Formális módszerek

A formális modellezés és a formális verifikáció alapjai

dr. Bartha Tamás
BME Közlekedés- és Járműirányítási Tanszék

dr. Majzik István
Dr. Pataricza András
BME Méréstechnika és Információs Rendszerek Tanszék

Formális módszerek

Matematikai technikák,

- elsősorban diszkrét matematika
- és matematikai logika

használata arra, hogy elkészítsük és ellenőrizzük hardver és szoftver rendszerek

- specifikációját,
- terveit (modelljeit),
- implementációját (viselkedését),
- dokumentációját.

Mire szeretnénk használni a formális módszereket?

Milyen problémákkal nézünk szembe?

Miben segíthetnek a formális módszerek?

Egy tanulságos történet…

- Vasa svéd hadihajó, 1628:
  - Elsüllyedt közvetlenül a vízrebocsátás után
- Problémák:
  - Változó követelmények (II. Gusztáv Adolf király)
  - Hiányzó pontos specifikáció (Henrik Hybertsson építő)
  - Ellenőrizetlen tervek (Johan Isbrandsson alvállalkozó)
  - Figyelmeztetések figyelmen kívül hagyása (Fleming admirális)
- Irodalom:
  - The Vasa: A Disaster Story with Software Analogies. By Linda Rising. The Software Practitioner, January-February 2001.
  - Why the Vasa Sank: 10 Problems and Some Antidotes for Software Projects. By Richard E. Fairley and Mary Jane Willshire. IEEE Software, March-April 2003.

Az elsüllyedés okai röviden

- A Vasa-t eredetileg kis hajónak specifikálták, de végül nagy hajó lett.
- Az építés késésekkel indult és kapkodva fejezték be. Végig a határidők szorításában dolgoztak.
- Eredetileg egy ágyúfedélzettel tervezték, de kettővel készült el. Folyamatosan növelték a fegyverzet mennyiségét a stabilitásra való tekintet nélkül.
- A kivitelező betegsége és halála komoly gondokat okozott a projekt menedzsmentben.
- A hajótest nehezebb volt a szokásosnál. A hajótest alakja nem felelt meg a stabilitási követelményeknek a folytonos változtatások miatt.
- Nem alkalmaztak számításokat a stabilitás tervezése érdekében. A ballaszt elhelyezésére nem volt elég hely.
- A stabilitás tesztje komoly problémákat mutatott ki. A tesztet félbeszakították, az eredményről az építőket nem informálták.
- Az ágyúnyílásokat nem zárták le a próbaút során.

Komplex rendszerek tervezése (ma)

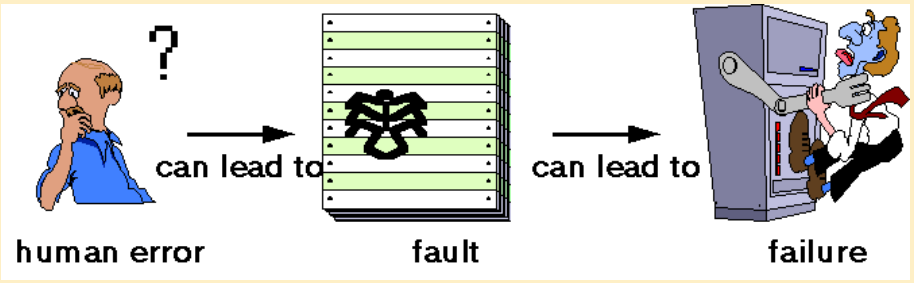
- Minimális tervezés
- Implicit folyamat
- Egyszerű eszközök

- Tervezés
- Definiált folyamat
- Hatékony eszközök

- Ellenőrzött tervek
- Meghatározott folyamat
- Automatikus eszközök

Szoftver minőségi krízis

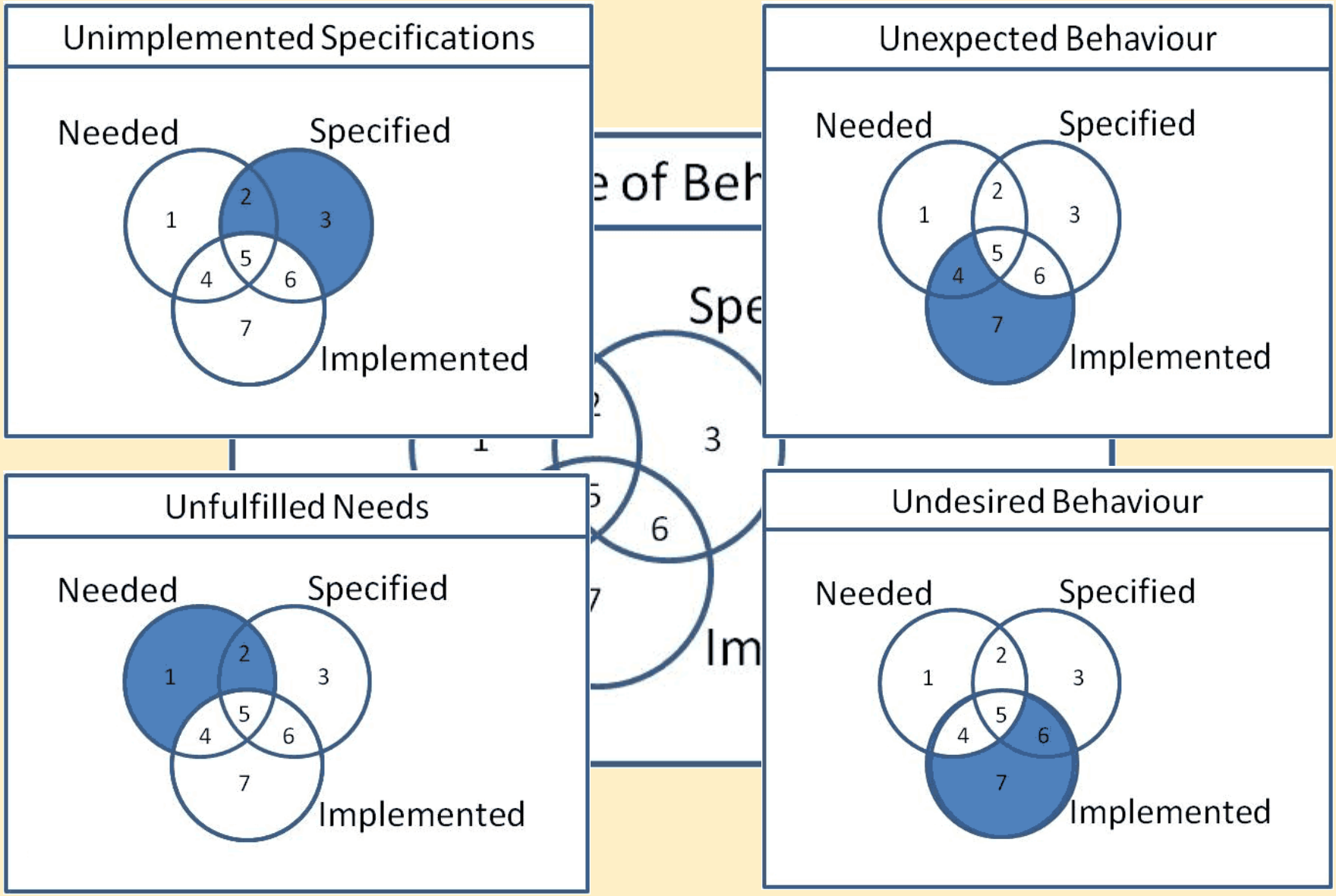
- Tipikus kódméret:
  - 10 kLOC … 1000 kLOC
- Fejlesztési ráfordítás:
  - 0,1 - 0,5 mérnökév / kLOC (nagyméretű szoftver)
  - 5-10 mérnökév / kLOC (kritikus szoftver)
- Hiba eltávolítás (ellenőrzés, tesztelés, javítás):
  - 45 - 75% ráfordítás
- Hibasűrűség változása:
  - 10 - 200 hiba / kLOC jön létre a fejlesztés során



