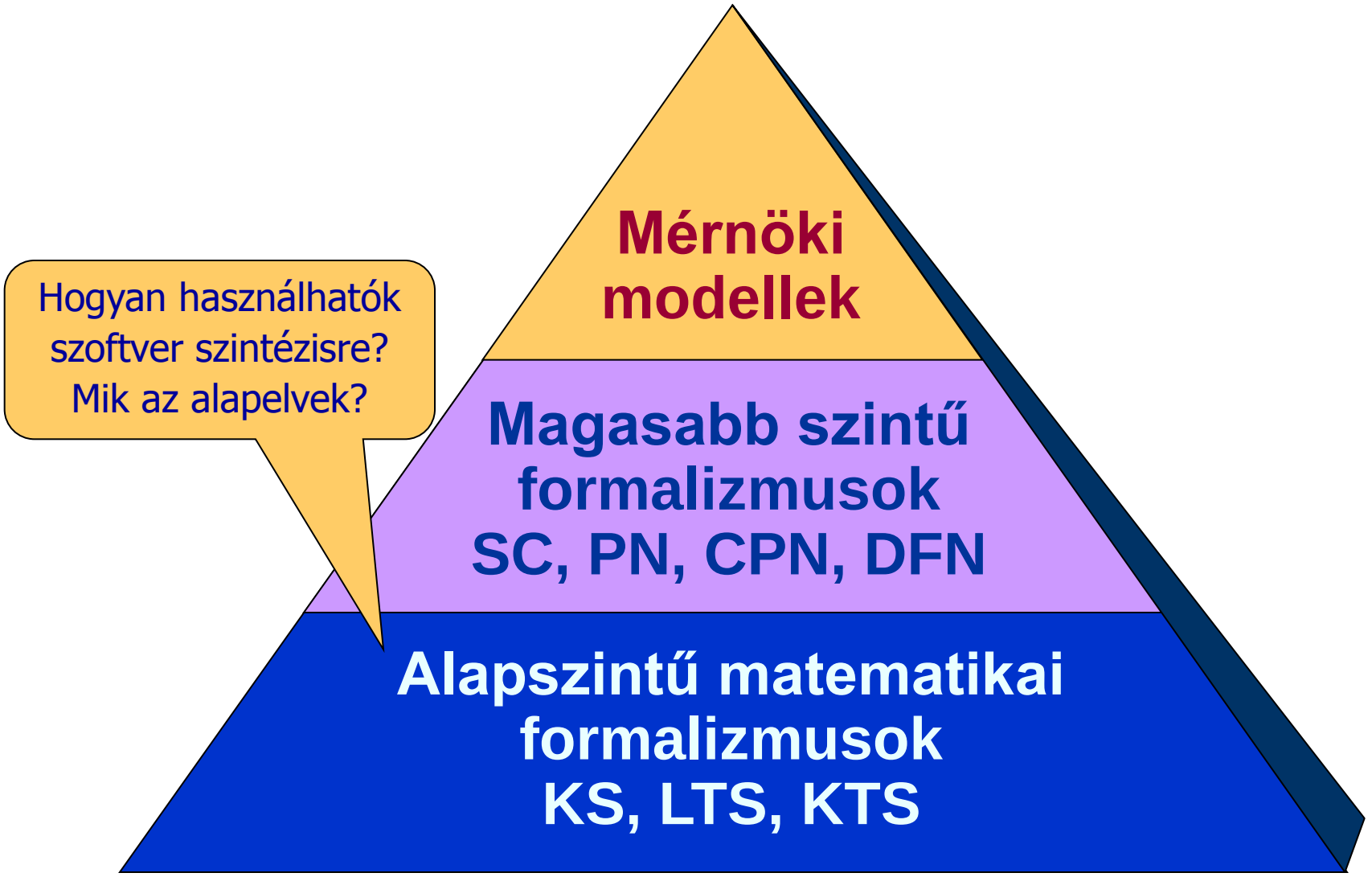
Forráskód generálás formális modellek alapján

dr. Majzik István
Horányi Gergő és Jeszenszky Balázs (TDK)
BME Méréstechnika és Információs Rendszerek Tanszék

Modellek a formális ellenőrzéshez



Tartalomjegyzék

Forráskód szintézis
időzített automata modellek alapján

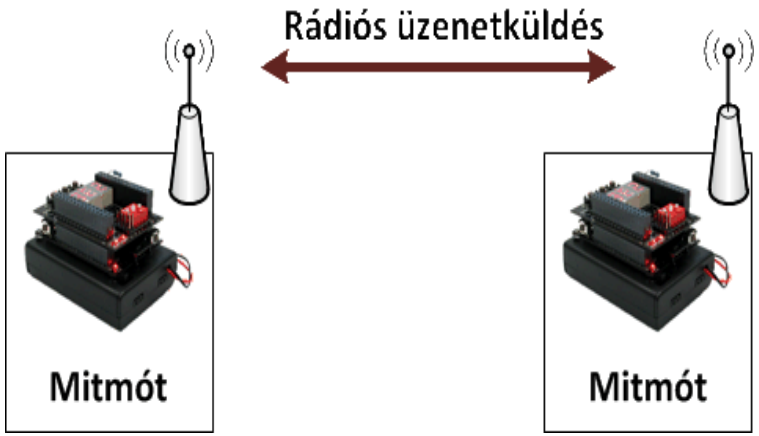
Az alkalmazás és a formalizmus

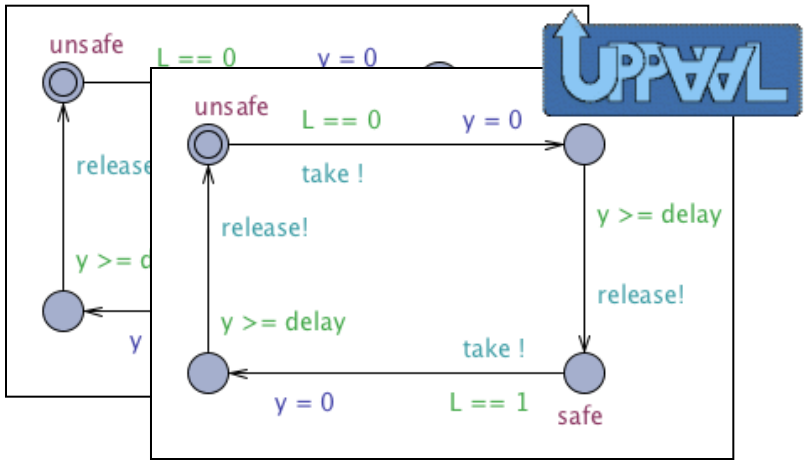
Beágyazott vezérlők:

- Eseményvezérelt, állapot alapú
- Egyszerű akciók
- Elosztott is lehet
- Kommunikáció
- Valósidejű működés

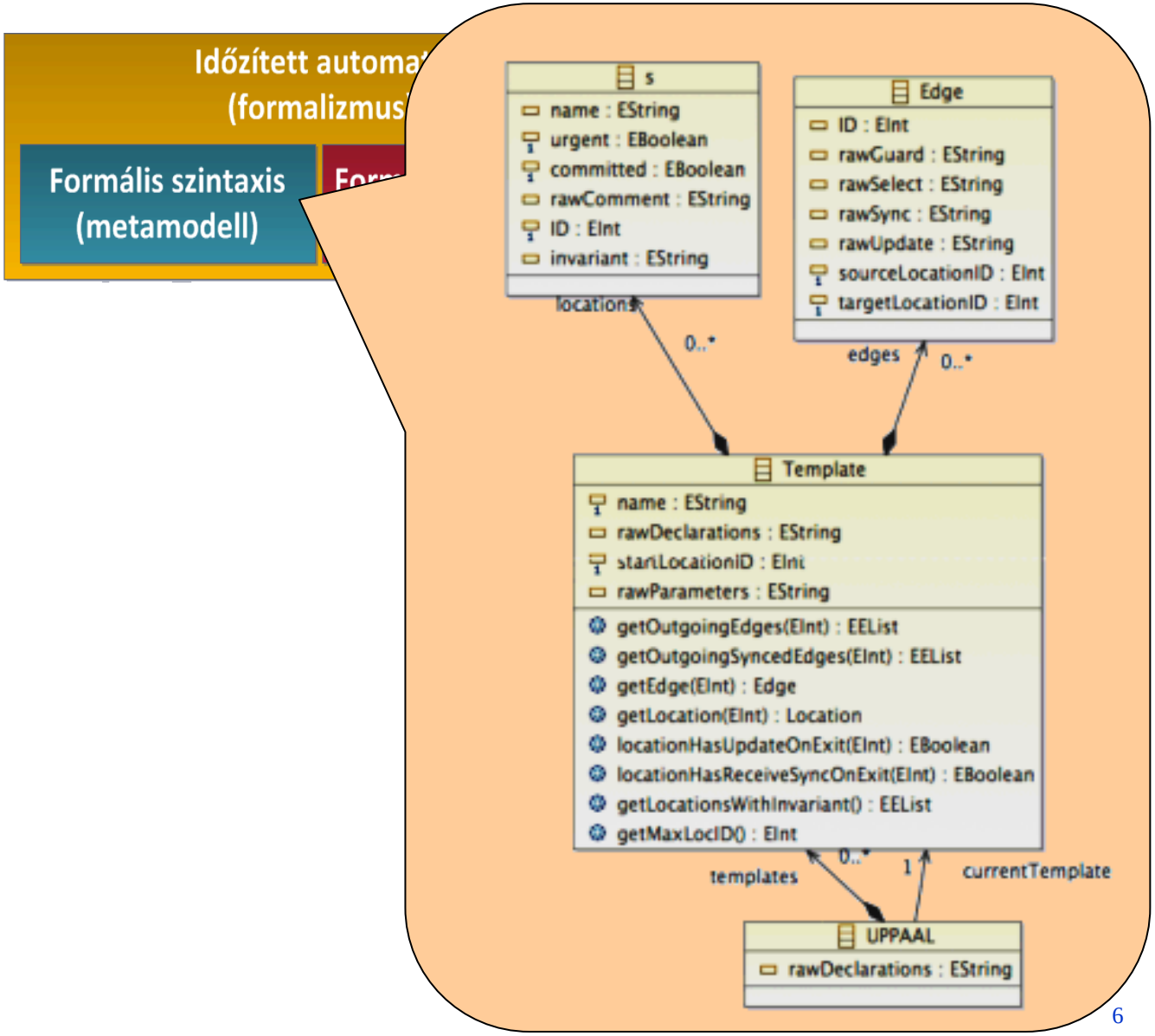
Időzített automaták:

- Véges automata alapú (állapotok, átmenetek)
- Akciók változókon
- Automaták hálózata
- Szinkron kommunikáció
- Óraváltozók használata

