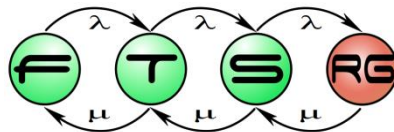


Modellek ellenőrzése

**Budapest University of Technology and Economics
Fault Tolerant Systems Research Group**



Ariane 5 hordozórakéta

- A legerősebb európai hordozórakéta



Ariane 5 hordozórakéta

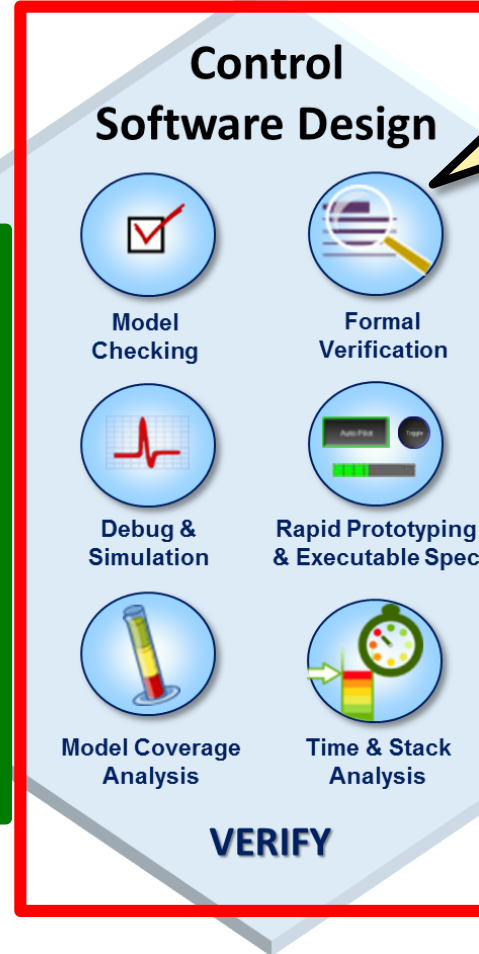
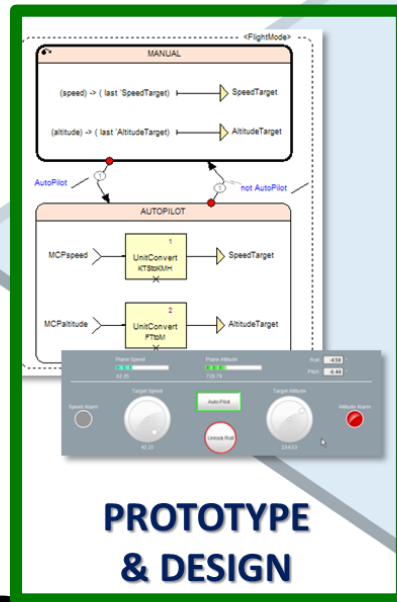
- 1996. június 4-én 37 másodperccel a kilövés után megsemmisítette magát
 - A szállított négy műhold is megsemmisült
 - \$370 milliós kár



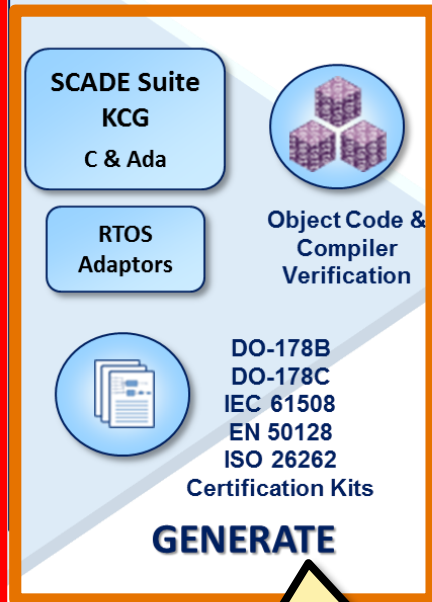
Ariane 5 hordozórakéta

- 1996. június 4-én 37 másodperccel a kilövés után megsemmisítette magát
 - A szállított négy műhold is megsemmisült
 - \$370 milliós kár
- A világ (egyik) legdrágább szoftverhibája
 - Elsődleges ok:
 - 64 bites szám sikertelen konverziója 16 bitesre
 - Másodlagos ok:
 - Soha nem tesztelték együtt a modulokat**

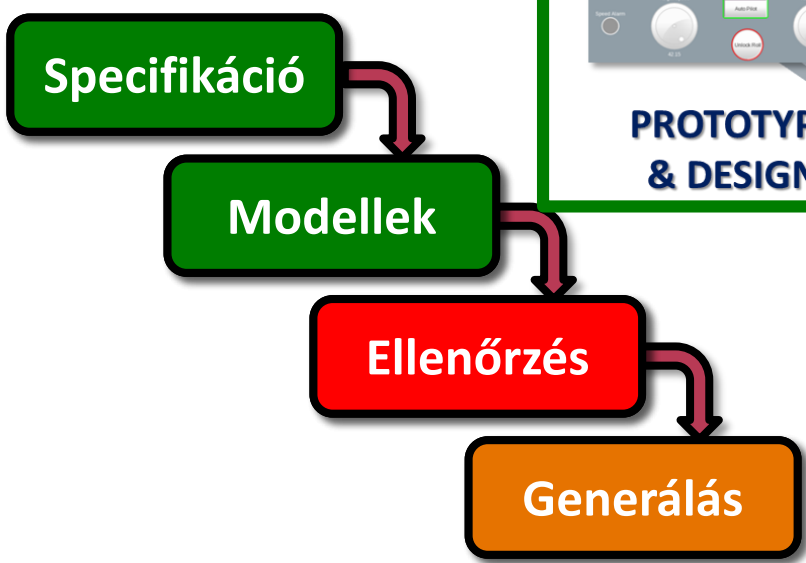
Példa: Esterel SCADE



Modellek ellenőrzése



Repülőgép,
kritikus ES,
autó



Tartalom

Alapfogalmak

Statikus ellenőrzés

Tesztelés

Formális verifikáció

Kódgenerálás esetén

Modellek fejlesztése

Szoftverfejlesztés

Követelmények, specifikáció

Követelmények, specifikáció

Kezdeti modellek

Ré

Implementáció

Tesztelés

Tesztelés

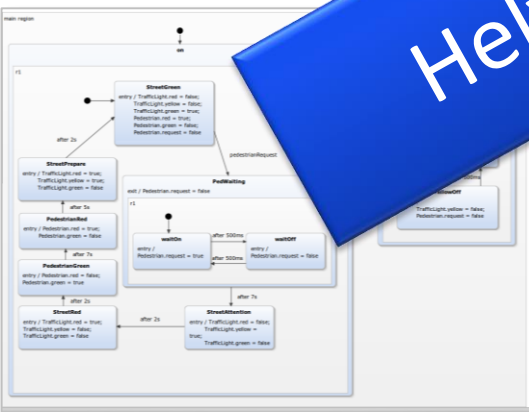
Karbantartás

Karbantartás

Helyes modell, helyesebb kód

```
import java.util.Vector;
import java.awt.*;
import java.awt.event.*;

public class InternetClient implements Serializable
{
    private Socket socket;
    private ObjectOutputStream stream;
    private ObjectInputStream stream;
    // Creates a connection to the internet
    public InternetClient(int port, String message)
    {
        // Connection to the internet
        // To read object
        // To write object
        // Int
        // Inte
        // Mess
    }
}
```



Alapfogalmak

Statikus ellenőrzés

Tesztelés

Formális verifikáció

ALAPFOGALMAK

Modellek és feladatok

- Szintézis:

Specifikációnak megfelelő modell?



- Analízis:

Modell viselkedése?



- Vezérlés:

Kívánt állapot hogy érhető el?



Helyesség

■ **Helyesség:**

modell vagy kód megfelelése a követelményeknek.

○ **Funkcionális helyesség:**

megfelelés a funkcionális követelményeknek

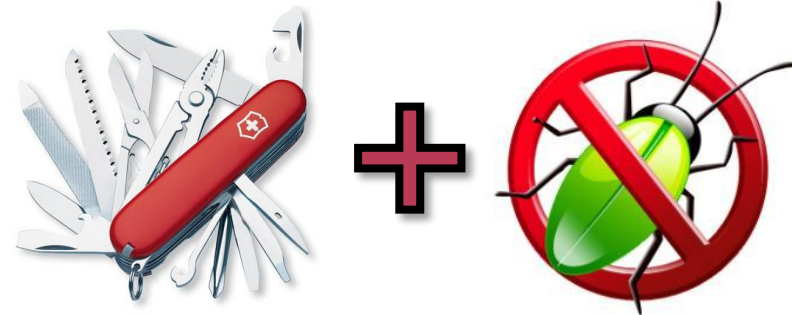
○ Nemfunkcionális követelmények ellenőrzése:
lásd teljesítménymodellezés előadás

■ **Szemponatok:**

○ Mindig képes teljesíteni a feladatot

○ Hibamentes

○ Nincs tiltott viselkedés



Funkcionális követelmények csoportosítása

- **Megengedett** viselkedés:
 - Milyen állapotokban lehet/nem lehet a rendszer
 - Milyen viselkedés tilos
 - Univerzális követelmények
 - Mindig igaznak kell lenniük
- **Elvárt** viselkedés:
 - Milyen állapotokba kell tudni eljutni
 - Milyen funkciókat kell tudnia a rendszernek
 - Egzisztenciális követelmények
 - Lehessen lehetőség a teljesülésükre

Funkcionális követelmények csoportosítása

■ **Megengedett** viselkedés:

- Milyen állapotokban lehet t/n
- Milyen viselkedés tilos
- Univerzális követelmények
 - Mindig igaznak kell lenniük

„Egy kereszteződés lámpái **soha nem lehetnek** egyszerre zöldek.”

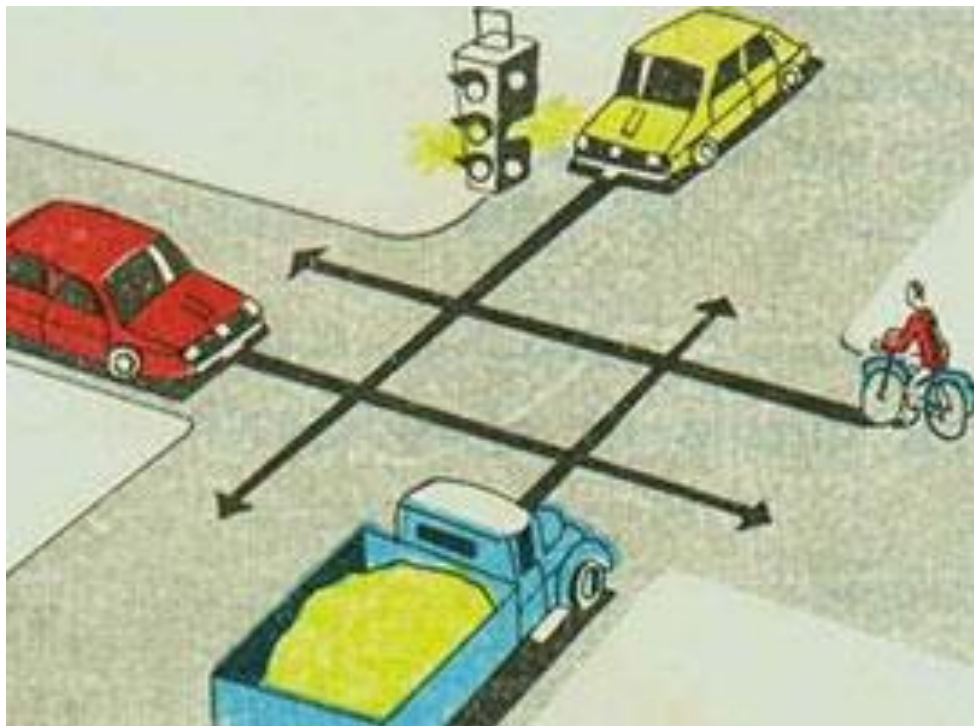
■ **Elvárt** viselkedés:

- Milyen állapotokba kell tudni
- Milyen funkciókat kell tudni
- Egzisztenciális követelmények
 - Lehessen lehetőség a teljesül

„A lámpa **legyen képes** zöldre váltani.”

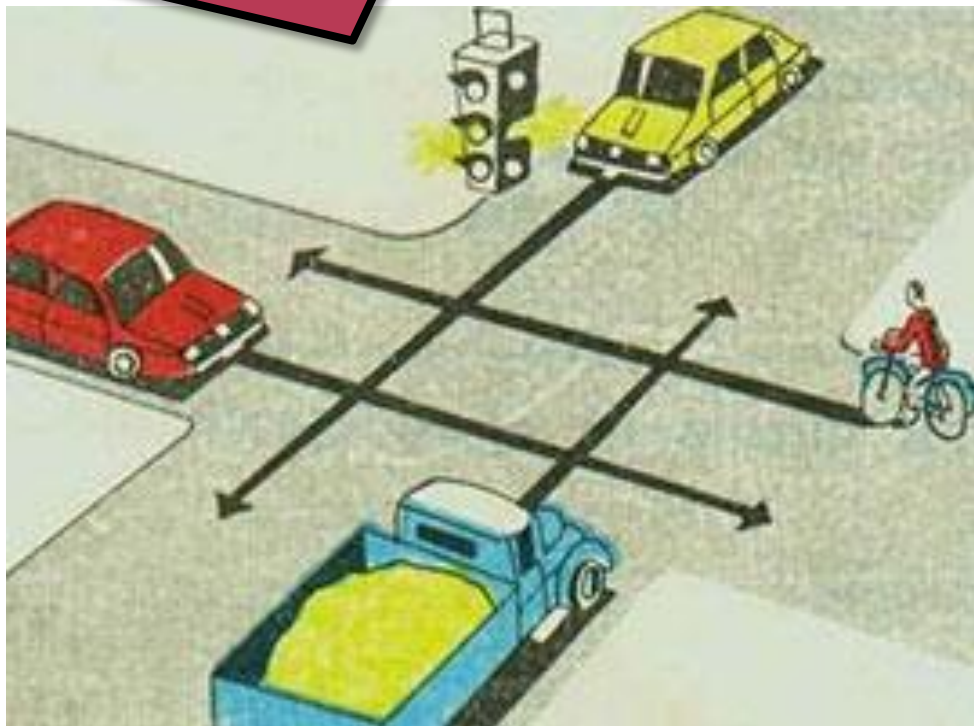
Holtpont

- Beragadt **állapot**:
a rendszer csak külső segítséggel képes kilépni.
 - Pl. egymásra várakozó folyamatok miatt



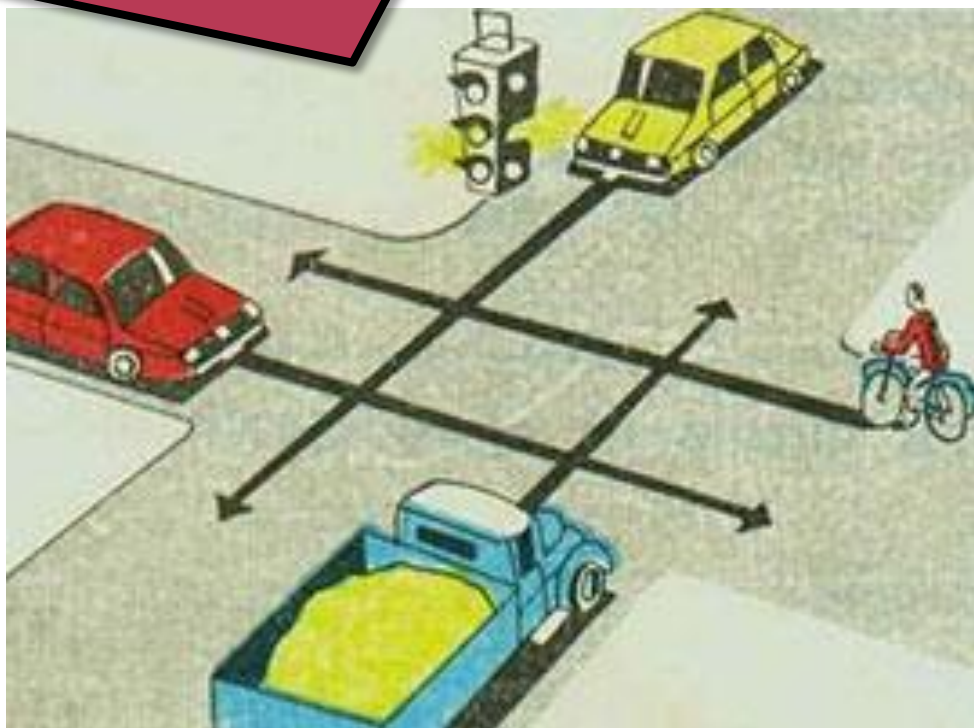
Holtpont (deadlock)

