

# Speciális tesztelési feladatok

Majzik István

Budapesti Műszaki és Gazdaságtudományi Egyetem

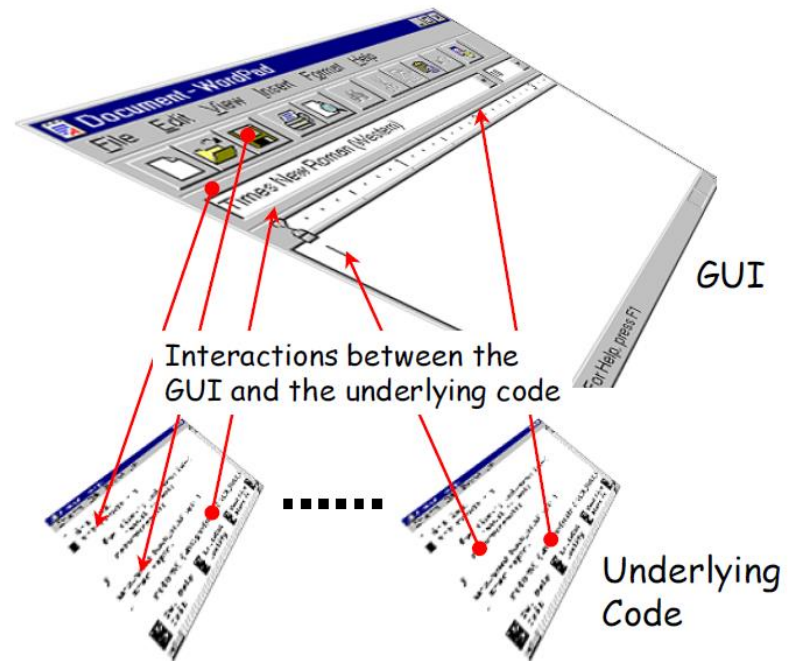
Méréstechnika és Információs Rendszerek Tanszék

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

# Felhasználói felületek tesztelése

# GUI jellegzetességek

- Felhasználói utasítások fogadása, eredmény megjelenítése
  - Kommunikáció grafikus elemeken keresztül
  - A program „logikájához” nem ad hozzá
- Eseményvezérelt működésű
  - Manipuláció főként egérrel
- Implementációk
  - GUI toolkitok (Qt, GTK+, Swing, WinForms, ...)
  - Webes GUI



# Tesztelési nehézségek

- Informális követelmények
  - Minták, ergonómiai ajánlások, konvenciók
- Nagyszámú bemenet, nagy állapottér
  - Ugyanabban a kontextusban sokféle esemény
  - Ugyanaz az esemény sokféle kontextusban
  - Váratlan események
- Bonyolult GUI funkciók (toolkit mint „fekete doboz”)
  - Rejtett, nem dokumentált funkciók
- Nehéz teszt végrehajtás
  - Egérkattintások reprodukálása
- Nehéz kiértékelés
  - Grafikus felület változásai és háttér működés azonosítása

# Szisztematikus tesztelés előfeltételei

- GUI modell felvétele

- Előnyök:

- Teszt fedettség definiálható
    - Automatikus tesztgenerálásra is lehetőséget ad

- Két tipikus GUI modell:

- Operátor alapú GUI modell
    - Állapotgép alapú GUI modell

- Teszt módszer rögzítése

- Előnyök

- GUI modellhez való illeszkedés biztosítható
    - Ad-hoc megoldásoknál jobban kézmentartható

- Két példa:

- Scenario alapú tesztelés: (Leggyakoribb) használat tesztelése
    - Kombinatorikus tesztelés: Teljes fedésre ad lehetőséget

# I. Operátor alapú GUI modell

- Program objektumok
  - Háttérműködés elemeihez kötött (pl. szövegrészek, fájlok, ...)
- Események
  - Menü események (MM)
    - Műveletek kibontása (pl. File/Save)
  - Fókusz kiterjesztő/kisajátító események (FKE)
    - Munkaablakok (pl. eszköztárak) megjelenítése, új ablak nyitása
  - Rendszerkapcsolati események (RKE)
    - Program objektumok (háttér állapot) megváltoztatása
- Operátorok
  - Rendszerkapcsolati operátorok: (MM,FKE)\*RKE
    - Program objektumok befolyásolása (pl. Edit/Cut és hatása)
  - Felületi menü/fókusz operátorok: MM, FKE kombinálása
    - Új ablak nyitása egy művelet hatására (pl. File/Open... esetén)
  - Összetett (absztrakt) operátorok: Elemi operátorok sorozata
    - Pl. fájl kiírása adott könyvtárba

# Scenario alapú tesztelés

## 1. Teszt cél meghatározás

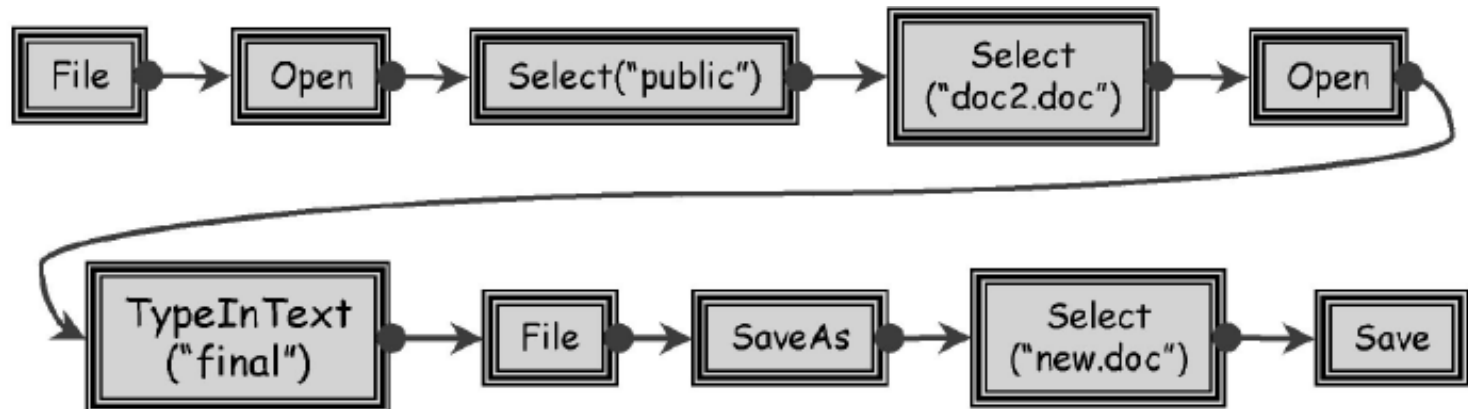
- Operátorok felmérése
- Objektumok felmérése
- Jellegzetes használat (kiindulási állapot, célállapot) meghatározása