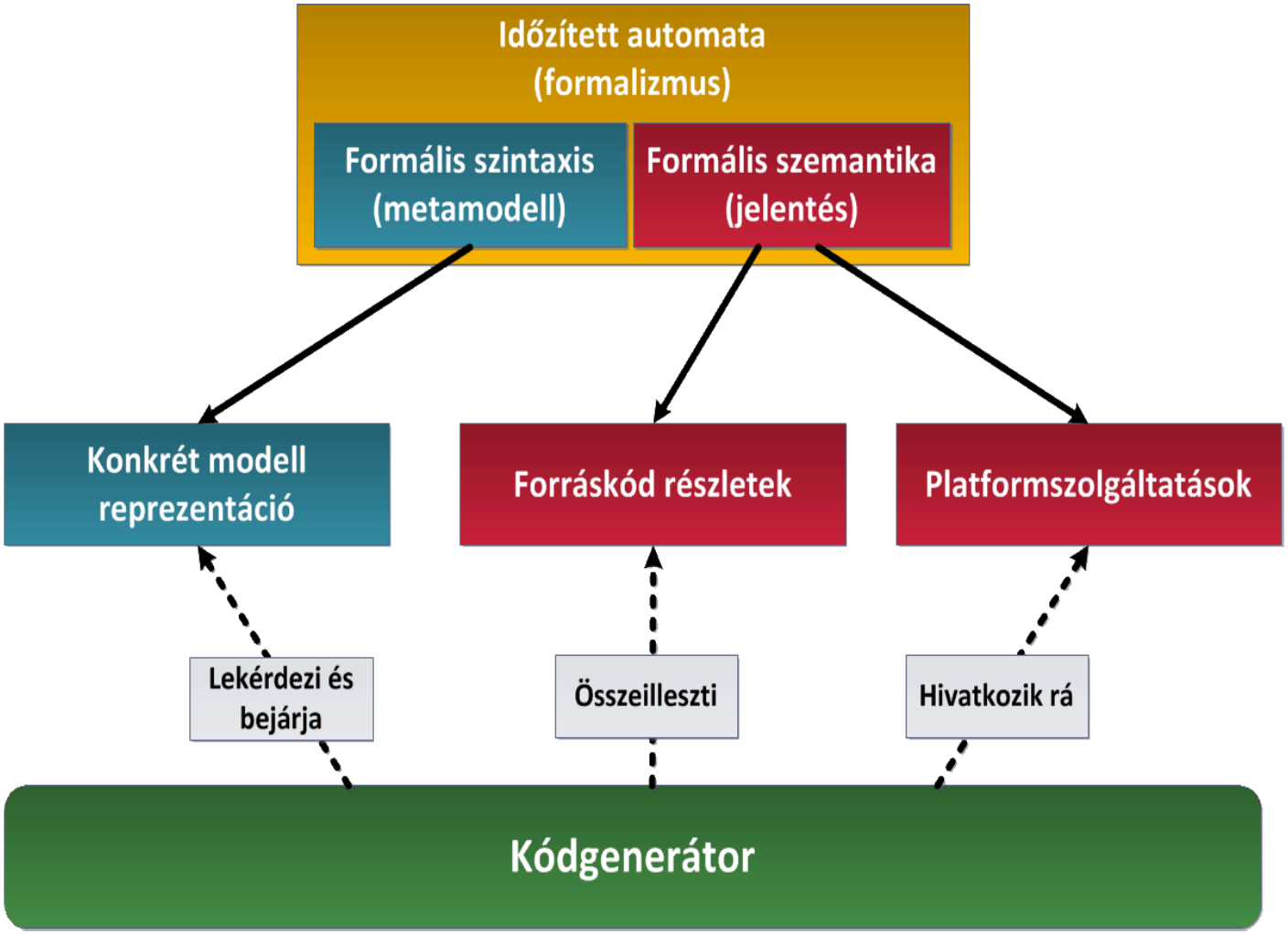




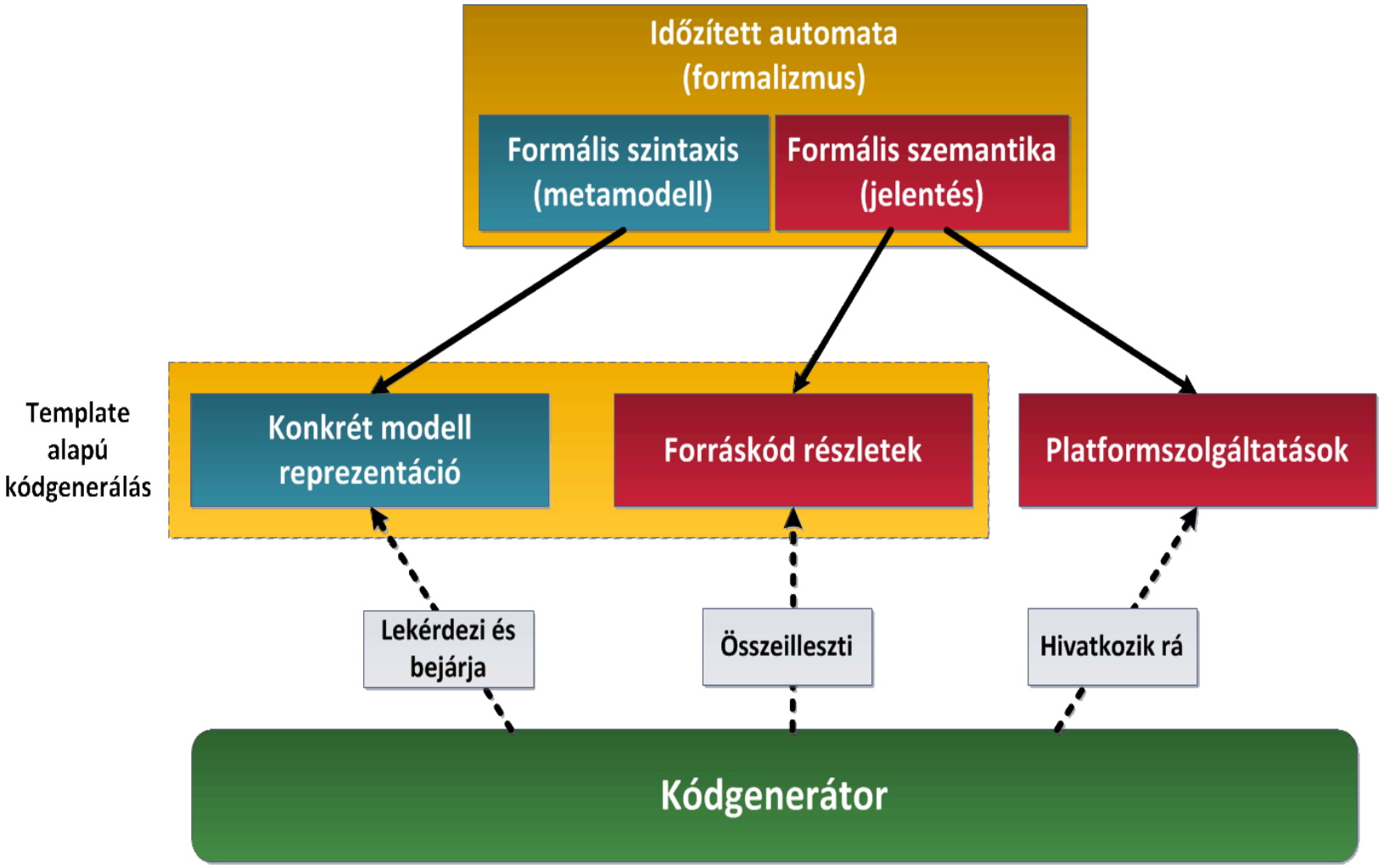
A kódgenerálás alapelvei



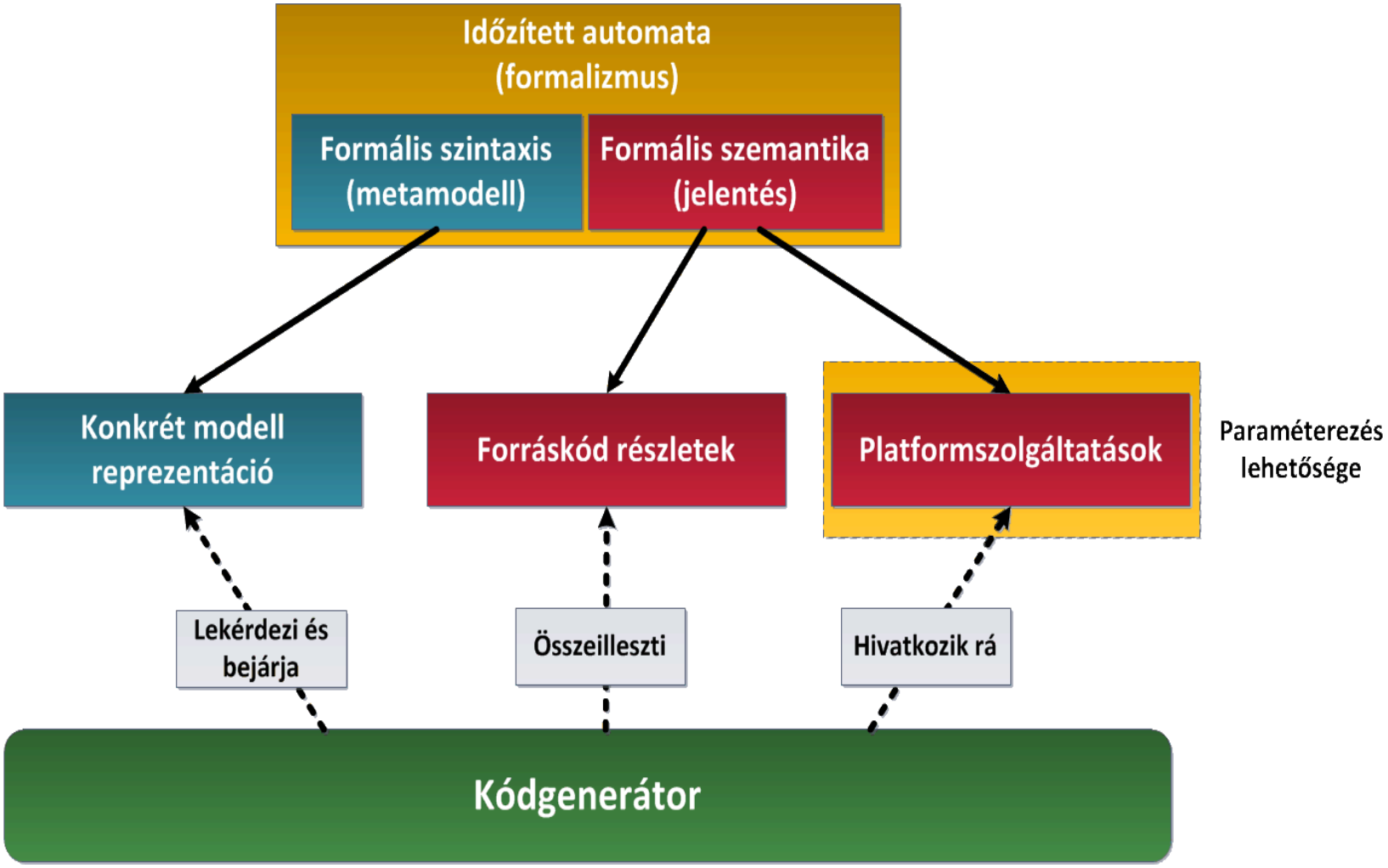
A kódgenerálás alapelvei



A kódgenerálás alapelvei

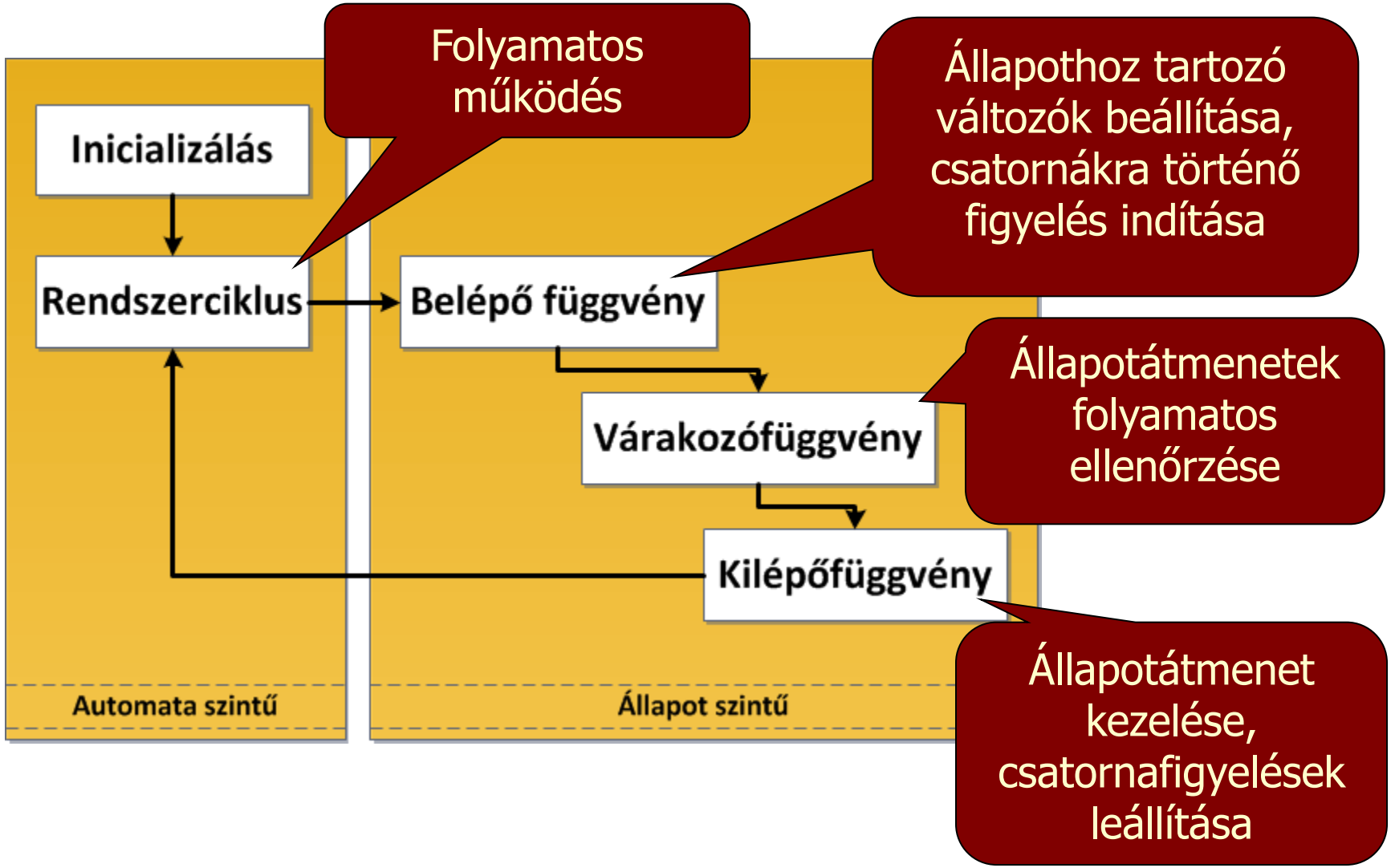


A kódgenerálás alapelvei



A formális szemantika leképzése forráskódra

- A generált kód felépítése egy automata esetén:



A kódgenerálás megvalósítása

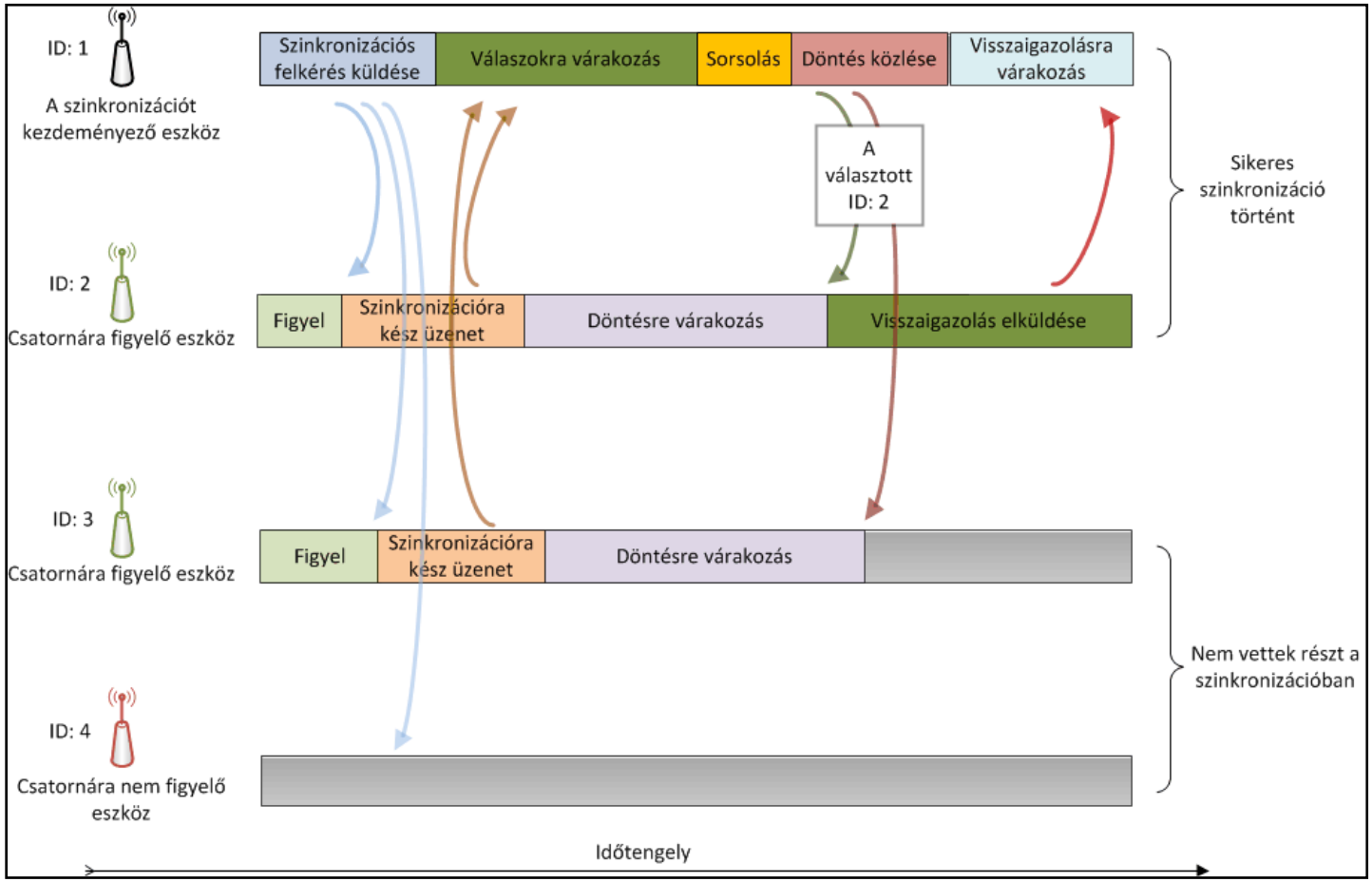
- Template alapú kódgenerálás:
  Példa technológia: Java Emitter Templates (JET)
  - Java utasítások: Modell bejárása és elemek azonosítása
  - Kódgenerálási minta: C kódrészletek kiírása

<% Java utasítás %>	<%= kiírandó eredmény (Java utasításból) %>

```
<%for (Location loc : template.getLocations()) { %>
  void enterToLocation<%= loc.getID() %> ( ) {