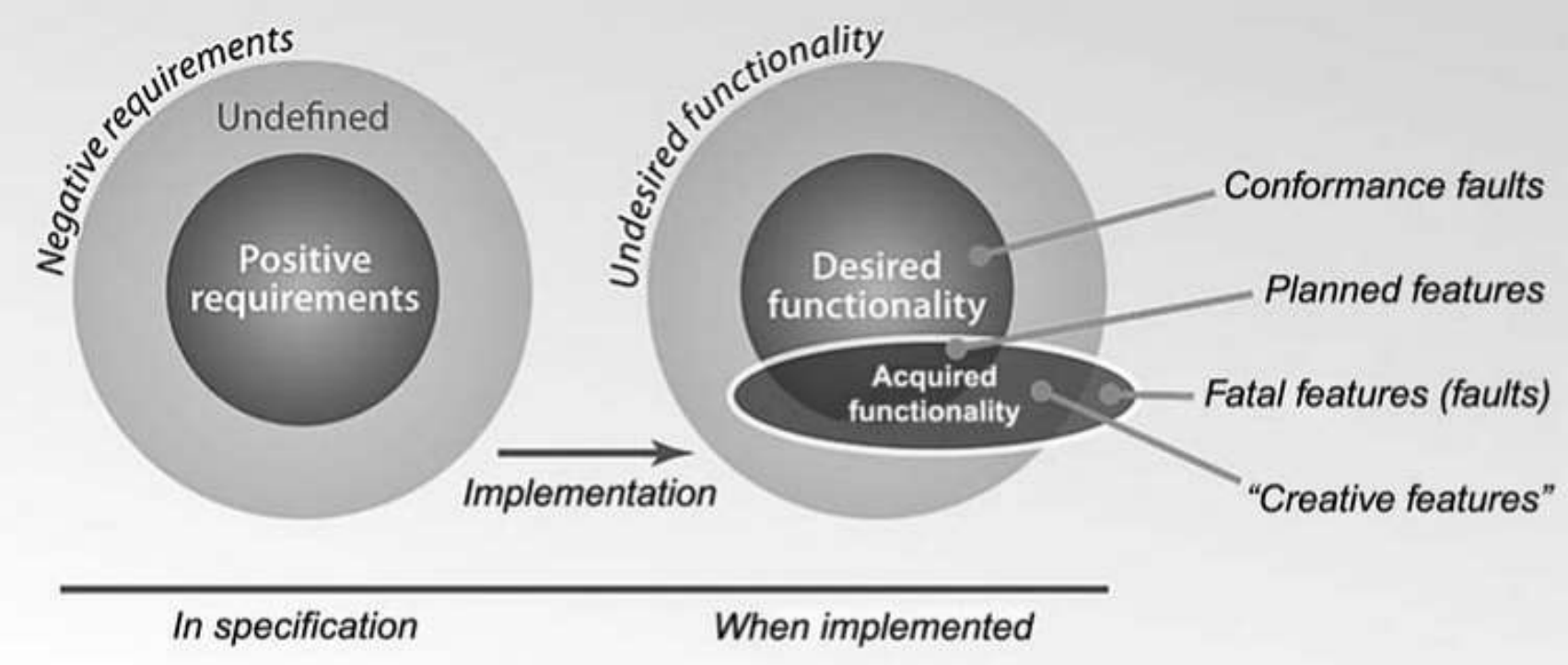
Ellenőrzés, debuggolás, javítás

  - 0,01 - 10 hiba / kLOC maradhat az üzembe helyezésig

A fejlesztés lehetséges hiányosságai

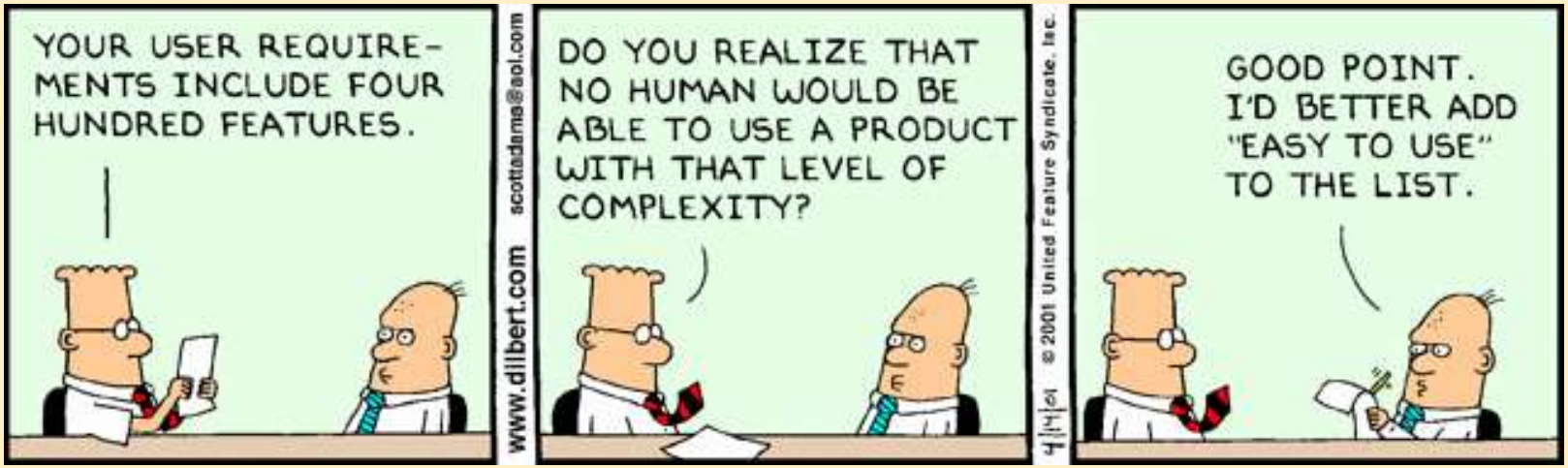


A fejlesztés lehetséges hiányosságai



Alkalmazások fejlesztésének kihívásai

- Jó minőségű specifikáció és tervek készítése
  - Teljes
  - Ellentmondás-mentes, egyértelmű
  - Ellenőrizhető
  - Redundanciát nélkülöző



Alkalmazások fejlesztésének kihívásai

- Tervek ellenőrzése
  - Tervezői döntések igazolása
    - Bizonyítottan helyes tervek a továbblépés alapjai
  - Hibák elkerülése vagy korai felderítése
    - Minőség ↔ költség ↔ fejlesztési idő optimalizálás
- Bizonyított helyességű eszközök használata
  - Forráskód, konfiguráció, teszt és monitor szintézis

Ezek alapjait a formális módszerek adhatják!

Fejlesztési megközelítés: MBE

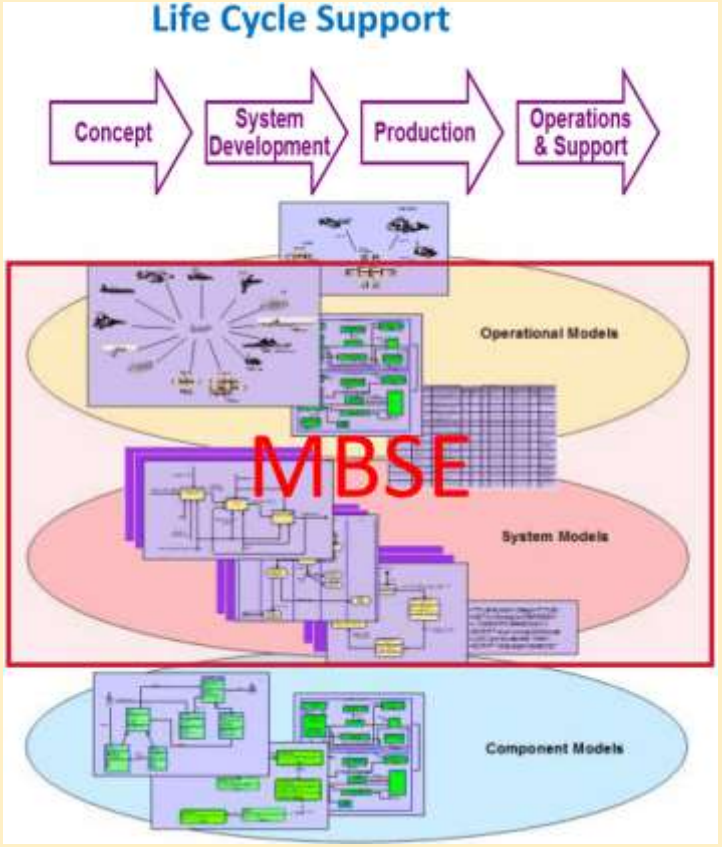
- Modell:
  - Egy koncepció, jelenség, kapcsolat, szerkezet vagy rendszer egyszerűsített leírása
  - Grafikus, matematikai vagy fizikai reprezentáció
  - A valóság absztrakciója a felesleges összetevők kiküszöbölésével
- A modell céljai közé tartozik
  - a megértés megkönnyítése,
  - segítségnyújtás a döntéshozatalban, a "mi van, ha" forgatókönyvek vizsgálatában,
  - az események magyarázata, ellenőrzése és előrejelzése.

„A modellalapú fejlesztés (Model Based Engineering, MBE): olyan mérnöki megközelítés, amely modelleket használ a tervezés technikai alapja szerves részeként, magában foglalva egy képesség, rendszer és/vagy termék követelményeit, elemzését, tervezését, végrehajtását és ellenőrzését a megvalósítás teljes életciklusa alatt.”
[Zárójelentés, Modellalapú Fejlesztés Albizottság, NDIA, 2011. február]

Modellalapú rendszerfejlesztés

„A modellalapú rendszerfejlesztés (MBSE) a modellezés formalizált alkalmazása a rendszerkövetelmények, a tervezés, az elemzés, az ellenőrzés és a validálási tevékenységek támogatása érdekében, a koncepcionális tervezési fázistól kezdve, a fejlesztés és a későbbi életciklusok során folytatva.” [INCOSE SE Vision 2020, 2007. nov.]

- Formalizálja a rendszerfejlesztés gyakorlatát a modellek használatán keresztül
- Széles tartományt lefed
- Több modellezési területet tartalmaz az életciklus során az komplex rendszerektől a komponensekig
- Minőség- és termelékenynövekedést eredményez a kockázat csökkentése mellett
  - Rigorózus és precíz
  - Fejlesztőcsapat és ügyfél közötti kommunikáció félreértések nélkül
  - A komplexitás kezelése

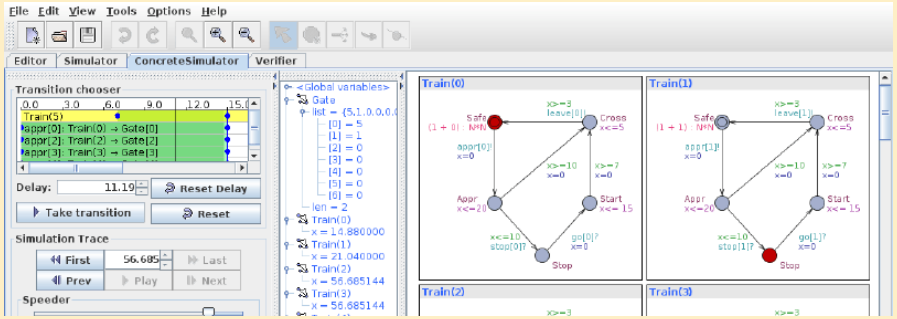


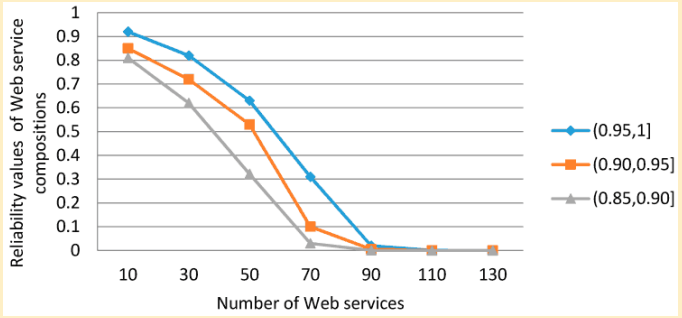
Formális módszerek

- Formális módszer
  - specifikáció (tulajdonságok/követelmények leírása) + modell (viselkedés leírása/végrehajtás) + analízis metódus/eszköz
  - *„A formális specifikáció tulajdonságok olyan csoportjának kifejezése egy adott formális nyelven és adott absztrakciós szinten, amiket valamilyen rendszernek el kell látnia*" [Lamsweerde, 2000]
  - A cél *„egy bizonyos (számítástechnikai) rendszer viselkedésének garantálása precíz/rigorózus megközelítés alkalmazásával*" [Almeida et al., 2011]
- Kulcsszavak
  - Egyértelmű, ellentmondásmentes leírások (szintakszis/szemantika)
  - Matematikailag bizonyítható helyesség, teljesség és ellentmondás-mentesség

Formális módszerek alkalmazási területei

- Diszkrét (időzítetlen/időzített) modellek, kvalitatív leírások
  - Verifikáció (helyességigazolás)
    - A rendszerfejlesztés különböző fázisai megfelelnek-e a specifikációnak?
    - „A rendszert megfelelő módon építjük-e?”
  - Formális verifikáció: formális módszerrel végrehajtott ellenőrzés
- Sztochasztikus modellek
  - Biztonsági elemzés, tranziensek és állandósult állapotbeli viselkedés
  - Megbízhatósági analízis, teljesítőképességi elemzés
    - dinamikus rendszerállapotokra, az FTA módszeren túl
  - Prediktív vizsgálatok, előrejelzések

