Méréstechnika és Információs Rendszerek Tanszék

Automatikus tesztgenerálás modell ellenőrző segítségével

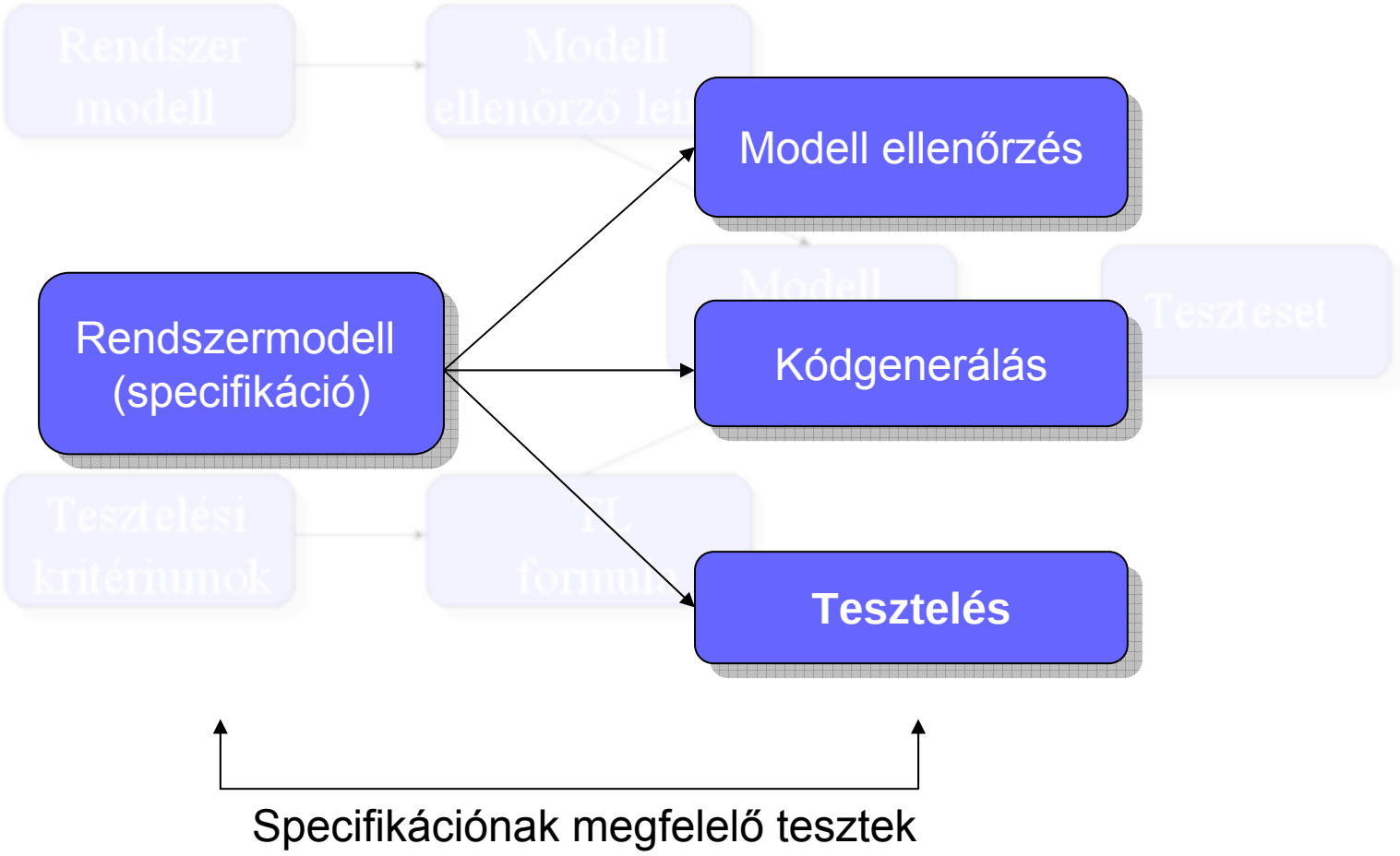
Micskei Zoltán
műszaki informatika, V.

Konzulens: Dr. Majzik István

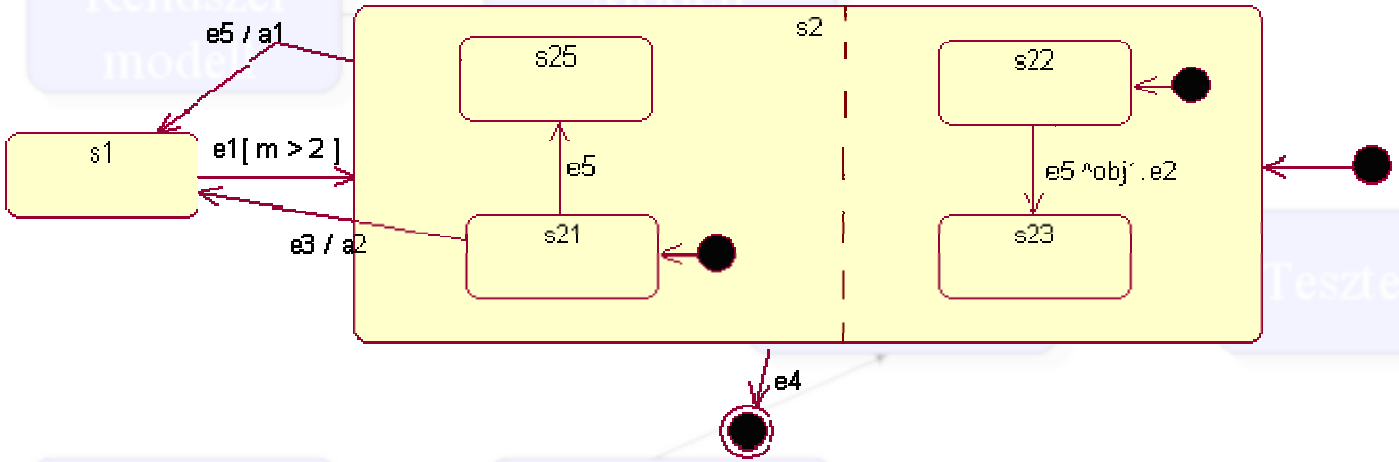
Tesztelés

- Célja: a rendszerben lévő hibák megtalálása
- Módszer: reprezentatív tesztesetek futtatása, és az eredmények összevetése a tesztesetben eltárolt várt kimenettel
- Problémák a klasszikus tesztelési módszerekkel:
  - Sok energia kézzel előállítani a teszteseteket.
  - A kész kódhoz készített tesztek nem garantálják a követelményeknek megfelelő működést.
  - Könnyű kihagyni lényeges teszteseteket.

