conditions”
- Annak jellemzője, mennyire helyesen működik a rendszer
  - rendkívüli bemenetek vagy
  - nagy igénybevételt jelentő környezeti feltételek mellett.

Robusztusság hiba:

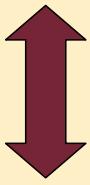
- Helytelen (nem elvárt) működés rendkívüli bemenetek és környezeti feltételek esetén

Robusztusság tesztelés:

- A robusztusság hibák aktiválása a tesztelés során

Robusztusság tesztelés

Funkcionális tesztelés
- Specifikációnak megfelelő működés vizsgálata
- Érvényes bemenet / elvárt kimenet



Robusztusság tesztelés
- Specifikációtól eltérő működés(képtelenség) vizsgálata
- Extrém vagy hibás bemenet / elvárt kezelés, hibajelzés

Tartalomjegyzék

Bemenetek robusztusság teszteléshez

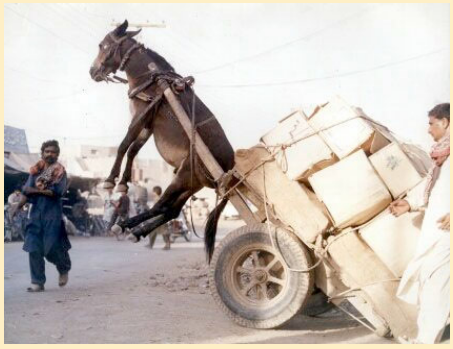
- Véletlen bemenetek
  - Van esélye robusztusság hiba aktiválásának
  - Egyszerű teszt adat generálás, de kis hatékonyság
- Típus-specifikus bemenetek
  - Típustól függően előre kijelölt extrém értékek
  - Komplex kombinációk lehetségesek
- Objektumok mint bemenetek
  - NULL értékek használhatók extrém értékként
  - Konstruktor hívása szükséges a létrehozáshoz
- Scenario mutációval generált bemenetek
  - Az extrém bemenetek állapottól függően adhatók ki
  - Sorrendi, kihagyási, időzítési hibák is definiálhatók

Bemenetek generálása típus-specifikus szabályokkal

Type	Parameter Mutation
String	Replace by null value
	Replace by empty string
	Replace by predefined string
	Replace by string with nonprintable characters
	Add nonprintable characters to the string
	Replace by alphanumeric string
	Add characters to overflow max size
Number	…
List	…
Date	…
Boolean	…

Munkaterhelés (workload) a tesztelés során

- Valódi terhelés
  - Pl. regisztrált scenariok visszajátszása
- Generált realisztikus terhelés
  - Jellegzetes scenariok generálása
  - Jobban hordozható, kézbentartható
- Szintetikus terhelés
  - Jellemző használat: Tervezett, névleges terhelés
  - Túlterhelés



Robusztusság teszt kimenetek értékelése

- Specifikációtól eltérő működés
  - Sokszor nincs előírt érték
  - Elvárt eredmények egyszerűsített kezelése szükséges
- Klasszikus kategóriák: CRASH
  - **C**atastrophic: A teljes rendszer összeomlik / újraindul
  - **R**estart: Az adott alkalmazás újraindulása
  - **A**bort: Az adott alkalmazás leáll
  - **S**ilent: Hibajelzés nélküli érvénytelen művelet
  - **H**indering: Érvénytelen hibakód
- Nem robusztusság hiba:
  - Érvényes hibakód visszaadása