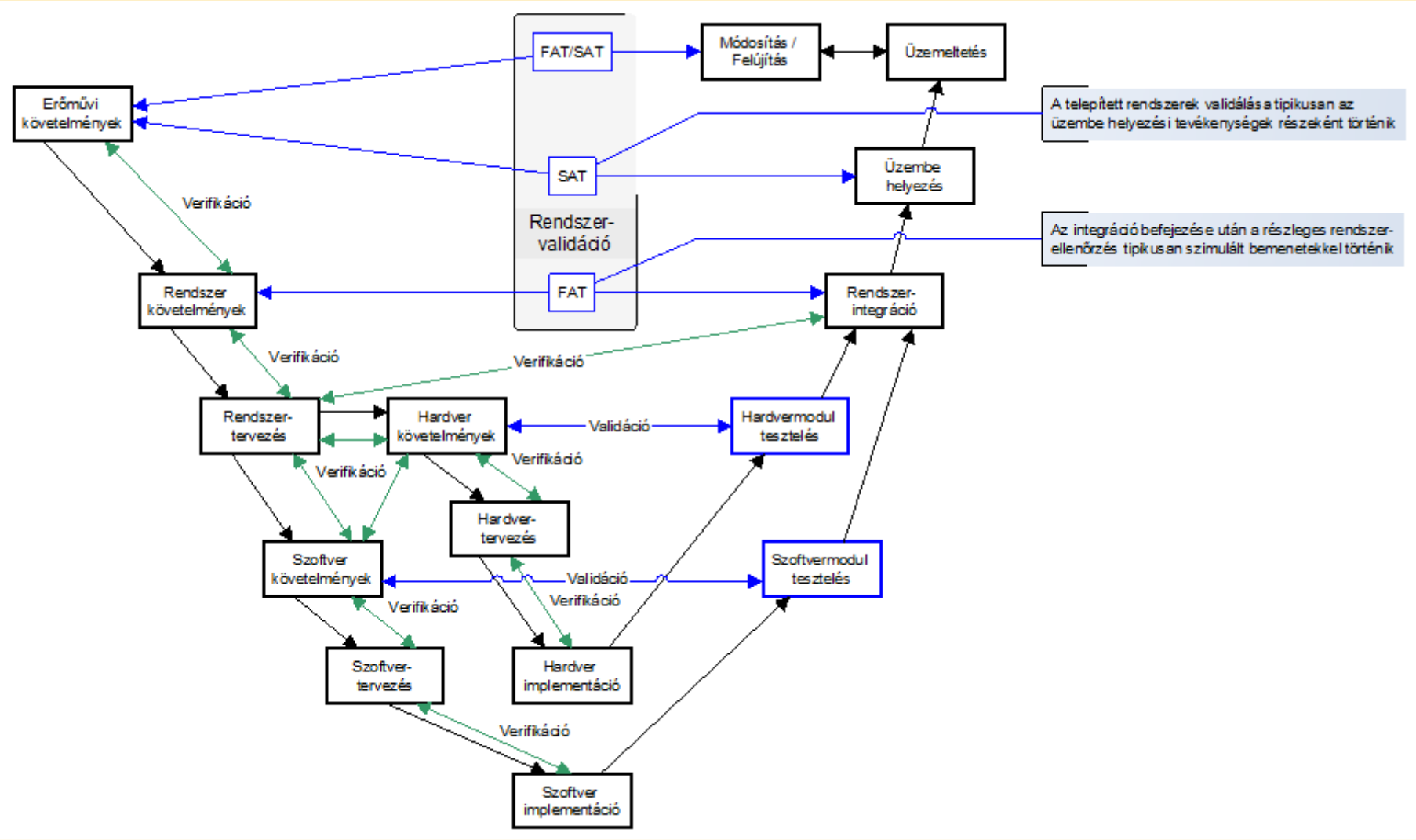




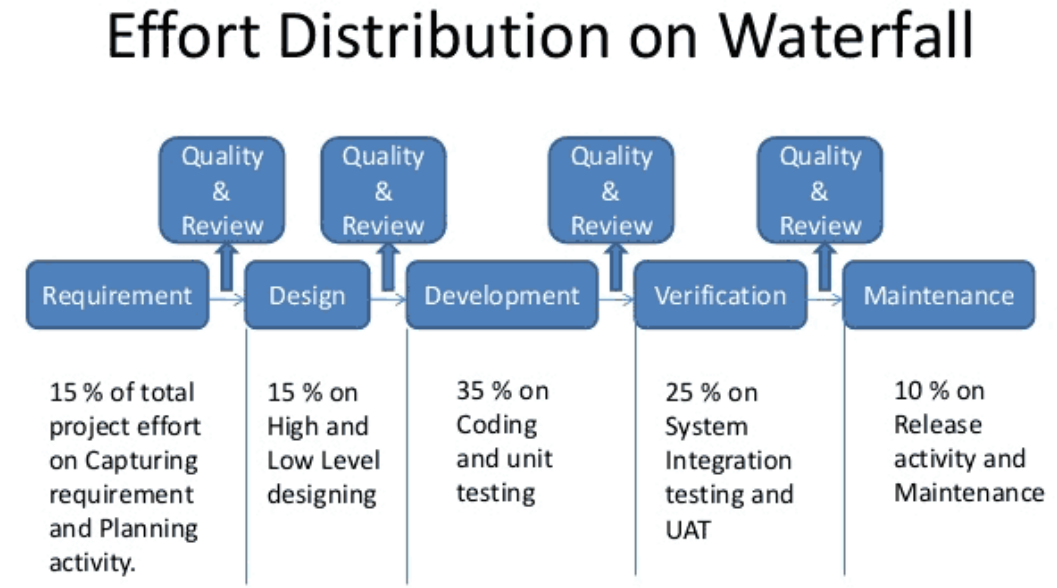
Verifikáció és validáció összehasonlítása

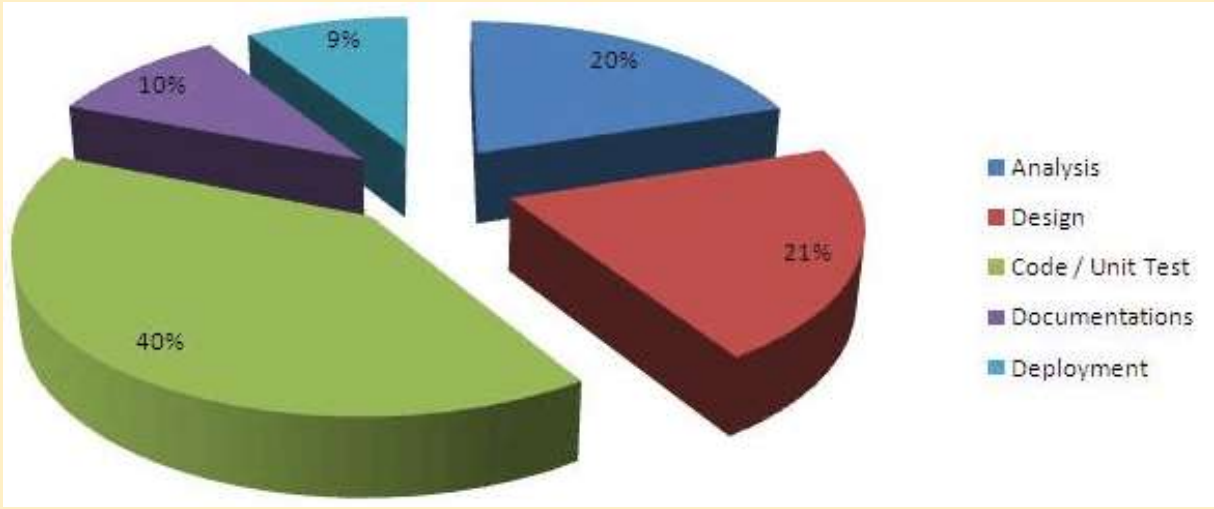
Verifikáció (igazolás)	Validáció (érvényesítés)
„Jól építjük-e a rendszert?”	„Jó rendszert építettünk-e?”
Összhang ellenőrzése a fejlesztési fázisokban, illetve ezek között	A fejlesztés eredményének ellenőrzése
Fejlesztési lépések során használt tervek (modellek) és specifikációjuk közötti megfelelés ellenőrzése	A kész rendszer és a felhasználói elvárások közötti megfelelés ellenőrzése
Objektív folyamat; formalizálható, automatizálható	Szubjektív elvárások lehetnek; elfogadhatósági ellenőrzés
Felderíthető hibák: Tervezési, implementációs hibák	Felderíthető hibák: Követelmények hiányosságai is
Nincs rá szükség, ha automatikus a leképzés követelmény és implementáció között	Nincs rá szükség, ha a specifikáció tökéletes (elég egyszerű)

Verifikáció és validáció a biztonsági életciklusban



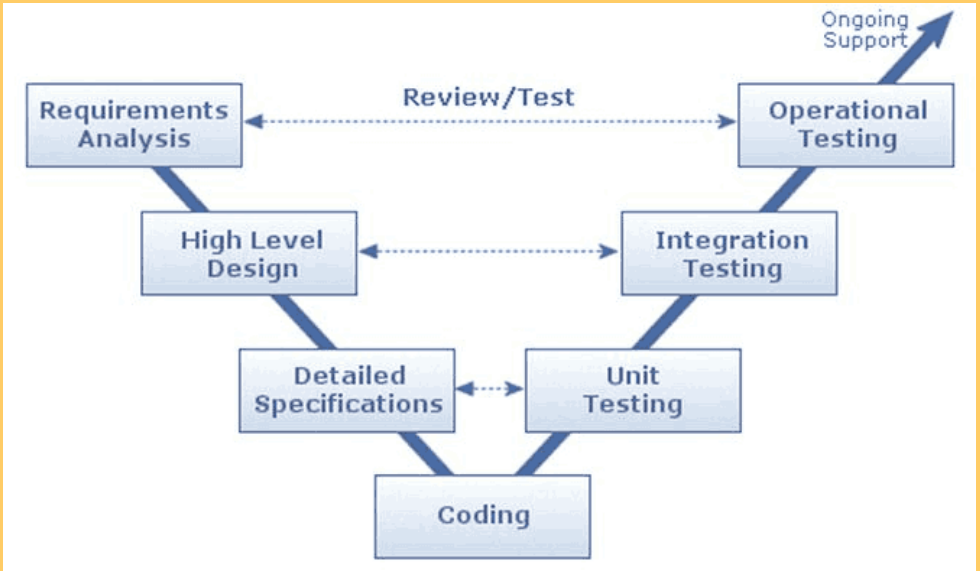
Ráfordítási igény projekt életciklus fázisokban