    stateReg = <%= loc.getID() %>;
    waitFunc = &waitInLocation<%= loc.getID() %>;
    exitFunc = &exitFromLocation<%= loc.getID() %>;
   <%if (settings.getLoggingMode() == SettingsHandler.LoggingModes.OFFLINE) { %>
            offlineLogFunction(<%=loc.getID()%>, locationLog);
    …
  }
…
```

A platformszolgáltatások megvalósítása

- Nemtriviális kódrészleteket igényel (függvénykönyvtár)
  - Időkezelés (hardver beállítása, interrupt, …)
  - Kommunikáció (pl. szinkron kommunikáció)



A platformszolgáltatások beillesztése

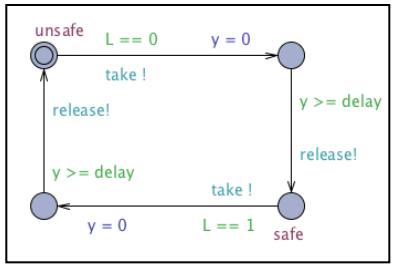
Szemantikához kötődő elvárások:
- Kommunikáció
- Időkezelés

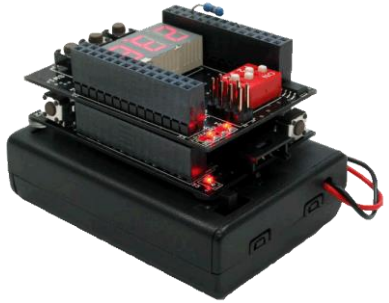
Alkalmazási terület szolgáltatásai:
- Fizikai bemenetek és kimenetek
- Platform funkció indítása