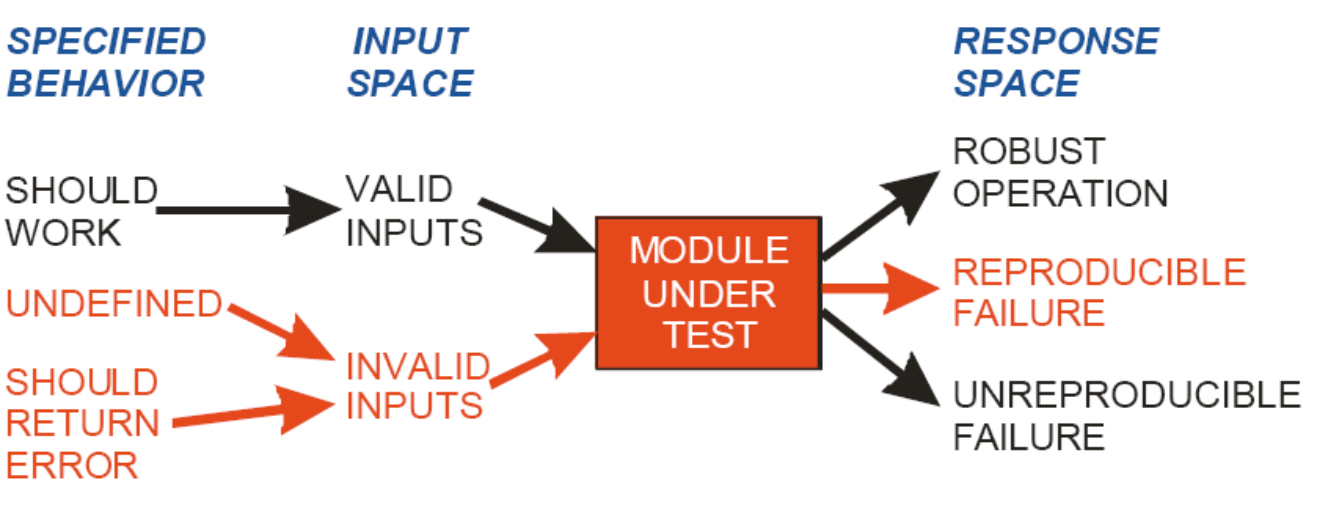
Tartalomjegyzék

Jellegzetes eszközök

- Hardver hibainjektáló eszközök
  - Közvetett robusztusság tesztelés: Környezet komponenseibe injektált hibák hatása vizsgálható a tesztelt komponensre
  - Belső hibák injektálása nem robusztusság tesztelés!
- Kombinatorikus robusztusság tesztelő eszközök:
  - Fuzz*:* Véletlenszerű bemenetek konzolos alkalmazásokhoz
  - Ballista*:* Típus-specifikus tesztelés POSIX, CORBA hívásokhoz
  - JCrasher*:* Extrém objektumok Java alkalmazásokhoz
- Forráskód mutációs eszközök
  - Funkcionális teszt szekvenciák mutálhatók
  - Szekvencia vagy hívási paraméter megváltoztatása
- Benchmark eszközök
  - Benchmark: Reprezentatív, megismételhető vizsgálat
  - DBench*:* Szolgáltatásbiztonság benchmarkok

Típus-specifikus tesztelés: Ballista – alapelvek

API tesztelés:

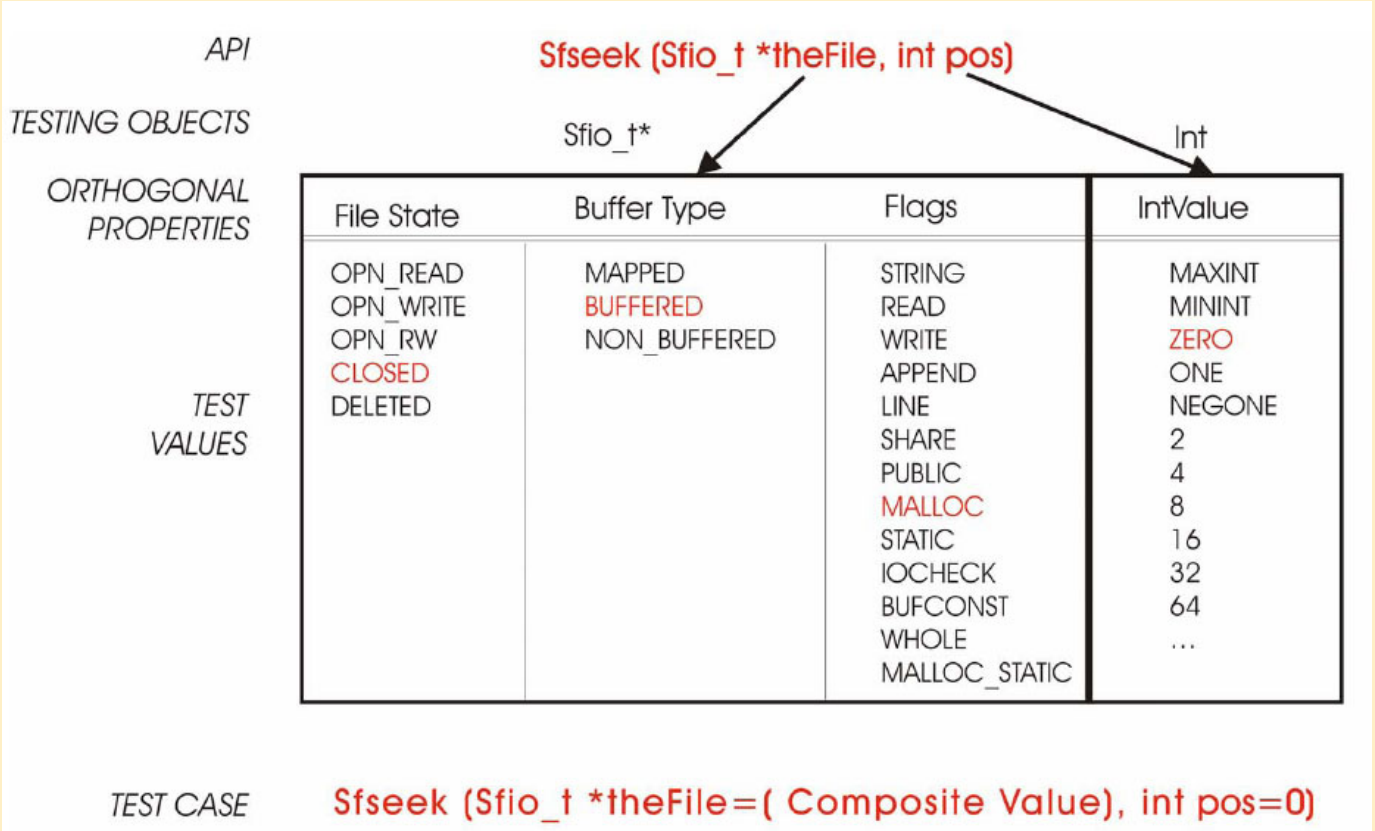


Típus-specifikus tesztelés: Ballista – bemenetek

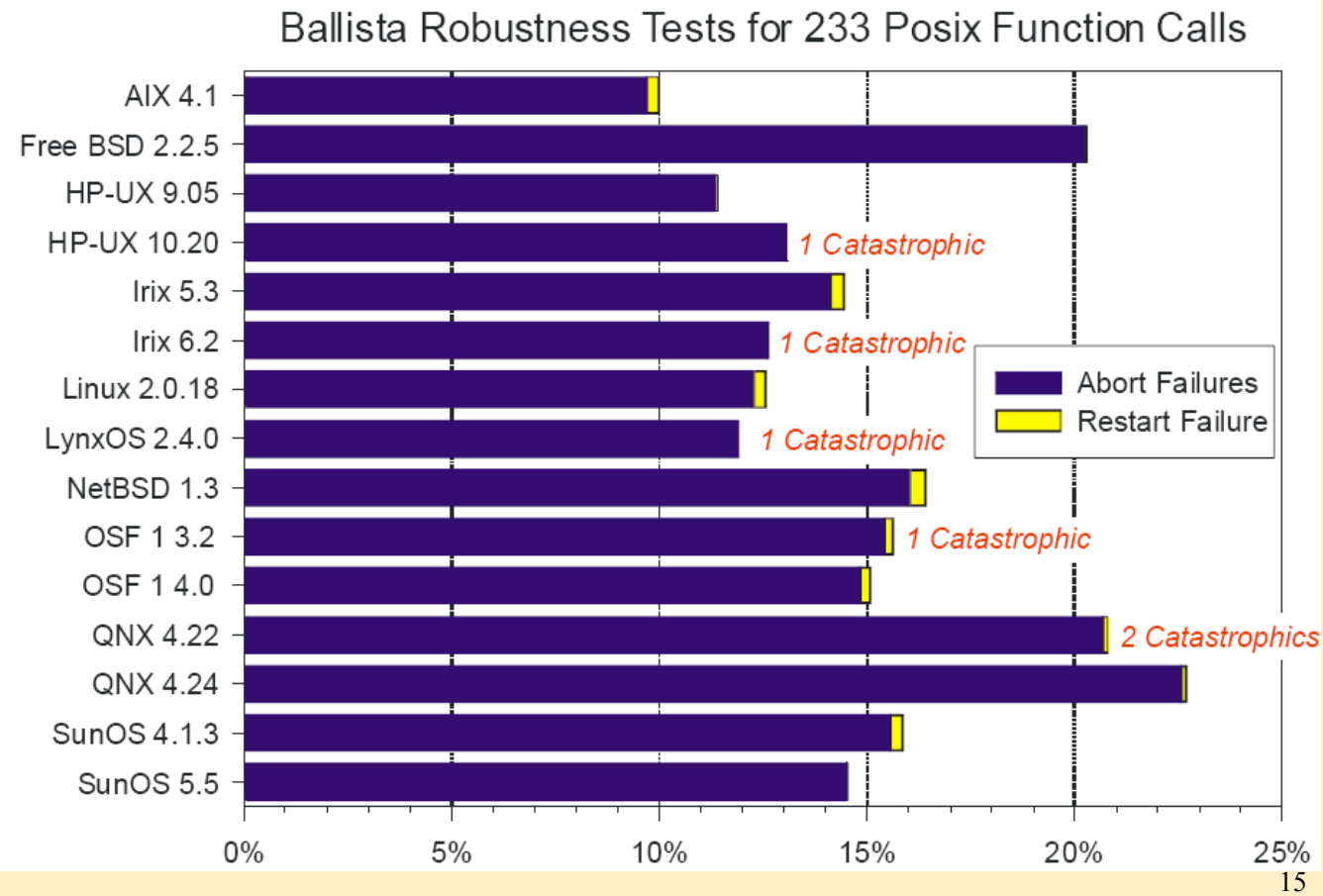
Szélső és extrém értékek kijelölése:

Data type	Substitution values					
Pvoid	NULL	0xFFFFFFFF	1	0xFFFF	-1	Random
Integer	0	1	MAX INT	MIN INT	0.5	
Boolean	0	0xFF (Max)	1	-1	0.5	
String	Empty	Large (> 200)	Far (+ 1000)			