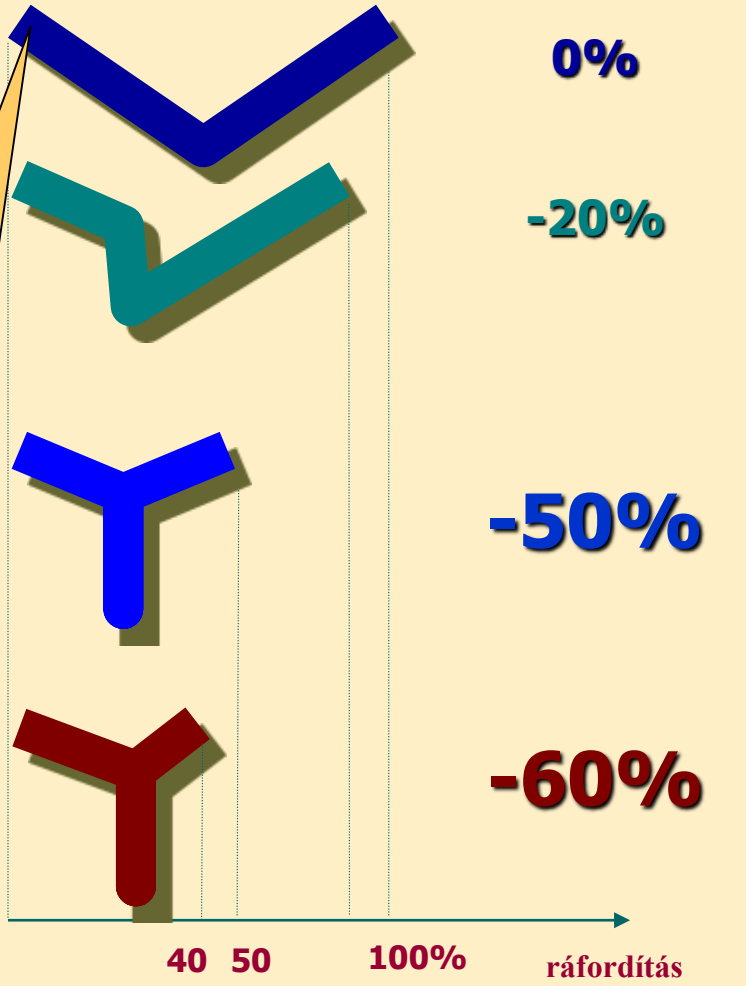




V-től az Y fejlesztési modellig

Szoftverfejlesztés a V-modell szerint





* Adatok: Esterel Technologies

V-től az Y fejlesztési modellig

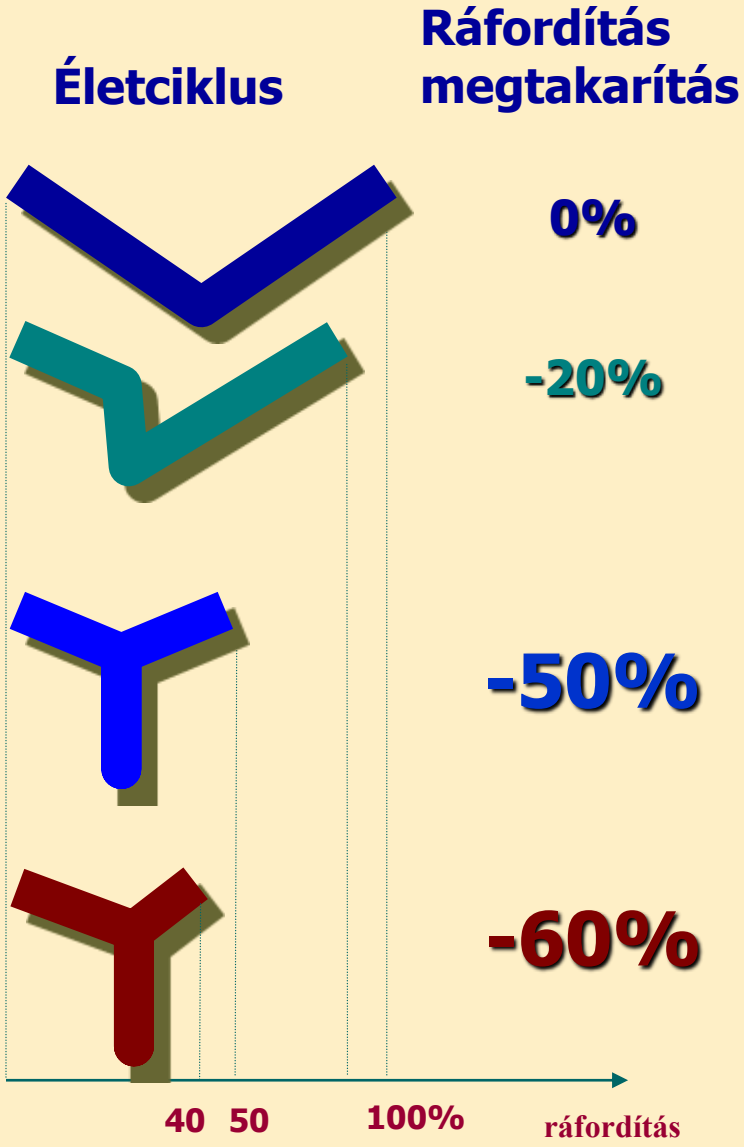
Kézi kódolás

"Közönséges" automatikus kódgenerátor használata

Minősített automatikus kódgenerátor használata

Formális verifikációval kiegészített tervezés

* Adatok: Esterel Technologies



Biztonságkritikus szoftverek fejlesztése

- IEC 61508: Szabvány előírások a fejlesztésre
  - Functional safety in electrical / electronic / programmable electronic safety-related systems
  - Szakterület-specifikus szabványok alapja
- Követelmény-specifikáció készítésének előírásai:

Table A.1 – Software safety requirements specification (see 7.2)

	Technique/Measure*	Ref.	SIL1	SIL2	SIL3	SIL4
1	Computer-aided specification tools	B.2.4	R	R	HR	HR
2a	Semi-formal methods	Table B.7	R	R	HR	HR
2b	Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	---	R	R	HR

NOTE 1 The software safety requirements specification will always require a description of the problem in natural language and any necessary mathematical notation that reflects the application.

NOTE 2 – The table reflects additional requirements for specifying the software safety requirements clearly and precisely.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

Biztonságkritikus szoftverek fejlesztése

- IEC 61508: Szoftver tervezés és fejlesztés előírásai

Table A.2 – Software design and development: software architecture design (see 7.4.3)

	Technique/Measure*	Ref	SIL1	SIL2	SIL3	SIL4
1	Fault detection and diagnosis	C.3.1	---	R	HR	HR
2	Error detecting and correcting codes	C.3.2	R	R	R	HR
3a	Failure assertion programming	C.3.3	R	R	R	HR
3b	Safety bag techniques	C.3.4	---	R	R	R
3c	Diverse programming	C.3.5	R	R	R	HR
3d	Recovery block	C.3.6	R	R	R	R
3e	Backward recovery	C.3.7	R	R	R	R
3f	Forward recovery	C.3.8	R	R	R	R
3g	Re-try fault recovery mechanisms	C.3.9	R	R	R	HR
3h	Memorising executed cases	C.3.10	---	R	R	HR
4	Graceful degradation	C.3.11	R	R	HR	HR
5	Artificial intelligence - fault correction	C.3.12	---	NR	NR	NR
6	Dynamic reconfiguration	C.3.13	---	NR	NR	NR
7a	Structured methods including for example, JSD, MASCOT, SADT and Yourdon.	C.2.1	HR	HR	HR	HR
7b	Semi-formal methods	Table B.7	R	R	HR	HR
7c	Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	---	R	R	HR
8	Computer-aided specification tools	B.2.4	R	R	HR	HR

NOTE – The measures in this table concerning fault tolerance (control of failures) should be considered with the requirements for architecture and control of failures for the hardware of the programmable electronics in IEC 61508-2.

* Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

Mi a helyzet a közlekedés/járműmérnöki területen?

<table>
<tr><th>Szakterület</th><th>Rendszer tanúsítás</th><th>Fejlesztési folyamat</th><th>Megbízhatóság-elemzés</th></tr>
<tr><td rowspan="2">Általános</td><td></td><td>IEEE-12207</td><td>IEC-61508</td></tr>
<tr><td colspan="3">CMMI</td></tr>
<tr><td>Autóipari</td><td></td><td colspan="2">ISO-26262</td></tr>
<tr><td>Repülési</td><td colspan="3">DO-178C, DO-278A, DO-333</td></tr>
<tr><td rowspan="2">Vasúti</td><td></td><td></td><td>IEC-50126</td></tr>
<tr><td></td><td>IEC-50128,
IEC-50129</td><td></td></tr>
<tr><td>Űr</td><td colspan="3">ECSS</td></tr>
</table>

Mik azok a formális módszerek?

A formális módszerek célja és eszközei

Formális módszerek

Formális módszer:

- A formális modellről ismeretet adó matematikai eljárás
- Eszközökkel támogatható
- A formális modell végrehajtása: Szimuláció
- A formális modell ellenőrzése: Formális verifikáció
  - „Önmagában való” vizsgálat
    - Konzisztencia, ellentmondás-mentesség
    - Teljesség, zártság
  - „Megfelelés” vizsgálata
    - Modellek között
    - Modellek és elvárt tulajdonságok között (implementáció $\leftrightarrow$ specifikáció)
- A formális modell alapján történő szintézis:
  - Szoftver (programkód, konfiguráció) generálása
  - Hardver tervek generálása

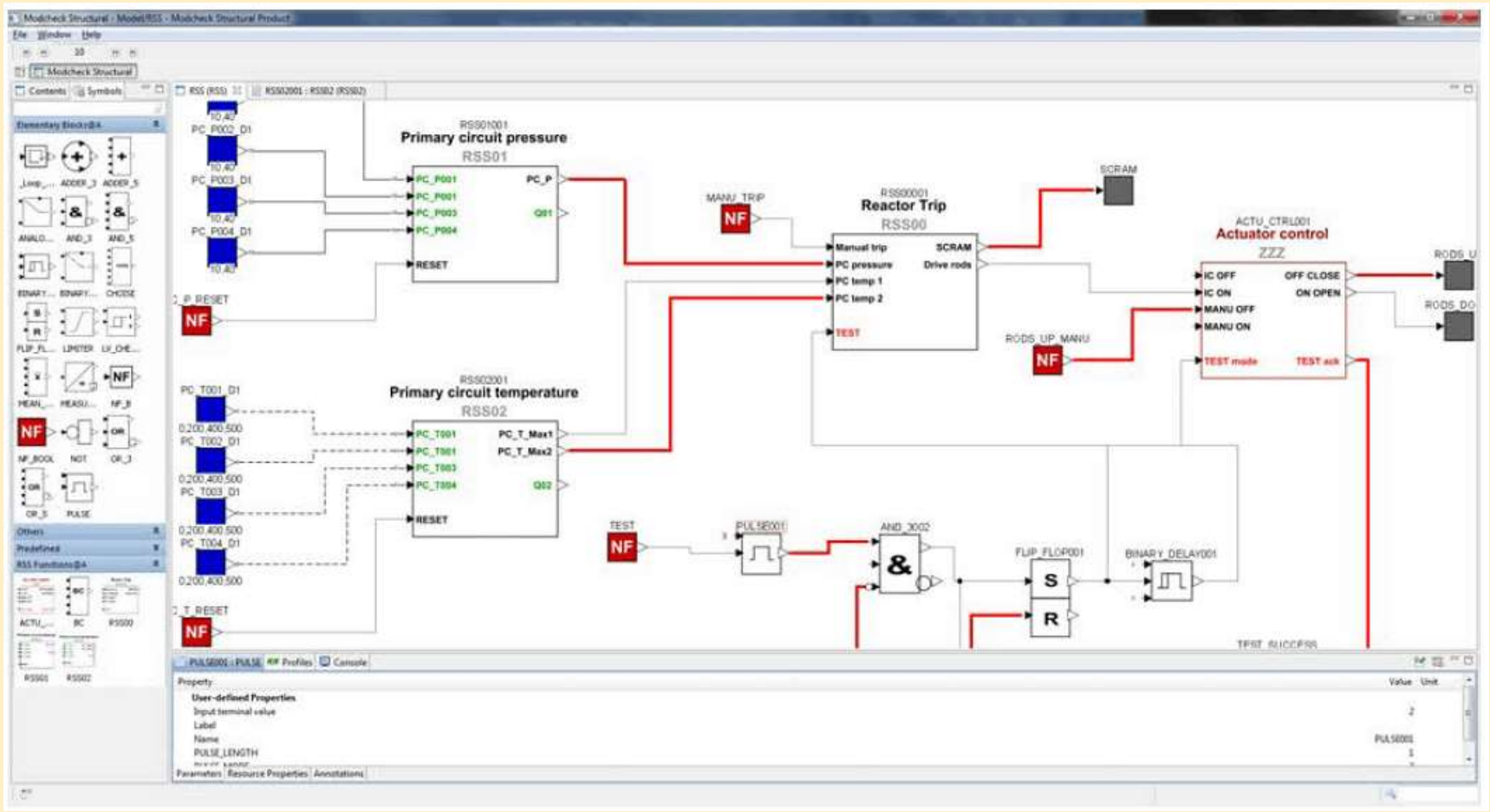
Hogyan ellenőrizhetjük a követelményeket?