„Útkereszteződésben - ha közúti jelzésből vagy forgalmi szabályból más nem következik - a jobbról érkező járműnek van elsőbbsége.” (1988. évi I. törvény a közúti közlekedésről)



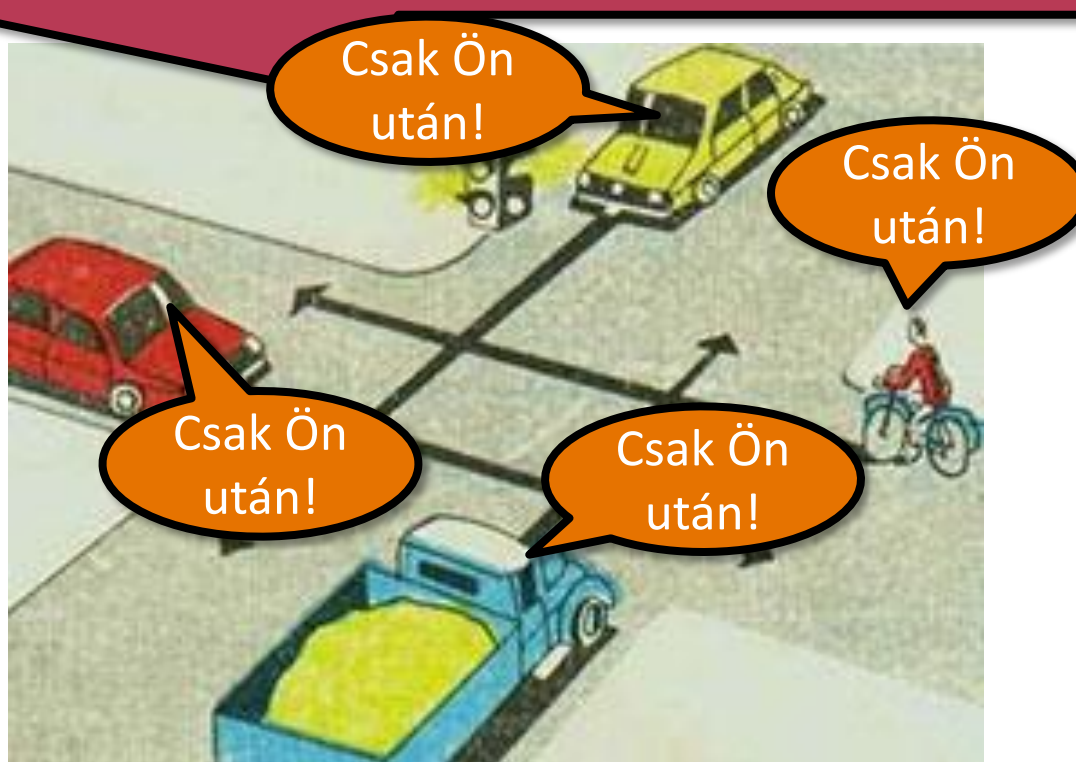
Holtpont feloldása

- „Ha 4 autó egyszerre ér a jobbkezes kereszteződésbe, akkor valamelyiknek le kell mondania az elsőbbségről és elengedni a másikat. Ha nem teszi, akkor a KRESZ szerint ott fognak állni örökké.” (gyakorikerdesek.hu)



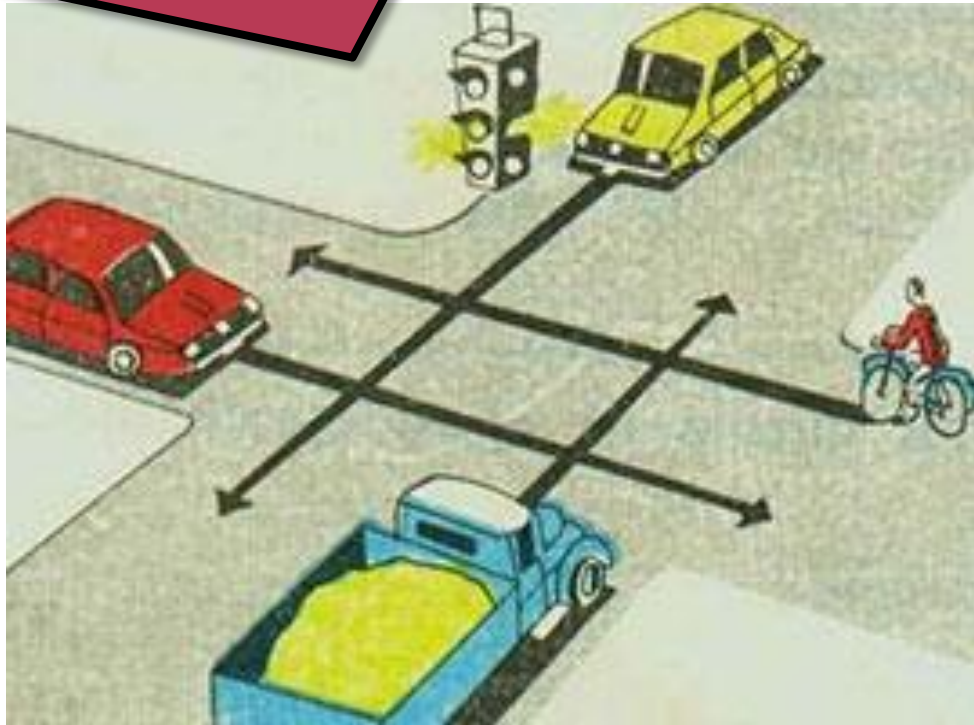
Holtpont feloldása

- „Ha 4 autó egyszerre ér a jobbkezes kereszteződésbe, akkor valamelyiknek le kell mondania az elsőbbségről és elengedni a másikat. Ha nem teszi, akkor a KRESZ szerint ott fognak állni örökké.” (gyakorikerdesek.hu)



Újabb holtpont

- „Ha 4 autó egyszerre ér a jobbkezes kereszteződésbe, akkor valamelyiknek le kell mondania az elsőbbségről és elengedni a másikat. Ha nem teszi, akkor a KRESZ szerint ott fognak állni örökké.” (gyakorikerdesek.hu)



Holtpont feloldása

„Útkereszteződésben - ha közúti jelzésből vagy forgalmi szabályból más nem következik - a jobbról érkező járműnek van elsőbbsége.” (1988. évi I. törvény a közúti közlekedésről)



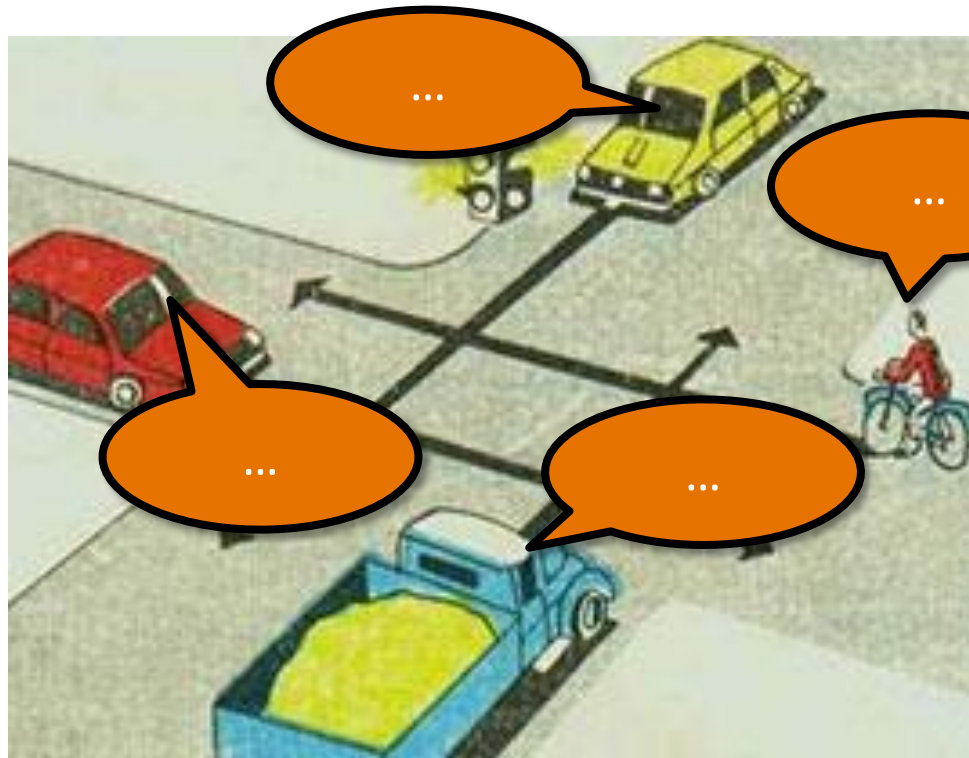
Holtpont feloldása

„Útkereszteződésben - ha közúti jelzésből vagy forgalmi szabályból más nem következik - a jobbról érkező járműnek van elsőbbsége.” (1988. évi I. törvény a közúti közlekedésről)



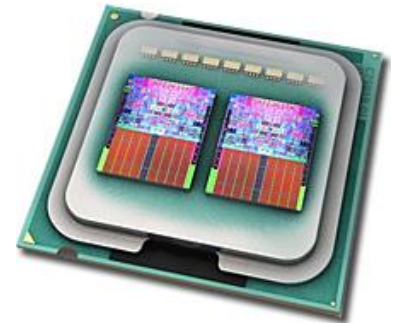
Végtelen ciklus (livelock)

- Beragadt **állapothalmaz**:
a rendszer csak külső segítséggel képes kilépni.
 - Pl. éppen a **holtpont feloldása** miatt



Holtpont (deadlock)

- Beragadt **állapot**:
a rendszer csak külső segítséggel képes kilépni.
 - Pl. egymásra várakozó folyamatok miatt
- Gyakori tervezési hiba párhuzamos rendszereknél
 - Sokszor nehéz elkerülni, feloldani
 - A jónak hitt megoldás is problémát okozhat
 - Nehéz tesztelni, látszólag véletlenszerű is lehet
 - „Többmagos CPU krízis”

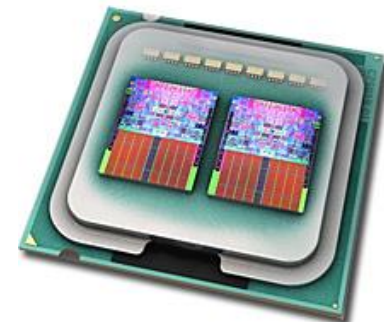


Holtpont (deadlock)

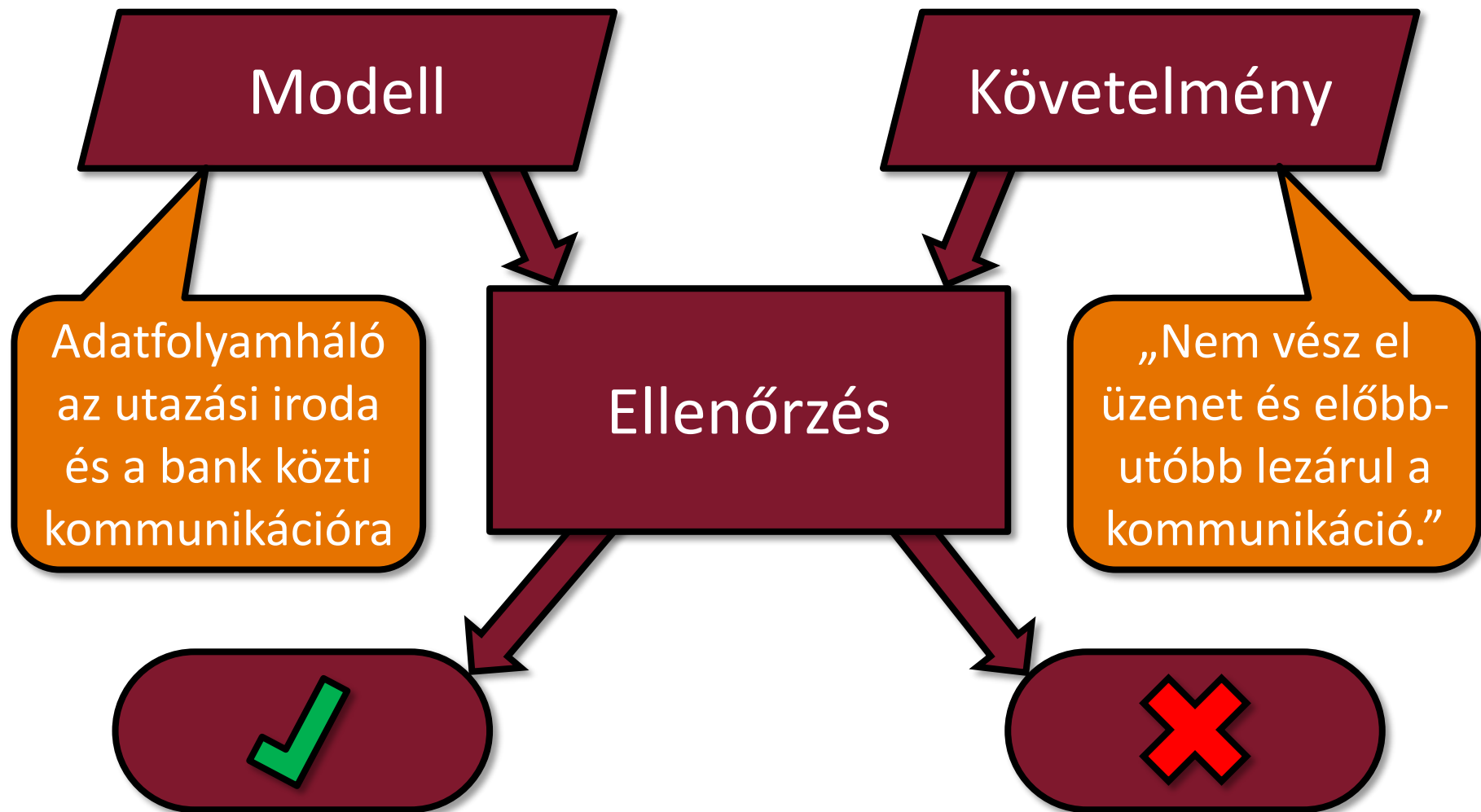
- Beragadt **állapot**:
a rendszer csak külső segítséggel oldható meg
 - Pl. egymásra várakozó folyamatok
- Gyakori tervezési hiba párhuzamos rendszereknél
 - Sokszor nehéz elkerülni
 - Két folyamat mindegyikének kétféle erőforrás kell a továbblépéshez, de mindegyikük egyet-egyét lefoglalt.

„Két folyamatnak üzenetet kell cserélnie, de mind a kettő a másik üzenetére vár.”

„Két folyamat mindegyikének kétféle erőforrás kell a továbblépéshez, de mindegyikük egyet-egyét lefoglalt.”



Modellek ellenőrzése



Vizsgálatok fajtái

■ Cél szerint:

○ **Verifikáció:**

jól csinálom a rendszert?

- Az implementáció megfelel a specifikációnak?

○ **Validáció:**

jó rendszert csinállok?

- A rendszer teljesíti a felhasználói követelményeket?