Típus-specifikus tesztelés: Ballista – kombinációk



Típus-specifikus tesztelés: Ballista – eredmények

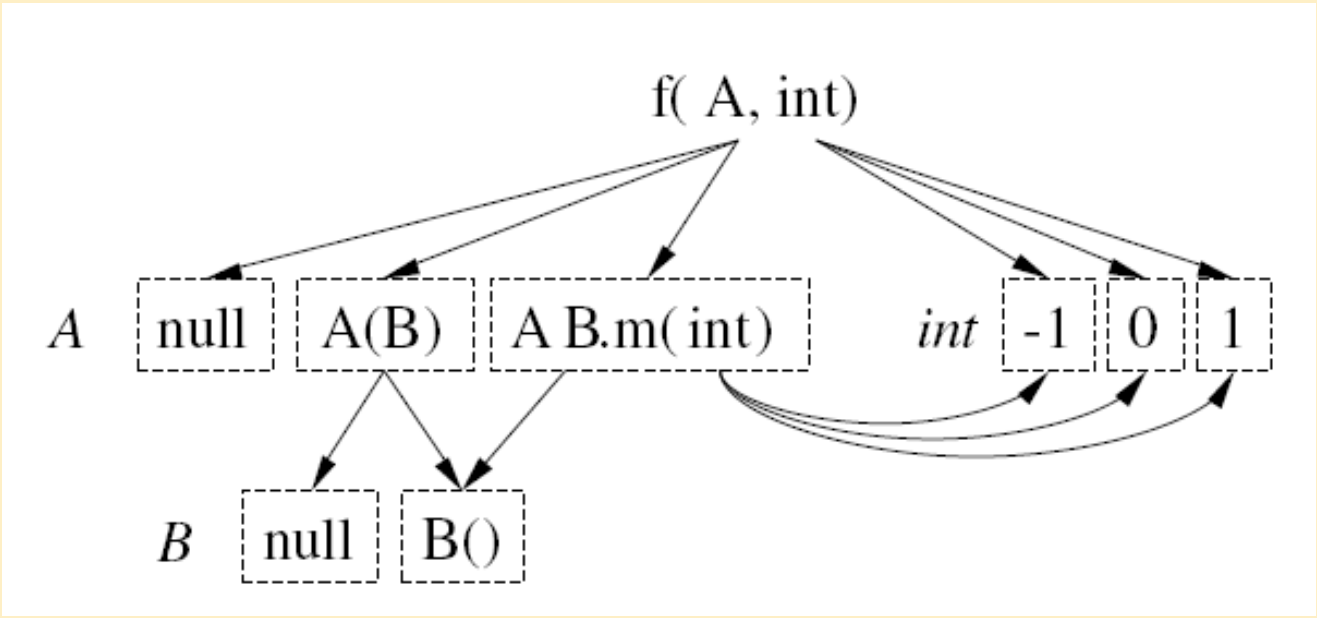


OO programok tesztelése: JCrasher – példa

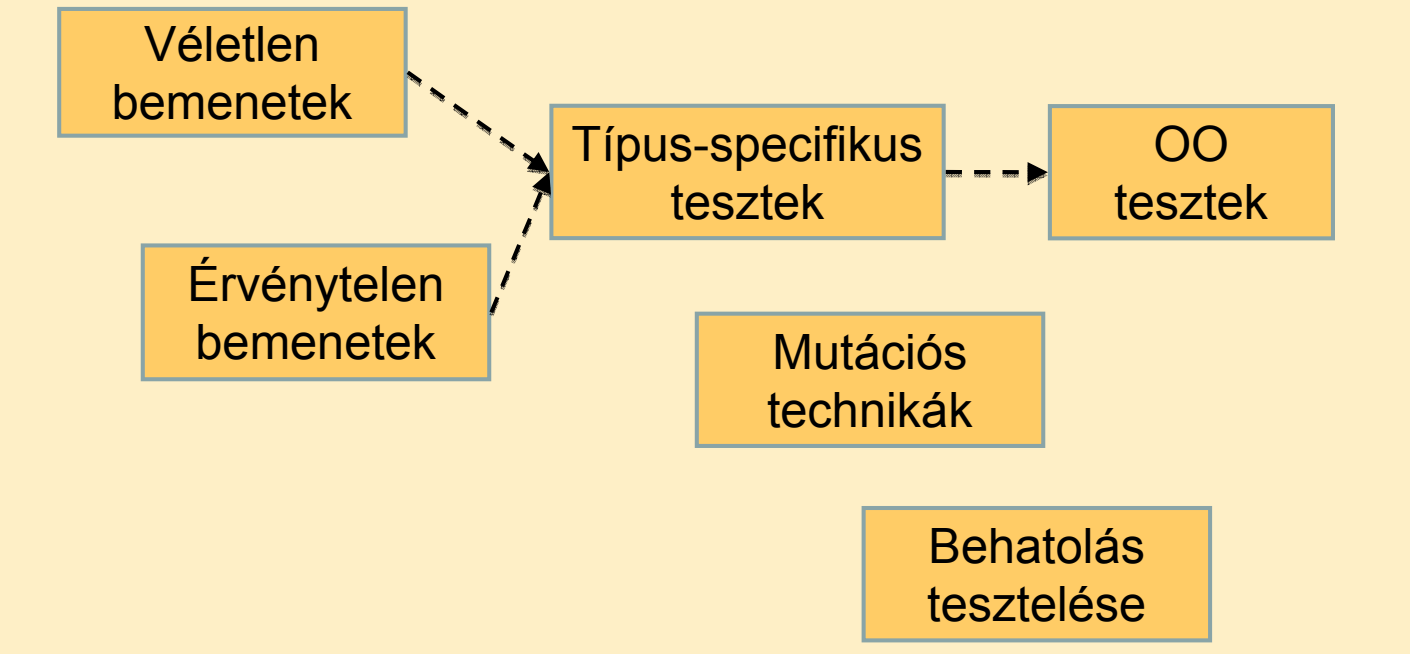
```
public void test2() throws Throwable {
        try {
                java.lang.String s1 = (java.lang.String)null;
                java.lang.String s2 = "Norm";
                Student s3 = new Student(s1, s2);
        }
        catch (Exception e) {dispatchException(e);}
    }

    public void test3() throws Throwable {
        try {
                java.lang.String s1 = (java.lang.String)null;
                java.lang.String s2 =
                    "~!@#$$%^&*()_+{}|[]';:/.,<>?`-=";
                Student s3 = new Student(s1, s2);
        }
        catch (Exception e) {dispatchException(e);}
    }
```