Absztrakt szolgáltatások

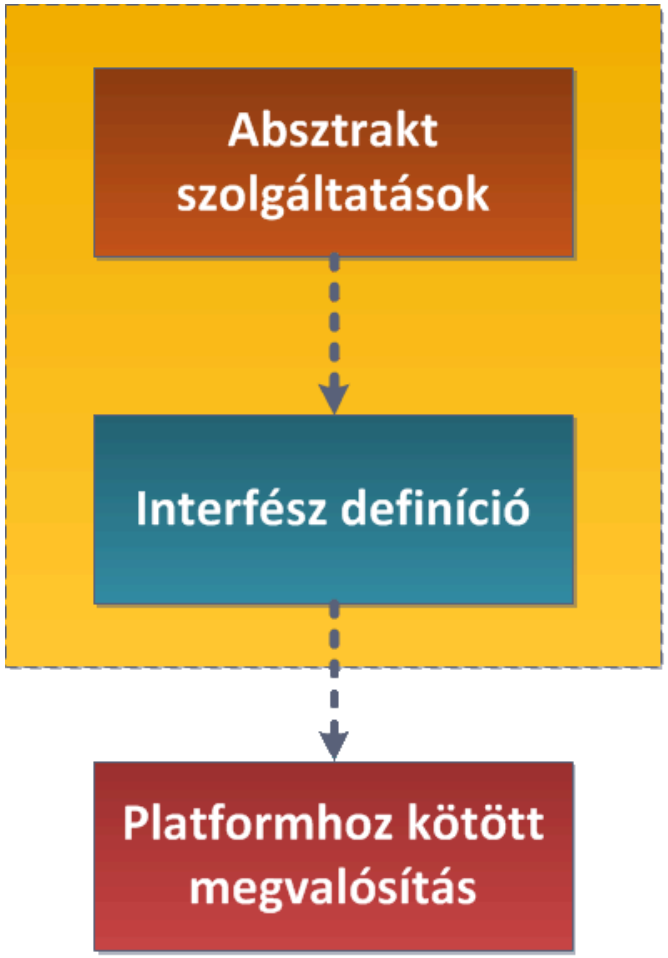
Interfész definíció

Platformhoz kötött megvalósítás

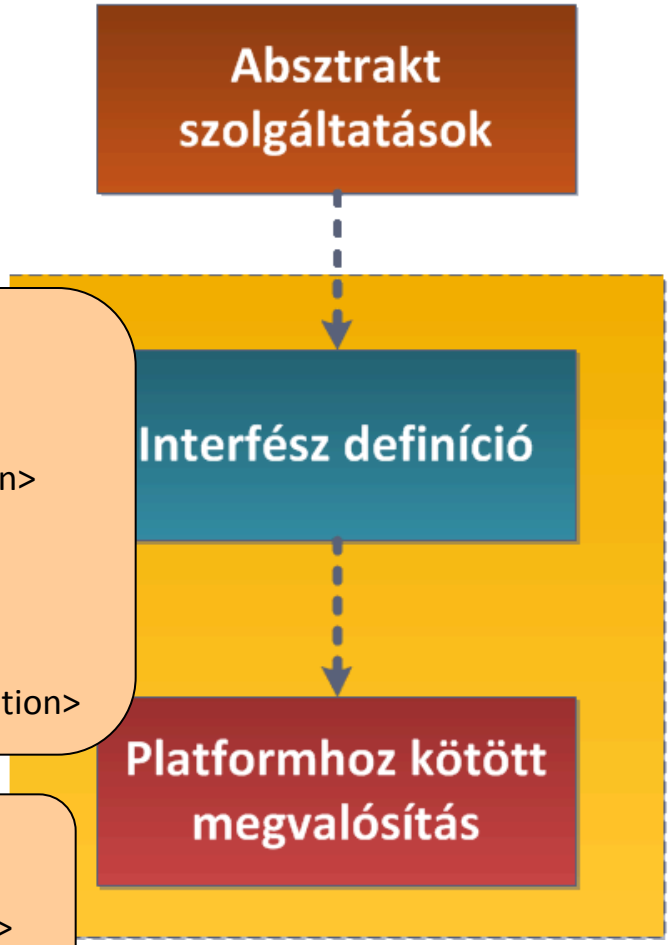




A platformszolgáltatások beillesztése



A platformszolgáltatások beillesztése

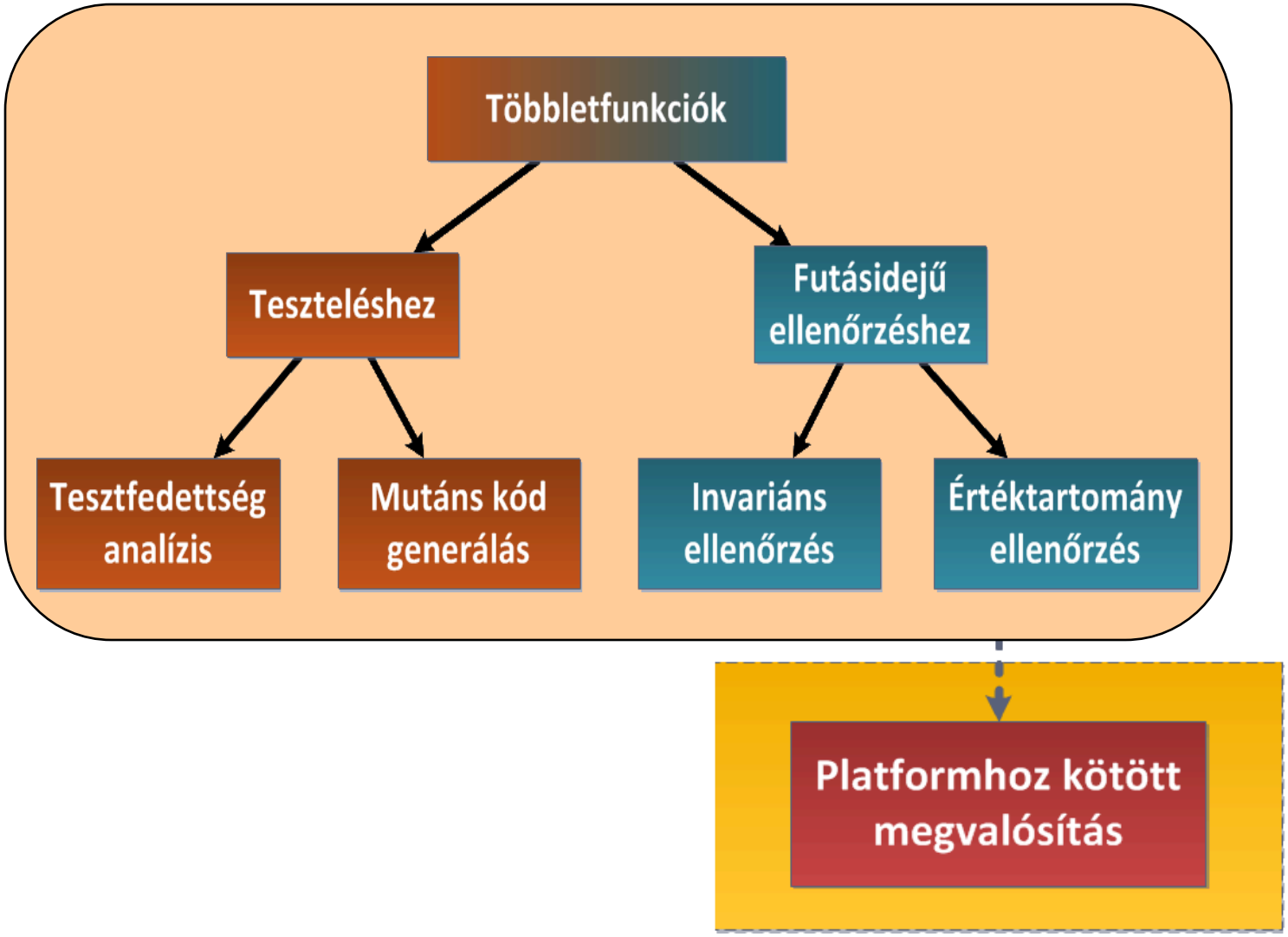


Konfigurációs állomány

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <communication>
    <implementation>../../API/API.c</implementation>
  </communication>
  <timing>
    <include>SDL/SDL.h</include>
    <waitFunction>SDL_Delay(100)</waitFunction>
    <implementation>clockFunctions.c</implementation>
```