- A megvalósítás tesztelésével
  - Létre tudunk-e hozni minden lehetséges végrehajtást lefedő teszt eseteket?
  - A problémás esetek figyeléséhez külön ellenőrző kell
  - A hiba már csak az implementáció után derülhet ki, drágán javítható
- Modellezéssel és szimulációval
  - Tudunk-e szimulálni minden lehetséges végrehajtást?
  - A problémás esetek detektálása nagy odafigyelést igényel
  - A hibák viszont olcsóbban javíthatók modell szinten
- Modellezéssel és az állapottér teljes ellenőrzésével
  - Szisztematikus algoritmus kell az állapottér teljes felvételéhez
  - Legyen automatikus a követelmények teljesülésének figyelése is
    - Ehhez egy olyan nyelv szükséges, amikkel a követelmények leírhatók
    - Általános módszer adható a követelmények modell alapú ellenőrzésére

A formális módszerek használata

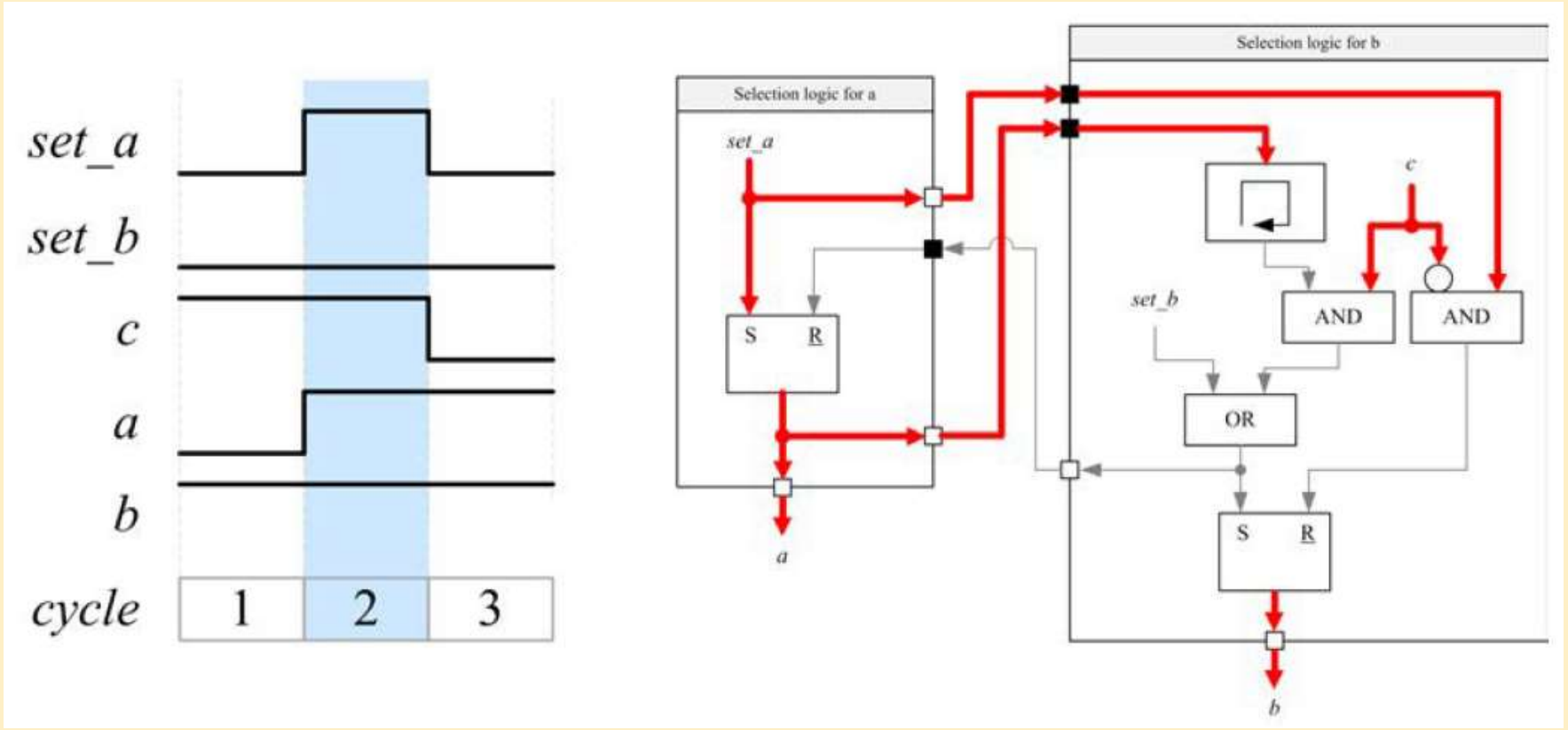
- Tervezők (nem csak matematikusok) által is használható, amennyiben a specifikus tudást eszközökbe integrálják
  - Modellező eszközök
  - Ellenőrző eszközök
    - Modellellenőrzők, ekvivalencia ellenőrzők, automatikus tételbizonyítók
  - Szintézis eszközök
    - Kódgenerátor, konfiguráció generátor az ellenőrzött modellek alapján
    - Teszt generátor (validáláshoz)
- Csökken a rendszerben maradó koncepcionális és tervezői hibák száma, javul a minőség
  - Szolgáltatásbiztonság növelhető
  - De garanciát nem ad a használhatóságra, a felhasználói elvárások teljesítésére!
  - Validációt nem helyettesíti

Gyakorlati példa: a MODCHK rendszer



A finn VTT kutatóközpont által kifejlesztett MODCHK eszköz lehetővé teszi a funkcióblokk diagramok grafikus modellezését és a (háttérben elvégzett modellellenőrzés által talált) ellenpéldák visszavetítését/megjelenítését.

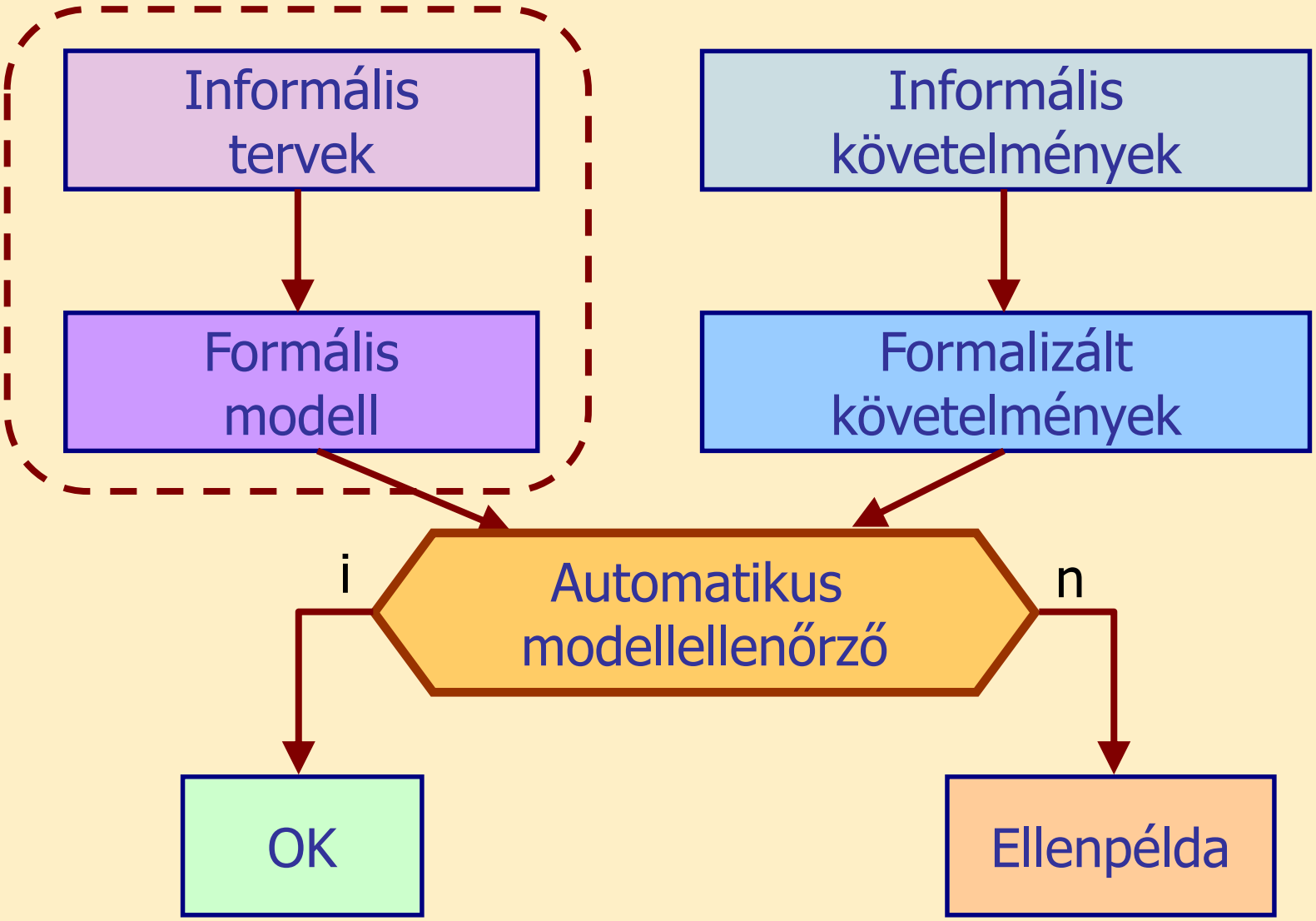
A MODCHK rendszer által talált ellenpélda