OO programok tesztelése: JCrasher – paraméterek

Adott típusú objektum létrehozása:



Összefoglalás: Az eszközök fejlődése



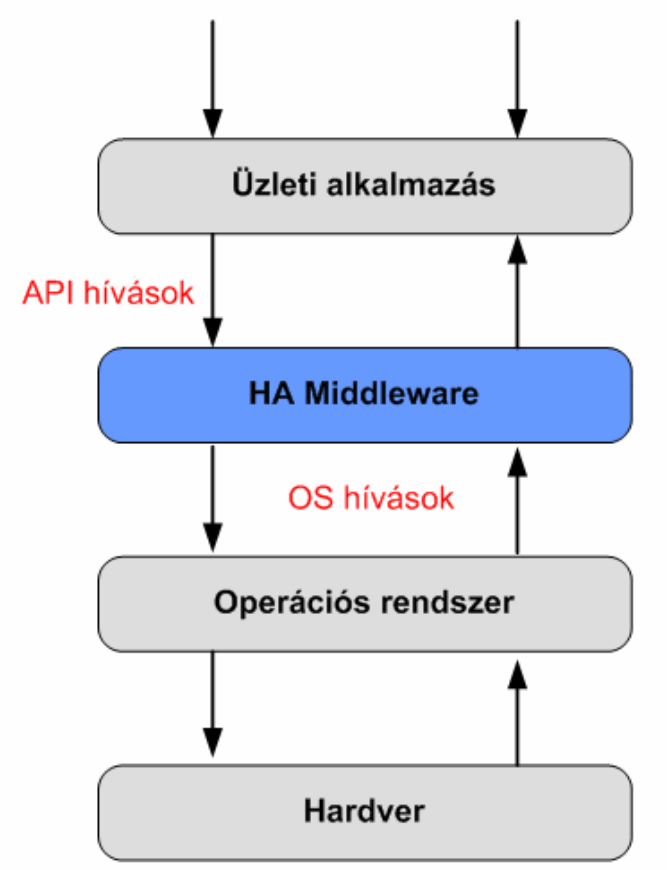
Tartalomjegyzék

- Definíciók
  - Robusztusság
- Robusztusság tesztek
  - Teszt bemenetek
  - Teszt eredmények értékelése
- Jellegzetes teszt eszközök
  - Ballista
  - JCrasher
- Mintapélda
  - SA Forum AIS API robusztusság tesztelése