```
<logging>
  <mode>ONLINE</mode>
  <implementation>logFunctions.c</implementation>
</logging>
```

Többletszolgáltatások a generált kódban

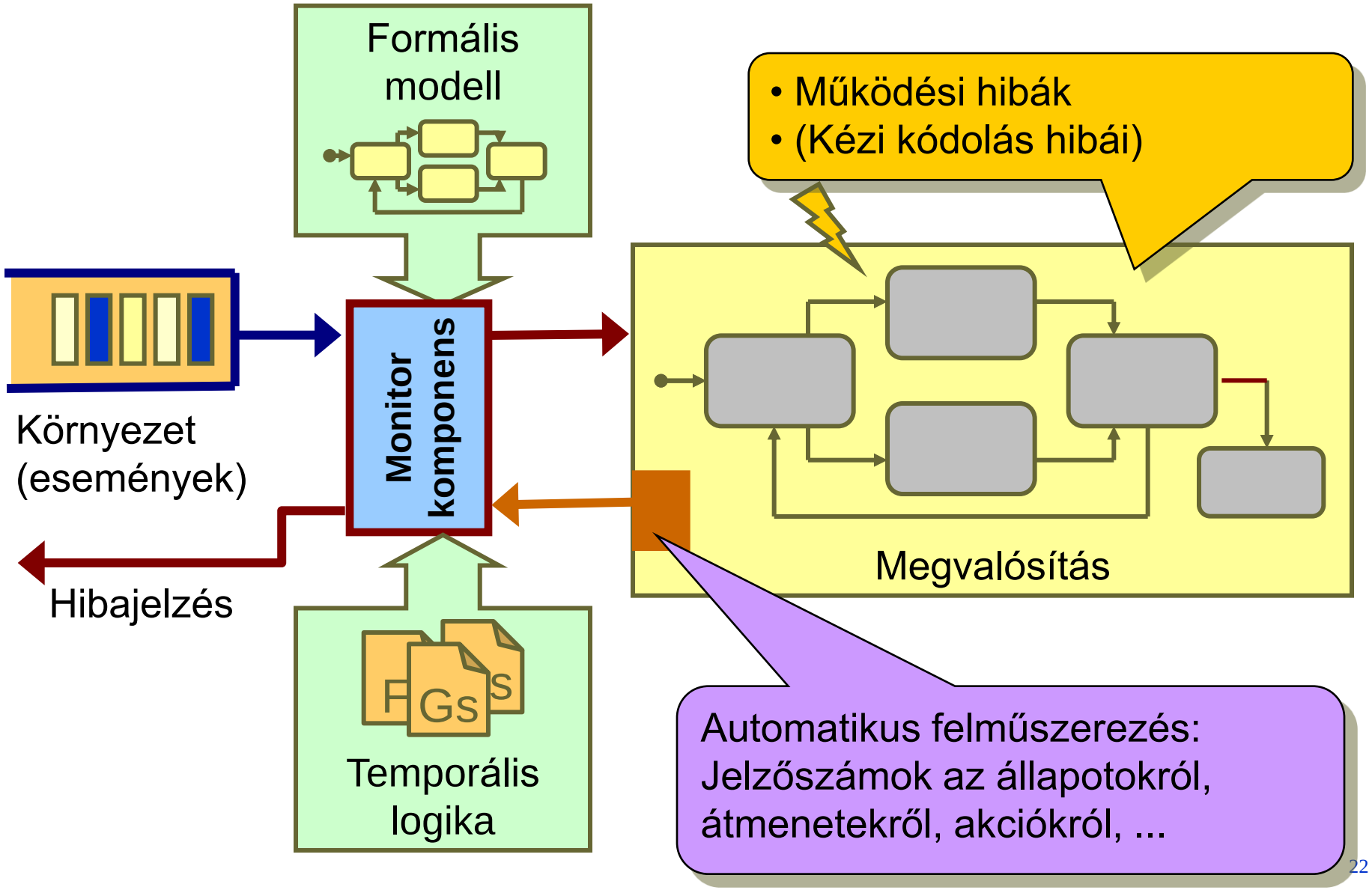


Futásidejű monitor szintézis a követelmények alapján

Bevezetés

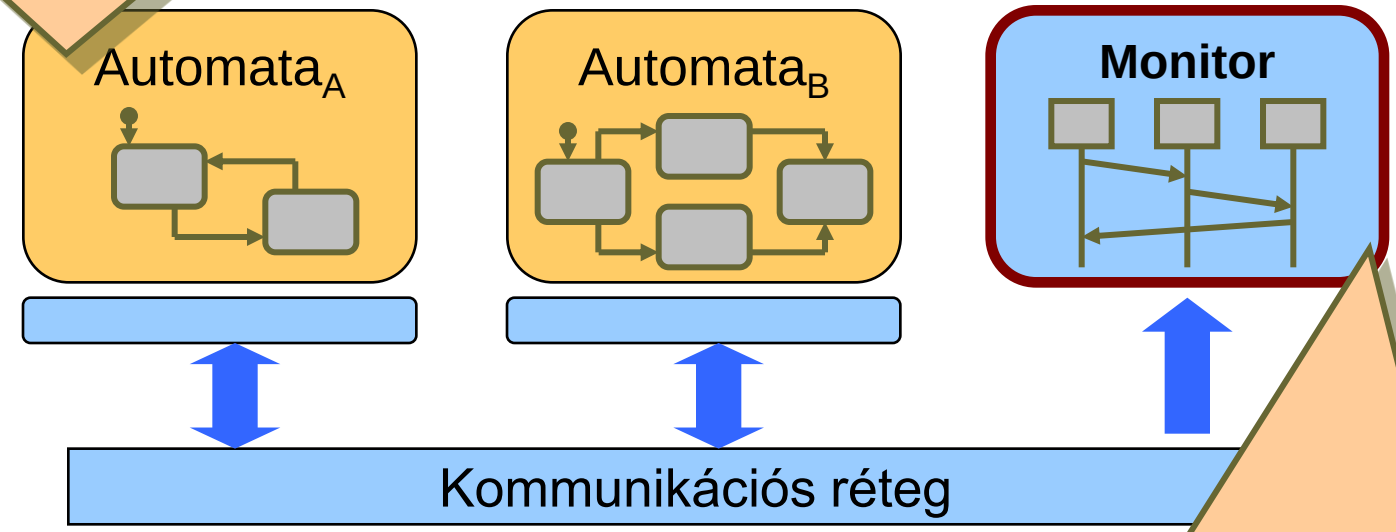
- Futásidőbeli verifikáció:
  - Tényleges viselkedés (programfutás) összevetése referencia viselkedéssel (modell vagy specifikáció)
    - A hibák okozta eltérések detektálhatók
    - Tényleges viselkedésről futásidejű információ: Passzív megfigyelés vagy jelzőszámok átvitele az alkalmazásból
    - Referencia viselkedésről tárolt információ: Formális modell vagy temporális logikai specifikáció
- Megvalósítási példák
  - Önellenőrzés:
    - Végrehajtható ellenőrző kódrészletek (assertions)
  - Független ellenőrzés:
    - Futásidejű monitor
    - Watchdog processzor
- Szabvány előírás biztonságkritikus rendszerekben

Belső viselkedés monitorozása



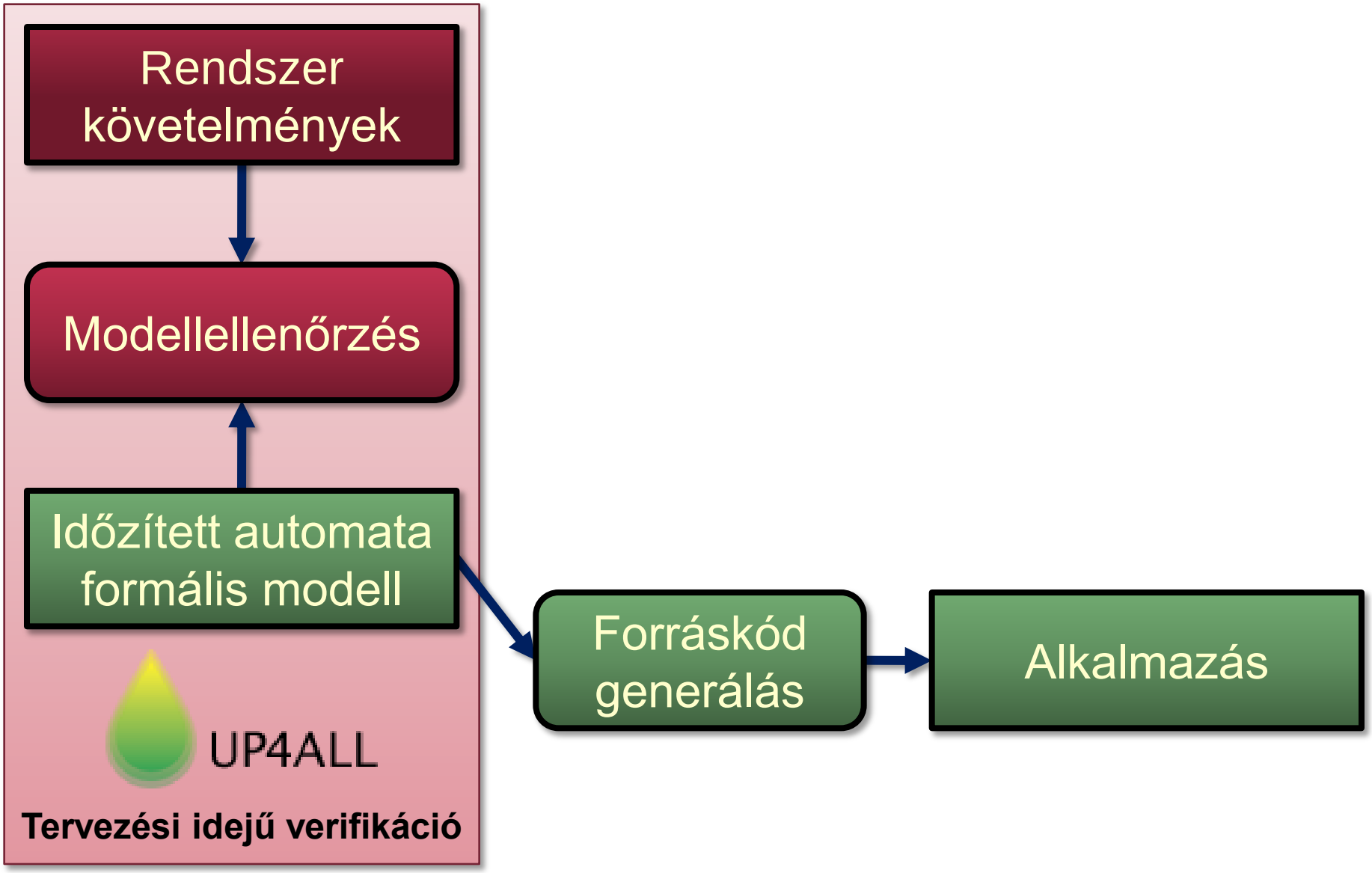
Automaták közötti interakciók monitorozása

- Működési hibák
- Kódolási hibák (protokoll)



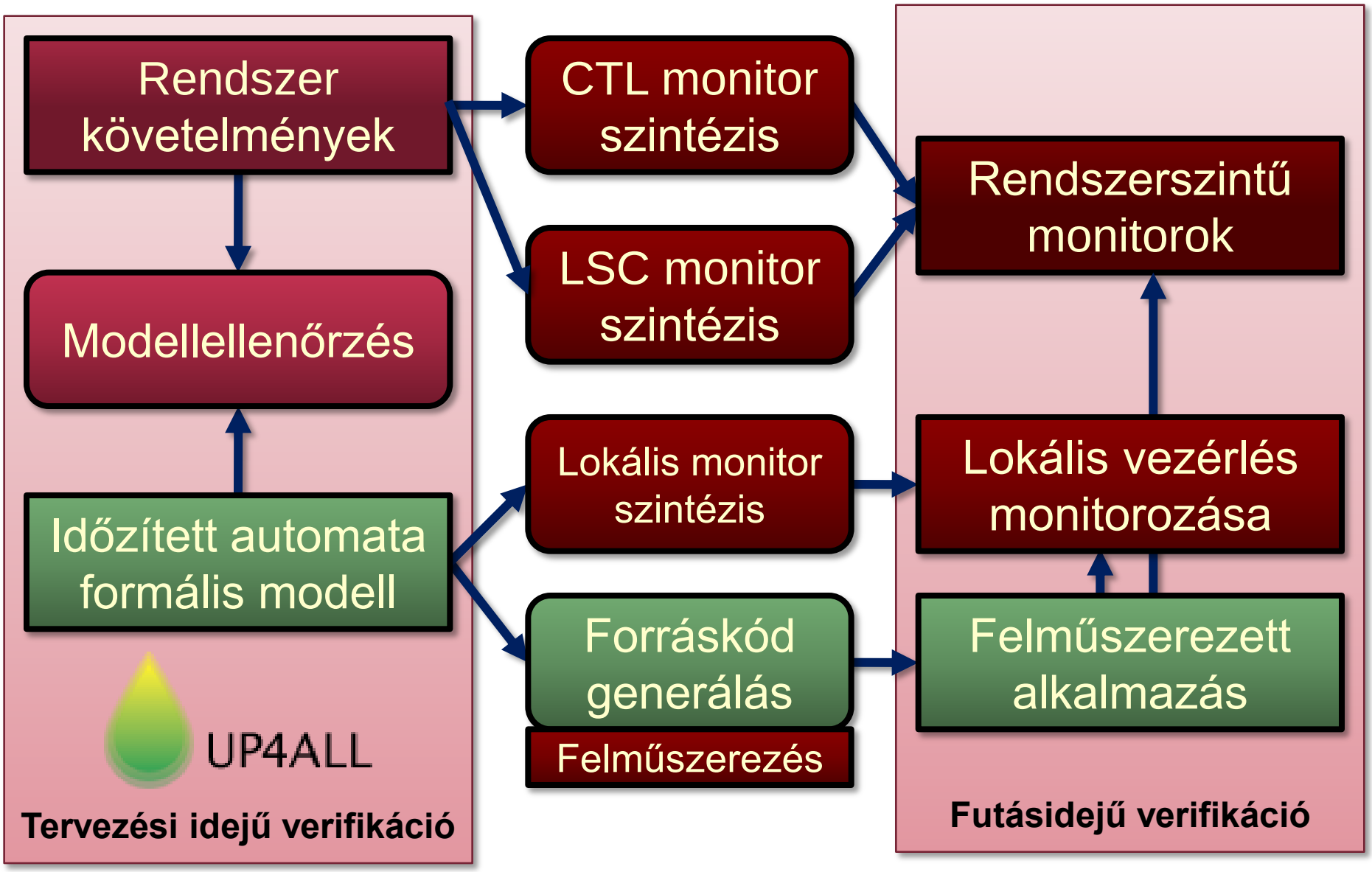
- Monitor a kommunikációs réteghez
- Referencia információ:
  - Temporális logikai követelmények
  - Szekvencia diagram (scenariok)

Kódgenerálás



24

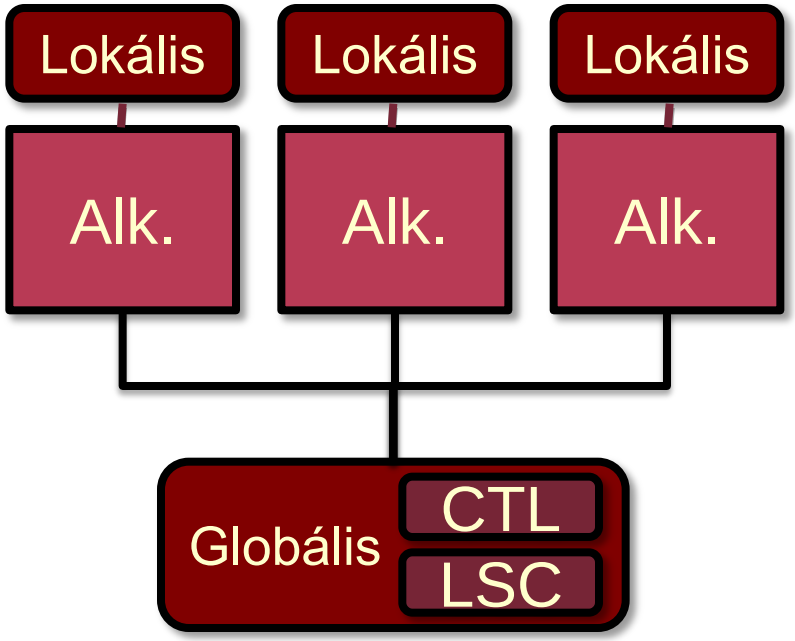
Kódgenerálás és futásidejű verifikáció



25

Hierarchikus monitorozás

- Két hierarchiaszint:
  - Lokális monitorozás