## 2. Operátor szekvencia konstruálása

- Jellegzetes (legvalószínűbb) operátorsorozatok lefedése

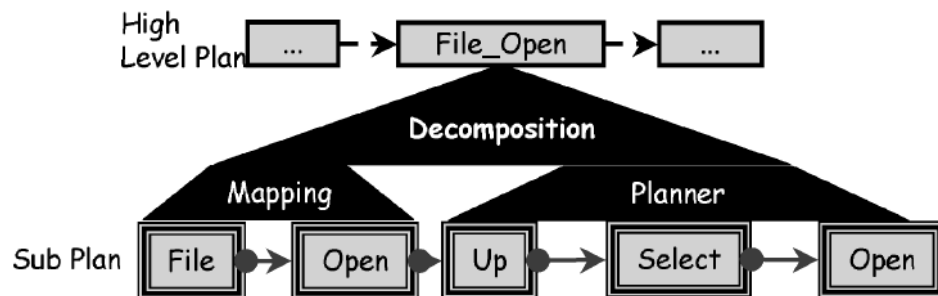
## 3. Konkrét esemény szekvenciára való leképezés

- Tesztesetek generálása
- Összetett operátorok automatikusan is leképezhetők



# Scenario generálás tervkészítő segítségével

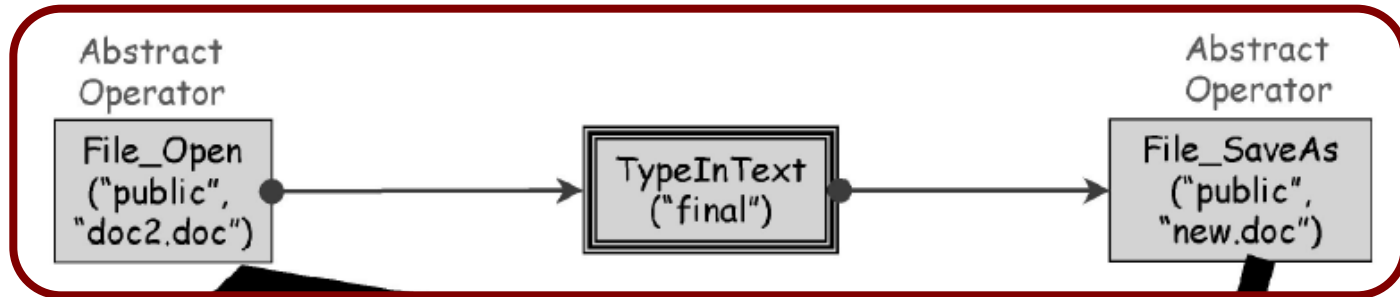
- A tervkészítés (planner) probléma elemei a GUI teszt generáláshoz:
  - Kezdőállapot: Kiindulási GUI és rendszerállapot (objektumok állapota)
  - Célállapot: Elérendő GUI és rendszerállapot
  - Operátorok (előfeltételek és hatások): GUI események alapján
    - Szabad változókat tartalmazhatnak, hierarchikusak lehetnek
  - Objektumok (lehetnek az operátorok változói): Program objektumok
- Megoldás: Terv (plan): Célállapot elérése a kezdőállapotból
  - Operátor példányok halmaza
  - Részleges rendezési reláció az operátorok között: sorrendi kötöttség
  - Ok-okozati kapcsolatok az operátorok között: hatások és feltételek kötése
  - Operátorok változóinak behelyettesítése: konkrét objektumok
- A terv teljes sorrendezéssel teszt szekvenciaként használható
  - Linearizálás