Modell alapú fejlesztés



Beágyazott rendszerek modellezése

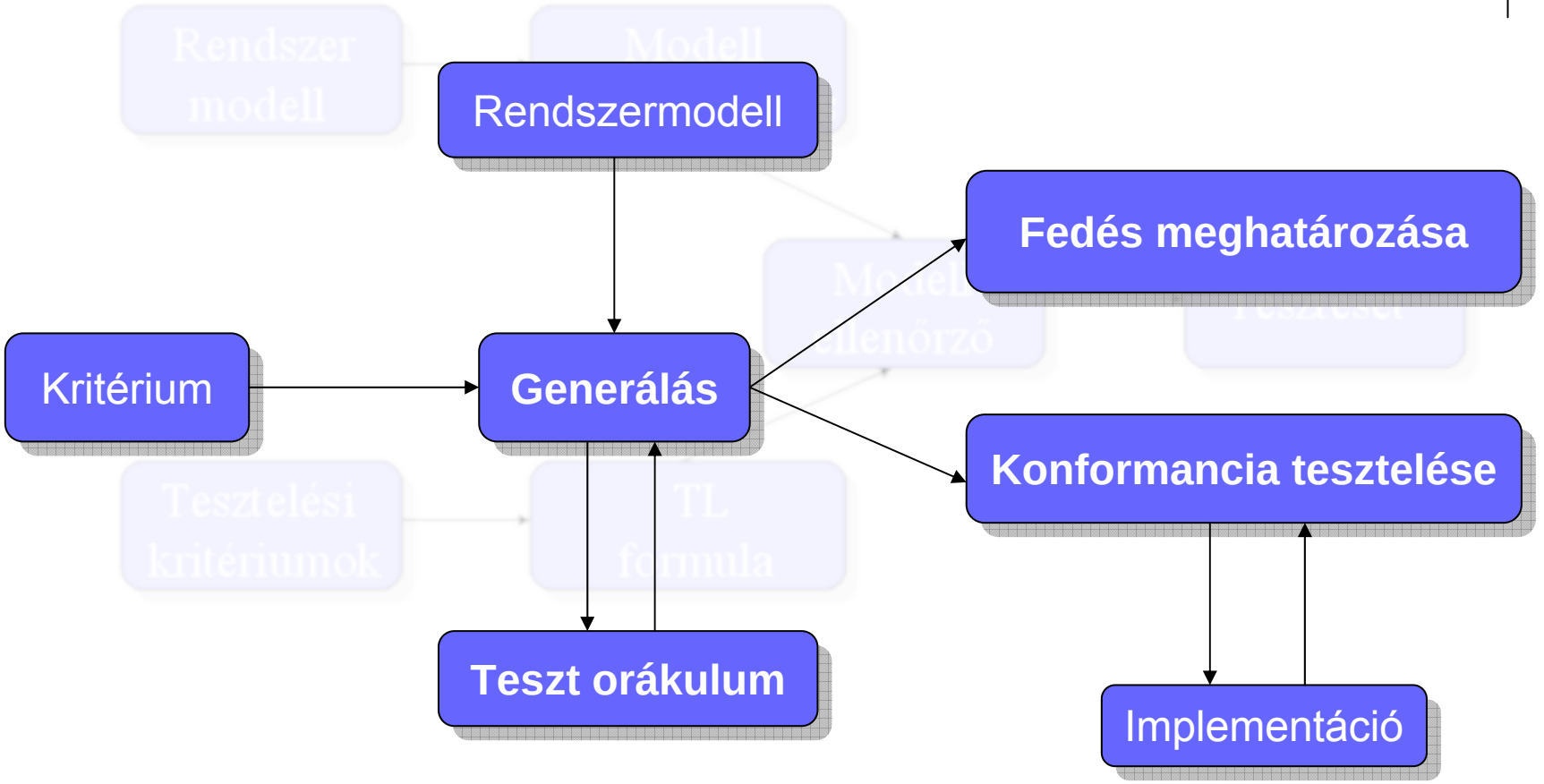


- Vezérlésorientált működés
- Esemény – válasz
- Állapottérkép használata a modellezéshez (UML, STATEMATE)

Rendszerek ellenőrzése: modell ellenőrző eszközök

- A rendszer állapotterének teljes bejárásával ellenőrzik bizonyos tulajdonságok meglétét
- A helyes működés bizonyítására szolgálnak
- Tipikus feladatok:
  - Élőség és holtpontmentesség vizsgálata
  - Egyedi, rendszerspecifikus kritériumok megadása
    - Állapotok, események elérhetősége, sorrendje
    - Általában temporális logikai formulákkal

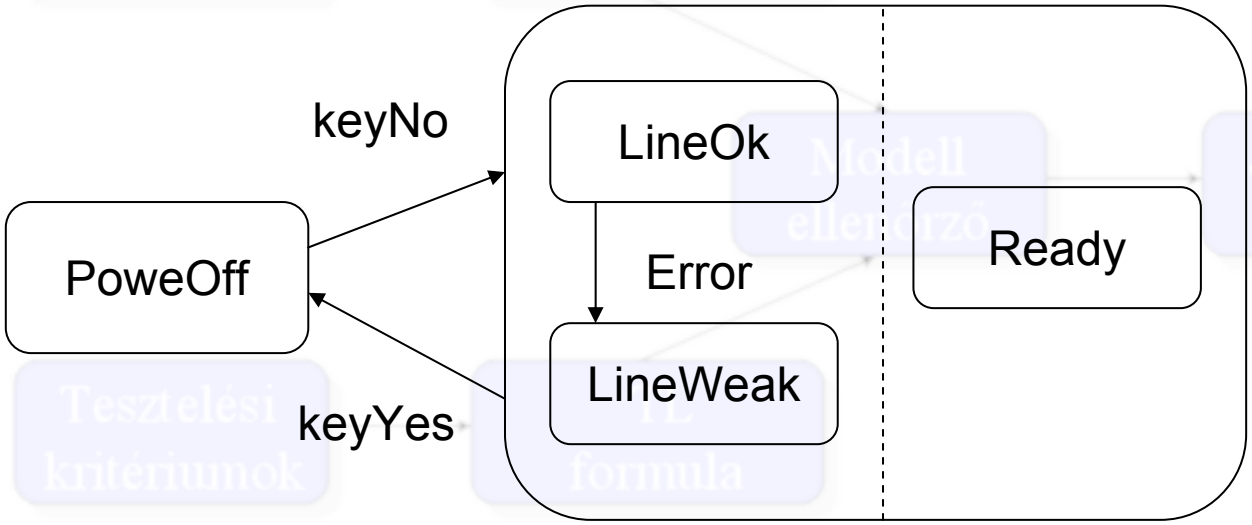
Modell alapú tesztelési módszerek



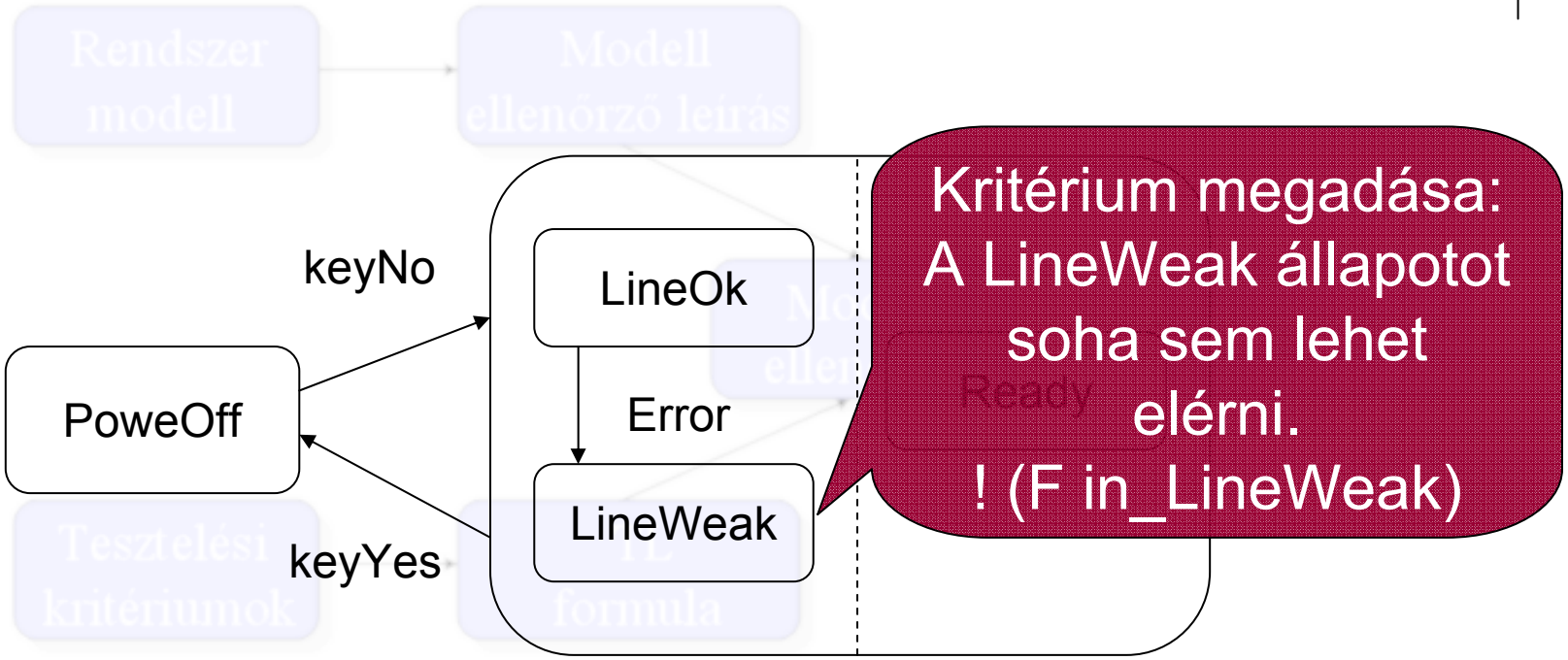
Tesztgenerálás modell ellenőrzővel

- Az állapottér bejárása:
  - Szabványos fedési kritérium alapján
  - Tesztelő által megadott kritérium alapján
- Tipikus kritériumok:
  - Állapotok, átmenetek lefedése
  - Változó definiálások és felhasználások lefedése
  - Be- és kimenő átmenet-párok lefedése egy-egy állapothoz