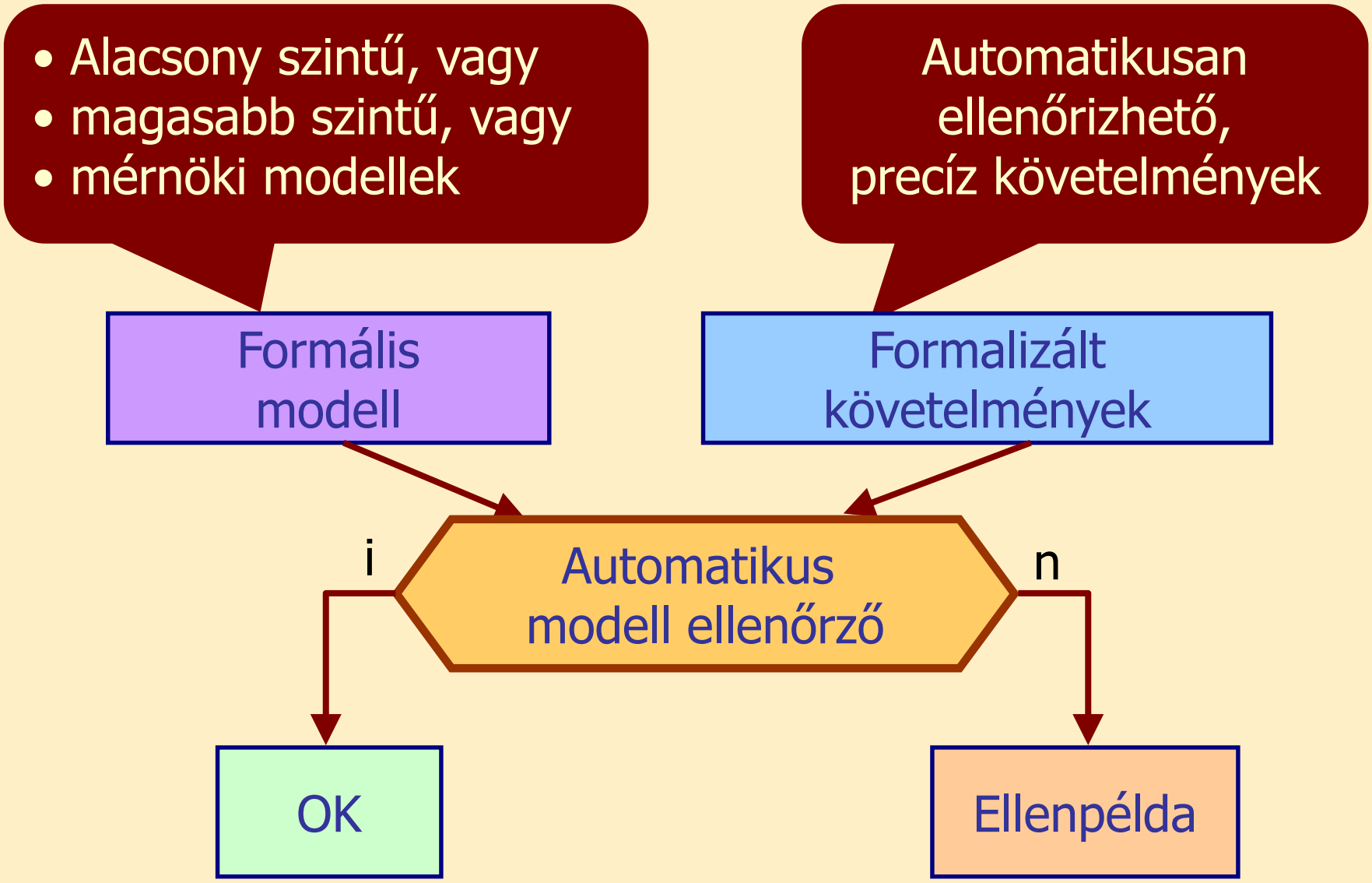
A formális módszerek használatának lépései

- Valós probléma formális vizsgálata:
  1. Fogalmi tér felépítése ← feltételezések
  2. Probléma formalizálása ← absztrakció
  3. Formális modell analízise ← automatikus lehet
  4. Eredmények értelmezése, felhasználása
- Alkalmazáshoz szükséges:
  - Feltételezések teljesülésének ellenőrzése
  - Modell validálása
  - Eszközök helyességének belátása

Mit szeretnénk elérni?



Mit szeretnénk elérni?



Modellellenőrzés

Előnyök

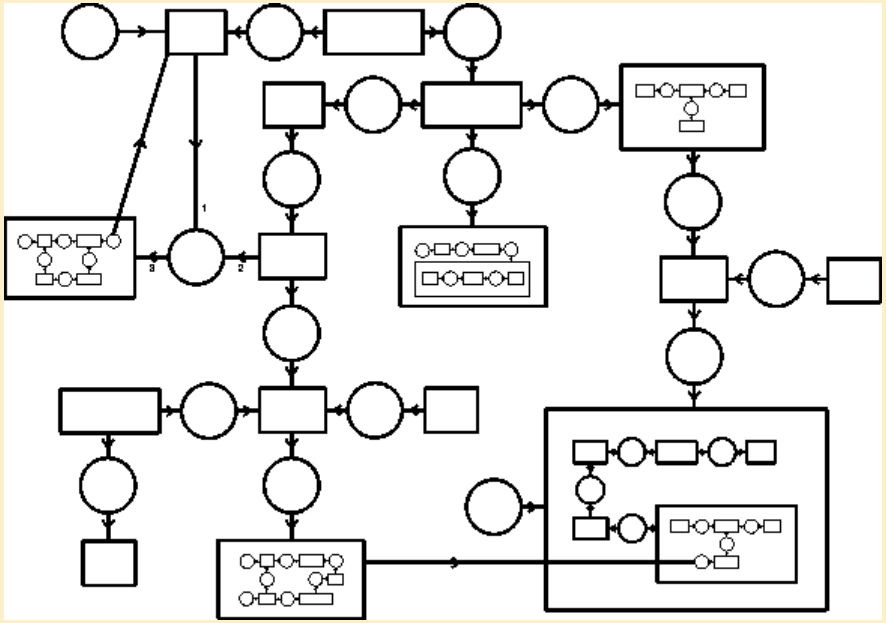
- Az összes lehetséges viselkedést felderíti
  - lehetővé teszi olyan tulajdonságok ellenőrzését is, amelyeket szinte lehetetlen tesztelni
- Ellenpéldák
- Lehetőség van az automatizálásra
  - így a formális módszerekhez nem értők is használhatják

Hátrányok

- A modellt ellenőrzi, nem a valós rendszert
  - hogyan lehetünk biztosak abban, hogy a valós rendszerre érvényesek az eredmények?
- Állapottér robbanás
- Formális specifikációra és formális modellre van szükség
  - ezek más ábrázolásokból is előállíthatók
  - de ezeknek is formálisnak kell lenniük

Alapmodellek

Alapszintű modellezési formalizmusok



Első lépés: modellezési formalizmus

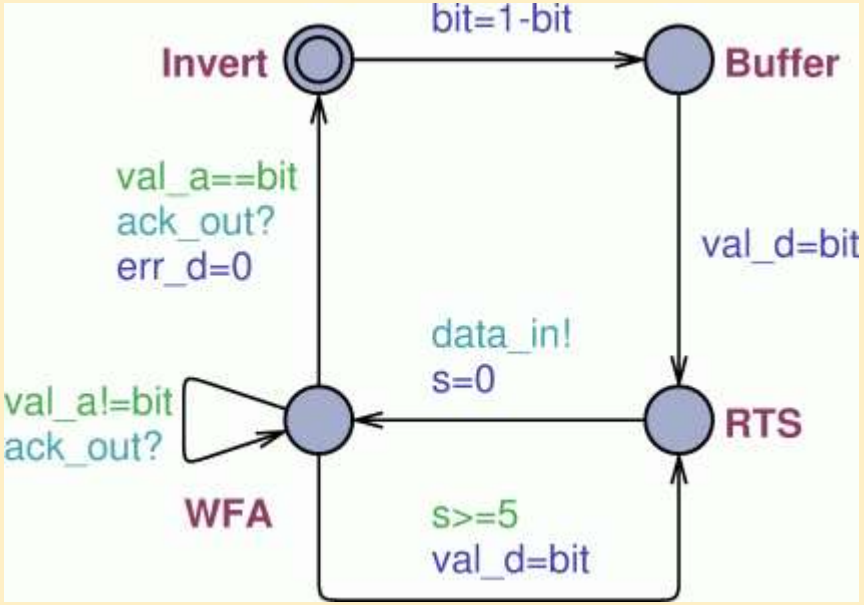
- A formalizálás célja: matematikai precizitással megadni
  - Modelleket: terveket, tervezői döntéseket (modellezési nyelv)
  - Követelményeket: elvárt tulajdonságokat (követelmény leíró nyelv)
- Formális nyelvek felépítése
  - Formális szintaxis
  - Formális szemantika

  } Jelölésmód: nyelvi elemek és kapcsolatok

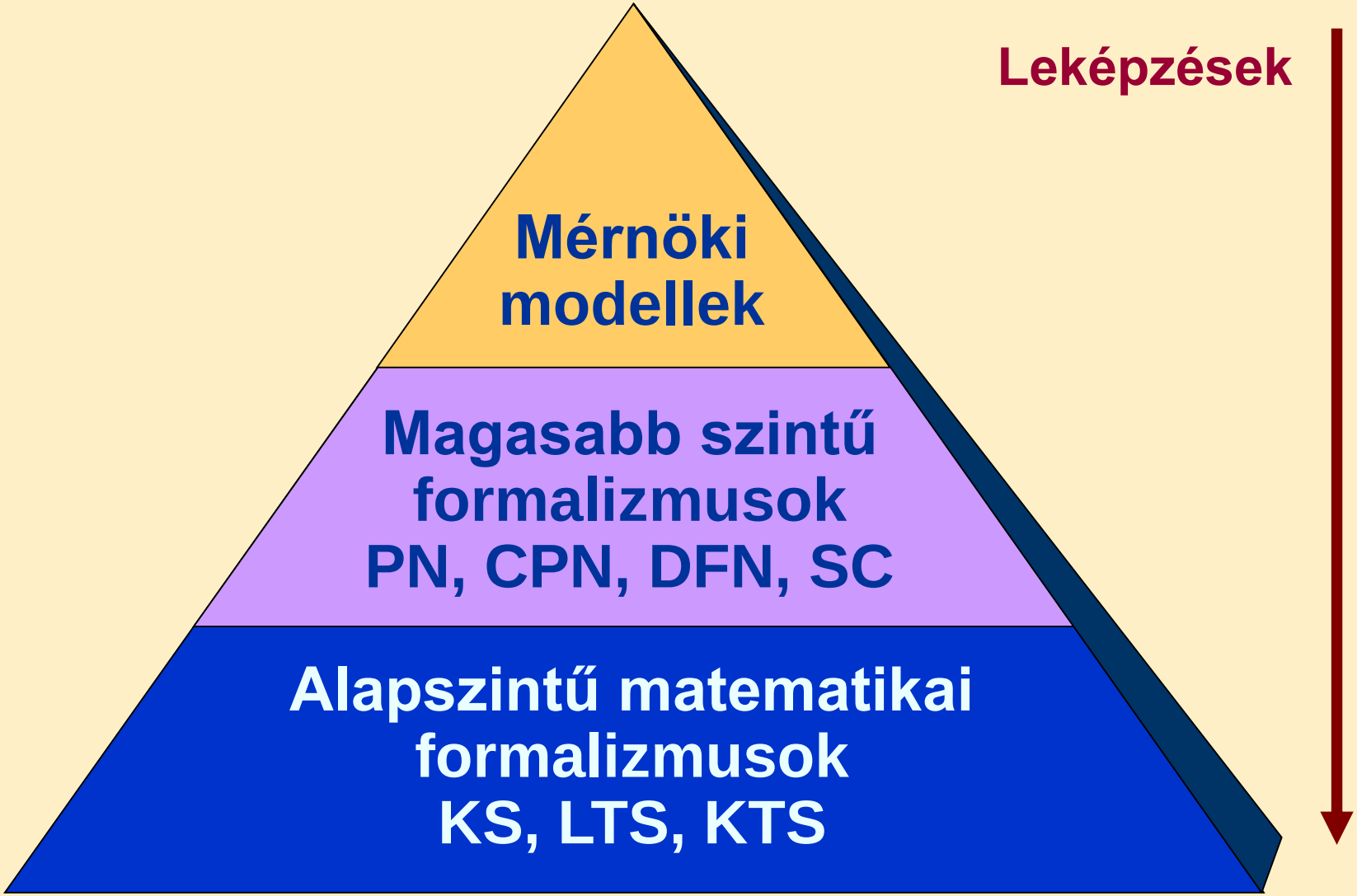
  } A jelölésmód interpretációja
- Mit szeretnénk leírni formális nyelvekkel?
  - Funkcionalitás (viselkedés, feltételek, elvárások, ...)
  - Struktúra, interfészek
  - Extra-funkcionális aspektusok is: teljesítmény, megbízhatóság, ...
- A formális nyelv használatának előnyei
  - Egyértelműség, ellenőrizhetőség
  - Automatikus feldolgozhatóság