■ Módszer szerint:

○ Statikus ellenőrzés

○ Dinamikus ellenőrzés

- Szűrőpróbaszerű (tesztelés, szimuláció)
- Kimerítő/teljes (modellellenőrzés)

Alapfogalmak

Statikus ellenőrzés

Tesztelés

Formális verifikáció

Alapfogalmak

Statikus ellenőrzés

Tesztelés

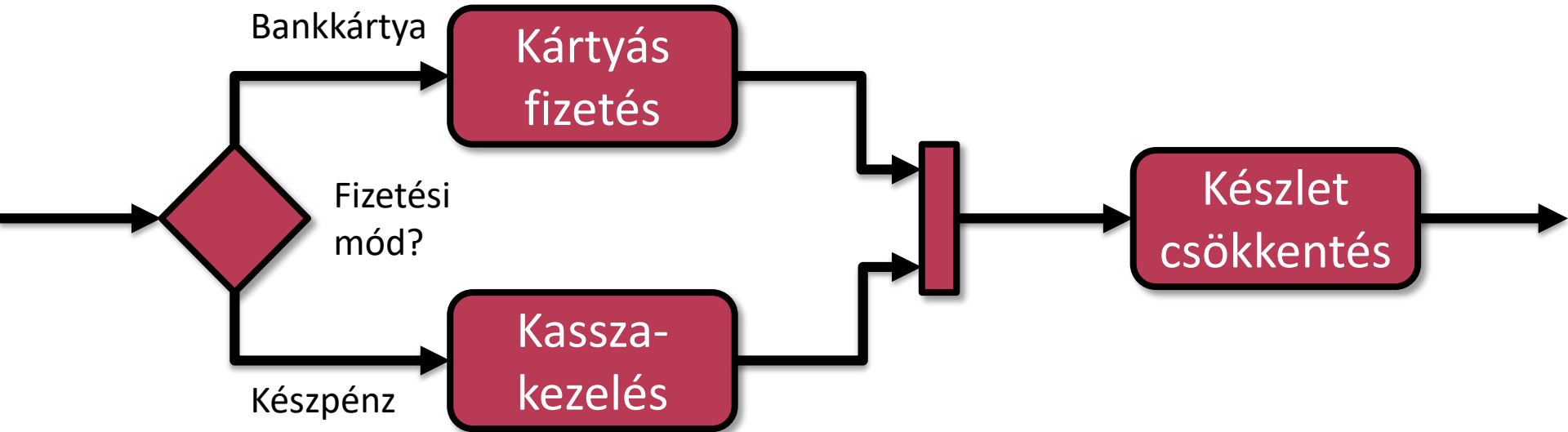
Formális verifikáció

STATIKUS ELLENŐRZÉS

Hibaminták

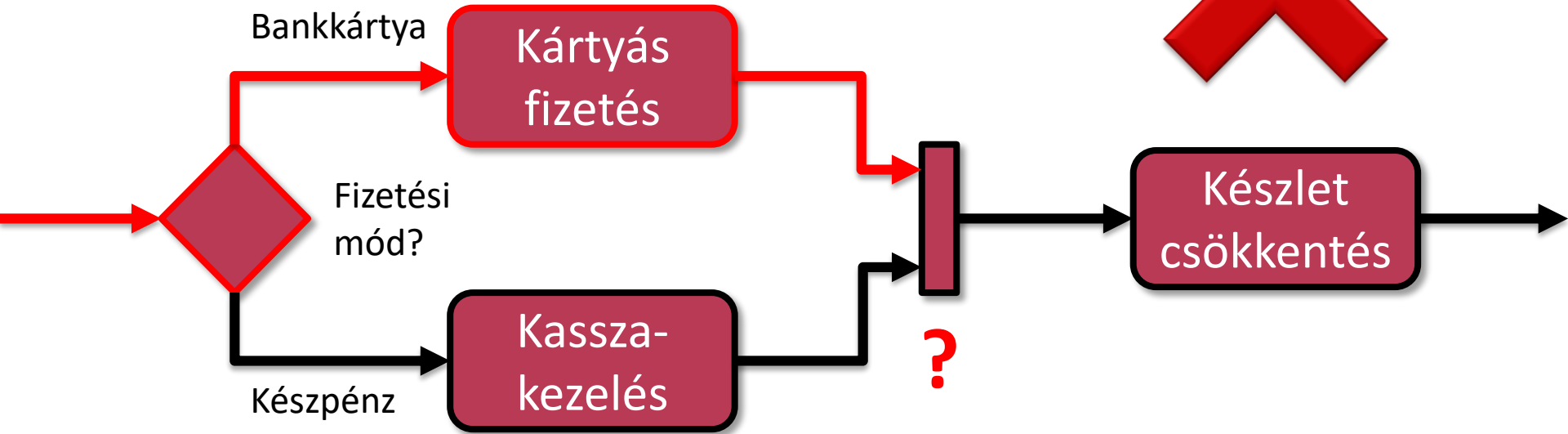
Decision és Join

- Helyes-e az alábbi modell?



Decision és Join

- Helyes-e az alábbi modell?

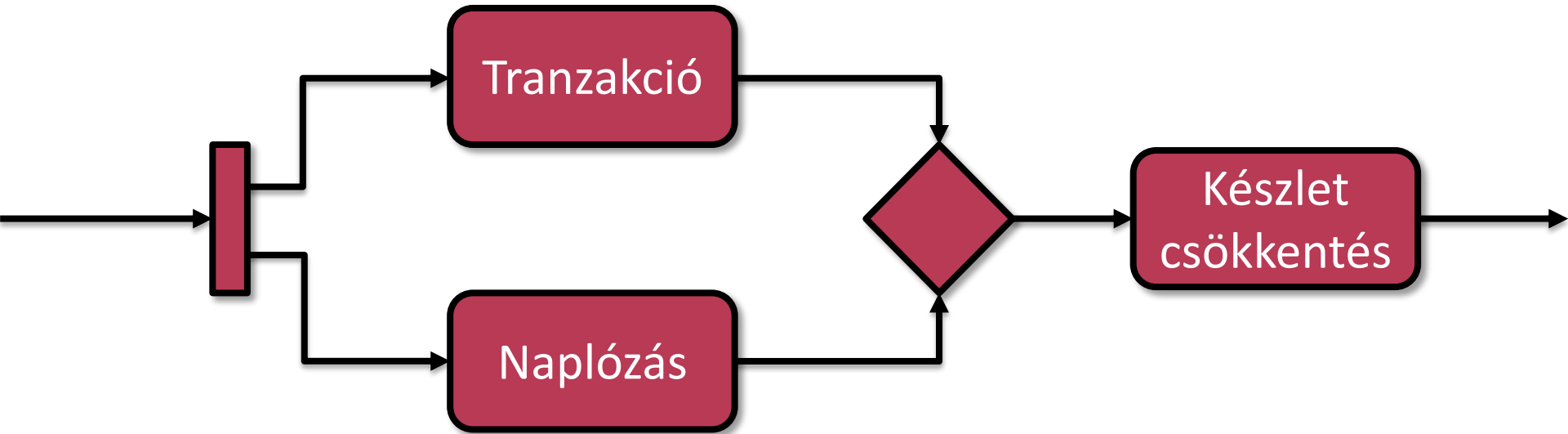


- Join: csak akkor léphet tovább, ha mindegyik bemeneten érkezett token

→ DEADLOCK

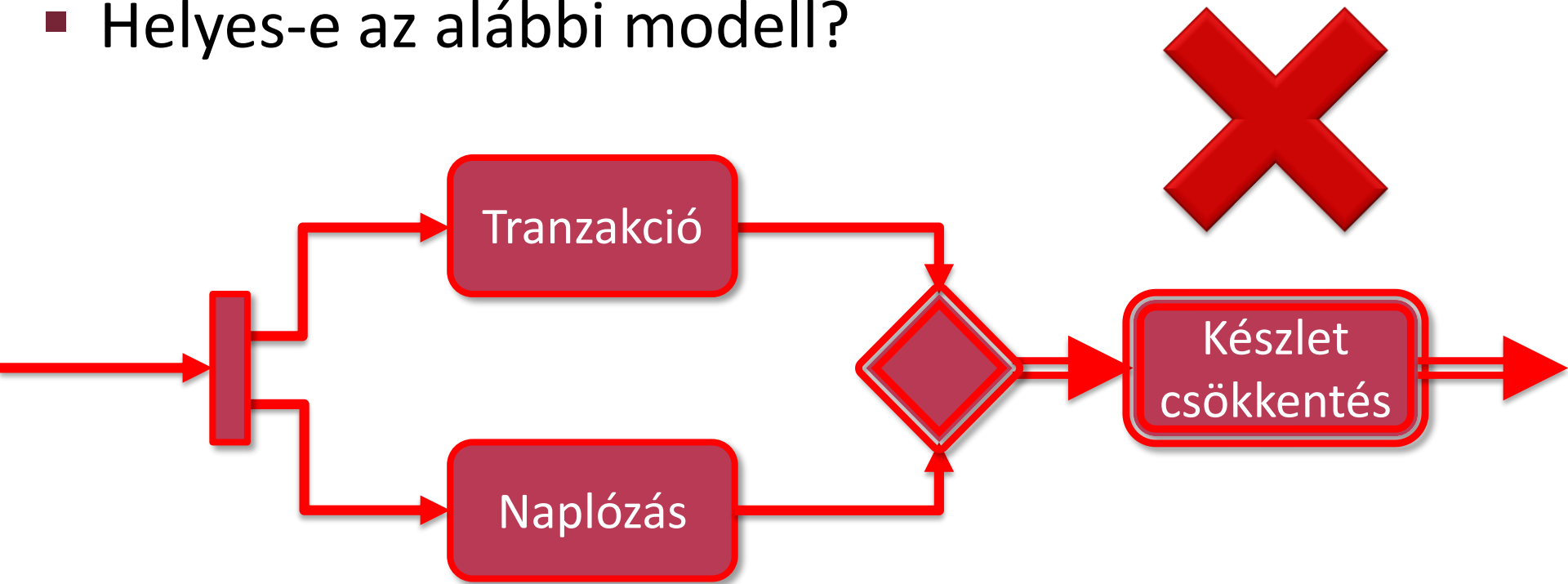
Fork és Merge

- Helyes-e az alábbi modell?



Fork és Merge

- Helyes-e az alábbi modell?

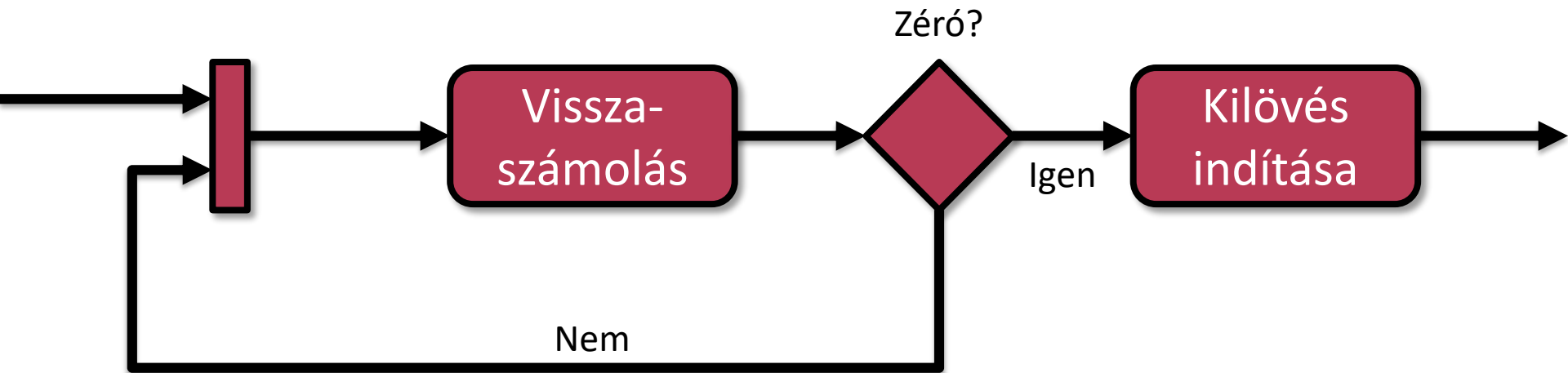


- Merge: bármelyik ágon érkező tokent átengedi
 - Nem szinkronizálja a szálakat

→ „Készlet csökkenés” kétszer fut le

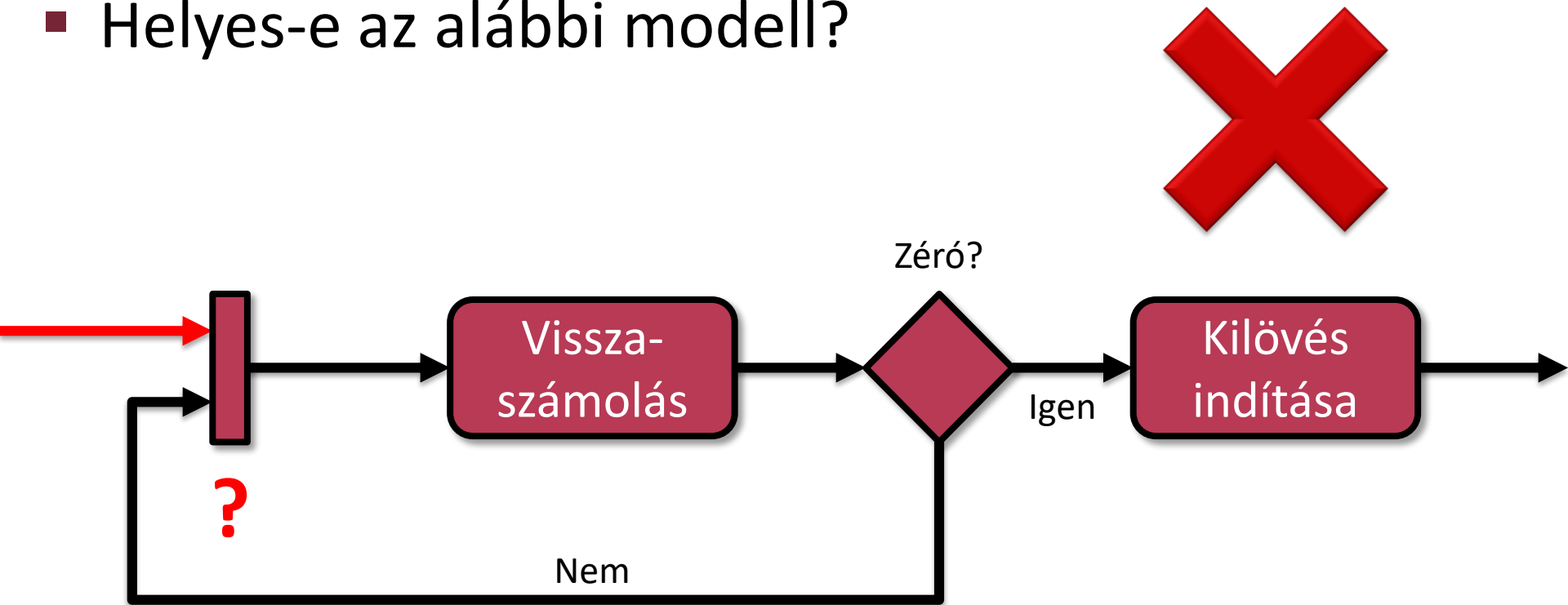
Ciklus 1.

- Helyes-e az alábbi modell?



Ciklus 1.

- Helyes-e az alábbi modell?