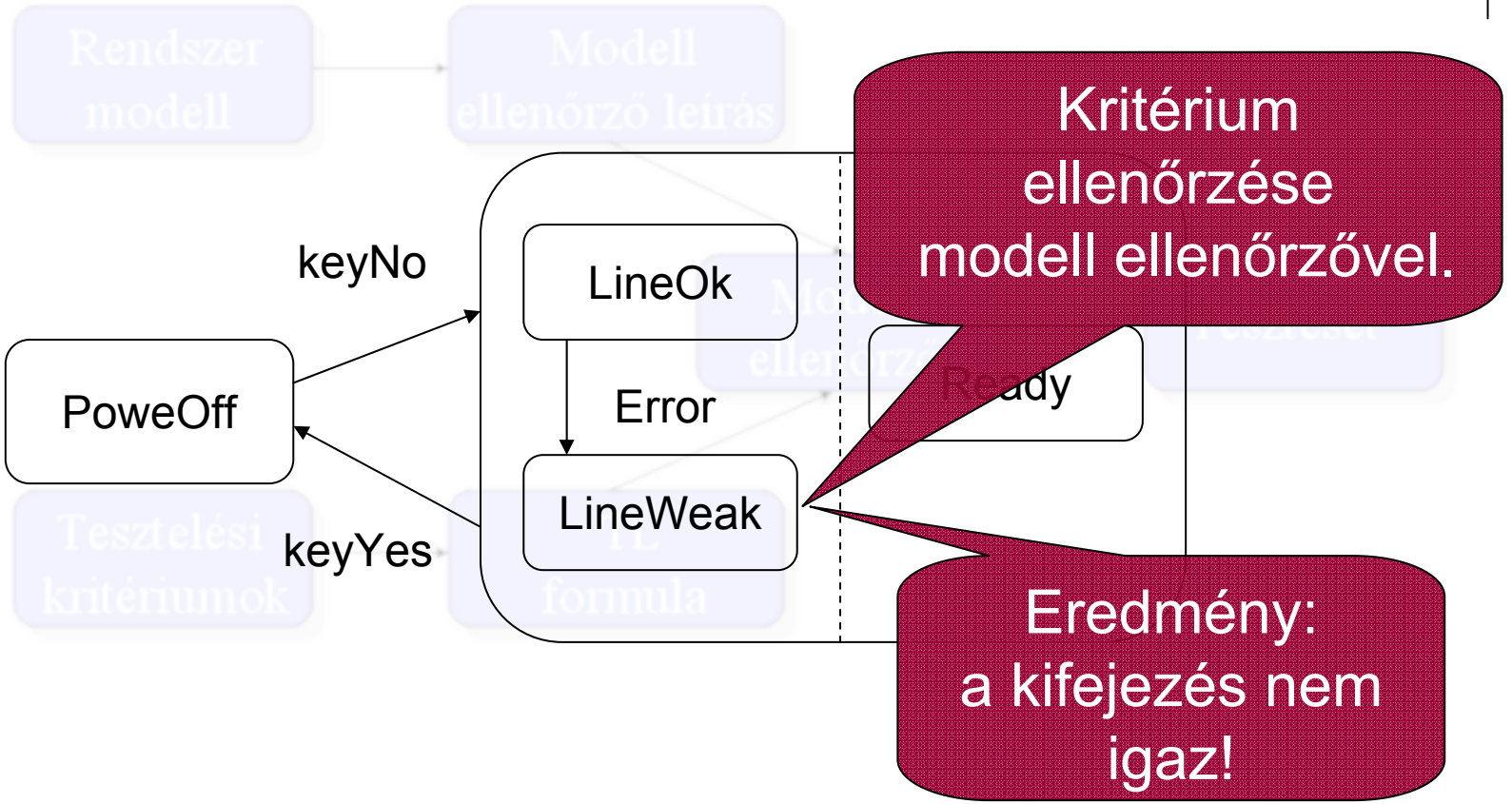
Hogyan használható a modell ellenőrző teszt generálásra?



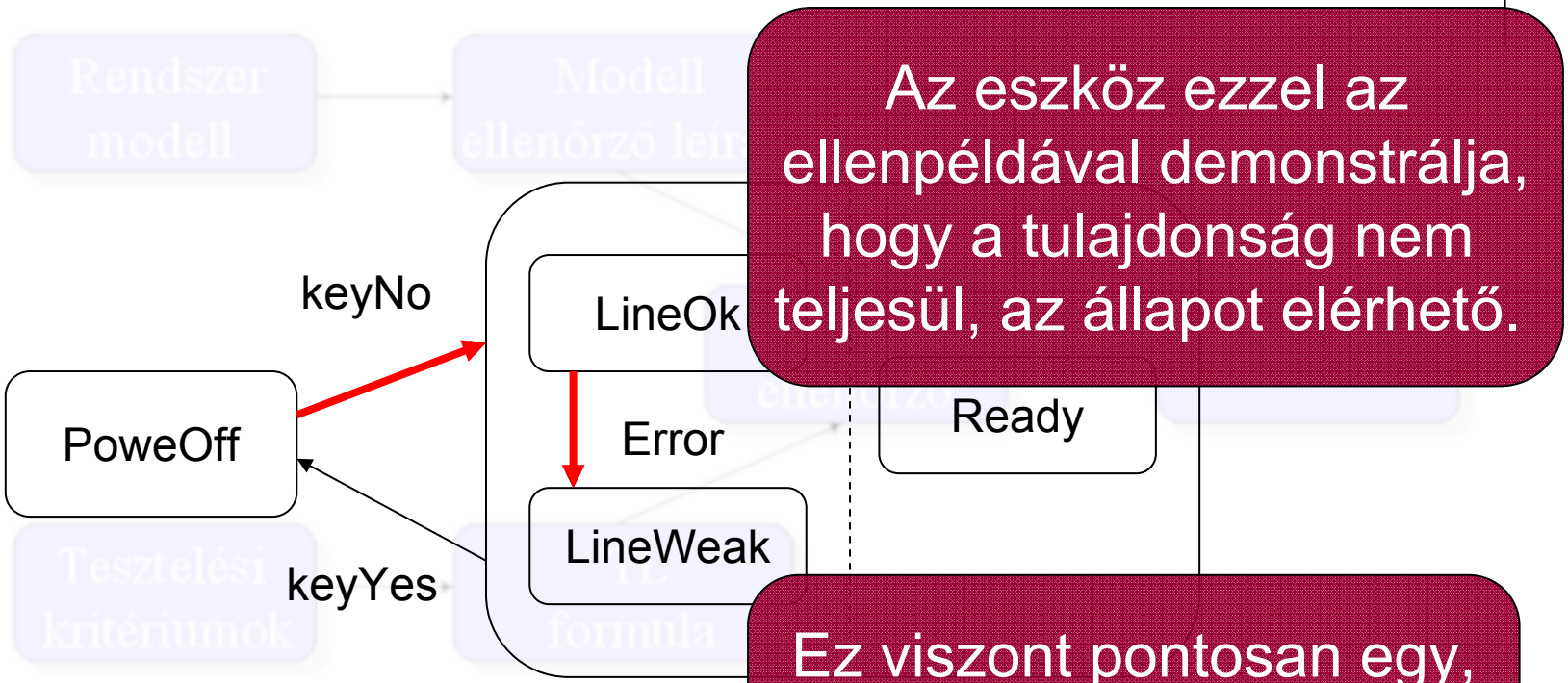
Hogyan használható a modell ellenőrző teszt generálásra?



Hogyan használható a modell ellenőrző teszt generálásra?



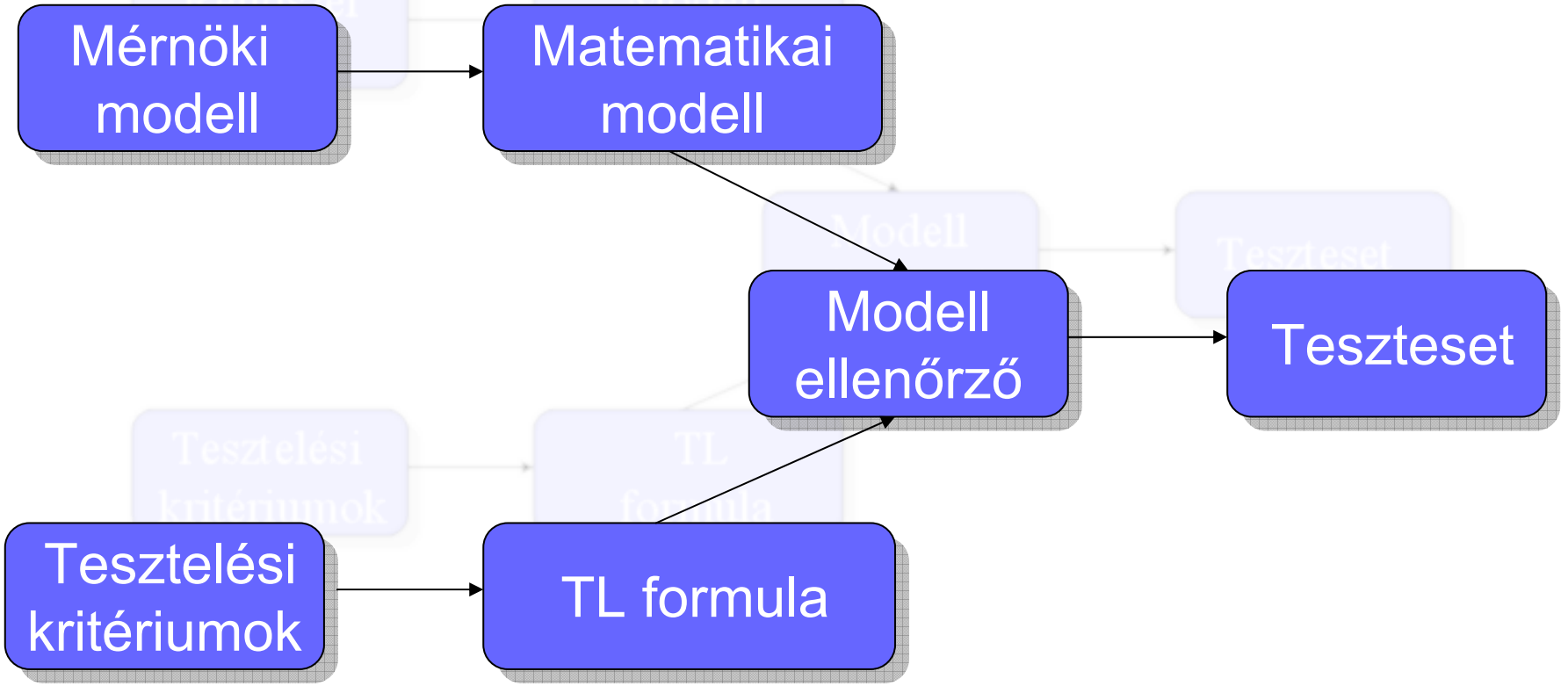
Hogyan használható a modell ellenőrző teszt generálásra?