    - Vezérlési folyam ellenőrzés
    - Lokális CTL kifejezések
  - Rendszerszitű monitorozás
    - Rendszerszintű CTL kifejezések
    - Rendszerszintű scenario-k (LSC)



- Előnyök:
  - Monitor szintézis és felműszerezés az ellenőrizendő követelményekhez optimalizálható (kis overhead)
- Használati esetek
  - Futásidőben: Hibadetektálás (egy-egy konkrét futásra)
  - Tesztelés közben: Teszt kiértékelés (itt több futásra is)

26

Vezérlési folyam ellenőrzése

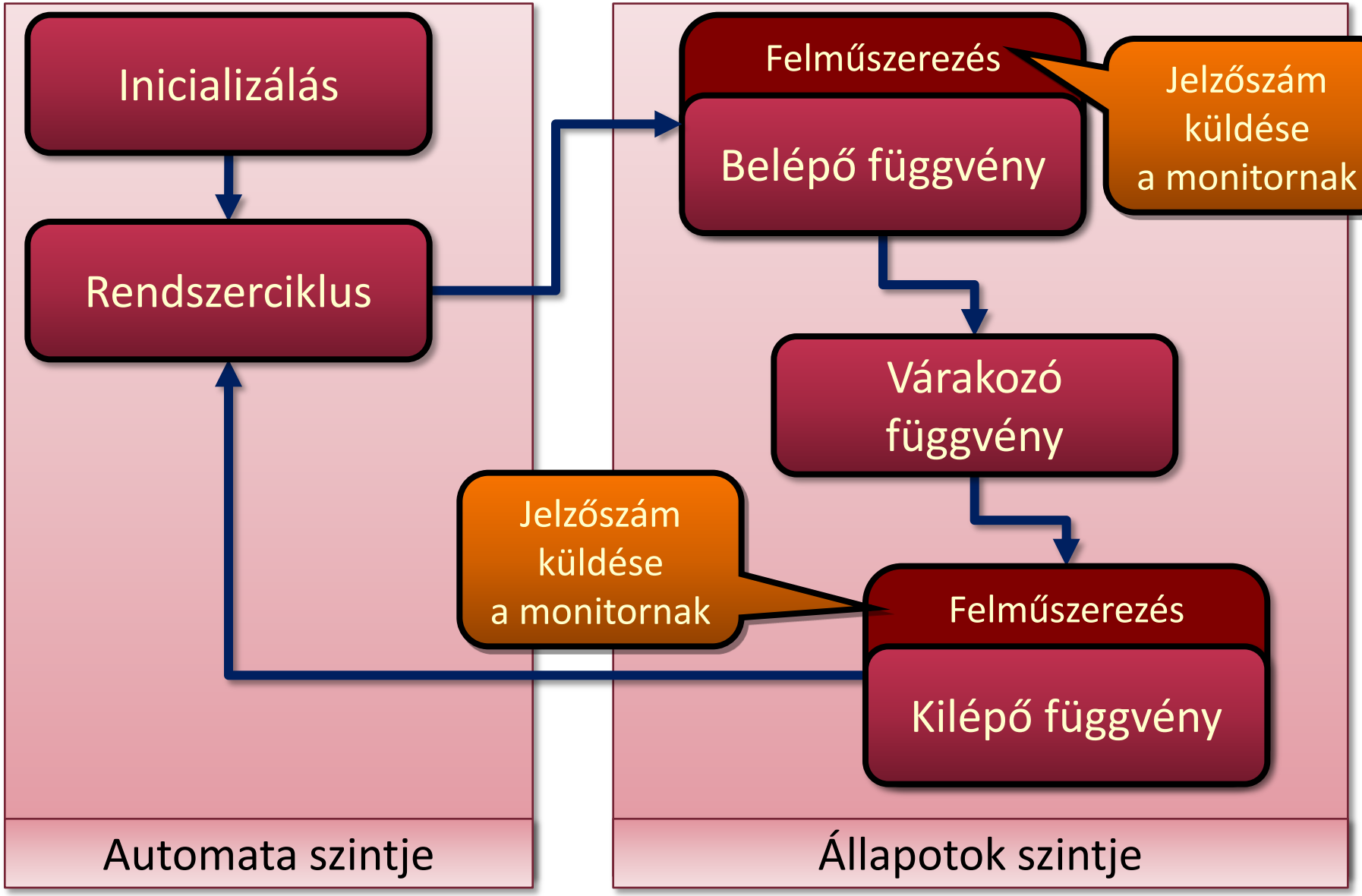
- A tranziens hibák többsége vezérlési hibát okoz

Monitor szintézis	Alkalmazás felműszerezés

▪ Az állapotok és átmenetek futásidejű szekvenciájának ellenőrzése
▪ Az időzített automata modell a referencia az ellenőrzéshez
▪ A monitor forráskódja automatikusan generálható az automata modellből
	▪ Minden állapot és átmenet felműszerezése megtörténik: Azonosítás a monitor számára
	▪ Kiterjesztések:
	  ▪ Idő invaránsok ellenőrzése
	  ▪ Deadlock ellenőrzés szívdobbanás (heartbeat) üzenetek alapján

27

Vezérlési folyam lokális ellenőrzése: Felműszerezés



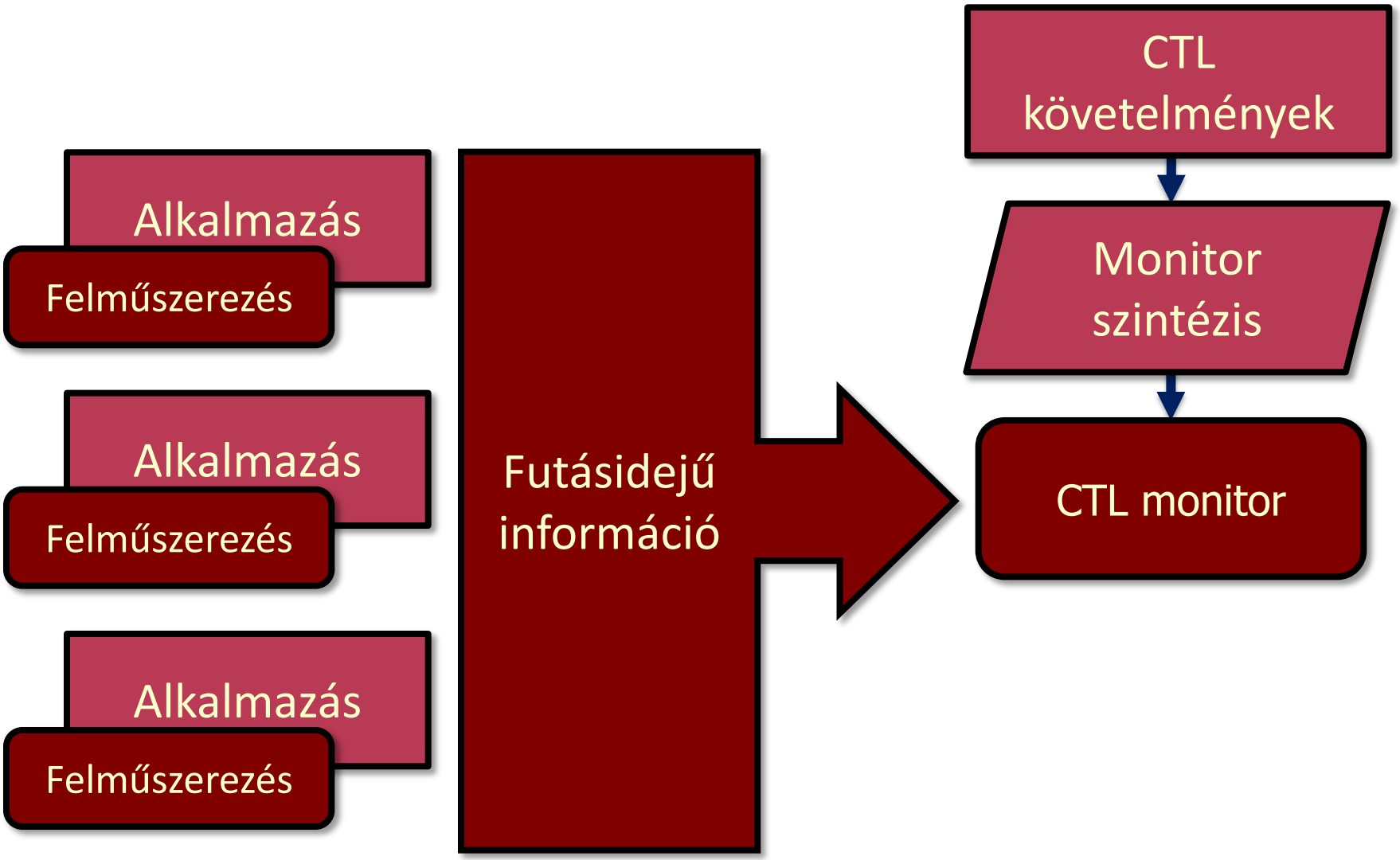
29

Temporális logika alapú monitorozás

- Követelmények: Temporális elérhetőség
  - Biztonsági követelmények: Invariáns kifejezések
  - Élő jellegű követelmények: Egzisztenciális kifejezések
- Elágazó idejű temporális logikai kifejezések
  - Timed CTL variáns az UPPAAL esetén (TCTL)