Egy egyszerű példa

- Automata formalizmus:
  - Állapotok és állapotátmenetek
  - Változók, konstansok
  - Feltételek az átmenetek végrehajtásához
  - Akciók az átmenetek végrehajtása során



Modellek a formális ellenőrzéshez



Alapszintű formalizmusok (áttekintés)

- Kripke-struktúrák (KS)
  - Állapotok, állapotátmenetek
  - Állapotok lokális tulajdonságai, mint címkék
- Címkézett tranzíciós rendszerek (LTS)
  - Állapotok, állapotátmenetek
  - Állapotátmenetek lokális tulajdonságai mint címkék
- Kripke tranzíciós rendszerek (KTS)
  - Állapotok, állapotátmenetek
  - Állapotok és állapotátmenetek lokális tulajdonságai, mint címkék
- Véges állapotú automaták időkezeléssel
  - Kiterjesztések: Változók, óraváltozók, szinkronizáció

1. Kripke-struktúra

KS, Kripke-structure:

- Állapotok tulajdonságait fejezzük ki: címkézés atomi kijelentésekkel
- Egy állapothoz sok címke rendelhető

Alkalmazás: Viselkedés, algoritmus leírása

$KS = (S, R, L)$ és AP, ahol

$\mathrm{AP} = \{\mathrm{P},\mathrm{Q},\mathrm{R},...\}$ atomi kijelentések halmaza (domén-specifikus)

$\mathrm{S} = \{\mathrm{s}_1, \mathrm{s}_2, \mathrm{s}_3, ...\mathrm{s}_n\}$ állapotok halmaza

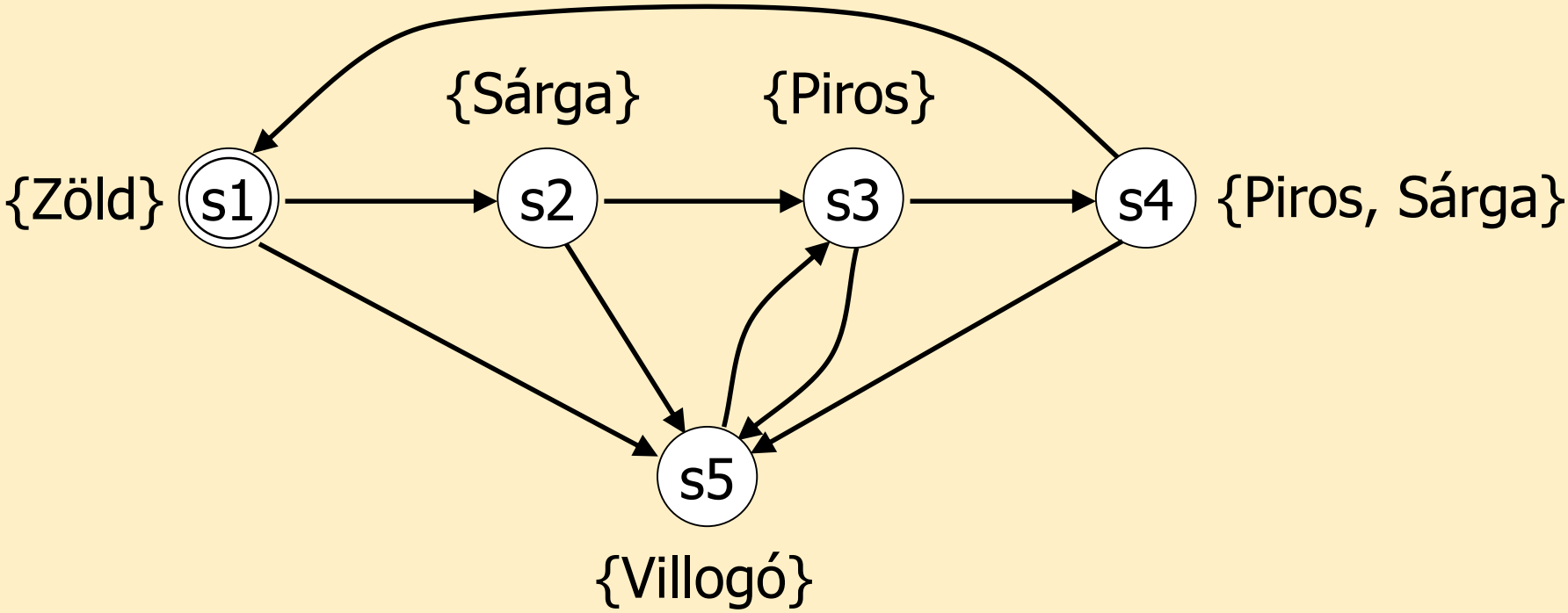
$\mathrm{R} \subseteq \mathrm{S} \times \mathrm{S}$: állapotátmeneti reláció

$\mathrm{L}: \mathrm{S} \to 2^{AP}$ állapotok címkézése atomi kijelentésekkel

Kripke-struktúra példa

Közlekedési lámpa viselkedése

- AP={Zöld, Sárga, Piros, Villogó}
- S = {s1, s2, s3, s4, s5}



2. Címkézett tranzíciós rendszer

LTS, Labeled Transition System:

- Állapotátmenetek tulajdonságait fejezzük ki: címkézés akciókkal
- Egy átmeneten csak egy akció szerepelhet

Alkalmazás: Kommunikáció, protokollok modellezése

$LTS = (S, Act, \rightarrow)$, ahol

$S = \{s_1, s_2, \ldots s_n\}$ állapotok halmaza

$Act = \{a, b, c, \ldots\}$ akciók (címkék) halmaza

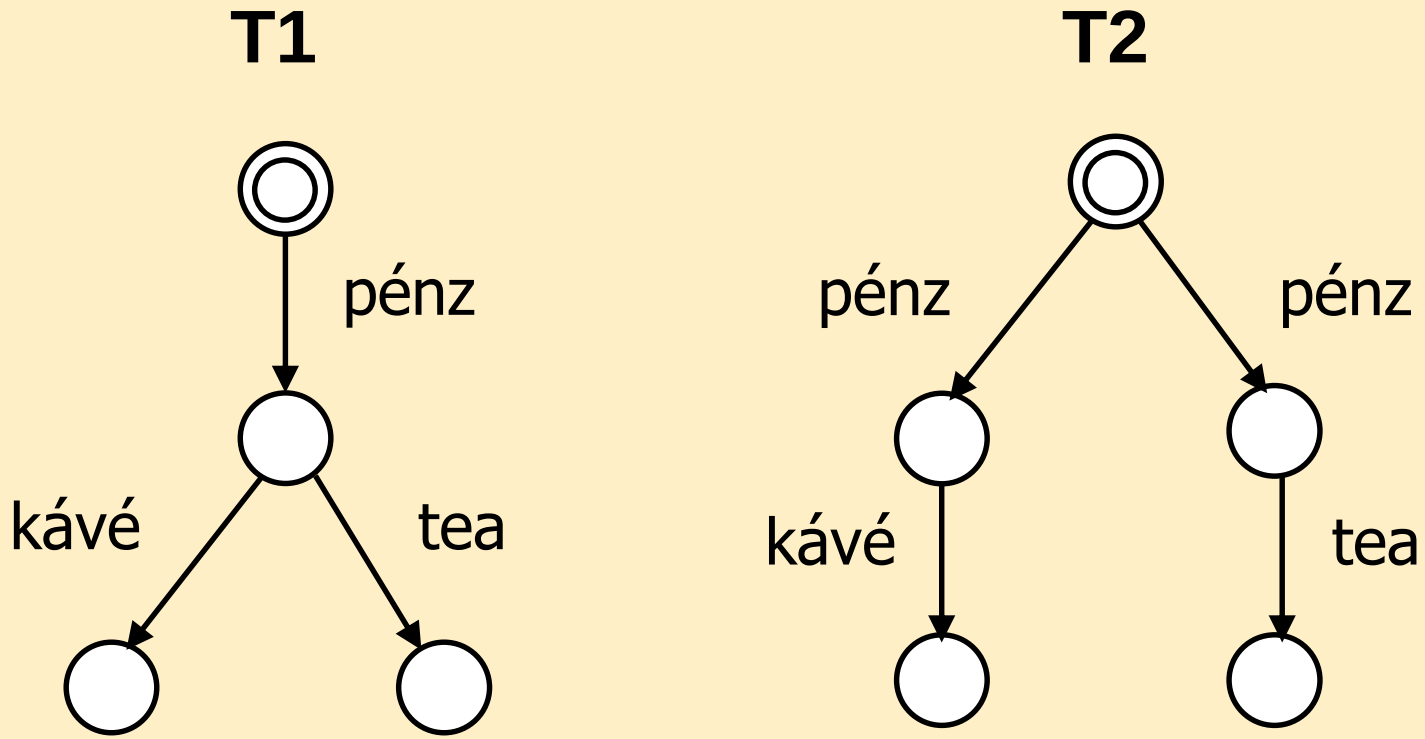
$\rightarrow \subseteq S \times Act \times S$ címkézett állapotátmenetek

Állapotátmenetek szokásos jelölése: $s_1 \xrightarrow{a} s_2$

LTS példák

- Italautomata modelljei

  Act = {pénz, kávé, tea}



3. Kripke tranzíciós rendszer

KTS, Kripke Transition System:

- Állapotok és átmenetek tulajdonságait is kifejezzük: címkézés atomi kijelentésekkel és akciókkal
- Egy állapothoz sok címke rendelhető, egy átmenethez egy címke rendelhető

$KTS = (S, \rightarrow, L)$ és AP, Act, ahol

$AP = \{P, Q, R, ...\}$ atomi kijelentések halmaza (domén-specifikus)