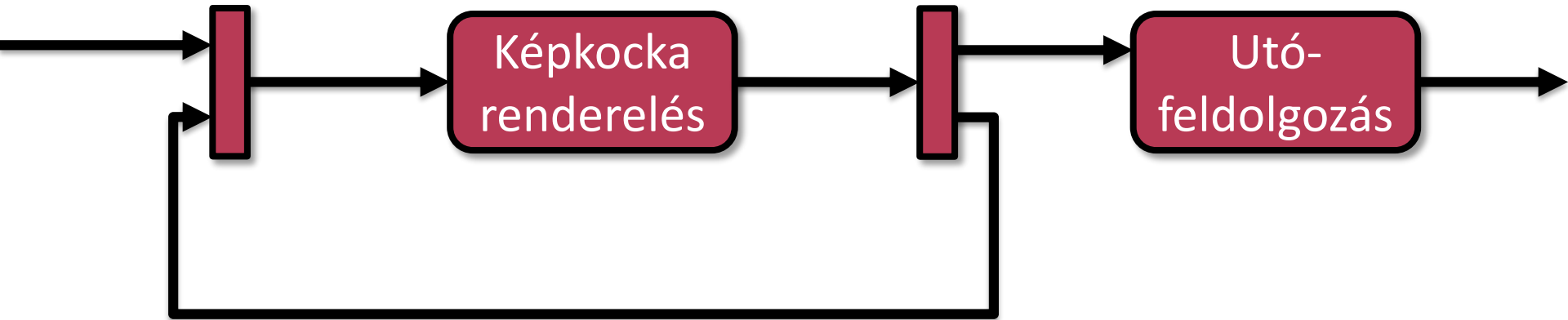


- Join: csak akkor léphet tovább, ha mindegyik bemeneten érkezett token

→ **DEADLOCK**

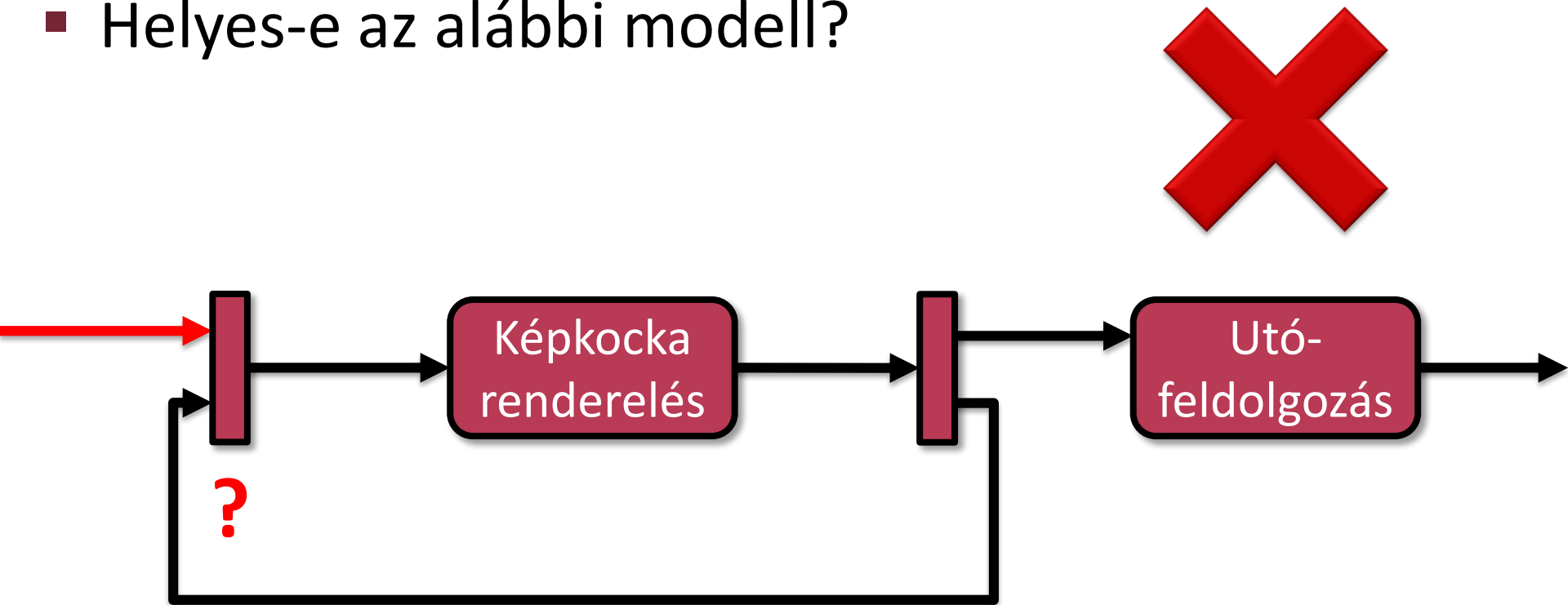
Ciklus 2.

- Helyes-e az alábbi modell?



Ciklus 2.

- Helyes-e az alábbi modell?

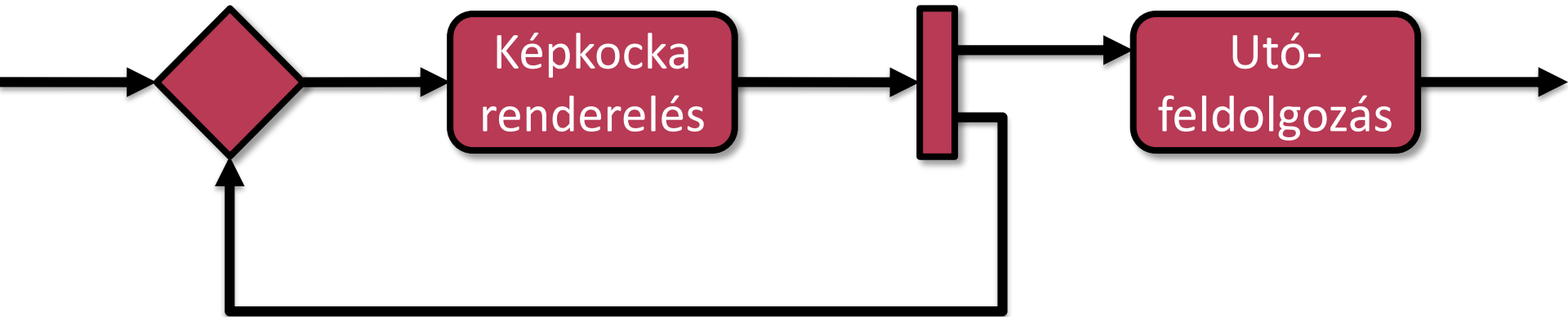


- Join: csak akkor léphet tovább, ha mindegyik bemeneten érkezett token

→ **DEADLOCK**

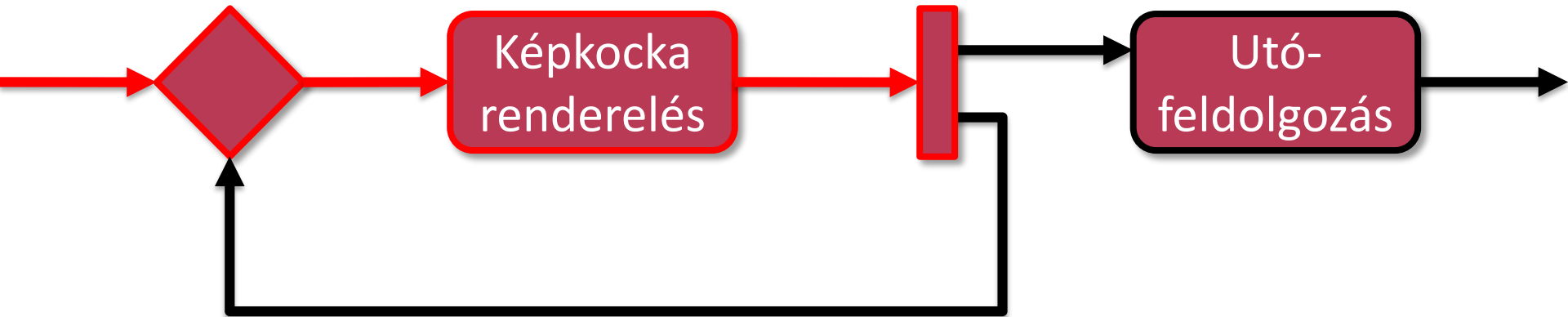
Ciklus 3.

- Helyes-e az alábbi modell?



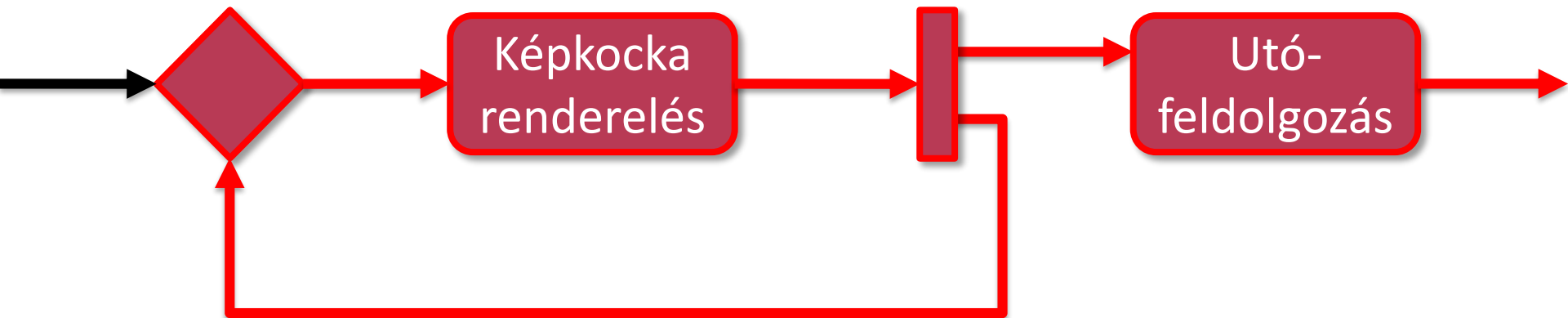
Ciklus 3.

- Helyes-e az alábbi modell?



Ciklus 3.

- Helyes-e az alábbi modell?

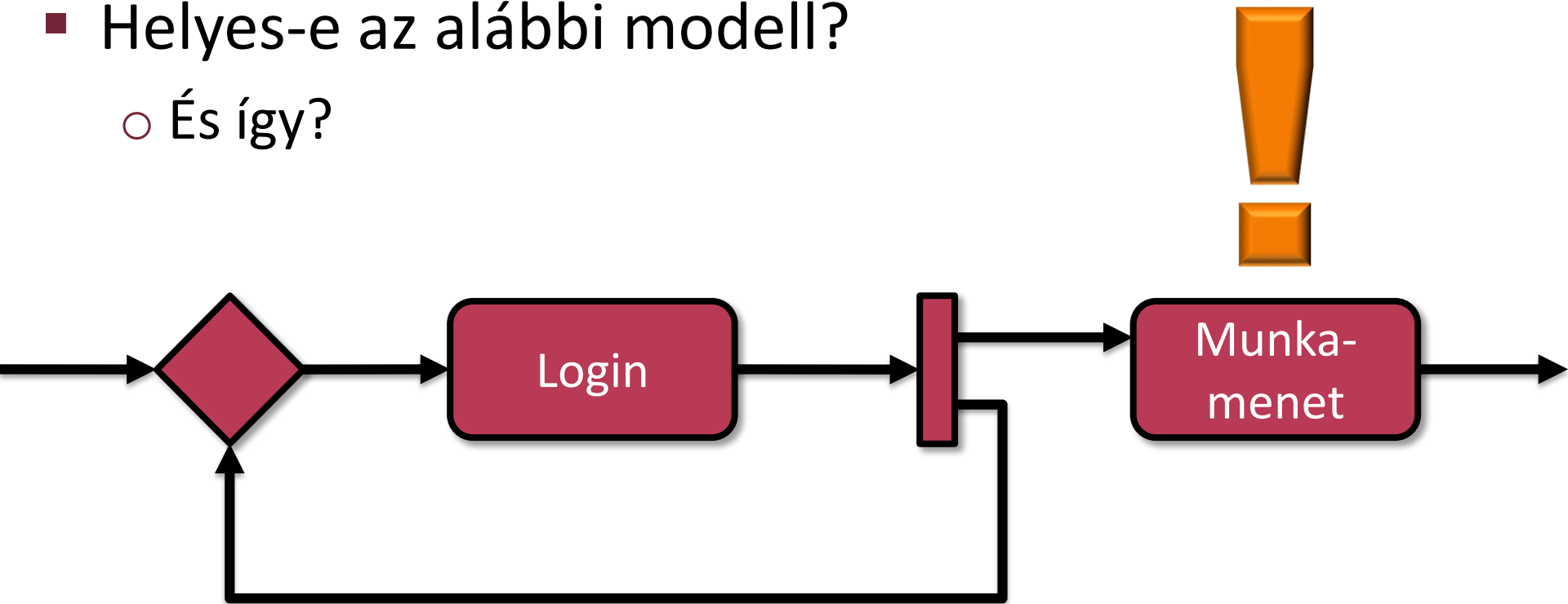


- Minden iterációban egy új képkocka
 - Mindegyikre utófeldolgozás (sokszor – hányszor?)

Határeset...

Ciklus 3.

- Helyes-e az alábbi modell?
 - És így?

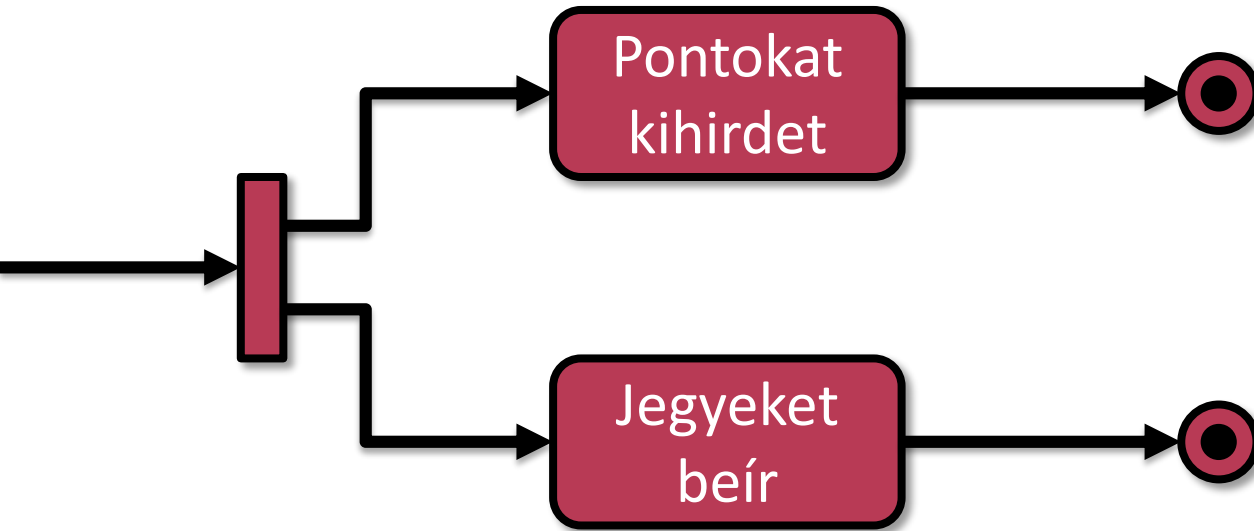


- Minden login után újabb login...
 - ...és egy munkamenet...?

→ Szálakat „termel” a hibás implementáció

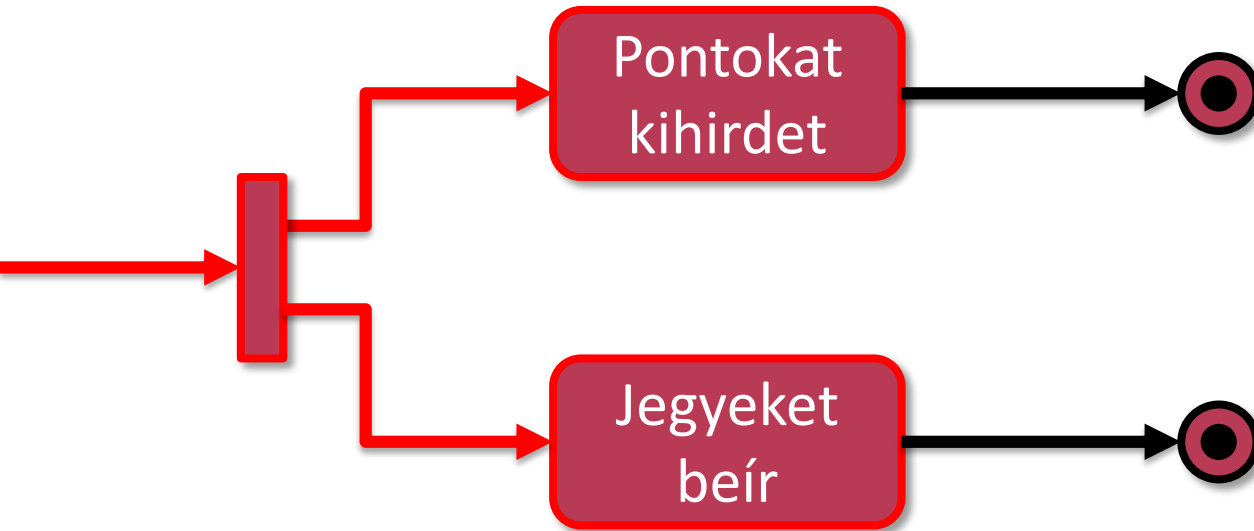
Termináló csomópont

- Helyes-e az alábbi modell?