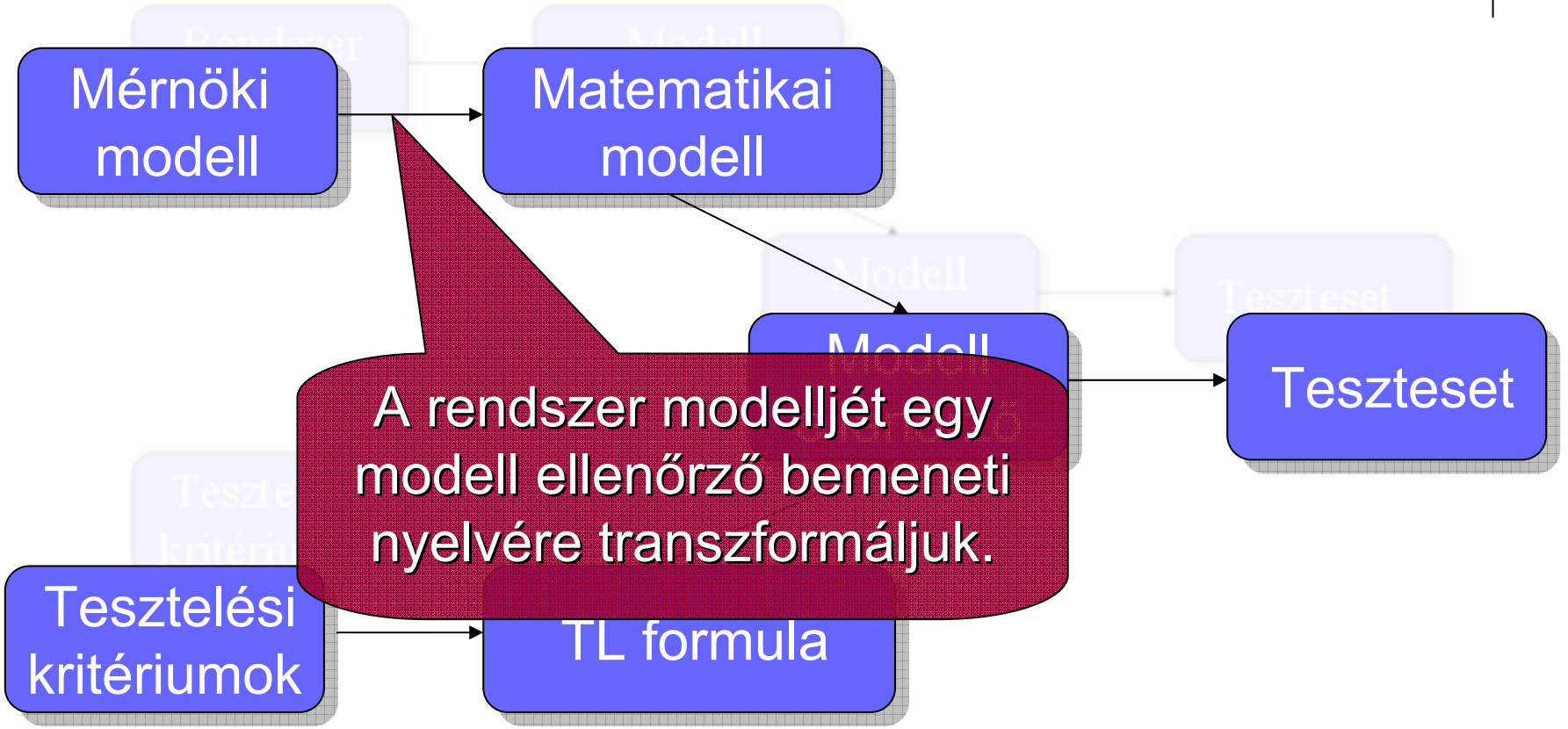
Az eszköz ezzel az ellenpéldával demonstrálja, hogy a tulajdonság nem teljesül, az állapot elérhető.

Ez viszont pontosan egy, a LineWeak állapotot lefedő teszteset!