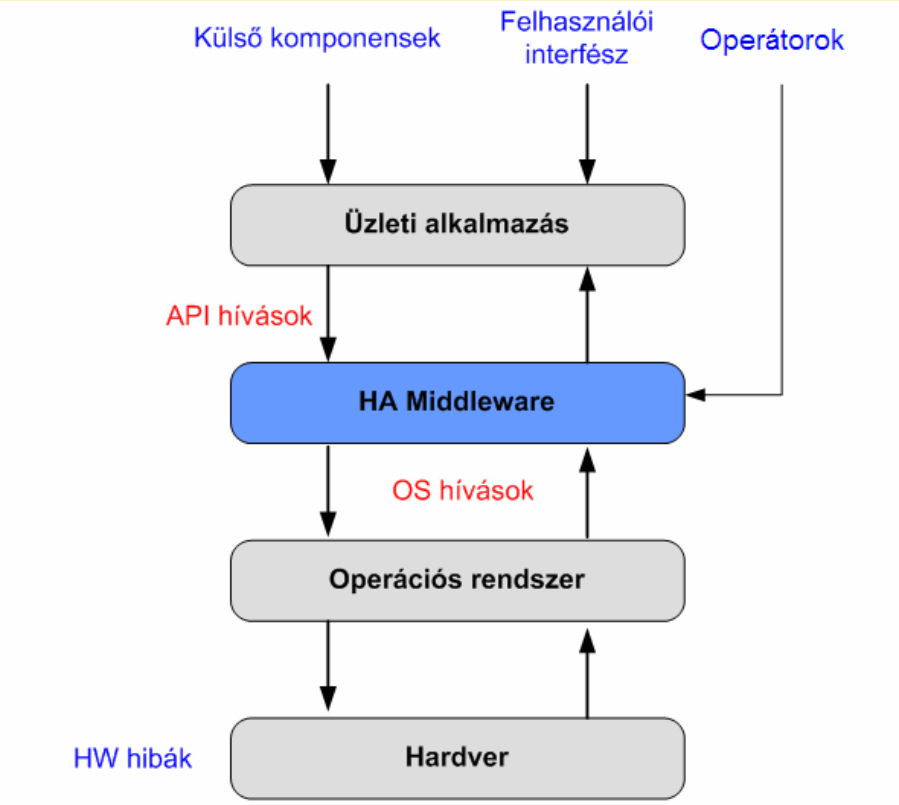
Mintapélda: HA köztesréteg tesztelése

- HA köztesréteg: Nagy rendelkezésreállás biztosítása
  - Komponens hibadetektálás, helyreállítás, újraindítás, failover
  - Szabványos operációs rendszerek felett
  - Elosztott alkalmazások
- Service Availability Forum: HA köztesréteg specifikáció
  - AIS: Redundancia menedzselési funkciók (AMF)
  - Szabványos interfészek (C nyelvű)
- Miért kritikus a robusztusság hiba?
  - Futtatott komponensek hibáira kell felkészülni
  - Egy hibás komponens a köztesréteg robusztusság hibájának aktiválásával az egész rendszer működését befolyásolhatja!
- Robusztusság tesztelés
  - Közös interfész alapján összehasonlítható implementációk
  - Nagyszámú hibaforrás és hibamód → automatikus tesztelés

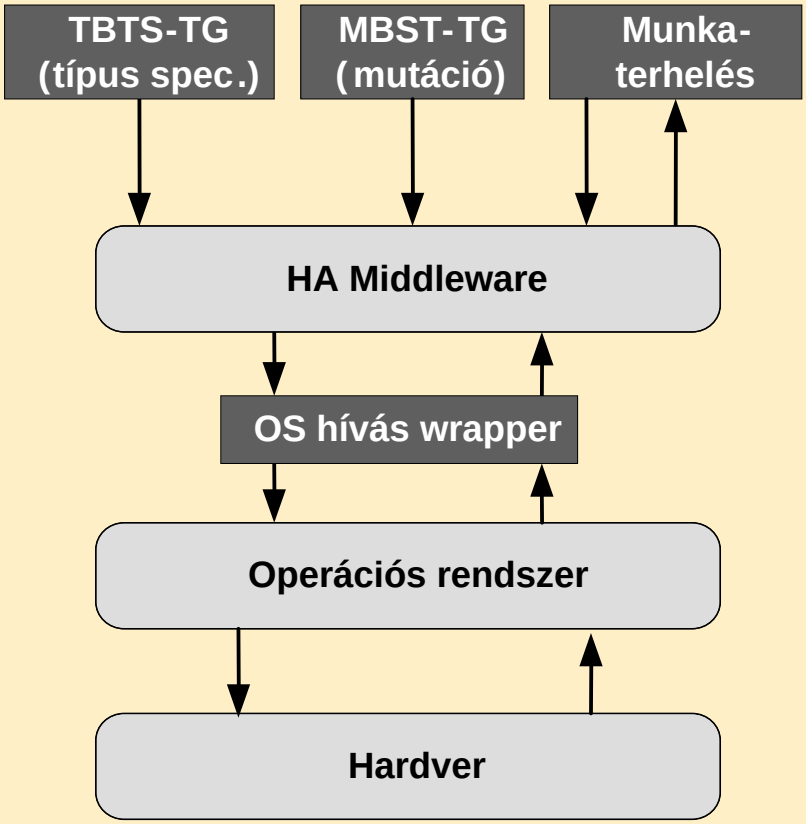
Robusztusság tesztek: Elsődleges források