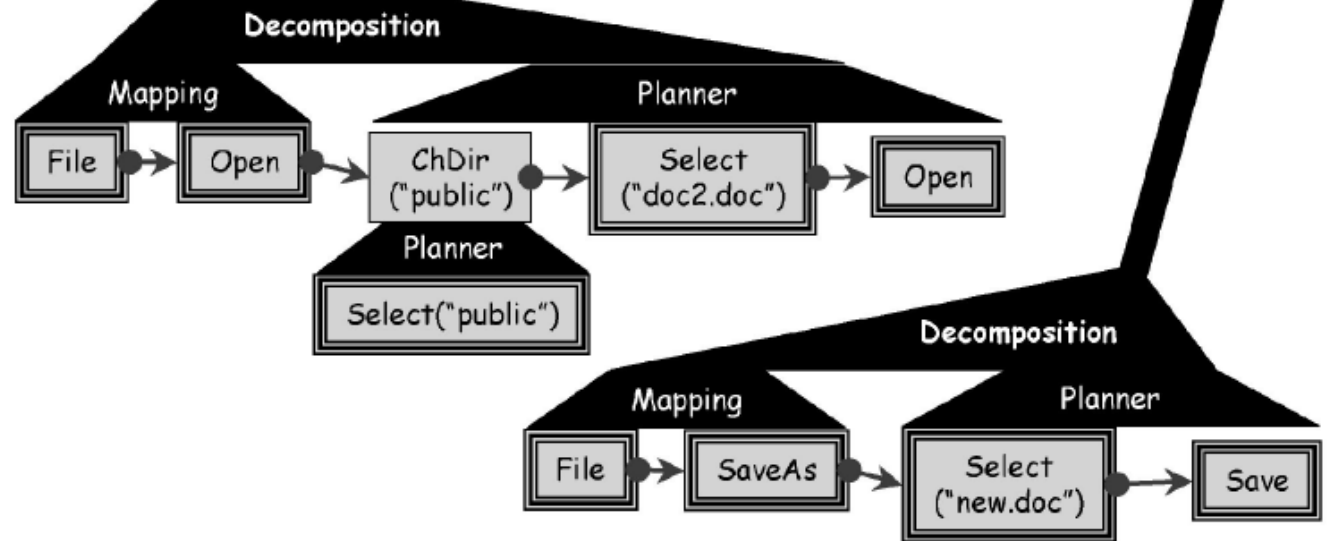


# Példa egy leképzésre

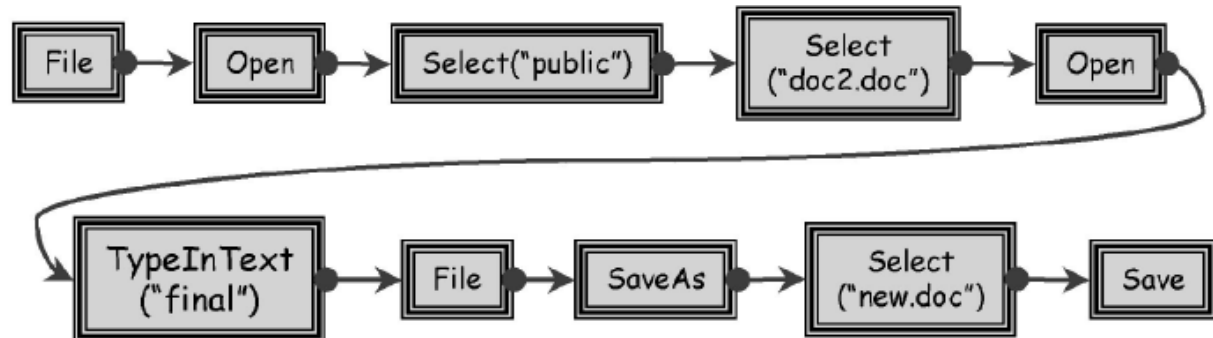
Kiindulás:



Leképzés:

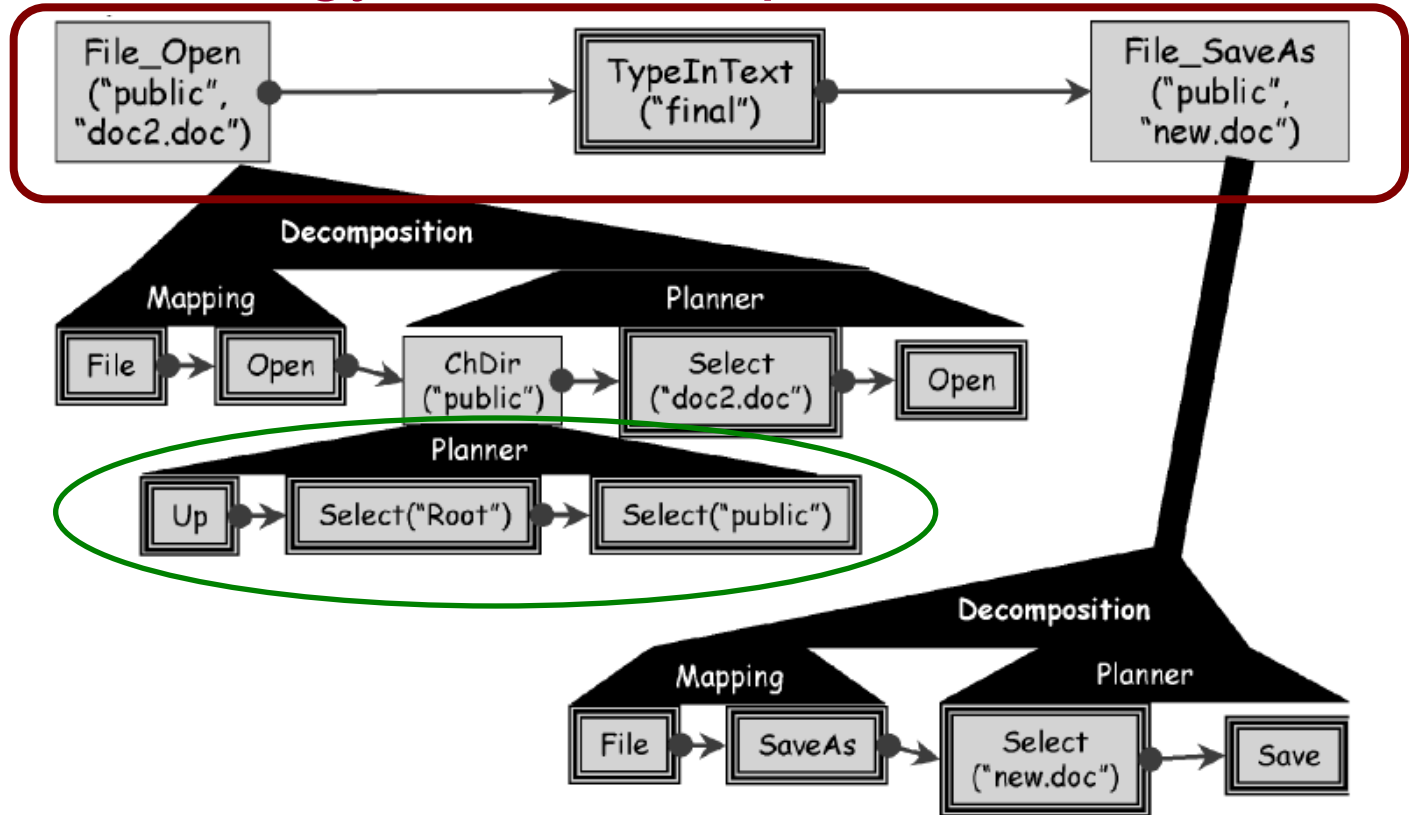


Eredmény:

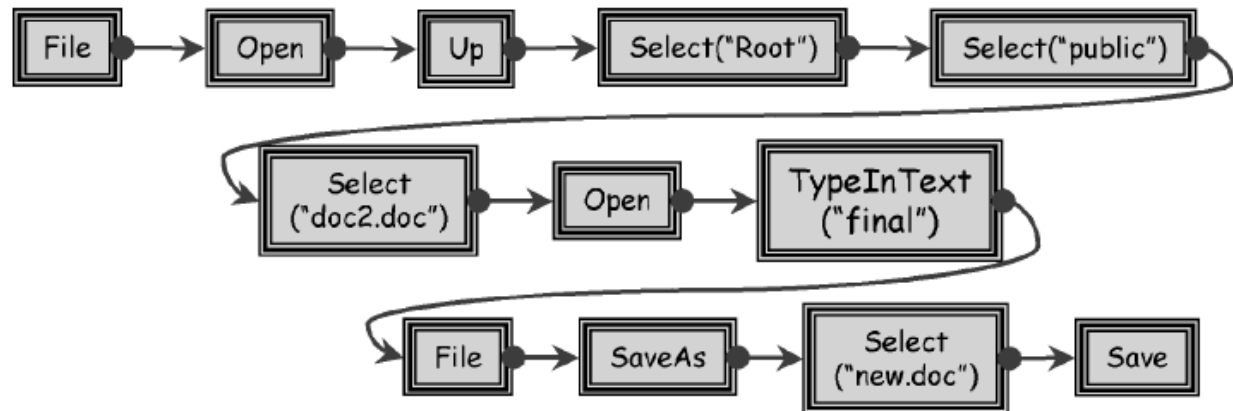


# Példa egy másik leképzésre

Leképzés:

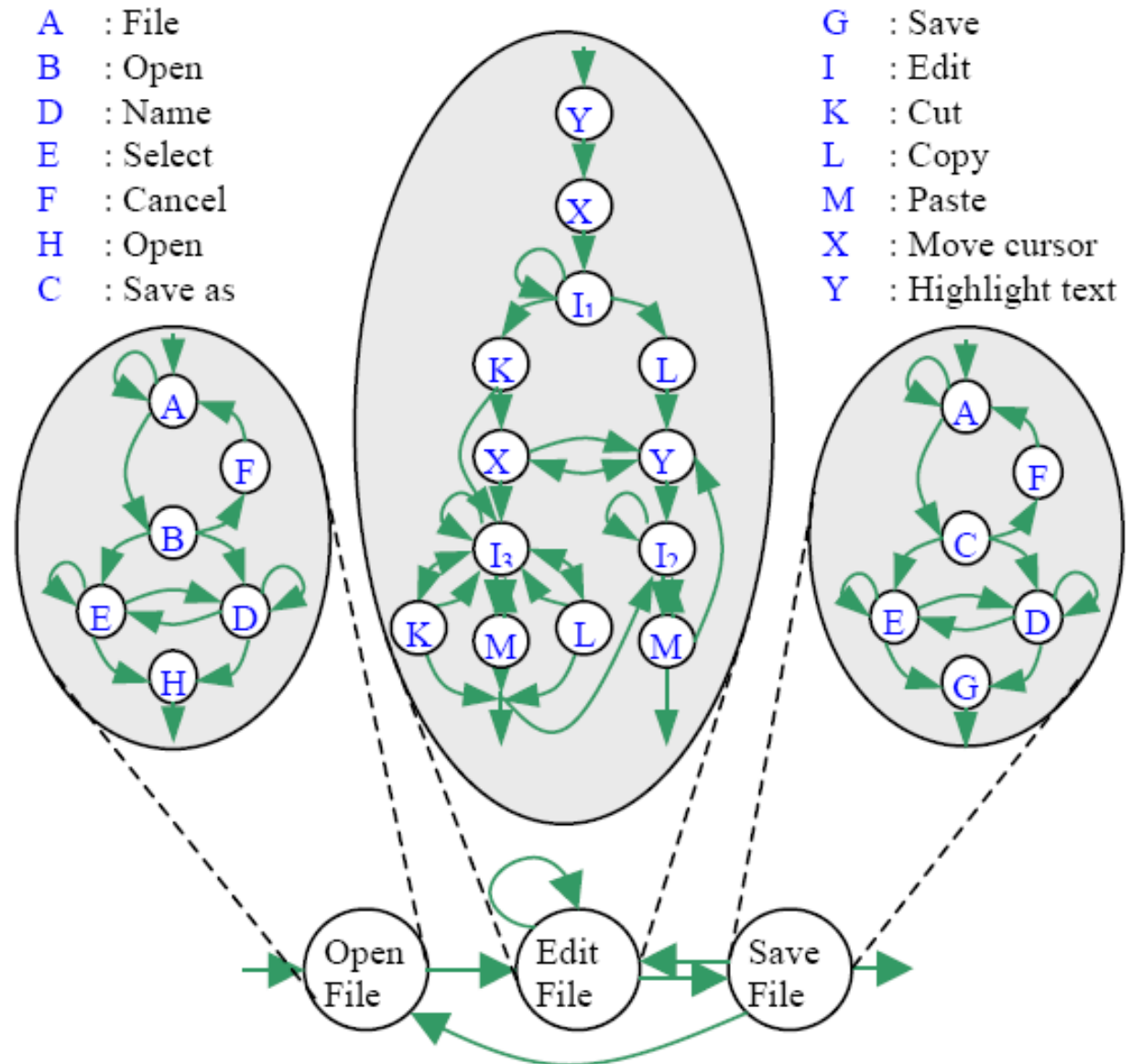


Eredmény:



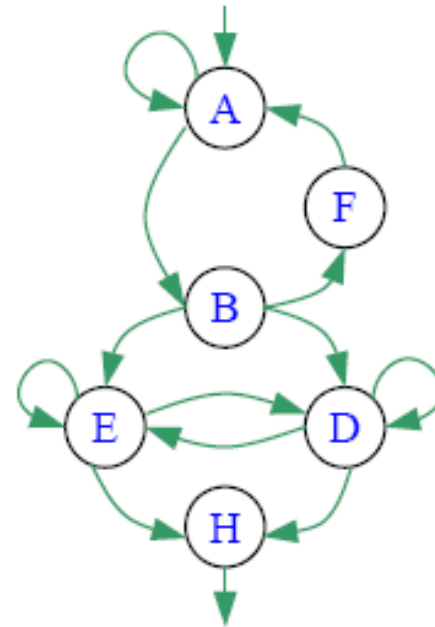
## II. Állapotgép alapú GUI modell

- GUI mint automata
- Esemény folyam (elemi műveletek)



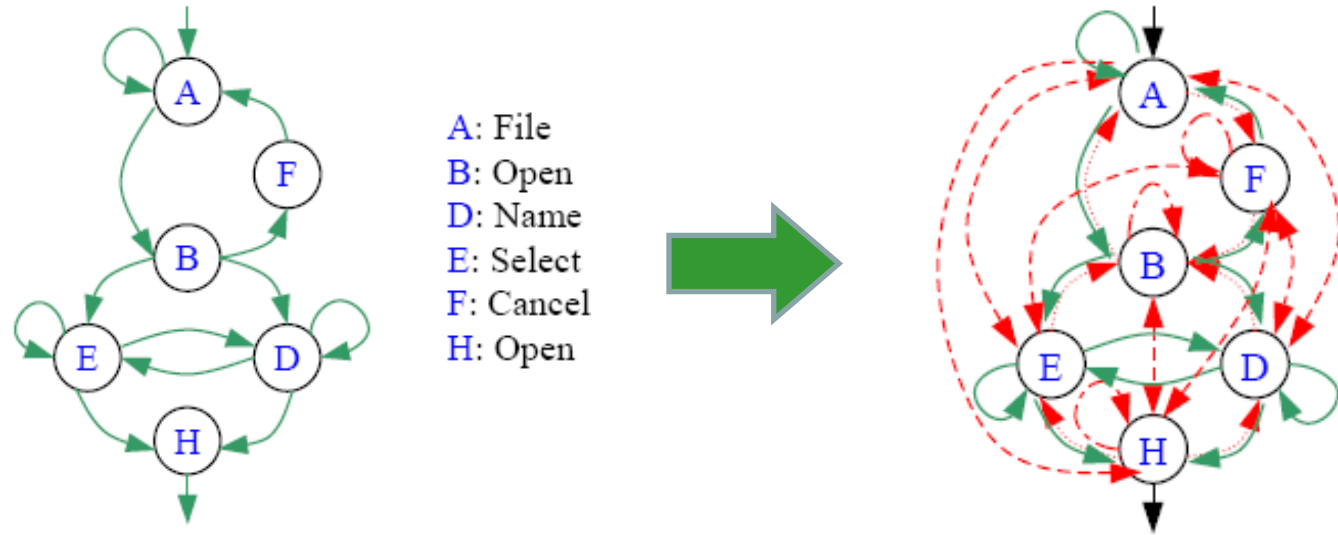
# A normál működés tesztelése

- Tesztelés fedettségi kritériumok alapján:
  - Átmenetek tesztelése
  - Átmenet párok tesztelése
  - Átmenet sorozatok tesztelése
    - Részleges bejárás
    - Teljes bejárás
- Valószínűségi tesztelés
  - Legvalószínűbb bejárásokat előre kell venni
  - Markov modell használható



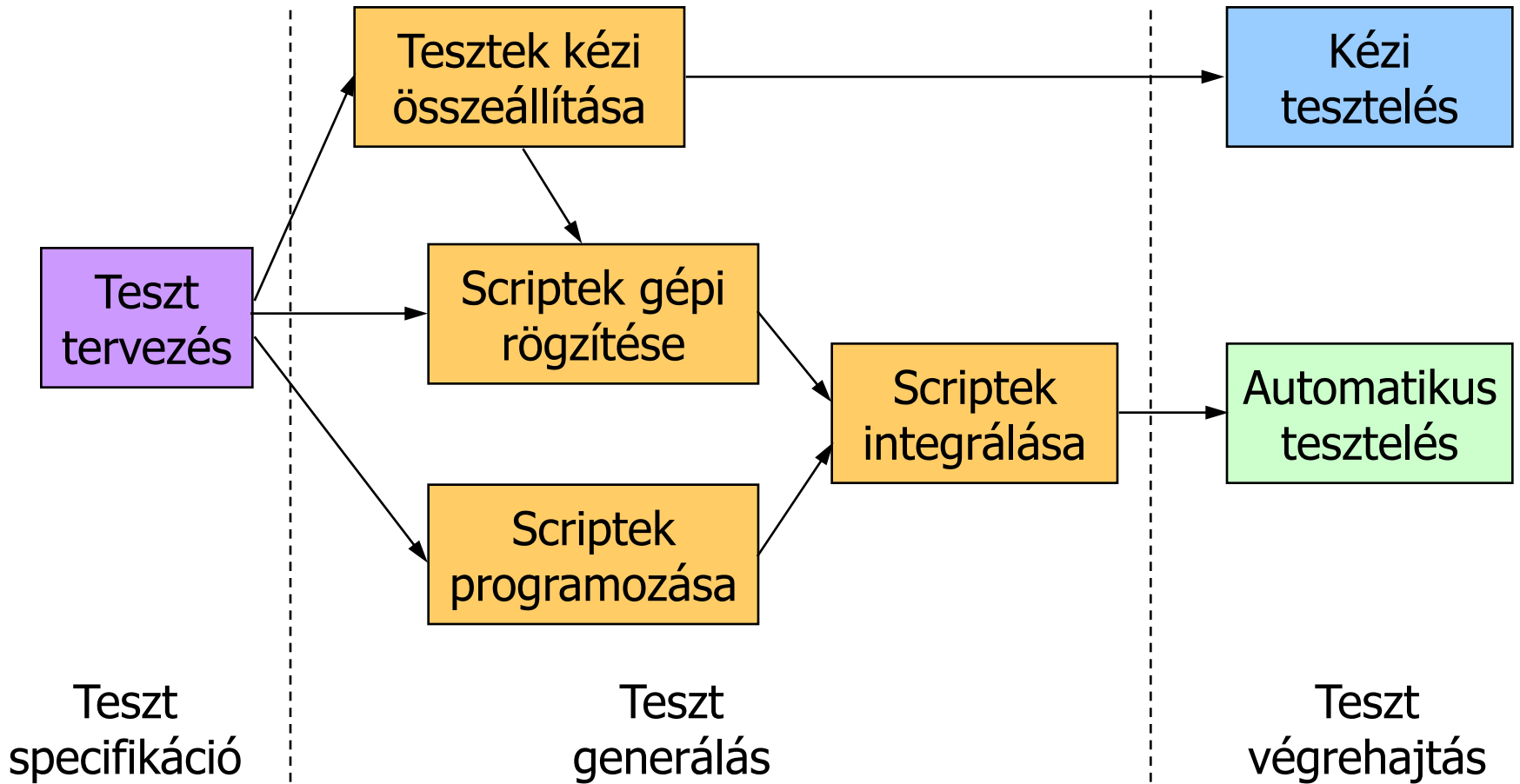
A: File  
B: Open  
D: Name  
E: Select  
F: Cancel  
H: Open