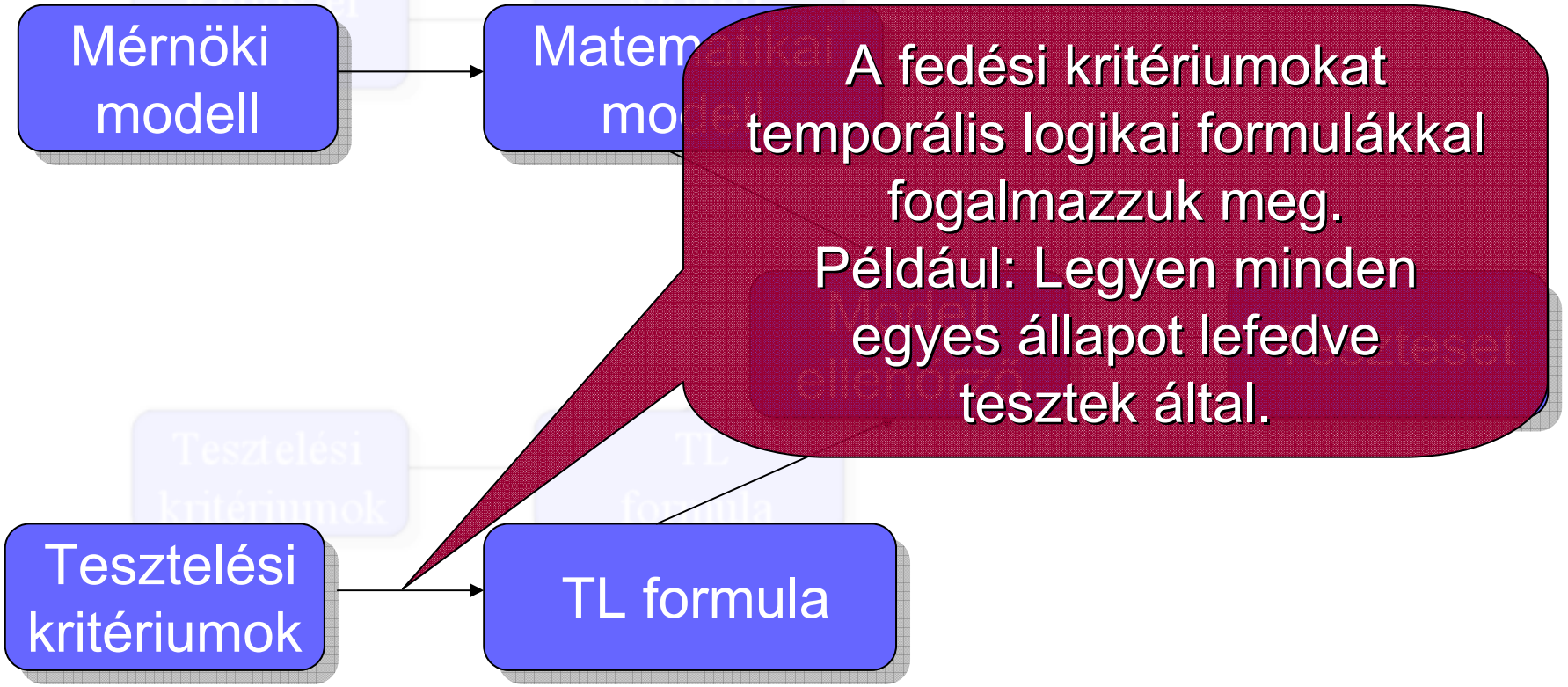
Automatikus tesztgenerálás



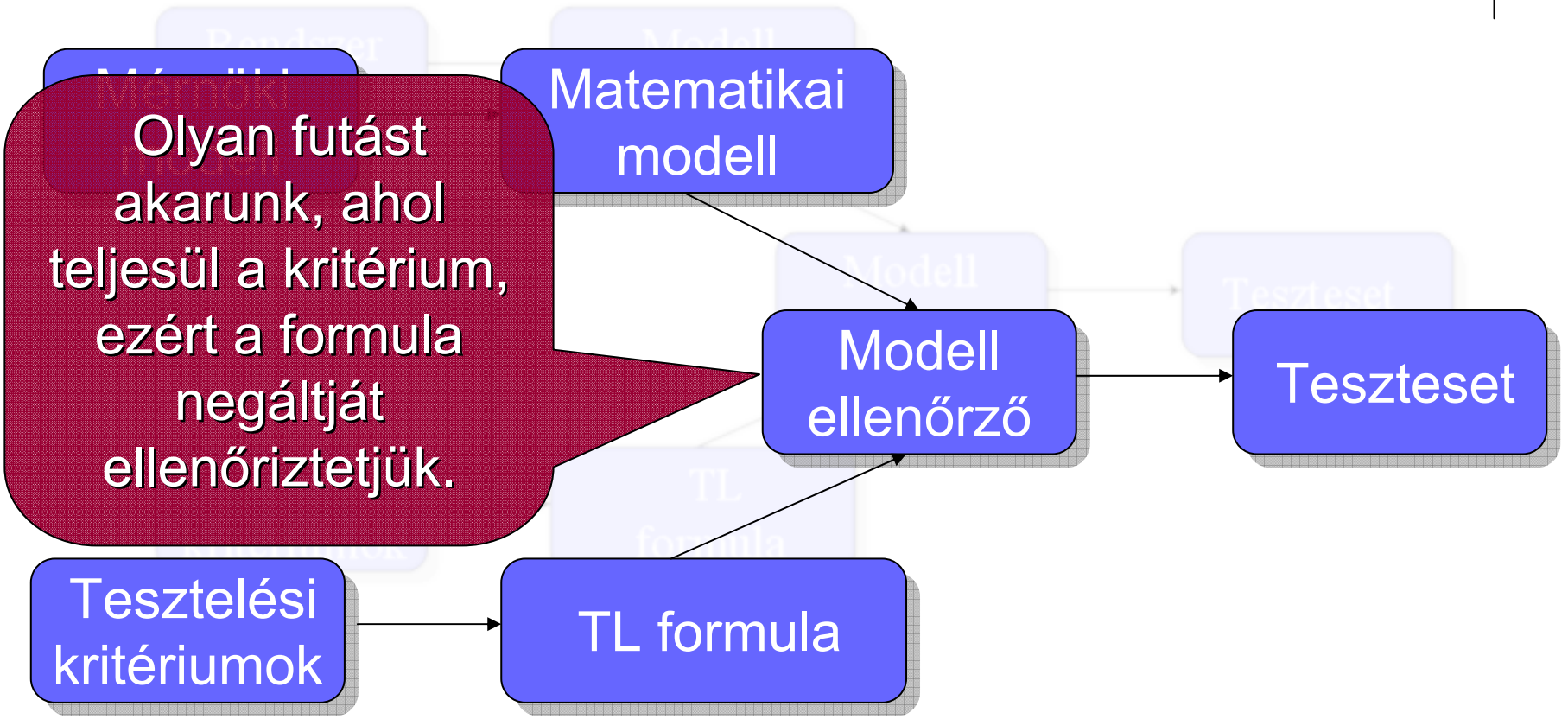
A működés menete – I.



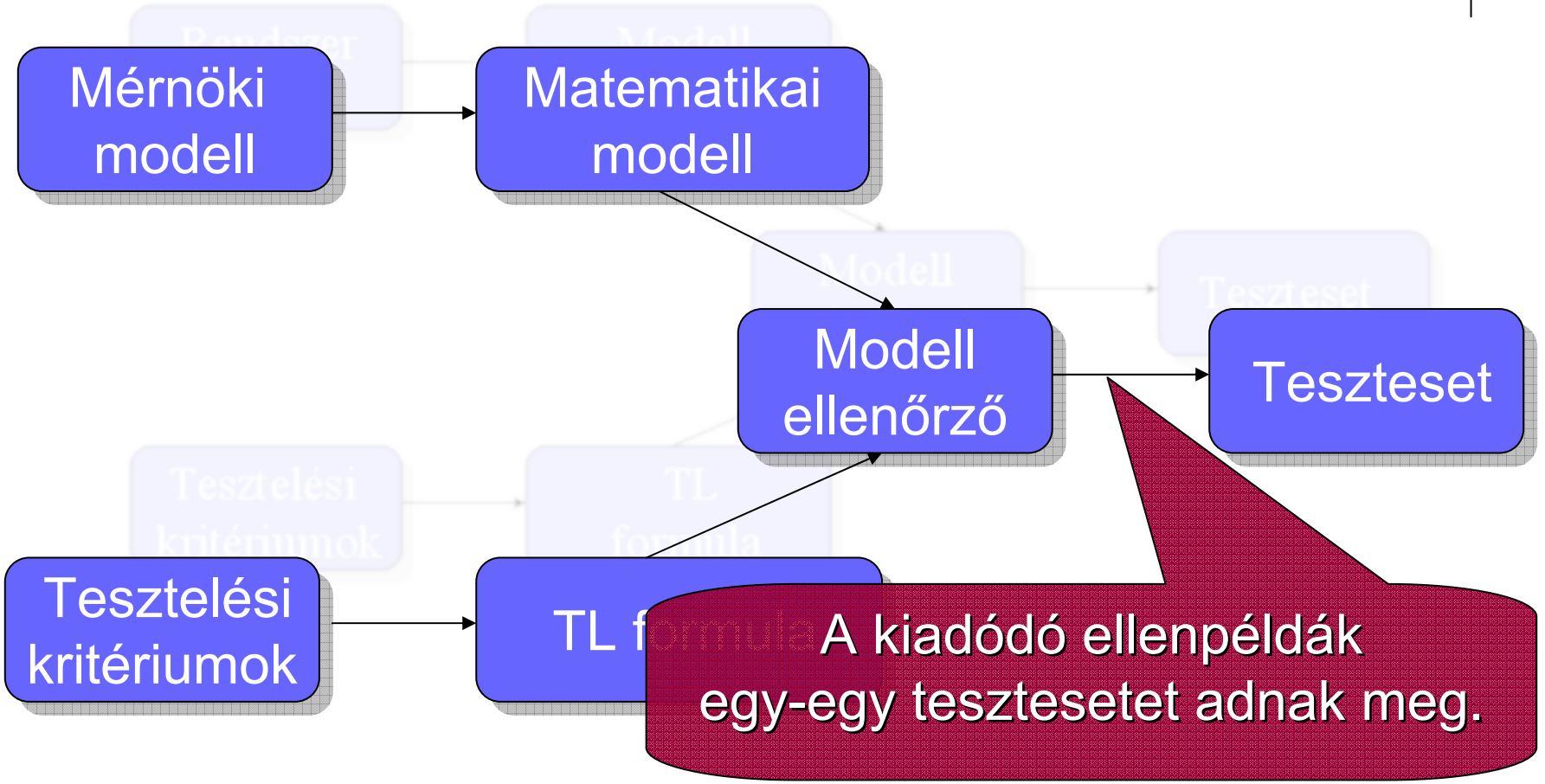
A működés menete – II.



A működés menete – III.