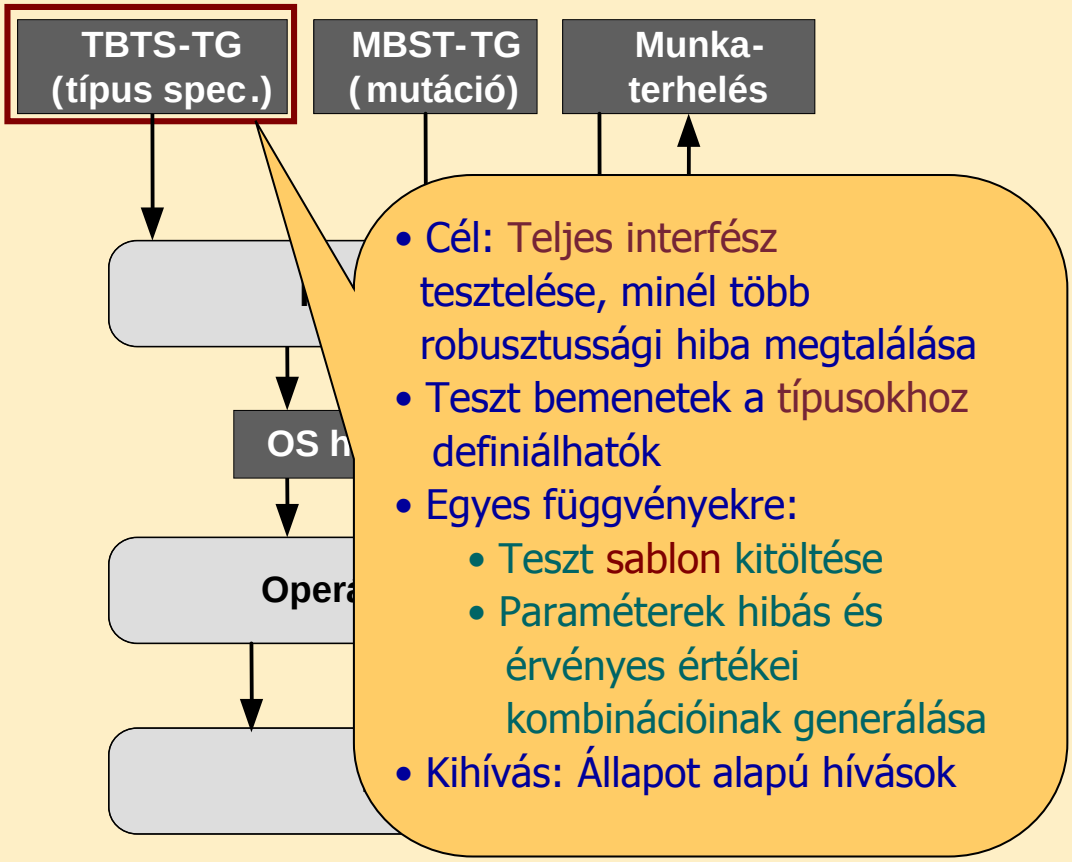
Robusztusság tesztek: Másodlagos források