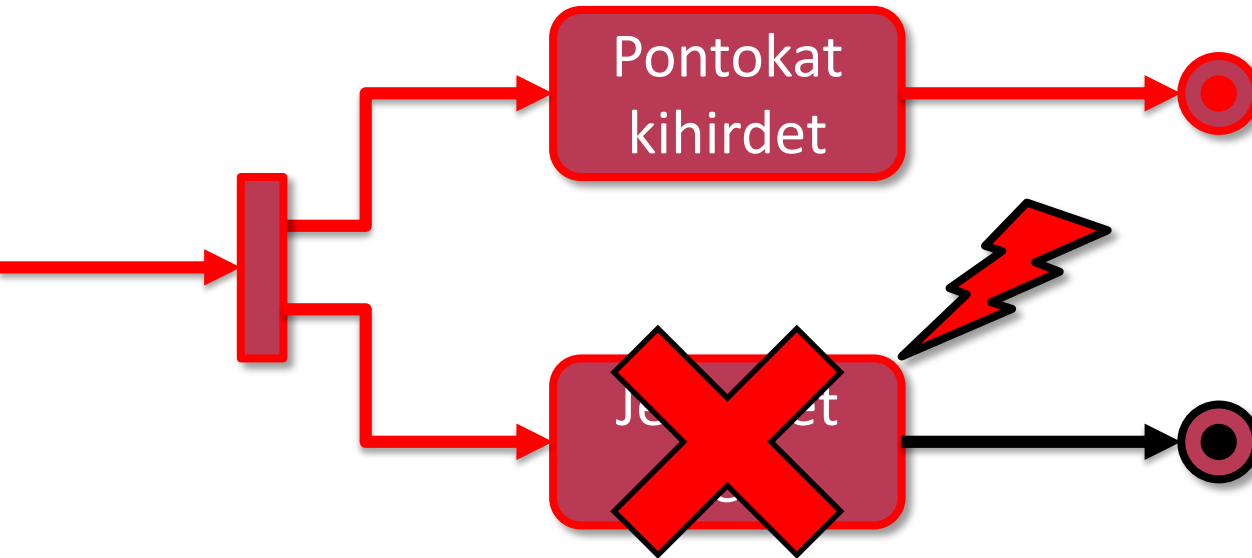
Termináló csomópont

- Helyes-e az alábbi modell?



Termináló csomópont

- Helyes-e az alábbi modell?



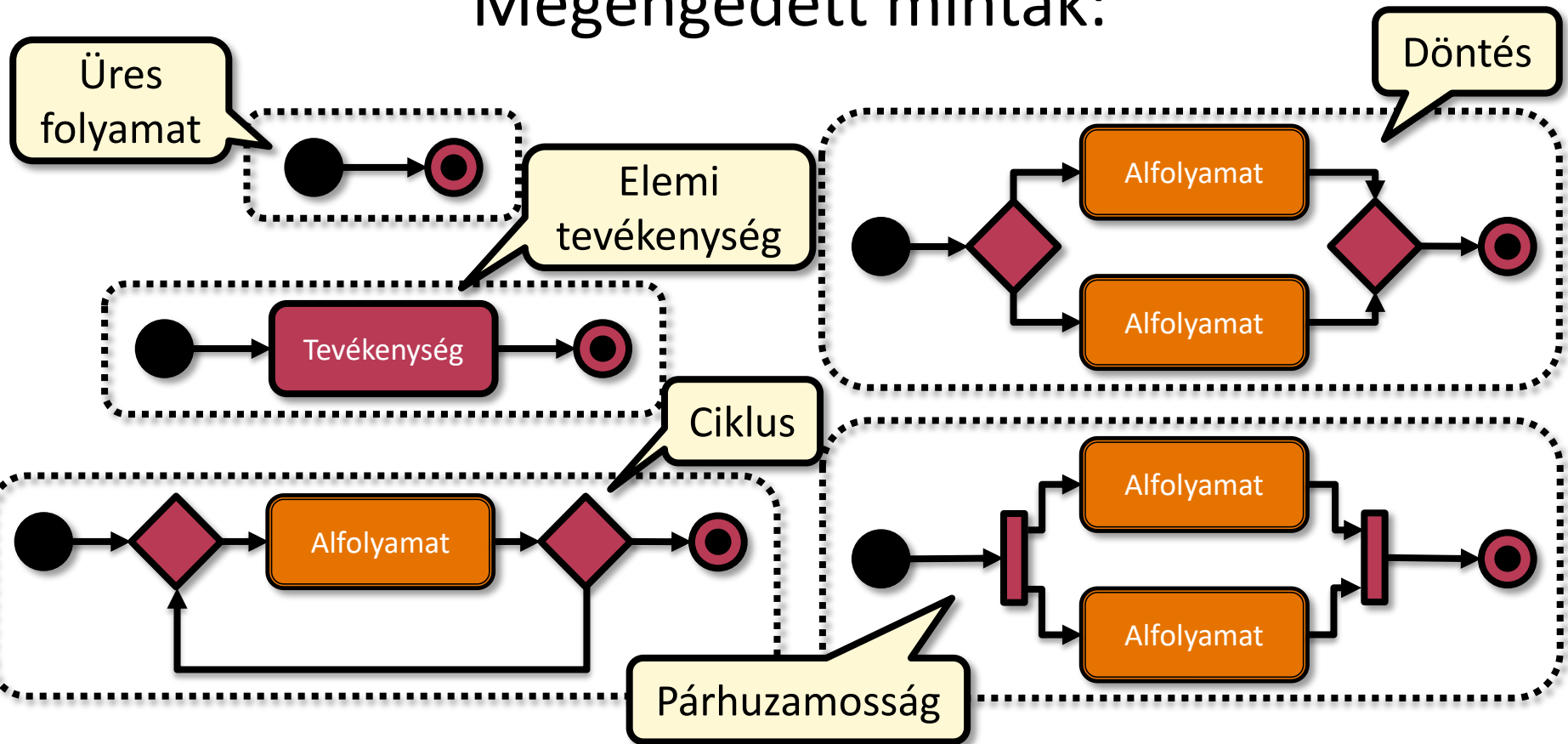
- Termináló csomópont: azonnal leállítja az **egész** folyamatot

→ **A másik tevékenység nem fog lefutni**

Jólstrukturált folyamatmodellek

- **Tanulság:** Jólstrukturált folyamatmodellekkel ezek a hibák elkerülhetők

Megengedett minták:



Adatkezelés statikus ellenőrzése

- Egy eljárás két egész számot szoroz
 - Származtatott követelmény:
 - „Ha legalább az egyik páros, az eredmény is az”
 - Végigkövethető a kódon
 - „Fejben végrehajtás”
- **Szimbolikus végrehajtás**
 - Konkrét értékek helyett értékhalmozokkal hajtjuk végre a programot
 - Érdekes bemenetek meghatározhatóak
 - Pl. ahol belső elágazások vannak
 - Milyen bemenettel érhetőek el az ágak?

Statikus ellenőrzés: szintaxisellenőrzés

- Szintaktikai ellenőrzés: modellező eszközök összekötik a logikailag egymásra épülő modellelemeket

Deklarálás interfészen:

```
var clock: integer = 60
```

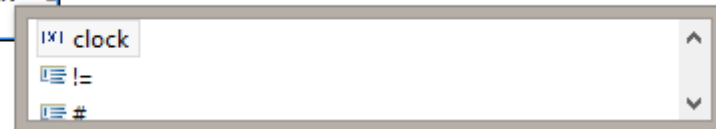
Használat modellben:

```
after 1 s [clock>0]/ clock -= 1
```

- Szintaxisvezérelt szerkesztő
 - Szerkesztés közben hiba → **Couldn't resolve reference**
 - Fejlett szerkesztőeszköz (például lehetőségek felkínálása)

- Kód és diagram együtt

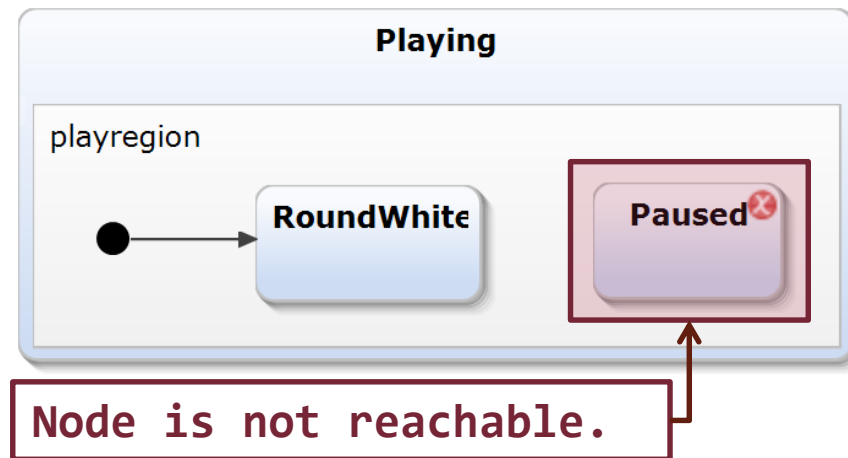
```
after 1 s [clock>0]/ clock-=1
```



- Programozás: szerkesztés közben **hibás**
- Modellezés: szerkesztés közben **helyes**

Statikus ellenőrzés: strukturális helyesség

- Strukturális ellenőrzés: modell gráf vizsgálata
- Hibaminták keresése szerkesztés közben
- Például elérhetetlen állapot:



- További ellenőrzések: hiányzó kezdőállapot, holtpont, változó értékadások, stb.

Alapfogalmak

Statikus ellenőrzés

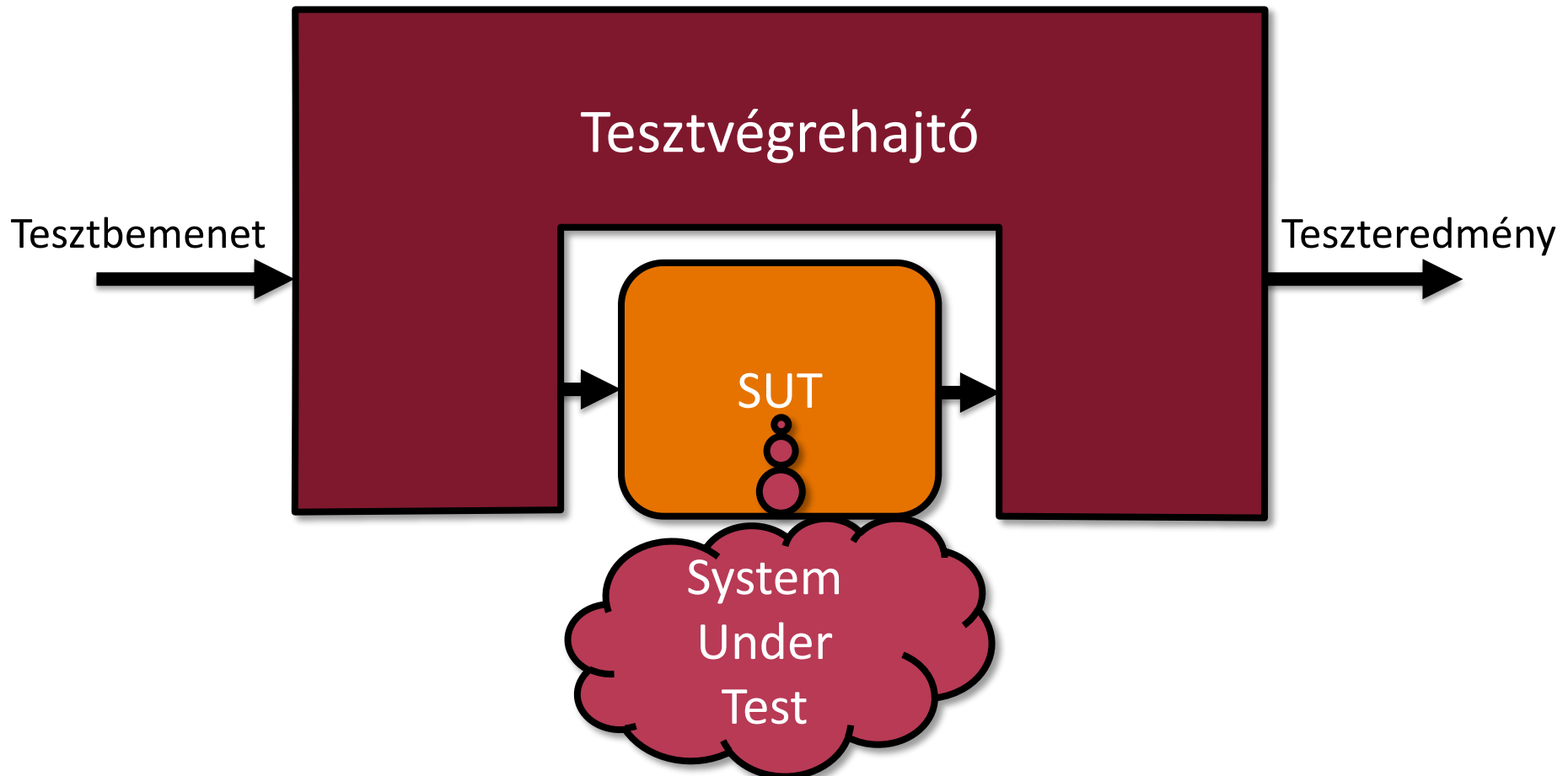
Tesztelés

Formális verifikáció

TESZTELÉS

Csupán „próbálgatás”?