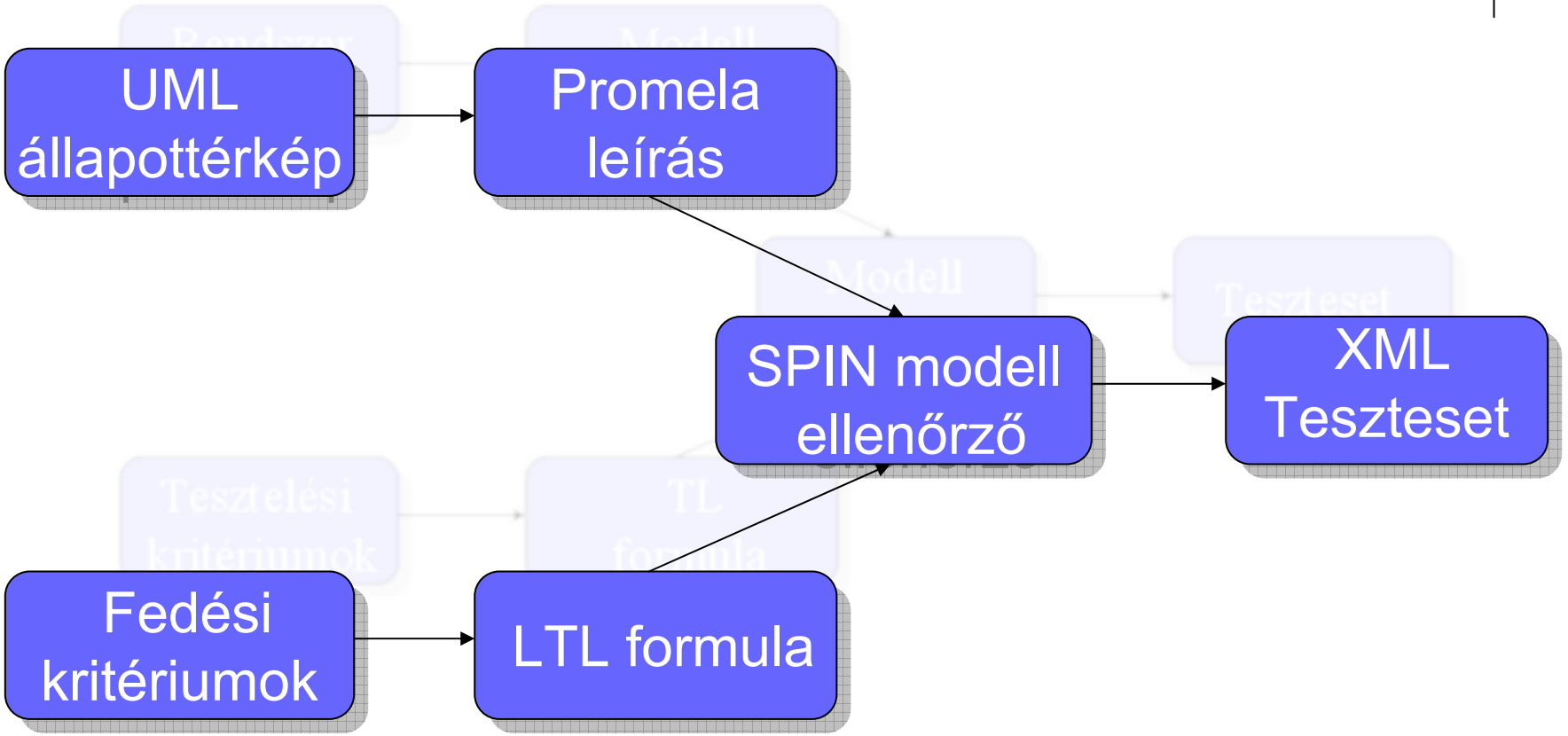
A működés menete – IV.



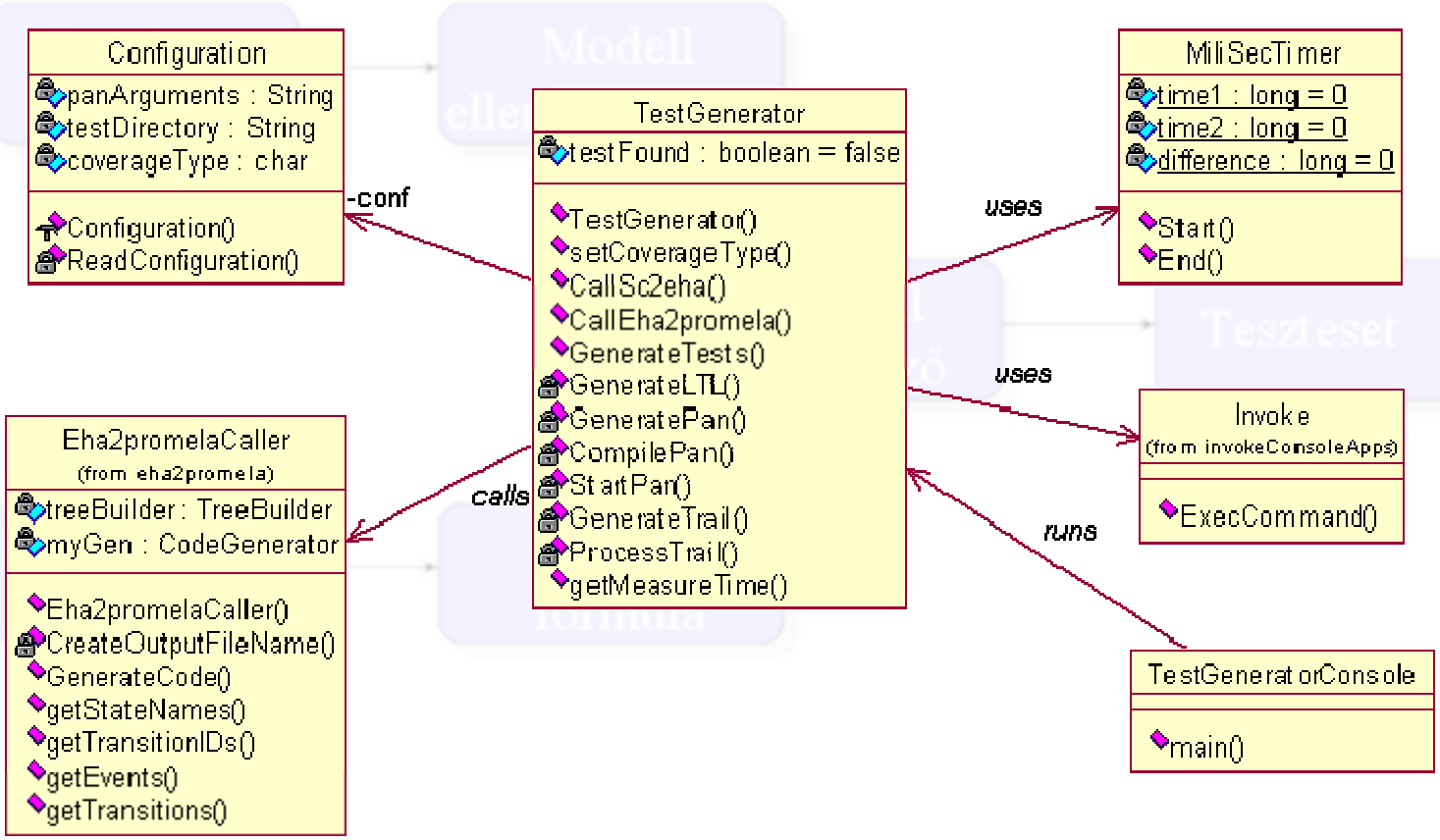
Implementáció

- Tesztgeneráló program készült:
  - A módszer működésének demonstrálása
  - Hatékonyság vizsgálata
- Működési lépései:
  - TL kifejezések generálása a fedési kritériumból
  - Rendszermodell transzformálása a modell ellenőrző bemeneti nyelvére
  - Modell ellenőrző paraméterezése, kifejezések ellenőrzése
  - Kimenet szűrése, és a tesztesetet leíró fájl előállítása

Implementált környezet



Tesztgenerátor osztálydiagram



Generált kimenet példa teszteset

Address G:\tests\t187.xml

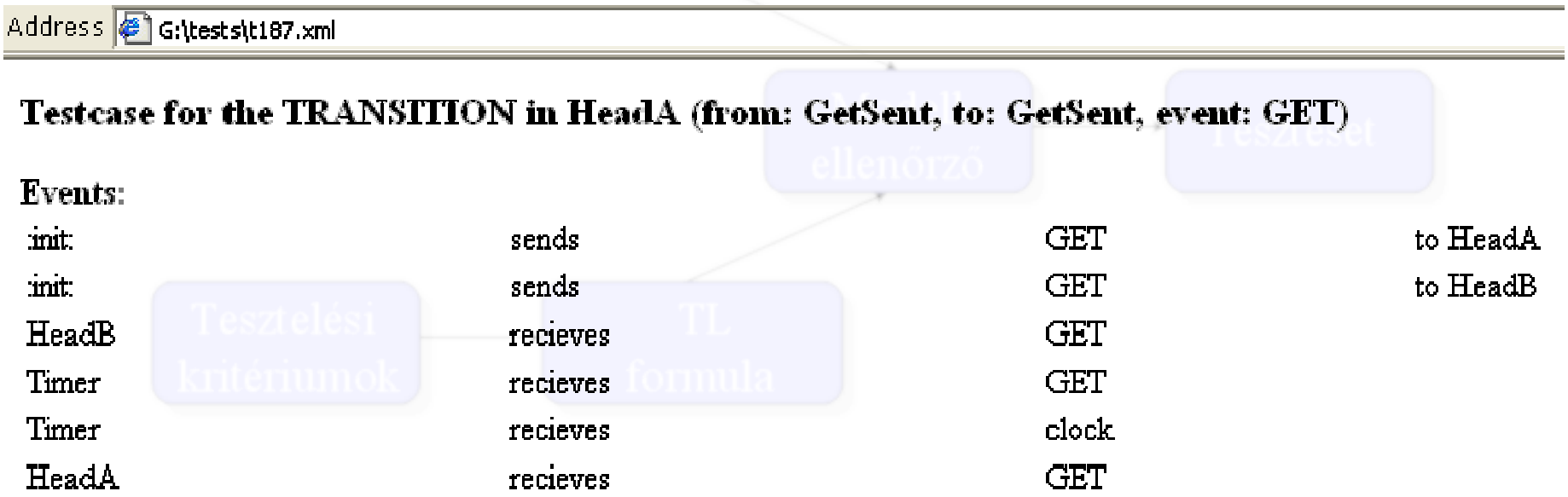
```
<?xml version="1.0" encoding="UTF-8" ?>
- <testcase>
  - <covering>
    - <transition event="GET" guard="" from="GetSend" to="HeadA">
        <action name="aGETACK_INV" />
      </transition>
    </covering>
  - <events>
      <event action="send" from=":init:" to="HeadA" name="GET" />
      <event action="send" from=":init:" to="HeadB" name="GET" />
      <event action="receive" from="" to="HeadB" name="GET" />
      <event action="receive" from="" to="Timer" name="GET" />
      <event action="receive" from="" to="Timer" name="block" />
      <event action="receive" from="" to="HeadA" name="GET" />
    </events>
</testcase>
```