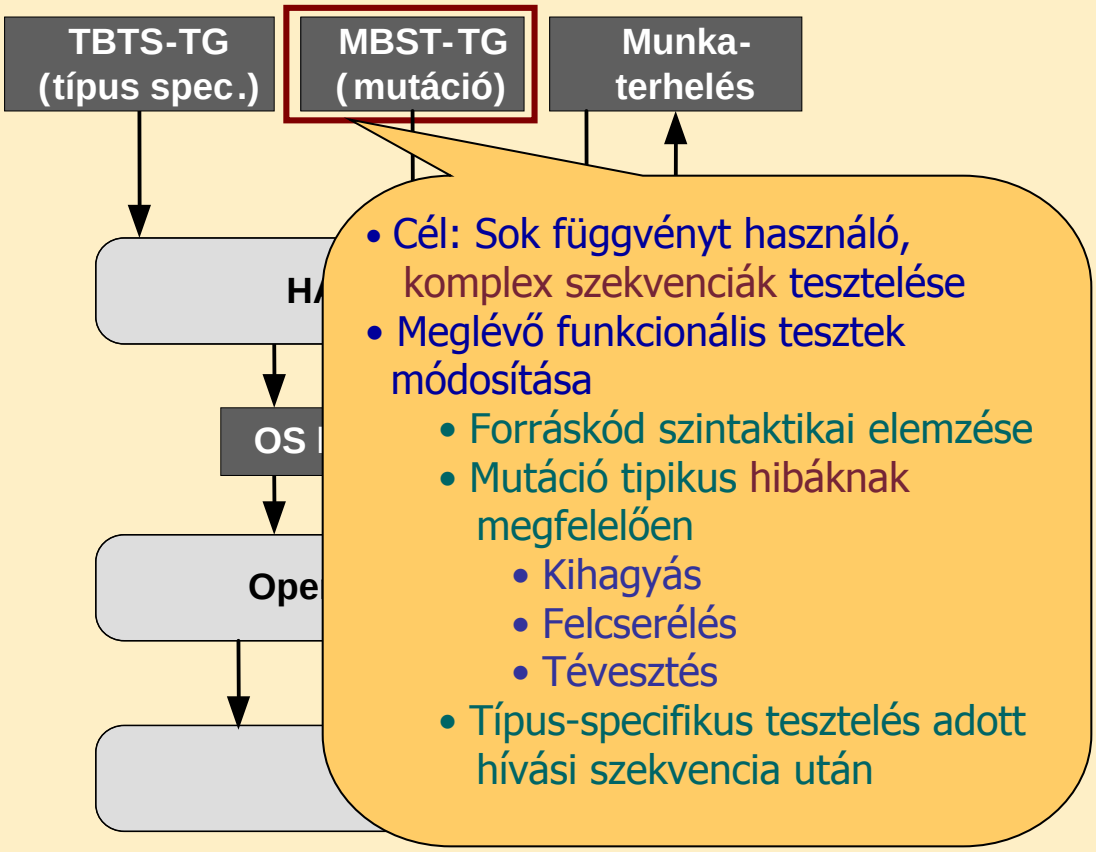
Az elkészült teszt eszközök



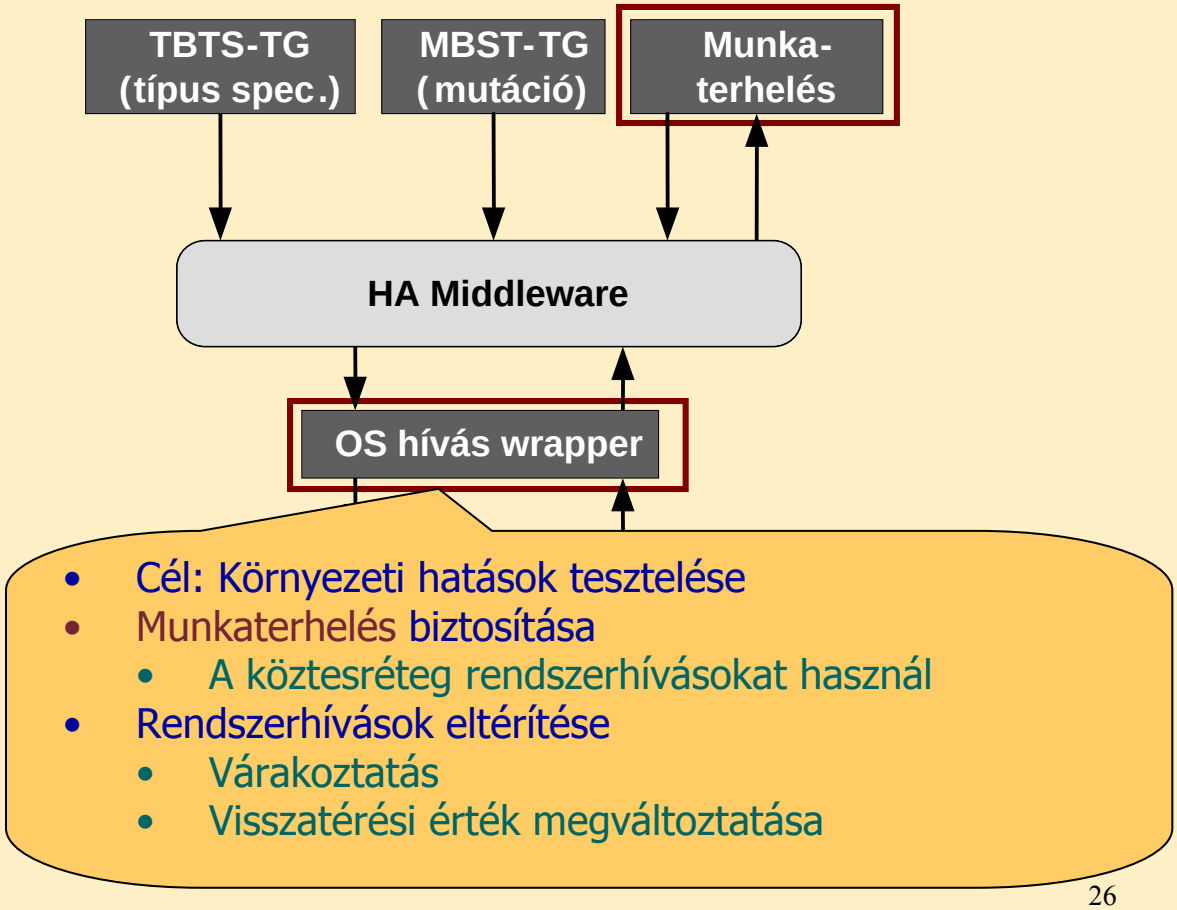
Az elkészült teszt eszközök



Az elkészült teszt eszközök



Az elkészült teszt eszközök



Robusztusság tesztelési eredmények

- Három köztesréteg vizsgálata
  - Openais 0.80.1
  - Openais trunk verzió
  - Fujitsu Siemens SAFE4TRY
- Teszt futtató keretrendszer
- Eredmények:
  - A szabványtól való eltérések detektálása
  - Gyakori hiba: Abortált teszt program
  - Kritikus hiba: Köztes réteg összeomlása

Eredmények áttekintése

SAFE4TRY robusztusabb

Típus specifikus	openais-0.80.1	openais-trunk	SAFE4TRY
Hibátlan lefutás	24568	26019	29663
Segmentation fault	1100	1468	0
Timeout	467	2178	2
Mutáció			
Hibás / összes	30 / 92	28 / 92	1 / 92
OS wrapper			
Nem volt hiba	6	5	5
Alkalmazás hiba	0	2	1
Köztesréteg hiba	3	2	3

Rendszerhívások hibájára mindegyik érzékeny