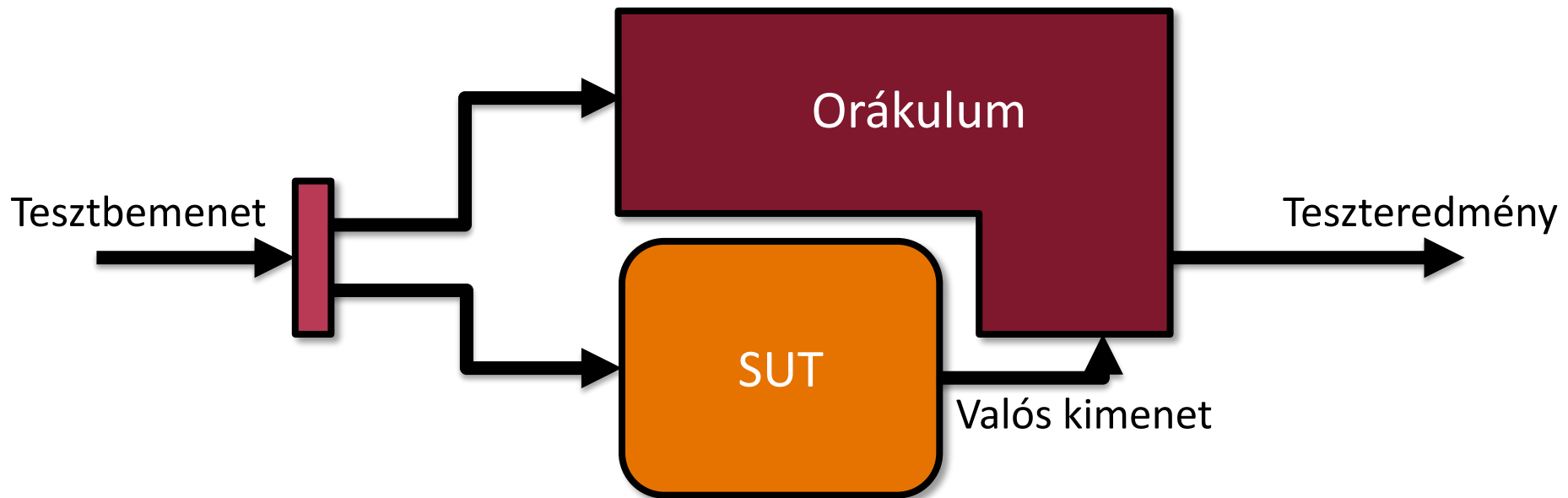
Modellek tesztelése



Modellek tesztelése

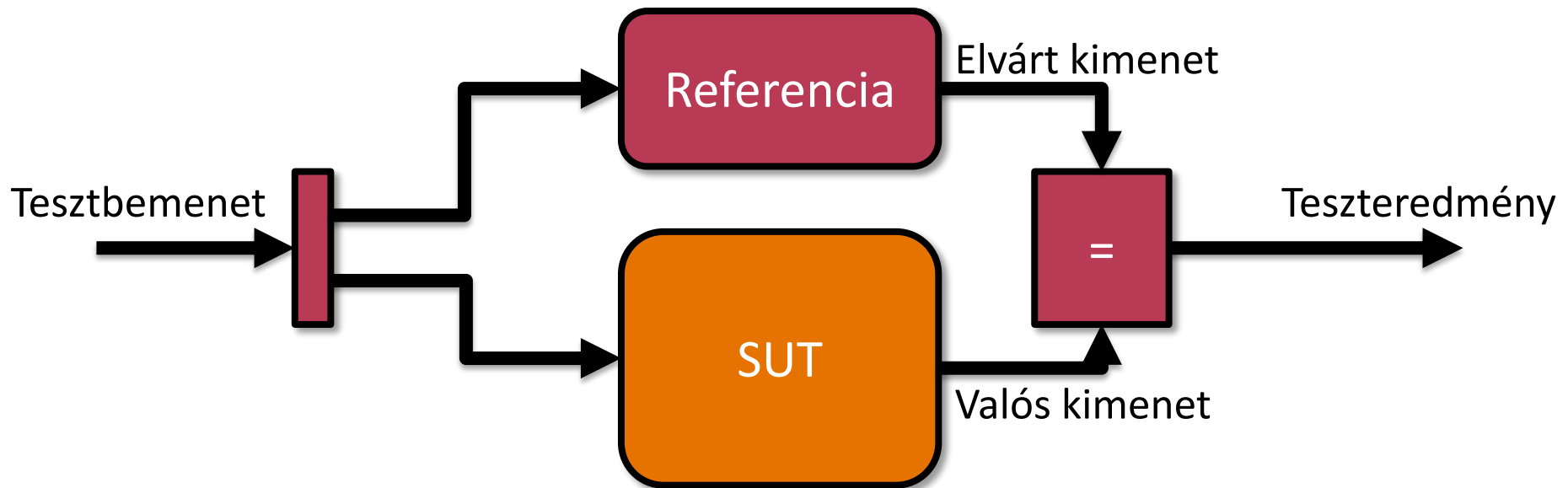
- **Orákulum:**

elvárt eredmények előállítása és összehasonlítása

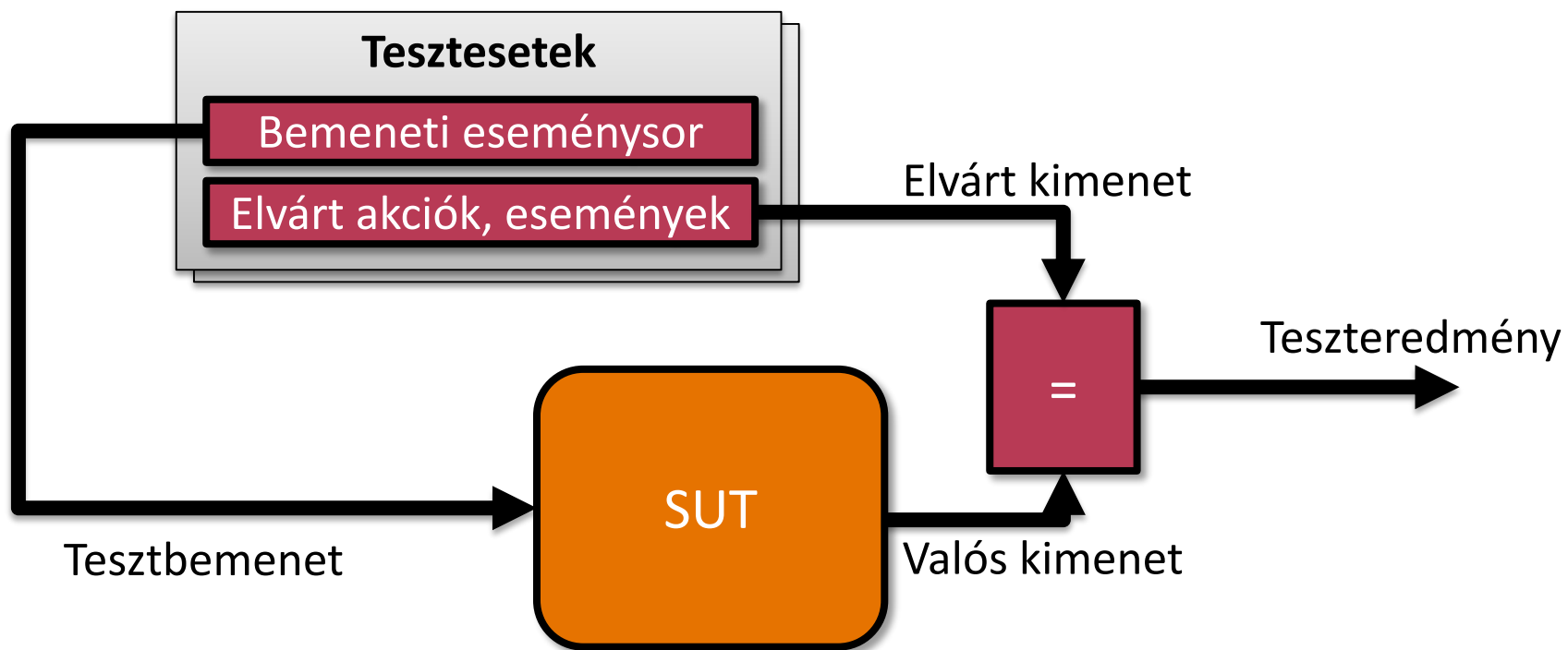


Modellek tesztelése

- **Referencia:**
tesztbemenet alapján elvárt kimenet



Modellek tesztelése példa: Yakindu állapotgép



Modellek tesztelése példa: Yakindu állapotgép

Példa teszteset: Beállítások menüben 1 és 3 perc között lehet 5 másodperc léptekkel állítani a kezdőidőt.

Bemenetek

Vizsgált automata

Elvárt kimenet leolvasása

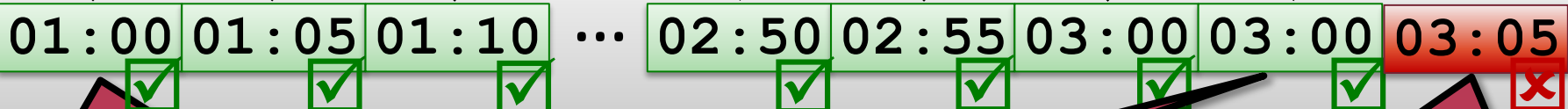
Modellek tesztelése példa: Yakindu állapotgép

Példa teszteset: Beállítások menüben 1 és 3 perc között lehet 5 másodperc léptekkel állítani a kezdőidőt

A + Gomb megnyomására 5 másodperccel nő



Vizsgált automata



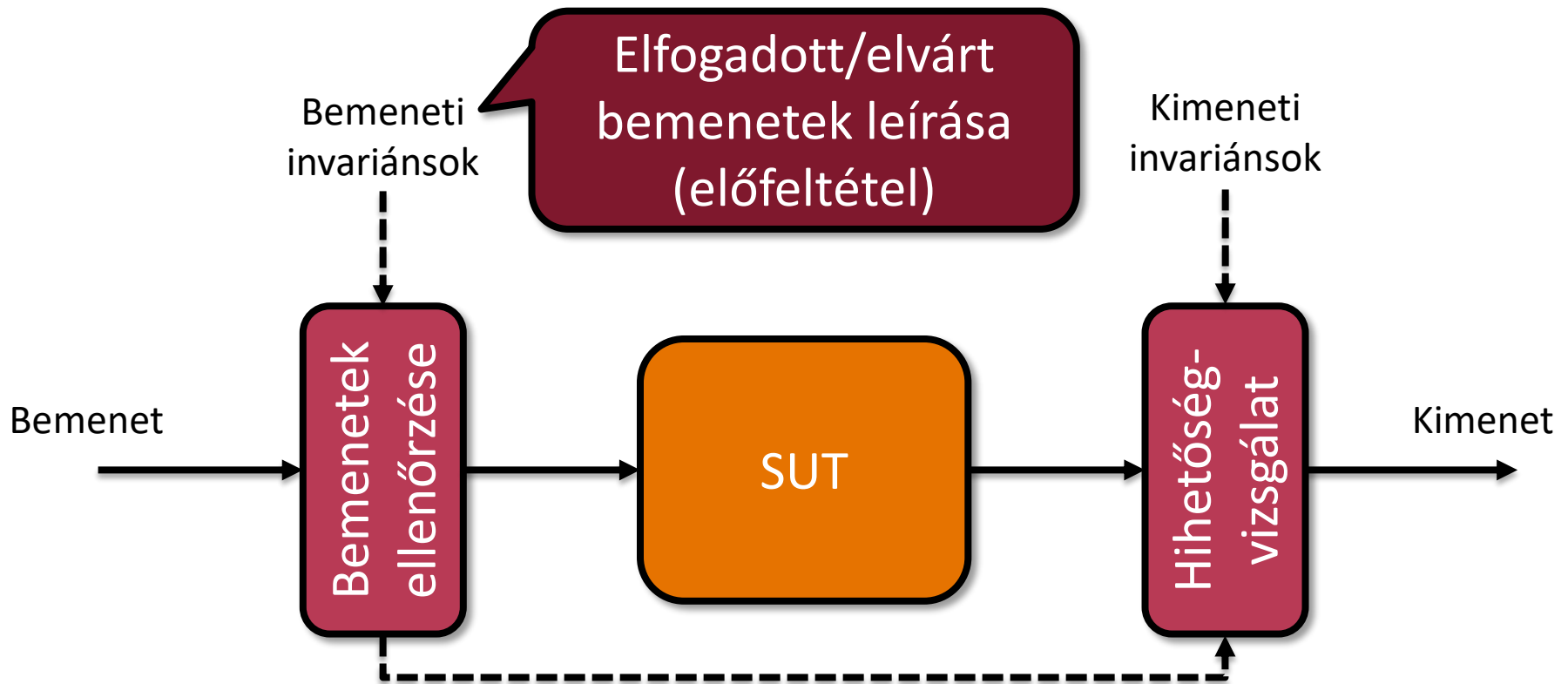
Elvárt kimenet leolvása

Kezdetben a kijelző 1 percet mutat

Tovább nem nő

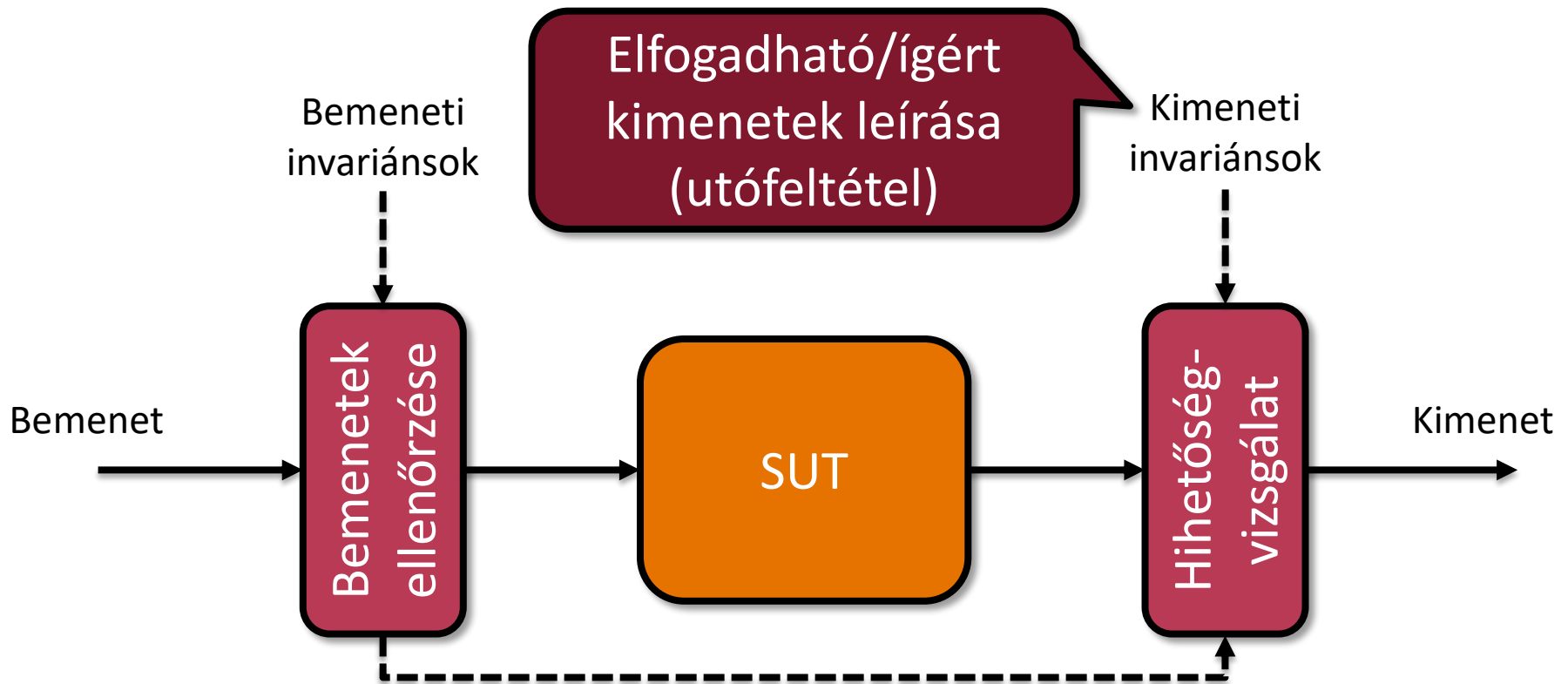
Hibás kimenet →
Jelzünk a fejlesztőnek

Öntesztelés (monitor)



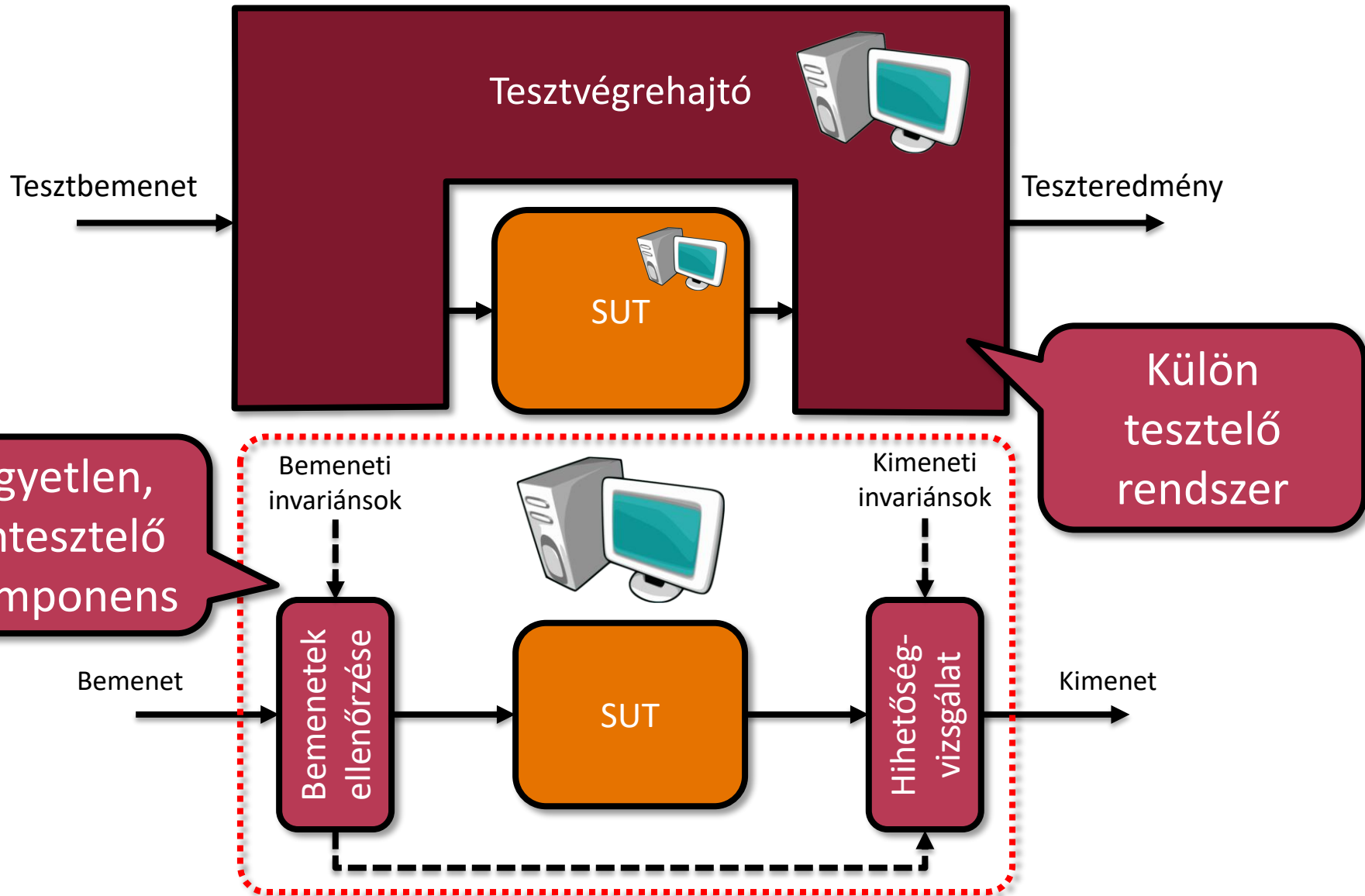
- **Invariáns tulajdonság:**
folyamatosan igaznak kell lennie