Megadja, hogy mit fed le a teszteset

A tesztesethez tartozó eseményszekvencia

Generált kimenet példa teszteset

XSLT segítségével HTML formátumra alakítva

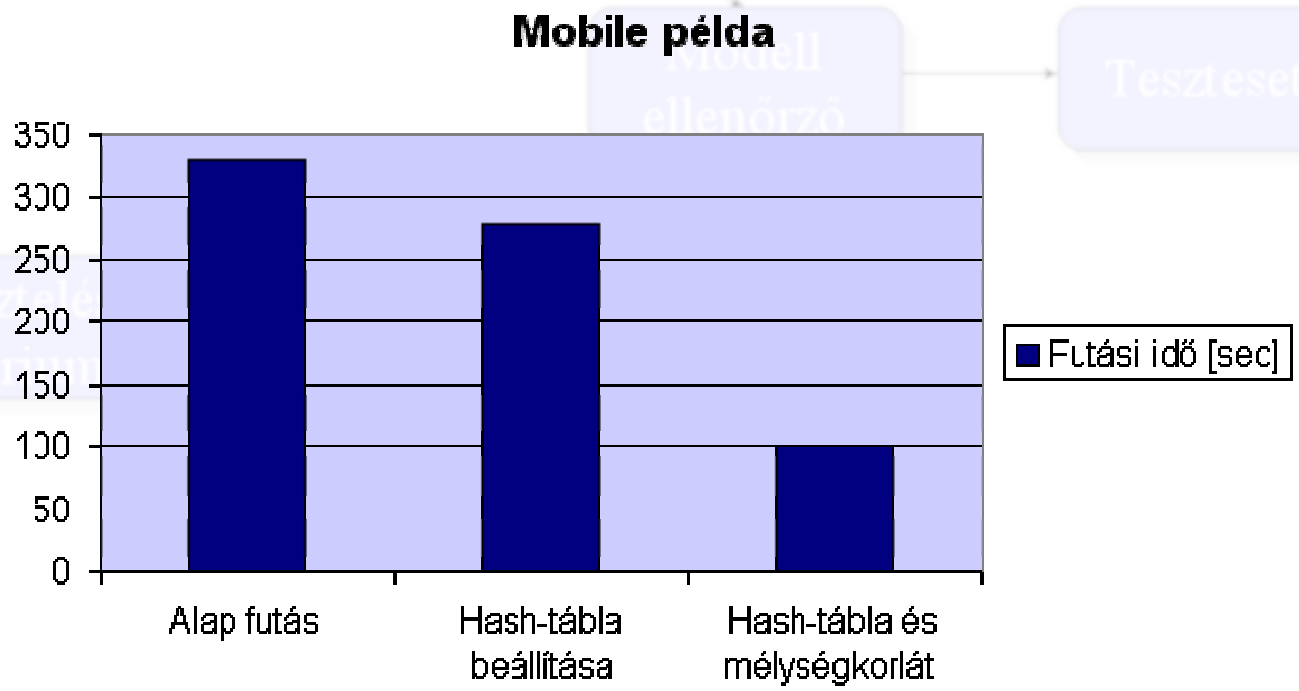


Optimalizáció

- A tesztgenerálás célja minél gyorsabban, minél rövidebb ellenpéldát találni
  → speciális beállítások szükségesek.
- Lehetőségek:
  - SPIN-ben a felesleges ellenőrzések letiltása
  - Megfelelő mélységkorlát definiálása
  - Állapotokat tároló hash-tábla méretének helyes megadása

Mobile példa: Teljes állapotfedésű tesztek

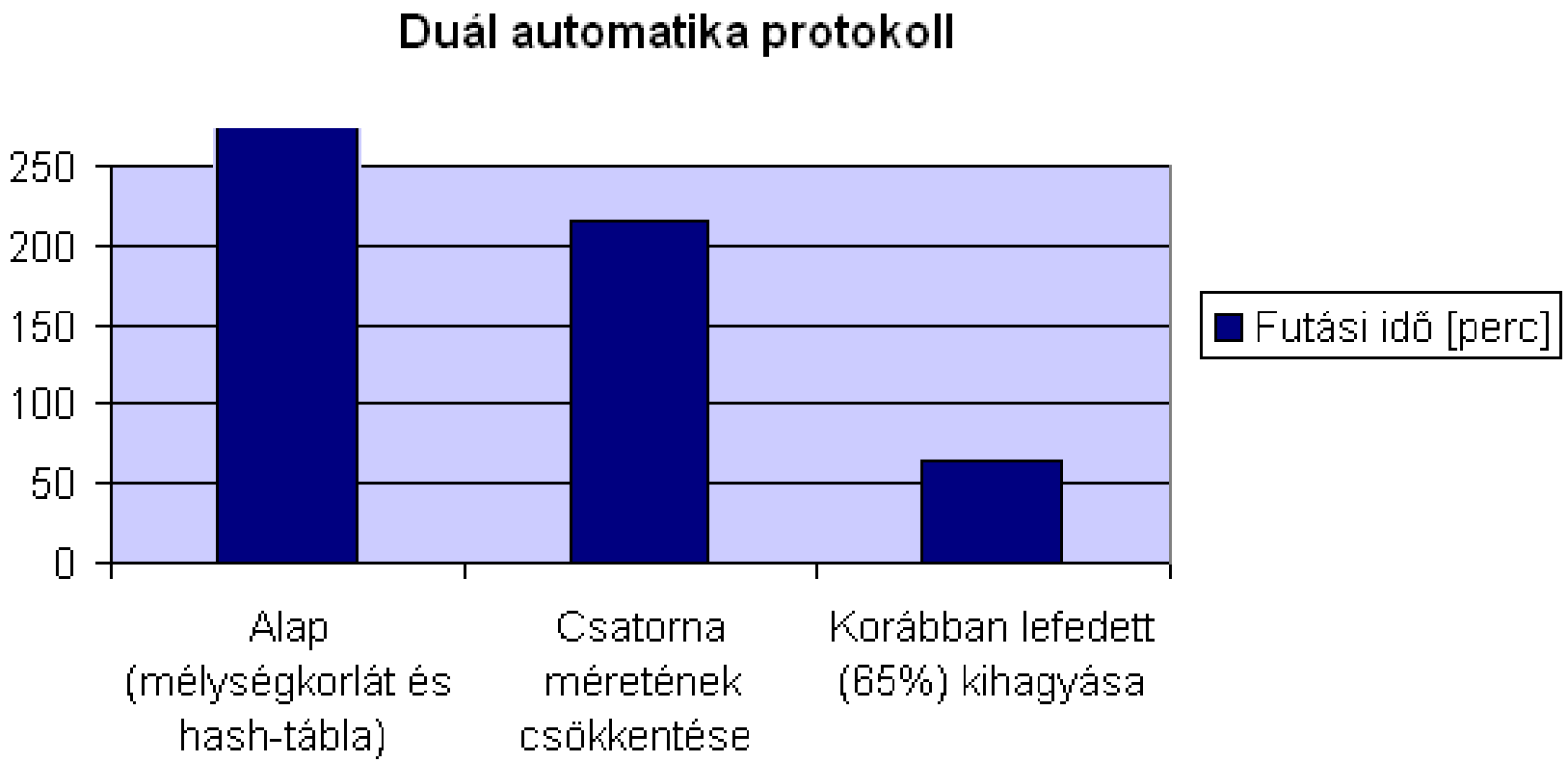
- Mobiltelefon vezérlését leíró állapottérkép
- 10 darab állapot, 21 darab átmenet



Duál automatika példa

- Valós protokoll
- 5 darab objektum, 31 állapot, 174 átmenet
- Komoly feladat, 2e+08 bejárandó állapot
- Más technikák is kellenek:
  - Állapottér tárolás tömörítve
  - Közelítő módszer alkalmazása (bitstate hashing)
  - Szűkítések a modellben:
    csatorna méret csökkentés
  - Korábban lefedett kritériumok kihagyása