    - Temporális operátorok korlátozott készlete
    - Temporális operátorok nem egymásba ágyazhatók
    - Óraváltozók használhatók
- Automatikus és optimalizált felműszerezés
  - Csak a követelményekben hivatkozott állapotok és átmenetek esetén kell a monitort értesíteni

30

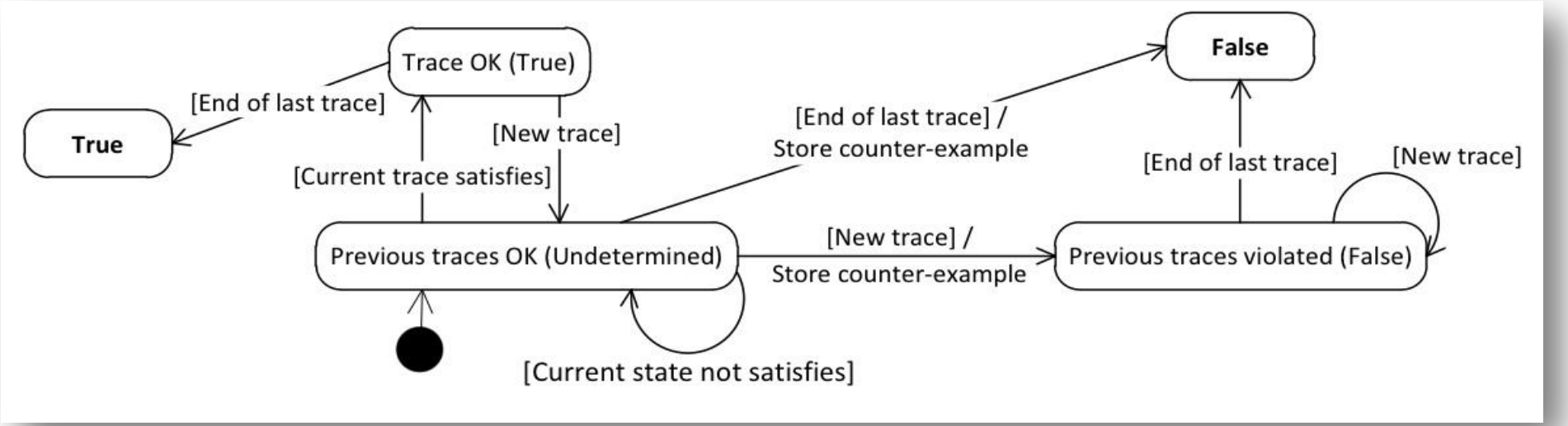
A monitorozás sémája



31

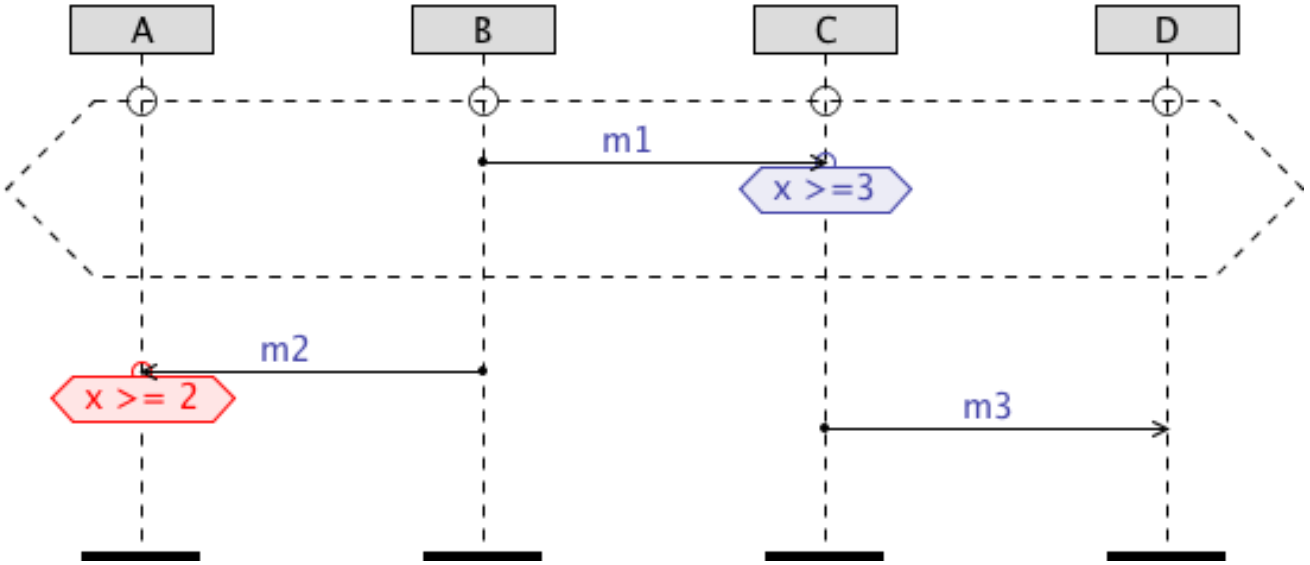
Monitorok szintézise CTL követelményekhez

- Cél: A követelmény alapján megfigyelő automata konstruálása a futásidejű szekvenciákhoz
  - Bemenetek: Jelzőszámok a hivatkozott lokális feltételek teljesüléséről, szekvencia kezdete (New trace), szekvencia vége (End of last trace)
  - Kimenet: Elfogadva (True), hiba (False), vagy nem meghatározott
- Példa: Az AF φ temporális operátor szemantikája alapján konstruált megfigyelő automata (itt φ adott állapotban lokálisan kiértékelhető)
  - Automatikus megvalósítás (C nyelven)



Scenario alapú követelmények

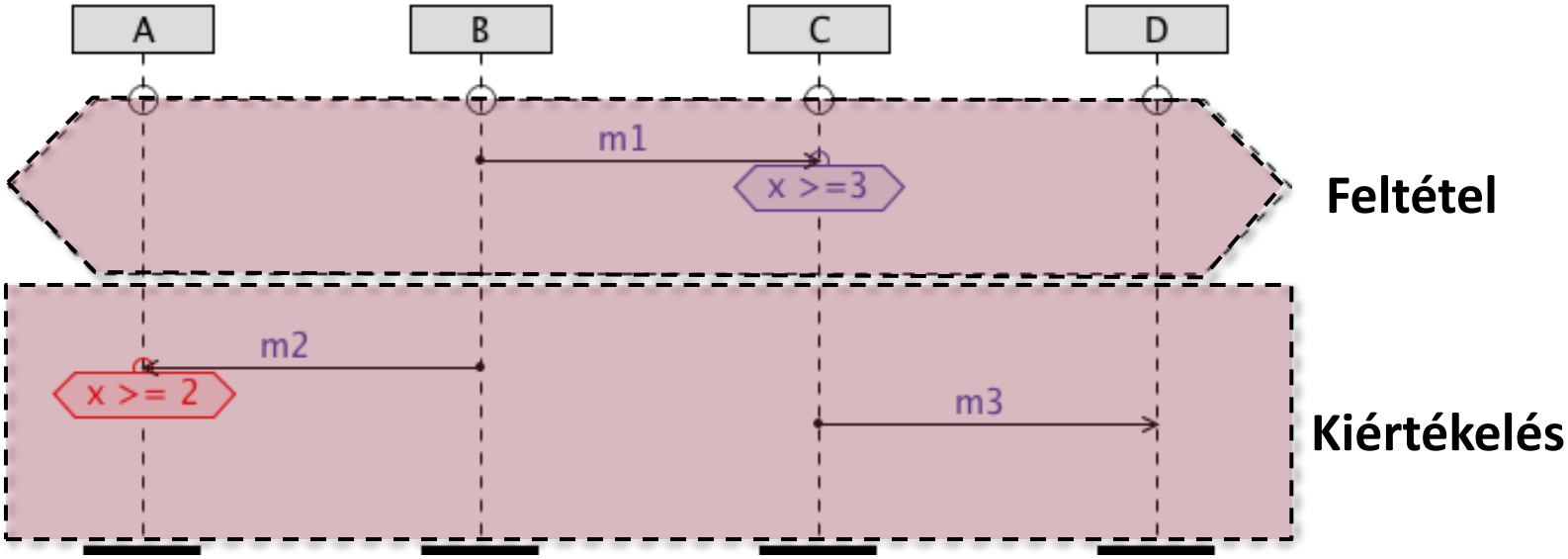
- Live Sequence Chart (LSC)
  - Az üzenet szekvencia diagram (MSC) kiterjesztése
  - Formalizált szemantika
- Intuitívabb, mint a temporális logika a lefutások követelményeinek megadására



33

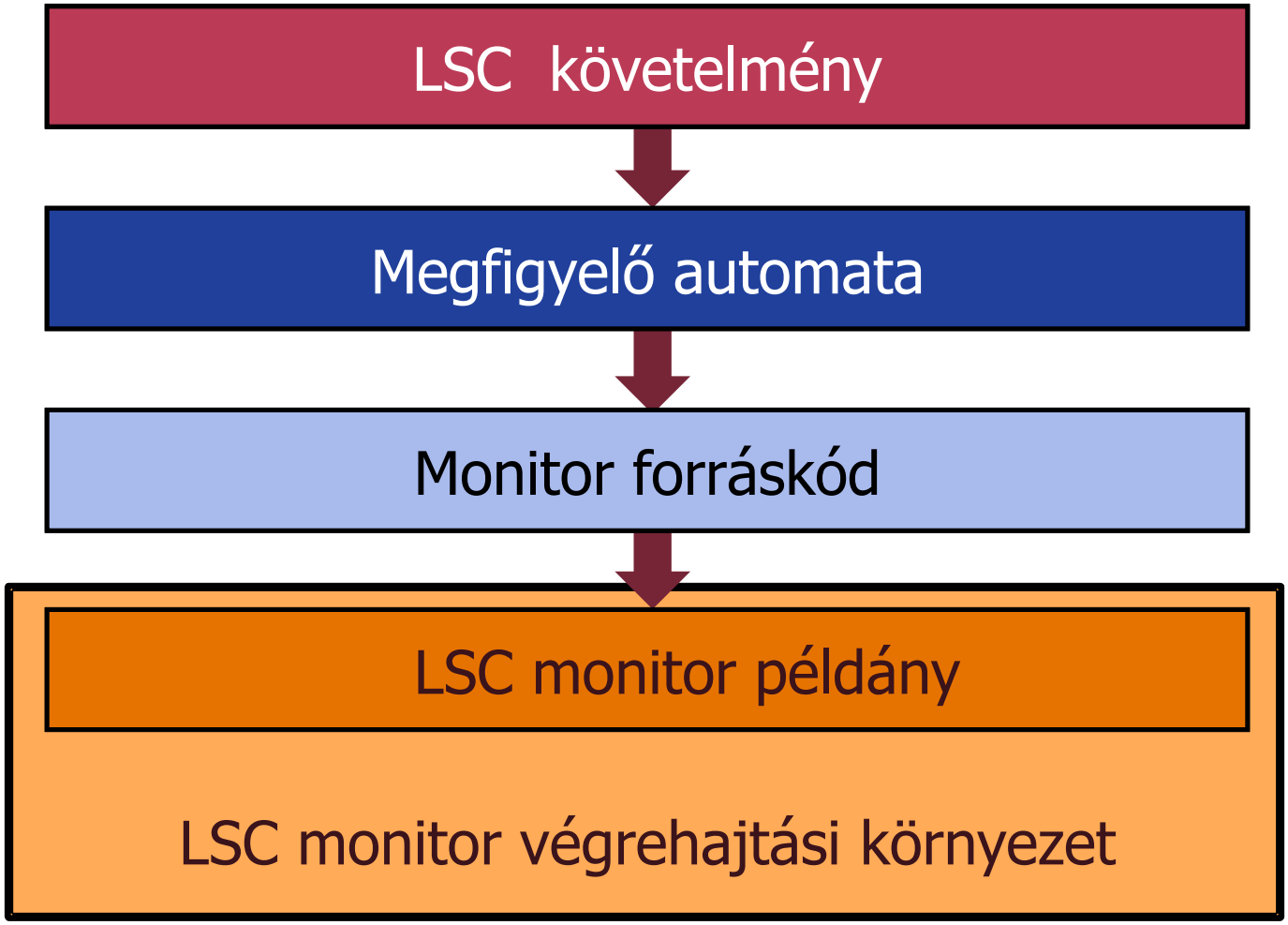
Scenario alapú követelmények

- Live Sequence Chart (LSC)
  - Az üzenet szekvencia diagram (MSC) kiterjesztése
  - Formalizált szemantika
- Intuitívabb, mint a temporális logika a lefutások követelményeinek megadására