Öntesztelés (monitor)



- **Invariáns tulajdonság:**
folyamatosan igaznak kell lennie

Öntesztelés vs. Külső tesztelés



Öntesztelő program

Előfeltétel: a diszkrimináns nemnegatív

```
void Roots(float a, b, c,  
           float &x1, &x2)  
{  
    float d = sqrt(b*b-4*a*c);  
  
    x1 = (-b + d)/(2*a);  
    x2 = (-b - d)/(2*a);  
}
```

Utófeltétel: mindkét megoldás zérushely

```
void RootsMonitor(float a, b, c,  
                 float &x1, &x2)  
{  
    // előfeltétel  
    float D = b2-4*a*c;  
    if (D < 0)  
        throw "Invalid input!";  
  
    // végrehajtás  
    Roots(a, b, c, x1, x2);  
  
    // utófeltétel  
    assert(a*x12+b*x1+c == 0 &&  
           a*x22+b*x2+c == 0);  
}
```

Öntesztelő program

Exception (kivétel):

A normálistól eltérő, váratlan helyzet, aminek

kezelését máshol valósítjuk meg.

Oka: hibás használat.

```
float d = sqrt(b*b-4*a*c);
```

Assert (feltételezés):

Hibás állapot, aminek kezelésére a kód

nincs felkészítve.

Oka: hibás implementáció vagy futásidejű hiba.

```
void RootsMonitor(float a, b, c,  
float &x1, &x2)
```

```
// előfeltétel
```

```
float D = b2-4*a*c;
```

```
if (D < 0)
```

```
throw "Invalid input!";
```

```
// végrehajtás
```

```
Roots(a, b, c, x1, x2);
```

```
// utófeltétel
```

```
assert(a*x12+b*x1+c == 0 &&
```

```
a*x22+b*x2+c == 0);
```

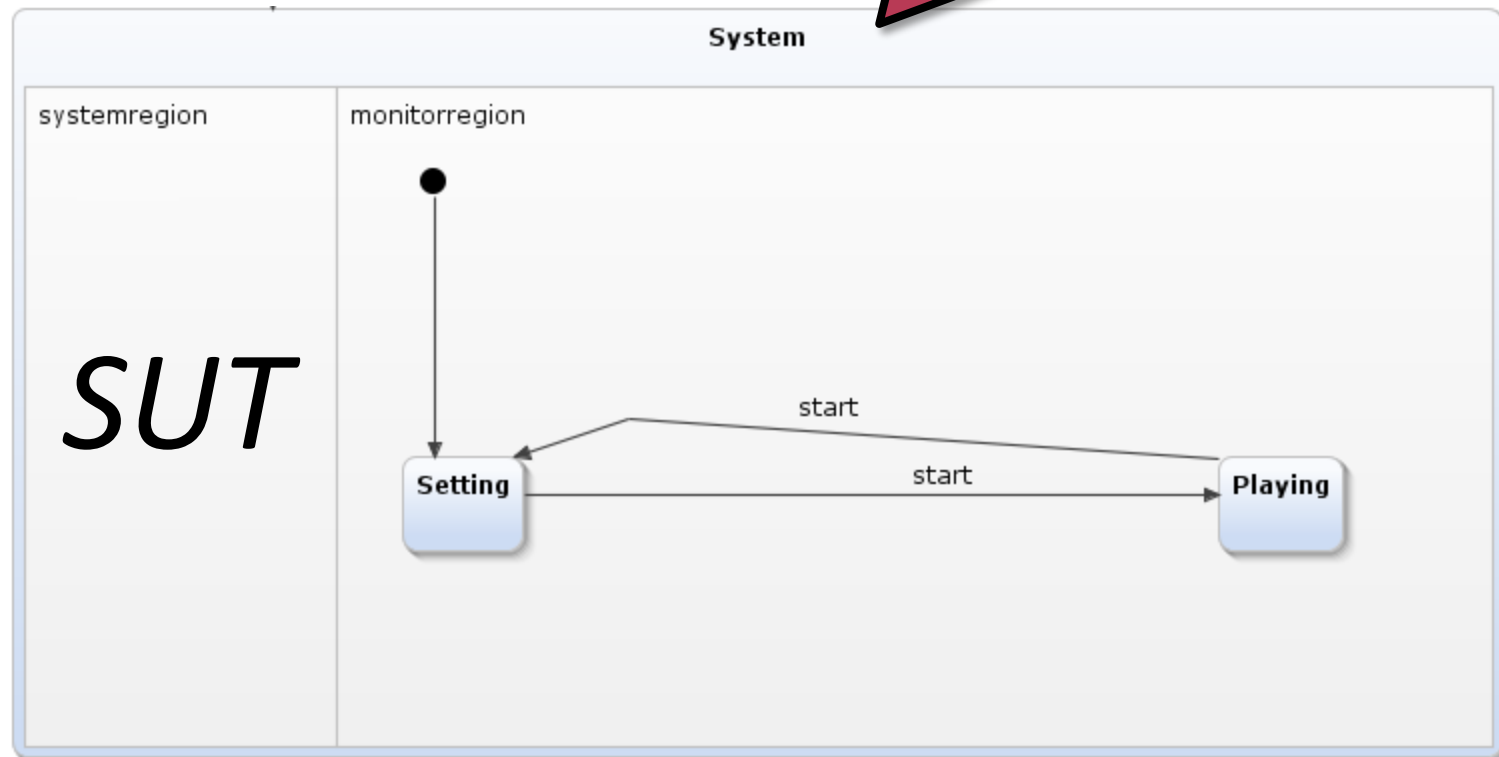
Öntesztelés Yakinduban

- Párhuzamosan fut a *SUT* és a *monitor* régió

- Jó eset:

- Érvényes bemenet (valid)
- Helyes üzemmód

A házi feladatban a beállítás és játék között válthatunk.



Öntesztelés Yakinduban

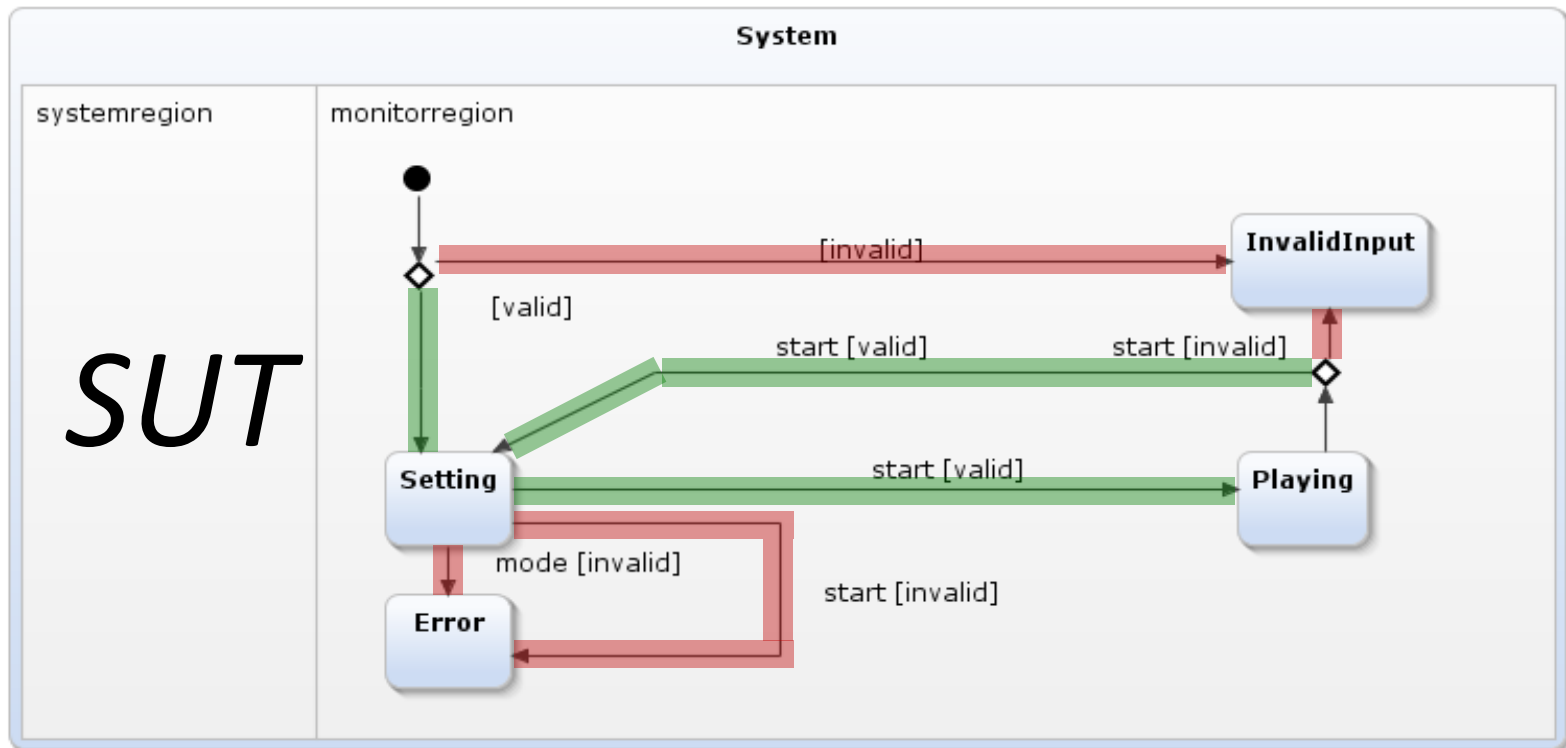
■ Párhuzamosan fut a *SUT* és a *monitor* régió

○ Jó eset:

- Érvényes bemenet (valid)
- Helyes üzemmód

○ Rossz eset:

- Hibás bemenet → InvalidInput
- Hibás kimenet → Error



Öntesztelés Yakinduban

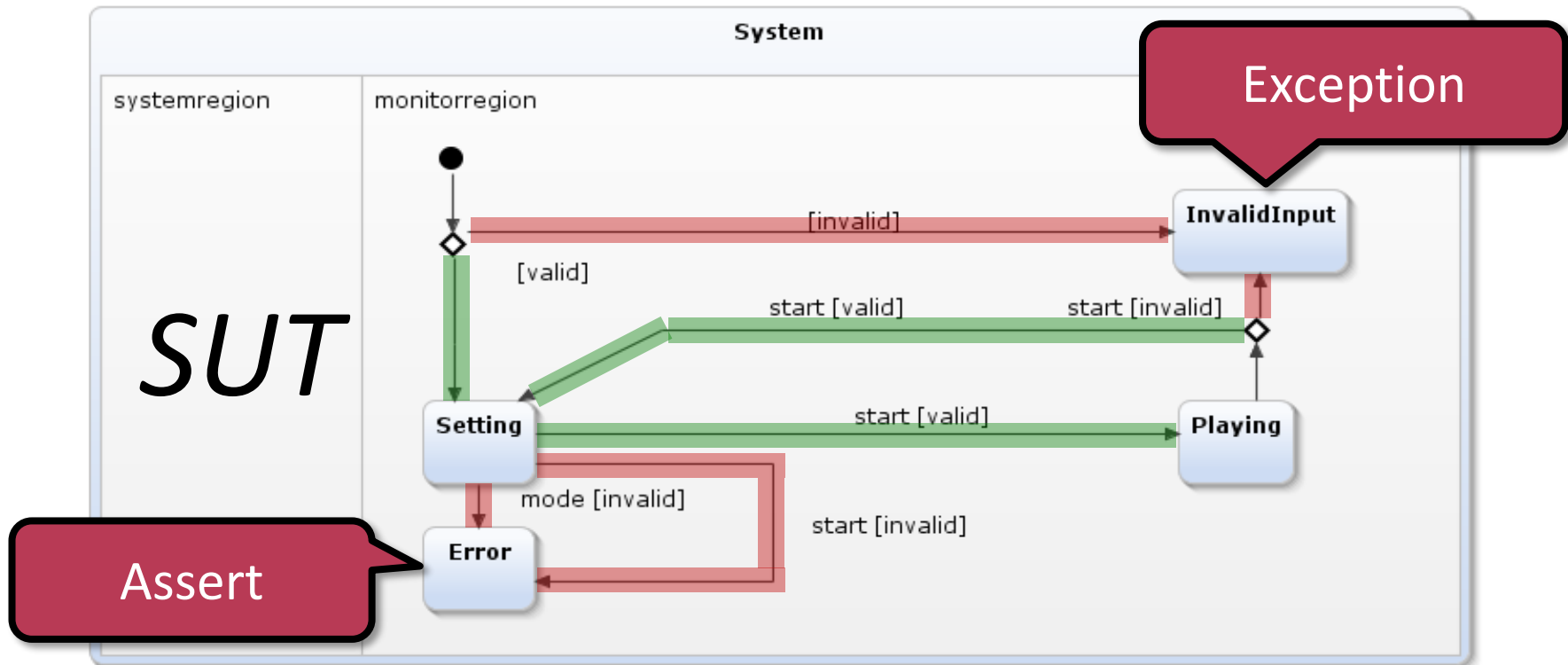
■ Párhuzamosan fut a *SUT* és a *monitor* régió

○ Jó eset:

- Érvényes bemenet (valid)
- Helyes üzemmód

○ Rossz eset:

- Hibás bemenet → InvalidInput
- Hibás kimenet → Error



Modellek tesztelése

- A modell futtatása: szimuláció
 - Adott bemenetekre vizsgált viselkedés
- Teszteset:
 1. Tesztbemenet
 - pl. értéktartomány közepe és két széle
 2. Elvárt kimenet

Milyen bemenetekkel teszteljük?

Fedettség

- Adott tesztkészlet esetén a **fedettség** a tesztek futtatása alatt érintett részek aránya a modellben.