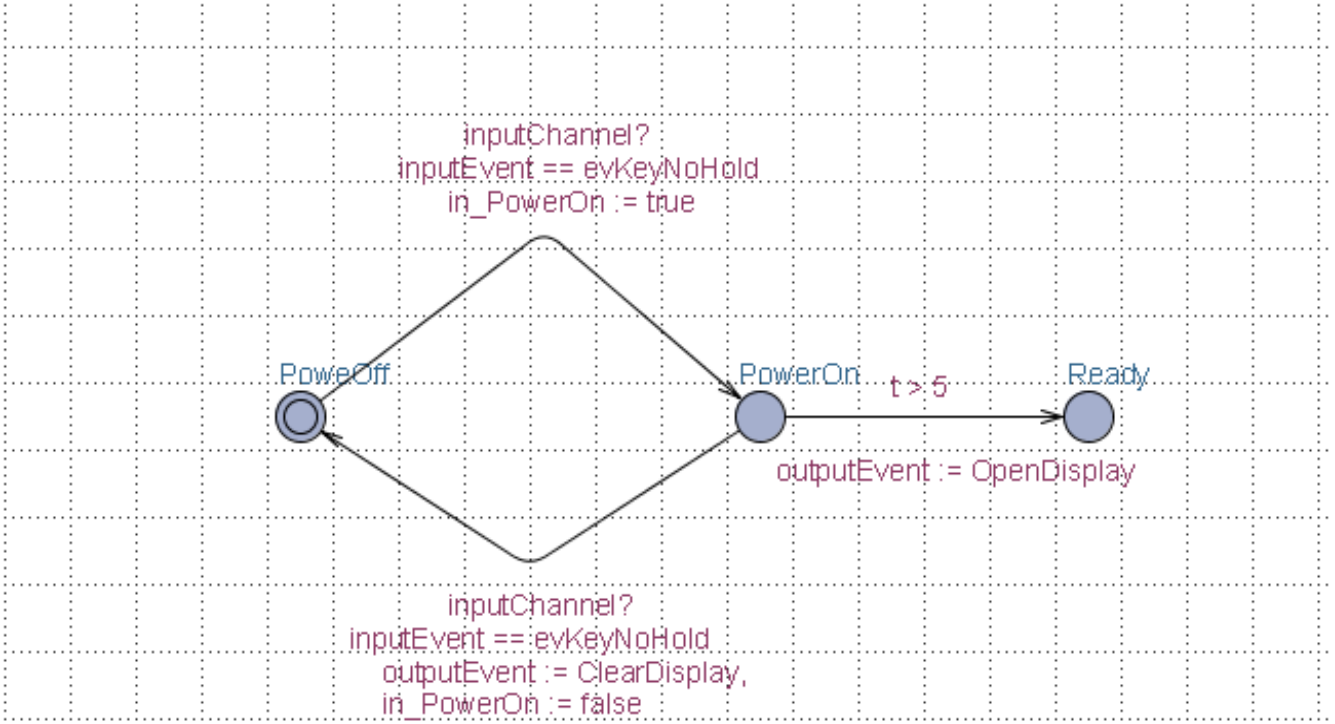
Duál automatika példa – Teljes állapotfedésű tesztek



Kiterjesztések

- Tesztkészlet optimalizálás
  - Legrövidebb/legkisebb tesztkészlet kiválasztása NP-teljes probléma
  - További heurisztikák alkalmazása: „mélyen” fekvő állapotok előbb, állapottér levágása
- Illesztés CASE eszközökhöz
  - Teszt menedzselő és futtató keretrendszerek pl. Rational TestManager

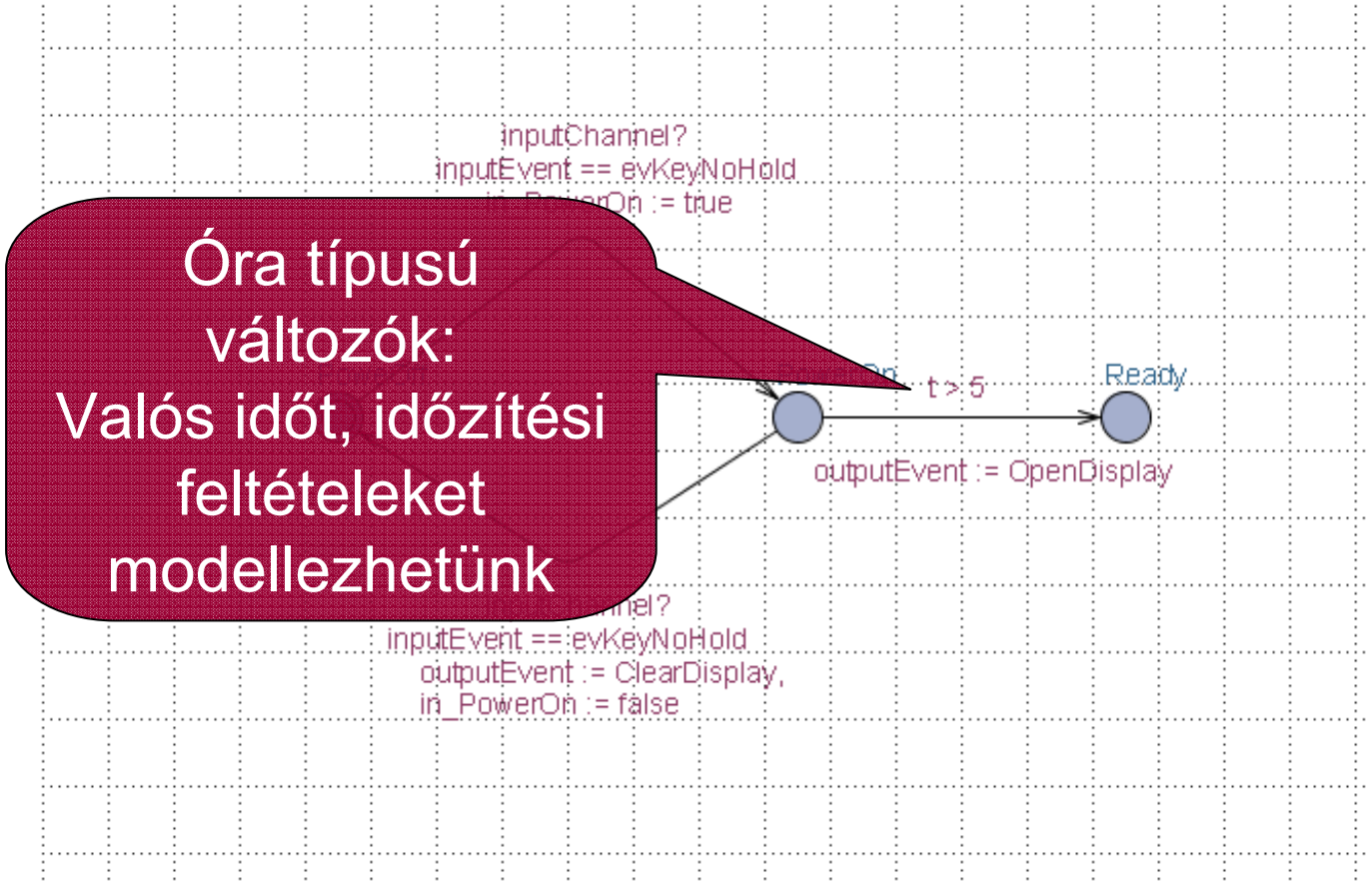
Valósidejű rendszerek



Speciális modell ellenőrző, Uppaal:

- Időzített automaták használata

Valósidejű rendszerek



Óra típusú változók:
Valós időt, időzítési feltételeket modellezhetünk

Generált tesztek

State:
(input.sending mobile.PowerOn mobile1.LineOK mobile2.CallWait)
t=0 inputEvent=28 outputEvent=14 in_PowerOn=1 #depth=5

Delay: 6

A teszt időzítési viszonyok is szerepelnek a generált tesztesetben

State:
(input.sending mobile.PowerOn mobile1.LineOK mobile2.CallWait)
t=6 inputEvent=28 outputEvent=14 in_PowerOn=1 #depth=5

Transitions:
input.sending->input.sendInput { 1, inputChannel!, 1 }
mobile2.CallWait->mobile2.VoiceMail { inputEvent == evKeyYes && t > 5
&& in_PowerOn, inputChannel?, 1 }

Eredmények összefoglalása

- Modell alapú tesztelés támogatása
- Automatikus tesztgeneráló eszköz eseményvezérelt, beágyazott rendszerekhez: tesztek (bejárási útvonalak) generálása
- Többféle teszt fedési kritérium figyelembe vétele
- Teszt időzítés automatikus generálása valósidejű rendszerek esetén
- Modell ellenőrző teszt-specifikus paraméterezése
- Méretkorlát: A modell ellenőrző képességei (kb. 10^8 állapot)

Köszönöm a figyelmet!

Kérdések