# A nem megengedett működés tesztelése



- Az állapotgép kiterjesztése „helytelen” átmenetekkel:
  - Sorrend megfordítása
  - Plusz önhurkok felvétele
  - Új sorrendi kapcsolatok felvétele (teljessé tétel)
- Helytelen átmenetek tesztelése
  1. Normál átmenetekkel a tesztelendő helytelen átmenetig
  2. Helytelen átmenet végrehajtásának kísérlete
  3. Elvárt hibajelzés vagy nem lehetséges végrehajtás ellenőrzése

# Kézi és automatikus tesztelés



# Automatizálási lehetőségek

- Teszt generálás: Script rögzítés („record”)
  - Felhasználói interakciók felvétele
  - Felvett script módosítható, többszörözhető
- Teszt végrehajtás: Script lejátszás („playback”)
  - Párhuzamosítható
  - Regressziós teszteléshez alkalmazható
- Eredmény ellenőrzés:
  - „Szöveg összevetés”: Csak karakteres felületekhez
  - „Képfeldolgozás”: Grafikus felületekhez
    - Widget alapú
    - Bitmap alapú

# Teszt automaták (példák)

Eszköz	Környezet	Licenz
Abbot	Java	CPL
Squish	Java + Web	Commercial (Eval)
SilkTest (Borland)	Multi	Commercial (Eval)
IBM RFT	Multi	Commercial (Eval)
BadBoy	Web	FFNP
JUnit Forms	.Net	BSD
QuickTest (HP)	Java + Windows	Commercial (Eval)
Ranorex	.Net+Web	Mixed
GUIDancer	Java	Commercial (Demo)
GTT	Java	GPL
Jemmy	Java	SPL
JFCUnit	Java	LGPL
Marathon	Java	LGPL
UISpec4J	Java	CPL
QF-Test	Java	Commercial (Eval)
Selenium	Web	Apache 2.0
WET	Web	BSD
Sahi	Web	Apache 2.0



# Példa: Rational Robot



- IBM Rational Functional Tester környezet
- Automatizált feladatok GUI komponensekhez:
  - Teszt szekvencia rögzítése („record”)
  - Teszt értékeléshez referencia (verifikációs pont) kijelölése
    - Menü, ablak, régió, clipboard, fájl, szöveg szintű elemek
    - Image mask megadható kép összehasonlításhoz
  - Teszt script mentése (módosítható SQABasic scriptek)
- Kiindulási információ:
  - (Grafikus) felhasználói felület felderítése, objektumok azonosítása
  - Object mapping: Felhasználói objektumokhoz
- Adatkészlet (data pool) megadható teszt sorozatokhoz
- Felhasználás:
  - Rögzített teszt szekvenciák lejátszása
  - Módosított szekvenciák lejátszása
  - Regressziós tesztelés

# Példa: Selenium



- Selenium IDE: Böngészőn keresztül történő tesztelés webes felületű alkalmazásokhoz
  - Rögzíti a felhasználói interakciókat
    - Módosítás: Szerkesztés, töréspontok
    - Mentés: Ruby, JavaScript, HTML
    - Kód generálás (Java JUnit)
  - Ezek teszteléshez újra lejátszhatók
    - Teszt bemenet: URL megnyitás, kattintás, szövegbevitel, ...
    - Teszt kimenet (assertion): Widget eltűnés, megjelenés, szöveg megjelenés,...
- Selenium Remote Control:
  - A tesztek több böngészőben futtathatók
    - Szerver komponens: Böngészők indítása, HTTP proxy funkció
    - Kliens könyvtár tesztek írásához: Java, PHP, Perl, Python, Ruby nyelvekhez
- Selenium Grid:
  - A tesztek több szerveren futtathatók a párhuzamos tesztelés érdekében
  - Selenium Hub: Több Remote Controlhoz

loginBookstore - Selenium IDE 1.0.7 \*

Base URL: http://localhost:8080/

Command	Target	Value
open	/bookstore/Login.jsp	
type	userName	user
type	password	user1
clickAndWait	//input[@value='Login']	
assertTextPresent	Welcome, user!	