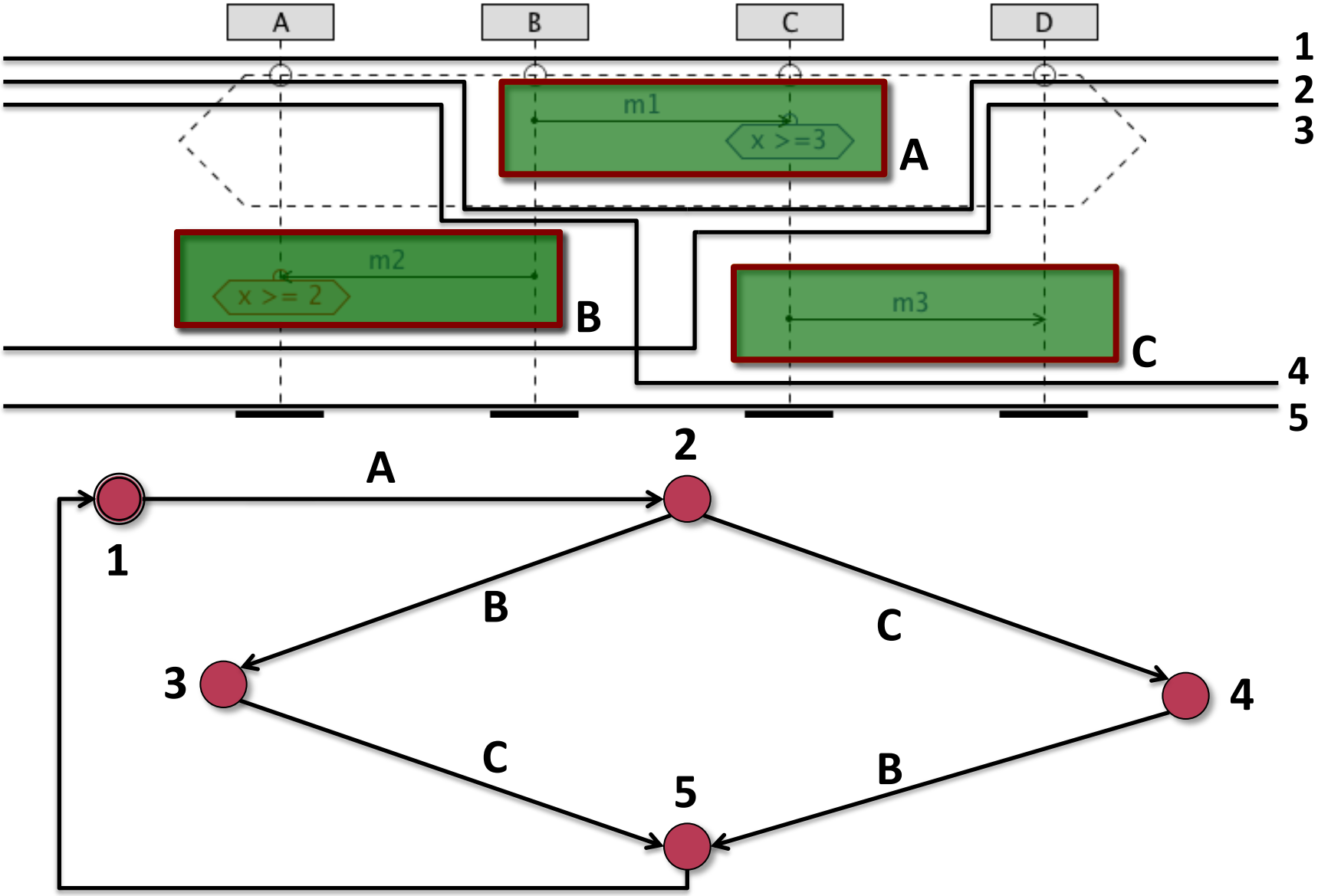
34

Scenario alapú monitorozás



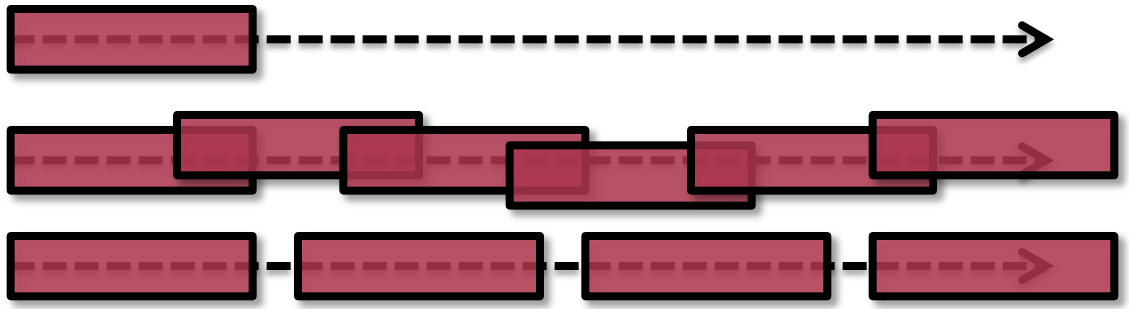
35

Megfigyelő automata konstrukciója LSC-hez



LSC monitor végrehajtási környezet

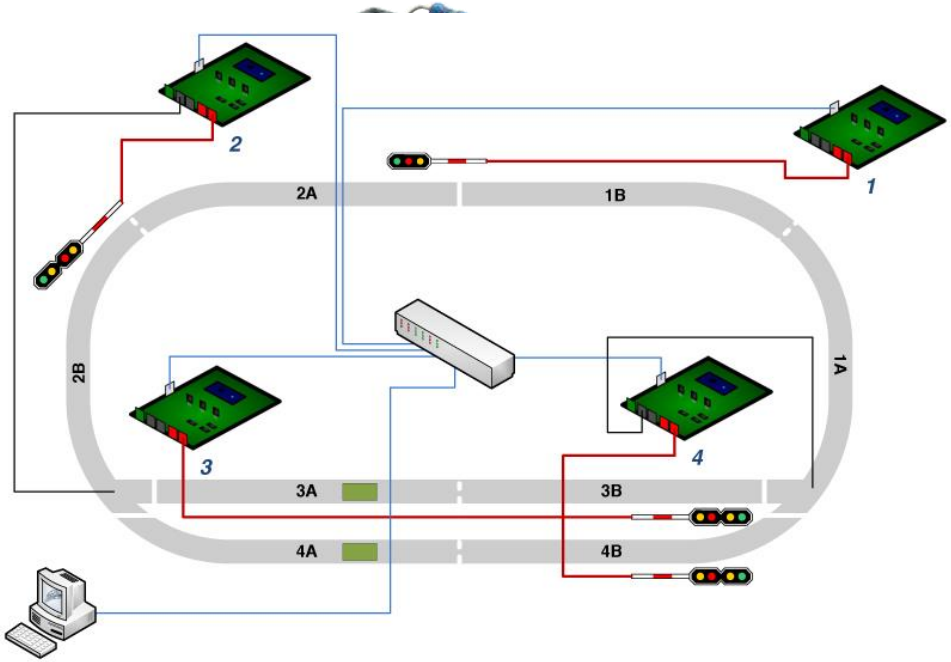
- Feladata: LSC monitorok indítása, leállítása
- A monitorok értesítik a státuszukról
- Támogatott LSC típusok:
  - Egzisztenciális
  - Univerzális
- Támogatott aktiválási módok
  - Kezdeti
  - Invaráns
  - Iteratív

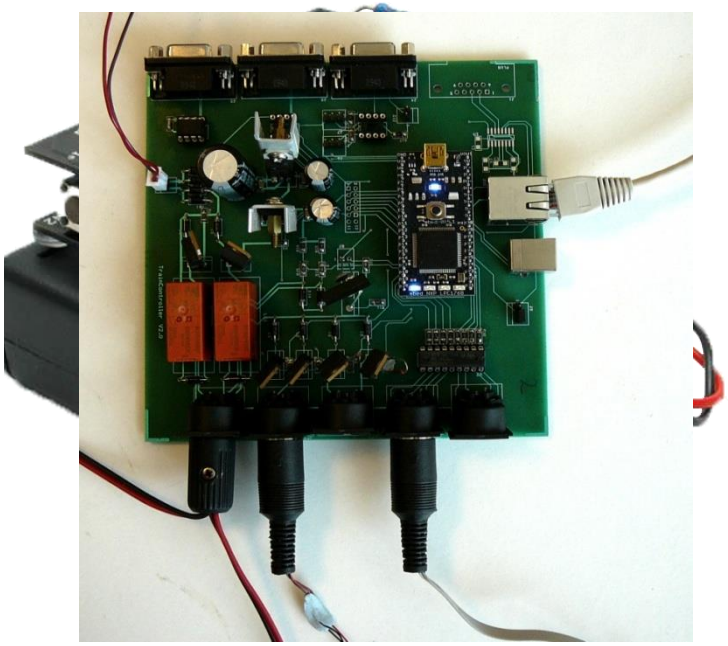


37

Megvalósítás

- Kétféle beágyazott platform
  - Moduláris mitmót vezetéknélküli kommunikációval
    - Ipari mintapélda: Bit szinkronizációs protokoll
  - mbed mikrokontroller (ARM Cortex-M3, 96 MHz)
    - Oktatási célú mintapélda: Modellvasút vezérlés

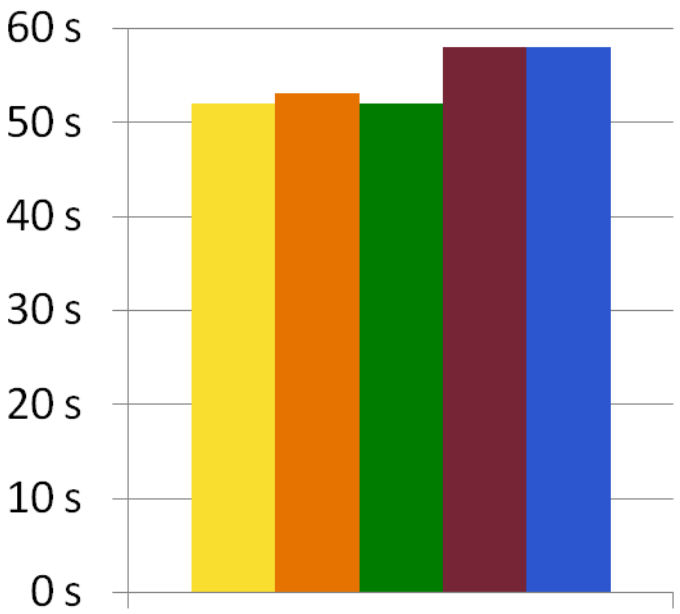




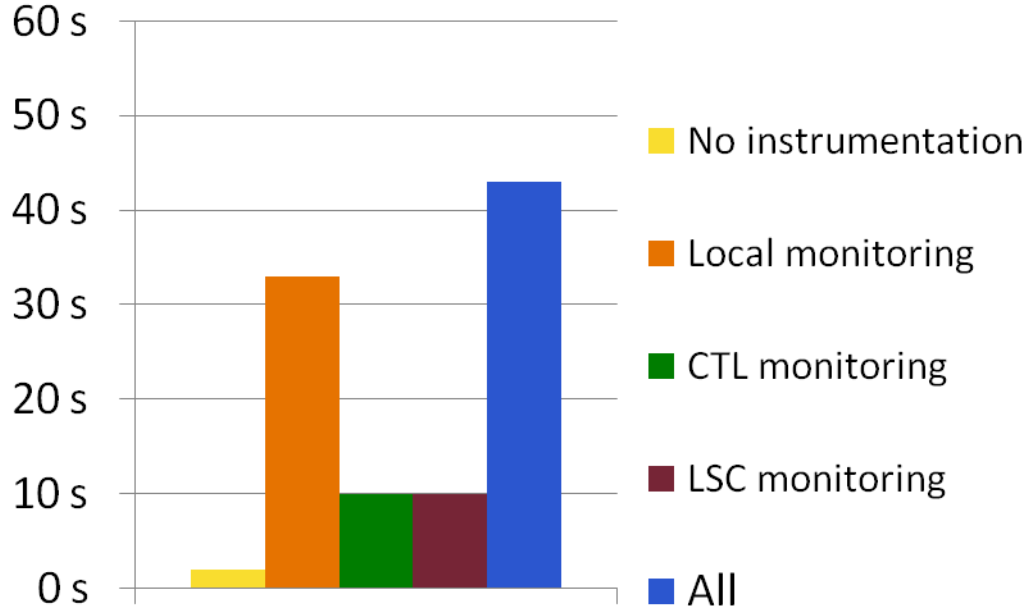
38

Mérési eredmények

- Az ellenőrzés időigénye az mbed platformon



With communication and control functions (50.000 state changes)

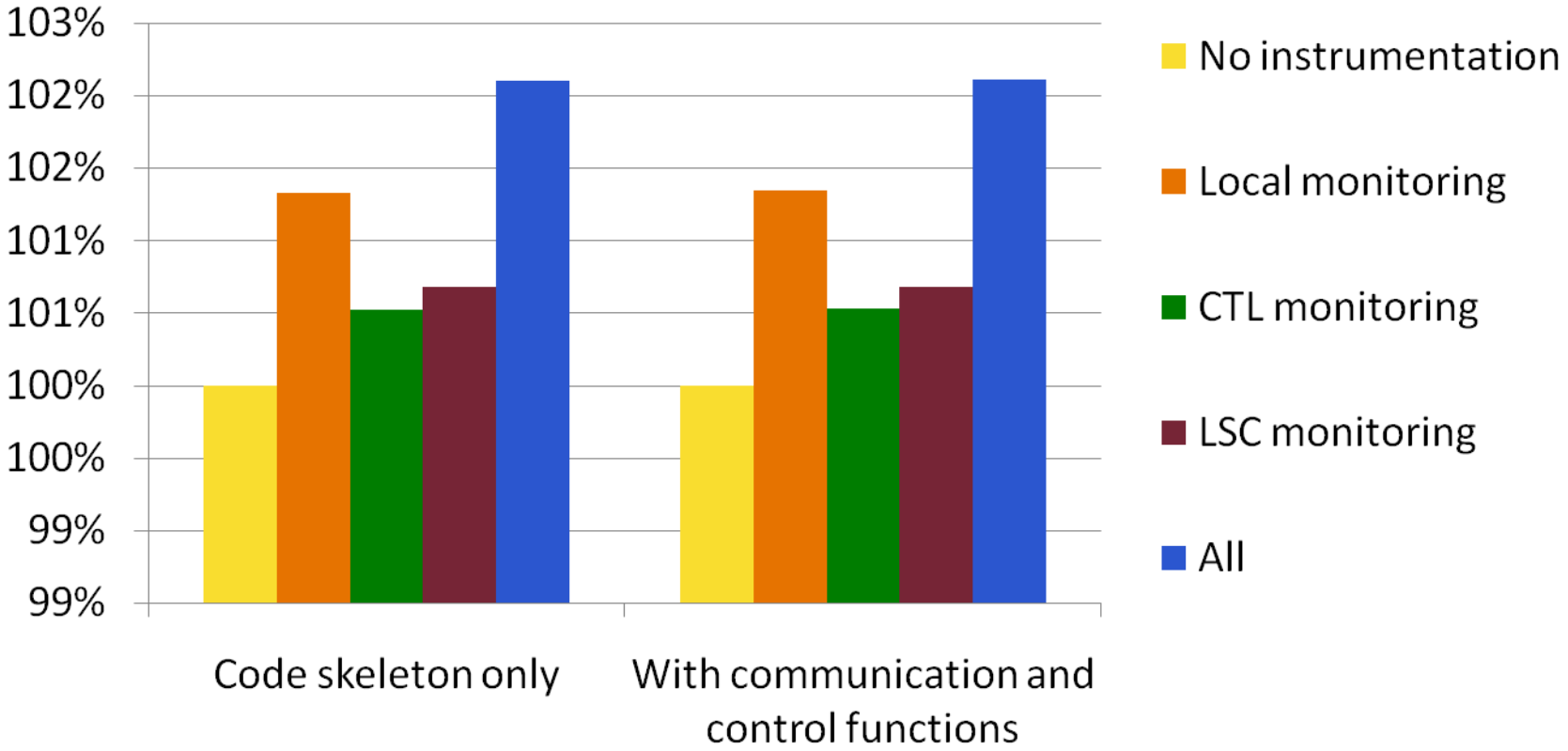


Code skeleton only (500.000 state changes)

- Kisebb, mint 12% többlet
- Nagyobb többlet „üres funkciók” esetén

Mérési eredmények

- Kódméret növekedés az mbed platformon



- Kisebb, mint 5% kódméret növekedés

40

Összefoglalás

- Alkalmazás forráskód szintézise
  - A formális szemantika szerepe
  - Platform szolgáltatások
- Monitor kód szintézise
  - Futásidőbeli verifikáció
  - Elfogadó automaták