$Act = \{a, b, c, ...\}$ akciók halmaza

$S = \{s_1, s_2, s_3, ...s_n\}$ állapotok halmaza

$\rightarrow \subseteq S \times Act \times S$ állapotátmeneti reláció

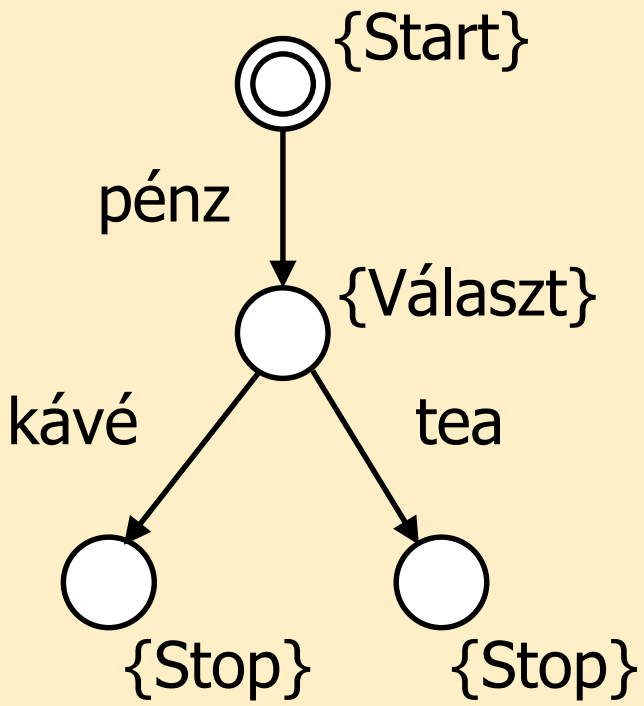
$L: S \rightarrow 2^{AP}$ állapotok címkézése atomi kijelentésekkel

KTS példa

- Italautomata modellje állapot címkékkel

  Act = {pénz, kávé, tea}

  AP = {Start, Választ, Stop}



Formális módszerek értékelése

Mit várhatunk?
Korlátok, sikertörténetek

Amik a formális vizsgálatot nehézzé teszik…

- Valósághű modellezés
  - Ismeretek hiánya, feltételezések (pl. a környezetről)
  - De: Formális módszerek használatától független ez a probléma
- Speciális ismereteket igényel a felhasználótól
  - Alacsony szintű matematikai modellek, jelölésrendszer
  - De: Mérnöki modellezési nyelvek eltakarhatják
- Bonyolultak az ellenőrzés és szintézis módszerei
  - Algoritmusok, technikák korlátait ismerni kell
  - Kézi beavatkozásra lehet szükség (pl. tételbizonyító rendszerek)
  - De: Terjednek a „gombnyomásra működő” eszközök
- Csak „kisméretű” problémákra alkalmazható
  - Modell, állapottér kezelhető-e a meglévő erőforrásokkal
  - De: Eszközök hatékonysága folyamatosan nő

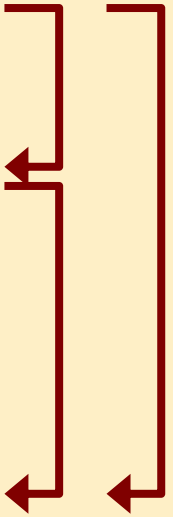
Jelen helyzet

- Megkötések
  - Diszkrét állapotú
  - Diszkrét idejű
  - Diszkrét eseményterű

  } rendszerek (DES)
- Kihívások, kutatási területek
  - Matematikai algoritmusok hatékonysága és szintje
  - Modell osztályok korlátai (pl. időzítés)
  - Modell készítés nehézsége (pl. absztrakció)
  - Nyelvek kifejezőképessége (pl. VHDL)
  - Nyelvek nem pontos definíciója (pl. UML)

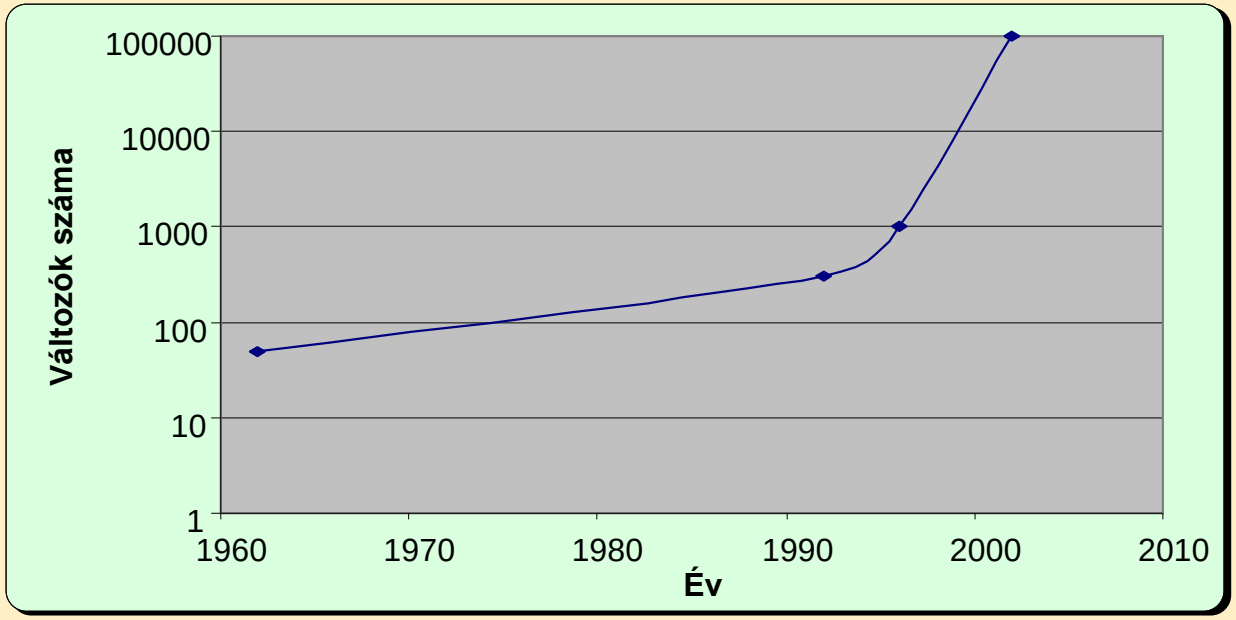
Modellek a formális ellenőrzéshez

- Rendszermodellek
  - Mérnöki modellek:
    - Pl. UML diagramok (fél-)formális szemantikával
  - Magasabb szintű modellek:
    - Vezérlés orientált: Automata, Petri-háló, …
    - Adatfeldolgozás orientált: Adatfolyam háló, …
    - Kommunikáció orientált: Processz algebra, …
  - Alapszintű matematikai modellek:
    - KS, LTS, KTS, automaták, Büchi automaták
- Tulajdonság leírások
  - Magasabb szintű:
    - Idődiagram, üzenet szekvencia diagram (MSC)
  - Alapszintű:
    - Elsőrendű logika, temporális logika, referencia automata



A formális verifikáció fejlődése

- A SAT eszközök (kielégíthetőség) lehetőségei:



- Modellellenőrző eszközök képességei:
  - $10^{20} \approx 2^{66}$ méretű állapottér (ROBDD-vel, 1990)
  - $10^{100} \approx 2^{328}$ méretű állapottér speciális esetben
  - 10^{62900} méretű állapottérre is volt példa (MIT TDK)

Sikeres megközelítés

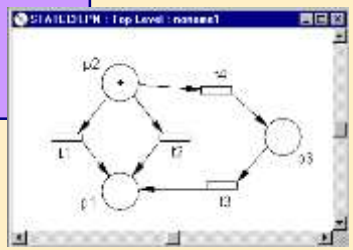
Rendszer tervezése

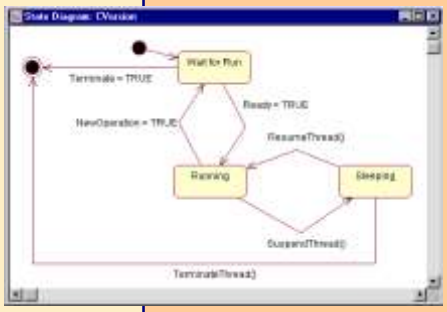
Formális ellenőrzés

Mérnöki modell (pl. UML)

Automatikus modellgenerálás

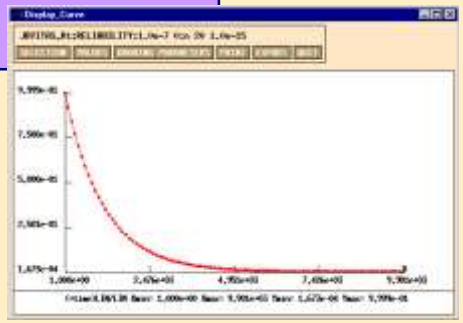
Formális modell





Eredmények visszavezetése

Analízis



Megvalósítás

Implementáció

```
int main() {
  while (i<z){
  a=x+x*b[i++];
  }
 }
```

Klasszikus alkalmazások

- USA TCAS-II forgalomirányító rendszer
  - RSML nyelven specifikált; teljesség és ellentmondás-mentesség ellenőrzése
- Philips Audio Protocol
  - 1994: manuális verifikáció, majd 1996: automatikus ellenőrzés (HyTech)
- Lockheed C130J repülési szoftvere
  - Programfejlesztés helyességbizonyítással (CORE spec. nyelv + Ada)
  - Költség nem nőtt a tesztelés egyszerűsödése miatt
- IEEE Futurebus+ szabvány
  - Carnegie Mellon SMV: cache koherencia protokoll hibájának kiderítése
- Hardver projektek: ACL2 automatikus tételbizonyító
  - Motorola DSP Complex Arithmetic Processor mag (250 regiszter): DSP algoritmusok ellenőrzése
  - AMD 5K86 processzor: Lebegőpontos osztás algoritmusának ellenőrzése
- Intel Core i7 processzor
  - *„For the recent Intel CoreTM i7 design we used formal verification as the primary validation vehicle for the core execution cluster"*
  - Szimbolikus szimuláció az adatutak teljes vizsgálatára (2700 mikroutasítás, 20 mérnökévnyi munka) – Binary Decision Diagram alkalmazása
- Modell alapú szoftverfejlesztéshez kapcsolódó eszközök
  - IBM, Esterel, Prover, Mentor, Verum, Telelogic, …

Forráskódhoz illeszkedő formális verifikáció

- Java
  - Bandera, PathFinder: modell absztrakció
  - Java VM formalizálása: Abstract State Machine
- Ada
  - SPARK Ada verification condition generator tételbizonyítóhoz
- C
  - BLAST: Szoftver modellellenőrző C programokhoz (absztrakció)
  - CBMC: C alapú korlátos modellellenőrző
- C#, Visual Basic .Net
  - Zing (MS Visual Studio-hoz): Konkurens szoftver modellellenőrzése
- Spec# (C# superset)
  - MS Research Boogie 2: Specifikációs nyelvi kiterjesztések
  - Helyességi kritériumok ellenőrzése: program absztrakcióval és tételbizonyítóval (Z3)
- Microsoft Windows Driver Kit (WDK)
  - Static Driver Verifier Research Platform, SLAM 2 eszköz
  - Windows API használati feltételeinek statikus ellenőrzése

Összefoglalás

- Mik a formális módszerek?
  - Formalizmus, formális nyelv
  - Formális módszerek és eszközök: Szimuláció, formális verifikáció, szintézis
- Mire használhatók?
  - Motiváció: Szoftver minőségi kihívások
  - A formális módszerek lehetőségei
- Mit várhatunk?
  - Korlátok
  - Sikertörténetek