Command: assertTextPresent |  
Target: Welcome, user! | Find  
Value: |

Log Reference UI-Element Rollup Info Clear

```
[info] Executing: |type | userName | user |  
[info] Executing: |type | password | user1 |  
[info] Executing: |clickAndWait |  
//input[@value='Login'] | |  
[info] Executing: |assertTextPresent | Welcome, user! |  
|  
[error] false
```

# Hibakezelés tesztelése hibainjektálással

# Hibainjektálás: Célkitűzések

- Biztonságkritikus rendszerek
  - Hibakezelés tesztelése is szükséges
    - Fail stop rendszerek: Hibadetektálás
    - Fail operational rendszerek: Hibatűrés
- Megvalósítási lehetőségek
  - Valós hibák hatásának megfigyelése (naplózás): Véletlen hibák esetén nehézségek
    - Hosszú idejű működtetés szükséges,
    - vagy nagyszámú megfigyelés szükséges
  - Hibainjektálás: Valóságban várható hibák bevitele
    - Prototípuson vagy modellen elvégezhető
    - Valós hibák „gyorsított módon” (stressz teszteléshez hasonló)

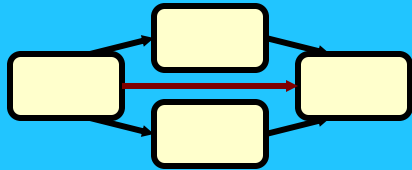
# Hibainjektálás: Módszerek

- **Hardver hibák injektálása:**
  - Hardver úton: Precíz (valós hibaokhoz közeli), de drága
    - Jelek közvetlen módosítása (tápfeszültség is)
    - Besugárzás (nehéz-ion, neutron), hőmérsékleti hatások
  - Szoftver úton: Olcsóbb, de kevésbé valóságos
    - **Hatások** szoftver emulációja: CPU regiszterek, memória módosítása
- **Szoftver hibák injektálása:**
  - Forráskód mutációja (pl. gyakori programozói hibák)
- **Modell alapú injektálás: Szimulációval vizsgálható**
  - Hardver: VHDL, Verilog modellek alapján
  - Szoftver: UML, állapotgép modellek

# Hibainjektáló eszközök

- **Hardver hibák:**
  - Közvetlen hozzáférés jelekhez: RIFLE, GOOFI
  - Besugárzás (nehéz-ion, neutron): speciális kamrák
  - Hatások szoftver emulációja: FIAT, FERRARI, FTAPE
- **Szoftver hibák (fault/error):**
  - Alacsony szintű emuláció: DOCTOR, Xception
  - Kód mutációs eszközök: FINE, DEFINE, G-SWFIT
  - Protokoll rétegek hibái: ORCHESTRA, Neko, WS-FIT
- **Modell alapú injektálás:**
  - VHDL, HDL szint: FOCUS, MEFISTO
  - Komponens szint (CPU, diszk, memória): DEPEND

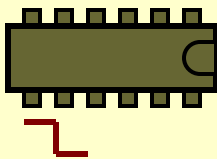
# A hibainjektálás valóságghűsége



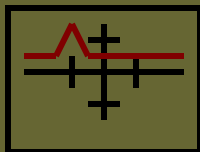
Hibaszimuláció a forráskódban  
vagy modell szinten a fejlesztőeszközben



Regisztartalom módosítása,  
memóriakép felülírása



Síneken haladó vagy áramkörök  
lábán megjelenő jelek módosítása



Radioaktív sugárzás, ioninjektálás,  
tápfeszültség zavarása, hőmérséklet

Ár,  
valóságghűség



# Robusztusság tesztelés



# 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

# 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 érték használható extrém értéként
  - Létrehozáshoz extrém paraméterek megadása
- **Scenario mutációval generált bemenetek**
  - Az extrém bemenetek állapottól függően adhatók ki
  - Sorrendi, kihagyási, időzítési hibák is definiálhatók

# Robusztusság teszt kimenetek értékelése

- Specifikációtól eltérő működés
  - Sokszor nincs előírt érték
  - Elvárt eredmények egyszerűsített kezelése szükséges
- Klasszikus kategóriák: **CRASH**
  - **Catastrophic:** A teljes rendszer összeomlik / újraindul
  - **Restart:** Az adott alkalmazás újraindulása
  - **Abort:** Az adott alkalmazás leáll
  - **Silent:** Hibajelzés nélküli érvénytelen művelet
  - **Hindering:** Érvénytelen hibakód
- Nem robusztusság hiba:
  - Érvényes hibakód visszaadása