- Állapot lefedettség (állapotgépben):

érintett állapotok

összes állapot

- Átmenet lefedettség (állapotgépben):

érintett átmenetek

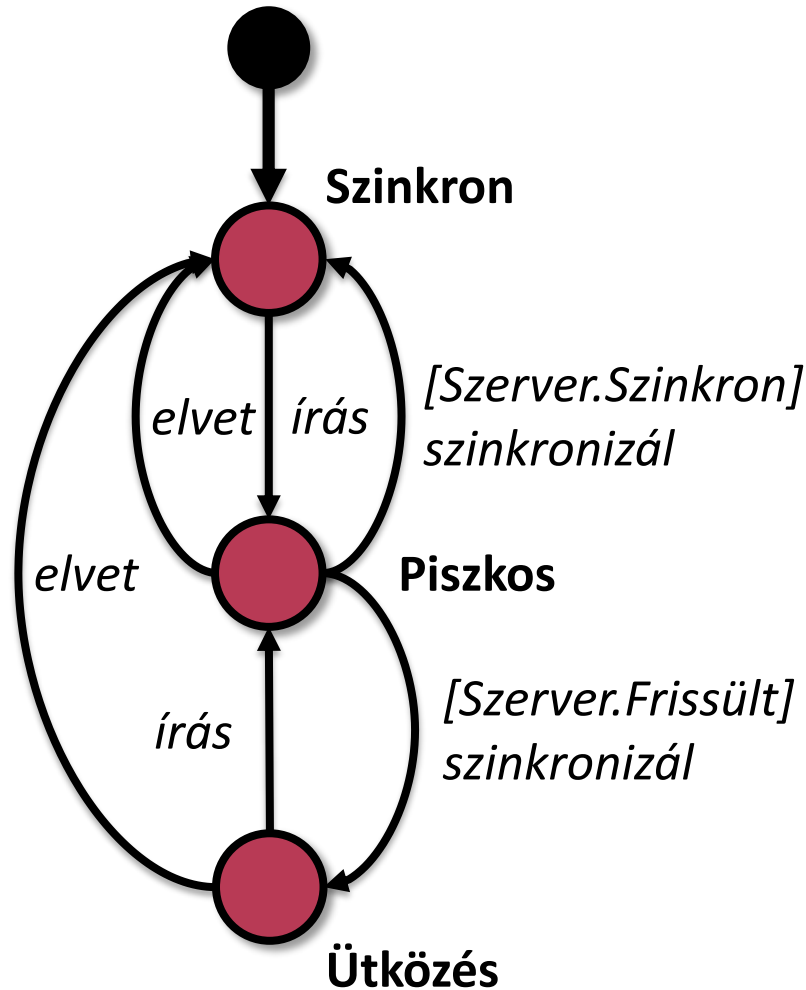
összes átmenet

- Utasítás lefedettség (vezérlési folyamban):

érintett tevékenységek

összes tevékenység

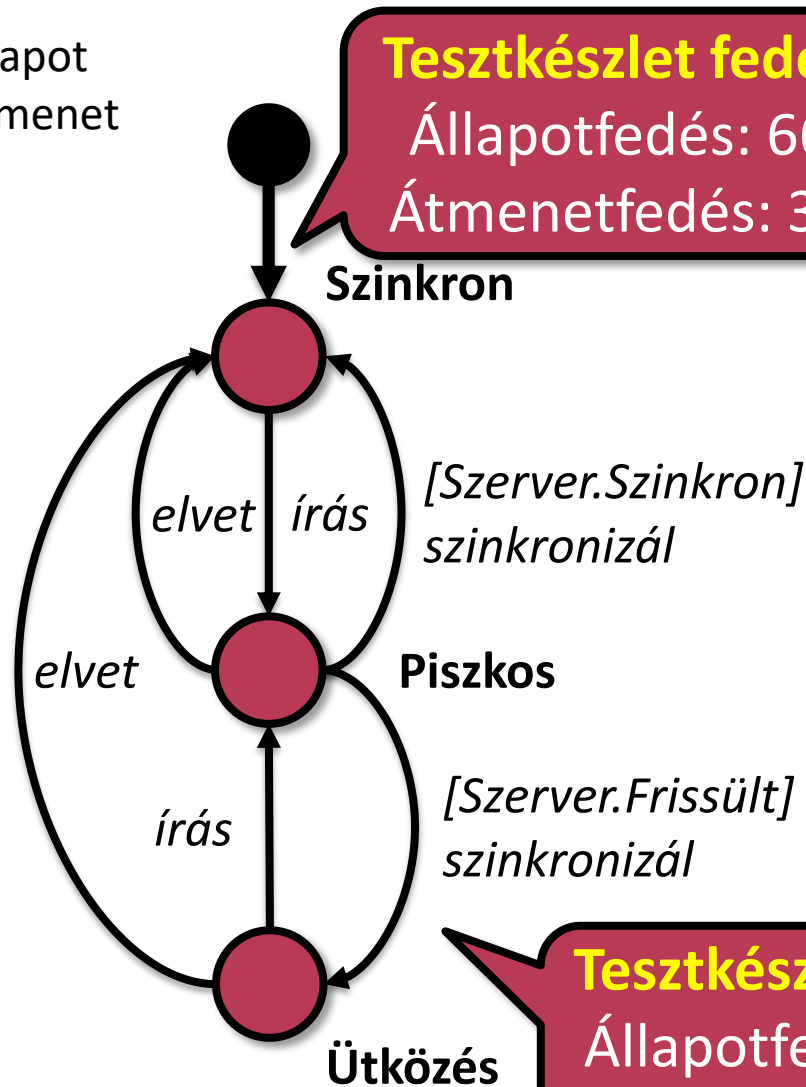
Példa: Felhőalapú adattárolás



„Felhő alapú adattárolást modellezünk, egyetlen állományra szorítkozva. A kliens írhatja az állományt, szinkronizálhat a szerverrel és elvetheti a helyi módosításokat. A szerveren tárolt változat verziójától függően a szinkronizálás ütközést is okozhat, ha más is módosította az állományt.”

Példa: Felhőalapú adattárolás

3 állapot
6 átmenet



Tesztkészlet fedése:

Állapotfedés: 66%

Átmenetfedés: 33%

1. Teszteset:

a) írá

b) elvet

2. Teszteset:

a) írá

b) Szerver = Frissült

c) szinkronizál

d) elvet

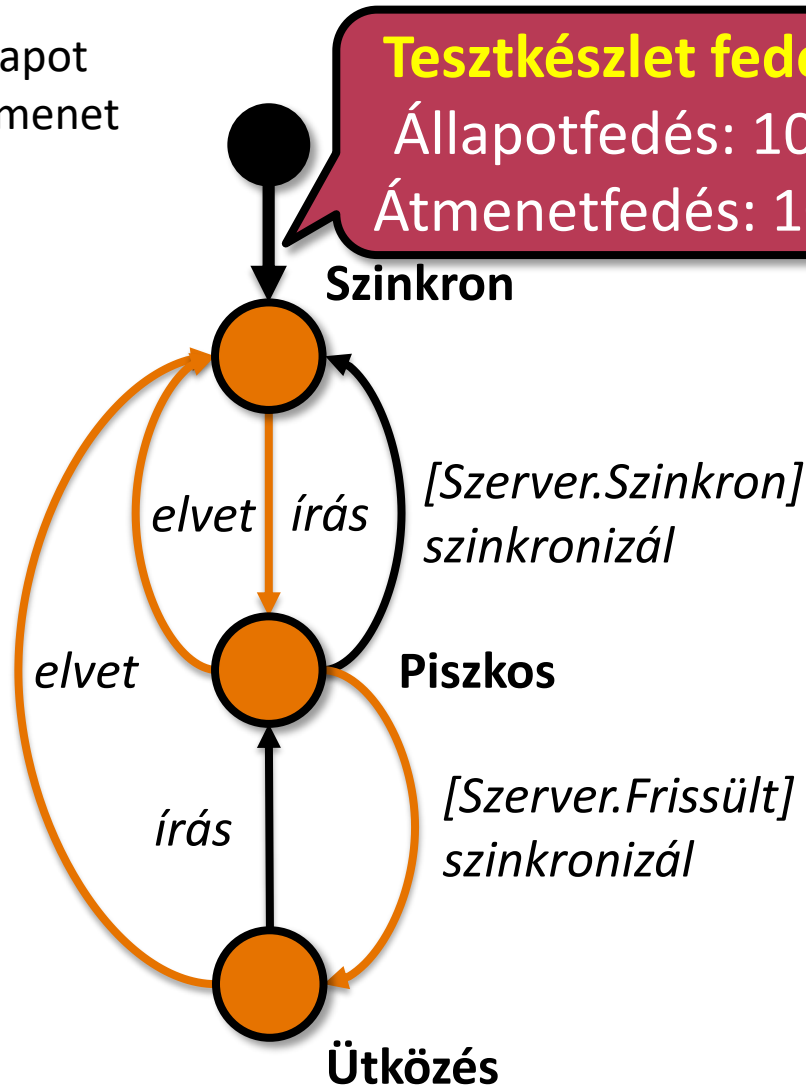
Tesztkészlet fedése:

Állapotfedés: 100%

Átmenetfedés: 66%

Példa: Felhőalapú adattárolás

3 állapot
6 átmenet



3. Teszteset:

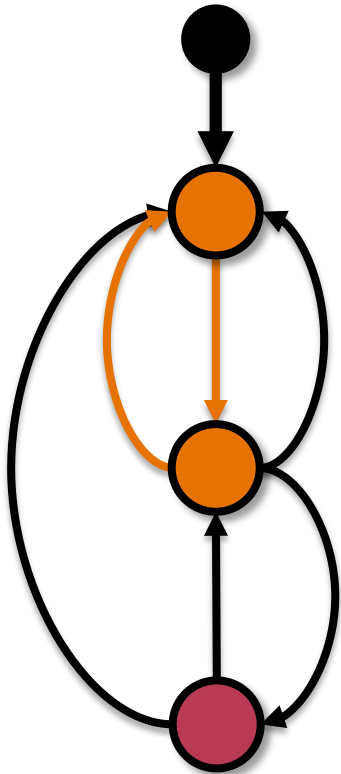
- írás
- Szerver = Frissült
- szinkronizál
- írás
- Szerver = Szinkron
- szinkronizál

Fedettség

1. Teszteset után:

Állapotfedés: $2/3=66\%$

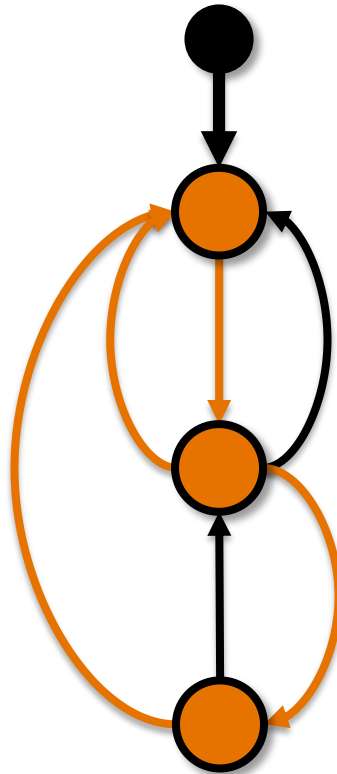
Átmenetfedés: $2/6=33\%$



2. Teszteset után:

Állapotfedés: $3/3=100\%$

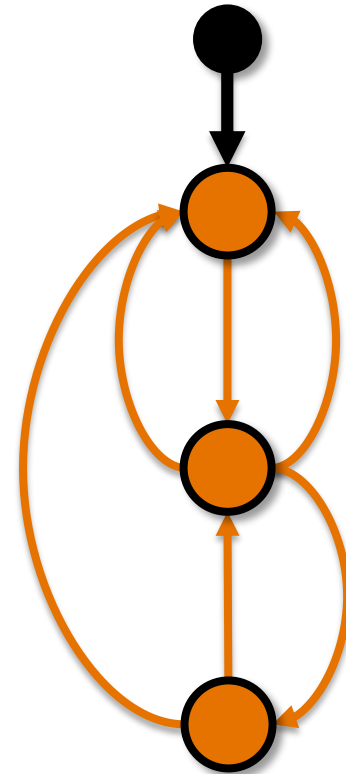
Átmenetfedés: $4/6=66\%$



3. Teszteset után:

Állapotfedés: $3/3=100\%$

Átmenetfedés: $6/6=100\%$



Tesztelt modellek használata

- **Szoftvertesztelés:**
 - (100%-osan fedő) tesztkészlet újrafelhasználása
 - Fedő tesztbemenetek (bemenet)
 - Modell által adott kimenetek (elvárt kimenet)
- **Monitorozás:** a modell szimulálása a szoftver futtatása mellett
 - Azonos bemenetek a modellnek és a programnak
 - Kimenetek összevetése → **hibadetektálás**
- **Logelemzés:**
 - Monitor futtatása naplózott bemenetek/kimeneten

Tesztelt modellek használata

■ Szoftvertesztelés:

- (100%-osan fedő) tesztkészlet újrafe
- Fedő tesztbemenetek (bemenet)
- Modell által adott kimenetek (elvárt kimenet)

Futtatás
előtt

■ Monitorozás: a modell szimulálása a szoftver futtatása mellett

- Azonos bemenetek a modellnek és a
- Kimenetek összevetése → **hibadetektálás**

Futtatás
közben

■ Logelemzés:

- Monitor futtatása naplózott bemene

Futtatás
után

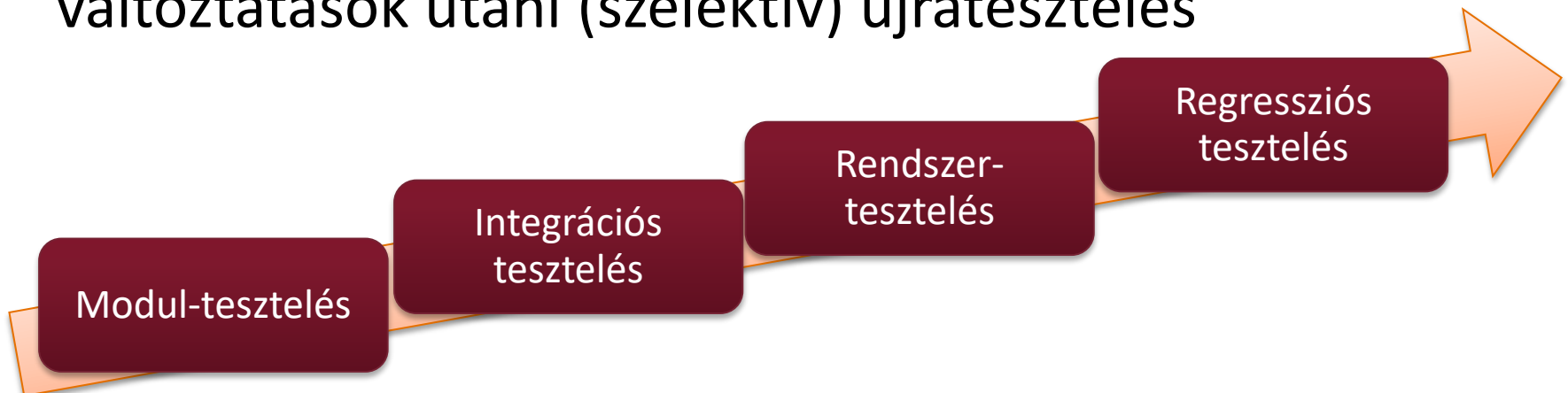
Teszt dokumentáció

- A teszteseteket és teszteredményeket is dokumentálni kell!
 - Mit vizsgál?
 - Milyen követelmény alapján?
 - Milyen bemenettel?
 - Futott-e már?
 - Ha igen, sikeres volt?
- Nyomonkövethetőség:
 - Teszteleetlen kódrészletek és ellenőrizetlen követelmények felderítése
 - Teszteredmények nyilvántartása



Tesztek fajtái, szakaszai

- **Modultesztelés:**
egy komponens leválasztása és tesztelése
- **Integrációs tesztelés:**
több komponens együttes tesztelése
- **Rendszertesztelés:**
a teljes rendszer együttes tesztelése
- **Regressziós tesztelés:**
változtatások utáni (szelektív) újratesztelés



Alapfogalmak

Statikus ellenőrzés

Tesztelés

Formális verifikáció

FORMÁLIS VERIFIKÁCIÓ

Modellellenőrzés

Formális verifikáció

- **Formális verifikáció:** modellek/programok helyességének bizonyítása matematikai eszközök segítségével
 - Bővebben lásd: Formális Módszerek MSc tárgy
- **Eszközök:**
 - **Modellellenőrzés**
 - Lehetséges viselkedések kimerítő vizsgálata
 - Automatikus helyességbizonyítás
 - Axiómarendszerek alapján automatikus tételbizonyítás
 - Konformancia vizsgálatok
 - Megfelelőség ellenőrzése modellek között

Modellellenőrzés

- **Modellellenőrzés:** a modell lehetséges viselkedéseinek kimerítő (teljes) vizsgálata adott követelmények alapján
 - Hibás működések keresése
- **ellenpélda**

| Tesztelés | Modellellenőrzés |
|------------------------------|-------------------------------|
| Szűrőpróbaszerű | Kimerítő/teljes |
| Elvárt kimeneteket ellenőriz | Állapotok sorozatát ellenőrzi |
| Kisebb számítási igényű | Nagy számítási igényű |
| Nem bizonyítóerejű | Formálisan bizonyít |