# 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

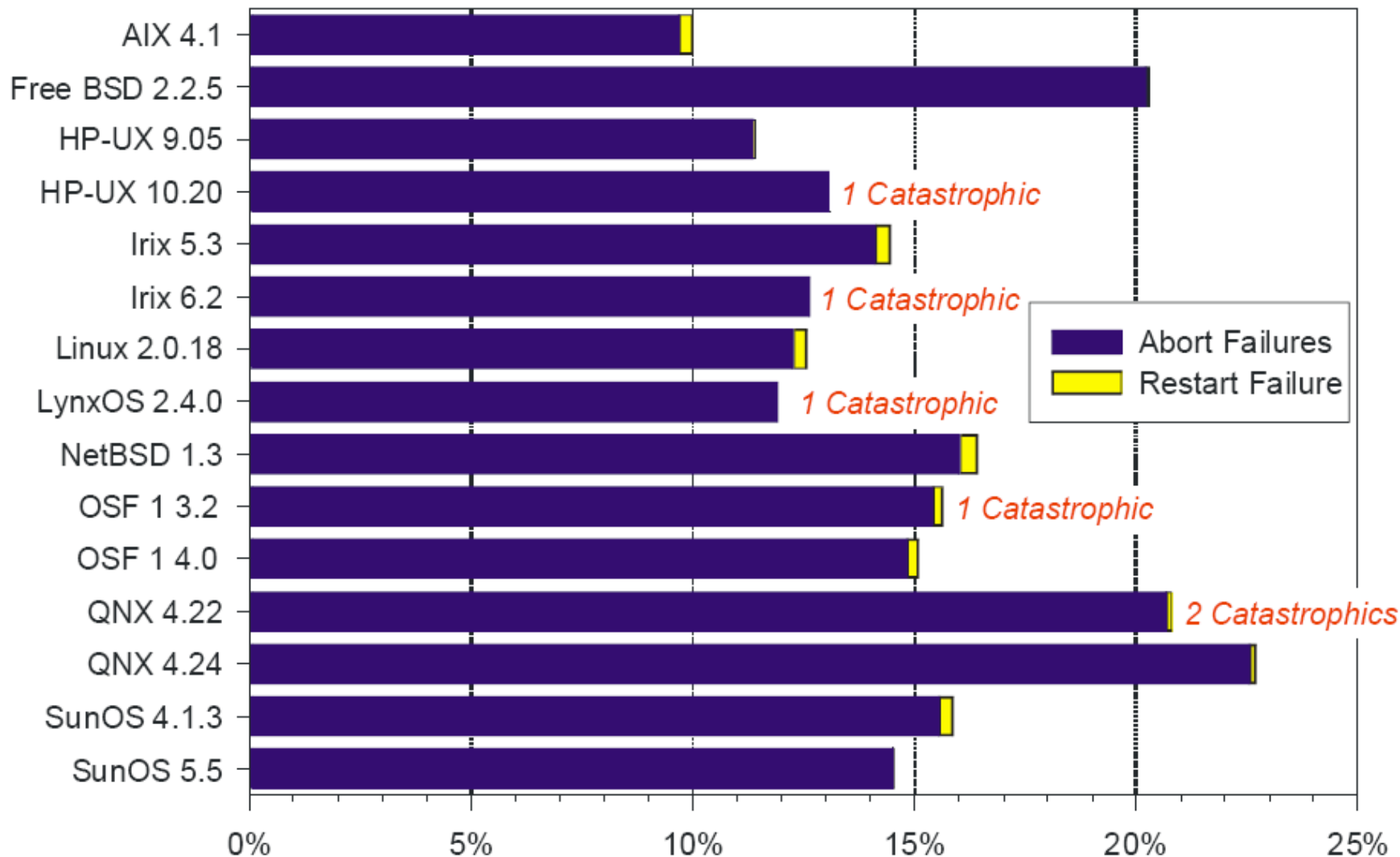
# Típus-specifikus tesztelés: Ballista – bemenetek

- Szélső és extrém értékek beállítása

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)			

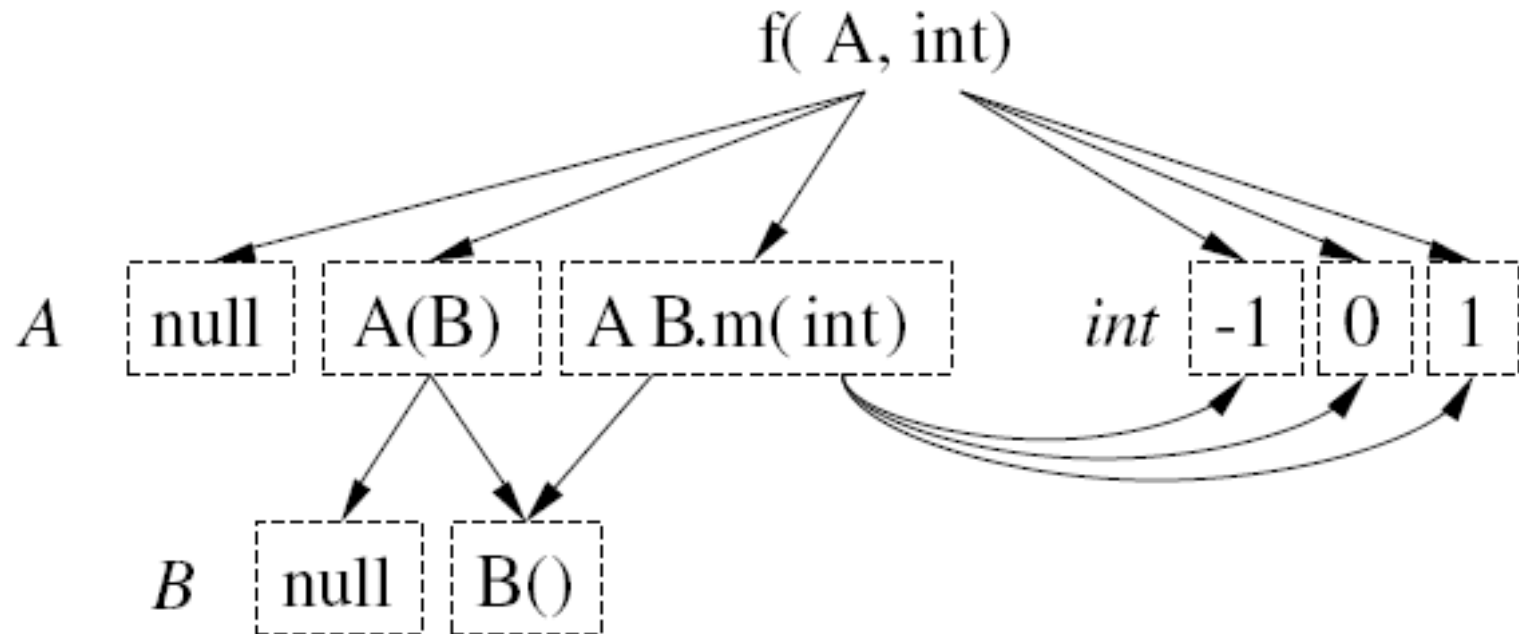
# Típus-specifikus tesztelés: Ballista – eredmények

Ballista Robustness Tests for 233 Posix Function Calls



# OO programok tesztelése: JCrasher – paraméterek

- Hogyan generáljunk adott típusú (extrém) objektumot?



# A módszerek fejlődése

