Modell alapú tesztelés

Majzik István és Micskei Zoltán
Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék
http://www.inf.mit.bme.hu/

Tartalomjegyzék

UML modellek tipikus használata a tesztelésben

- Használati eset diagram:
  - Validációs tesztelés: tesztelendő használati esetek
- Osztály- és objektumdiagram:
  - Modulltesztelés: komponensek, interfészek azonosítása
- Állapottérkép és aktivitás diagram:
  - Modulltesztelés: strukturális teszteléshez referencia
- Üzenet-szekvencia és együttműködési diagram:
  - Integrációs tesztelés: forgatókönyvek leírása
- Komponens diagram:
  - Rendszertesztelés: tesztelendő fizikai komponensek
- Telepítés diagram:
  - Rendszertesztelés: teszt konfiguráció

Tartalomjegyzék

Modell alapú tesztelés

Model-based testing (MBT)

- ~ explicit modell felhasználása valamilyen tesztelési tevékenység támogatására,
  - tesztesetek előállítása
  - teszt csonkok előállítása
  - teszt orákulumok előállítása
  - modell építése és validálása
  - …
- Szűkebb értelemben: modell alapú tesztgenerálás

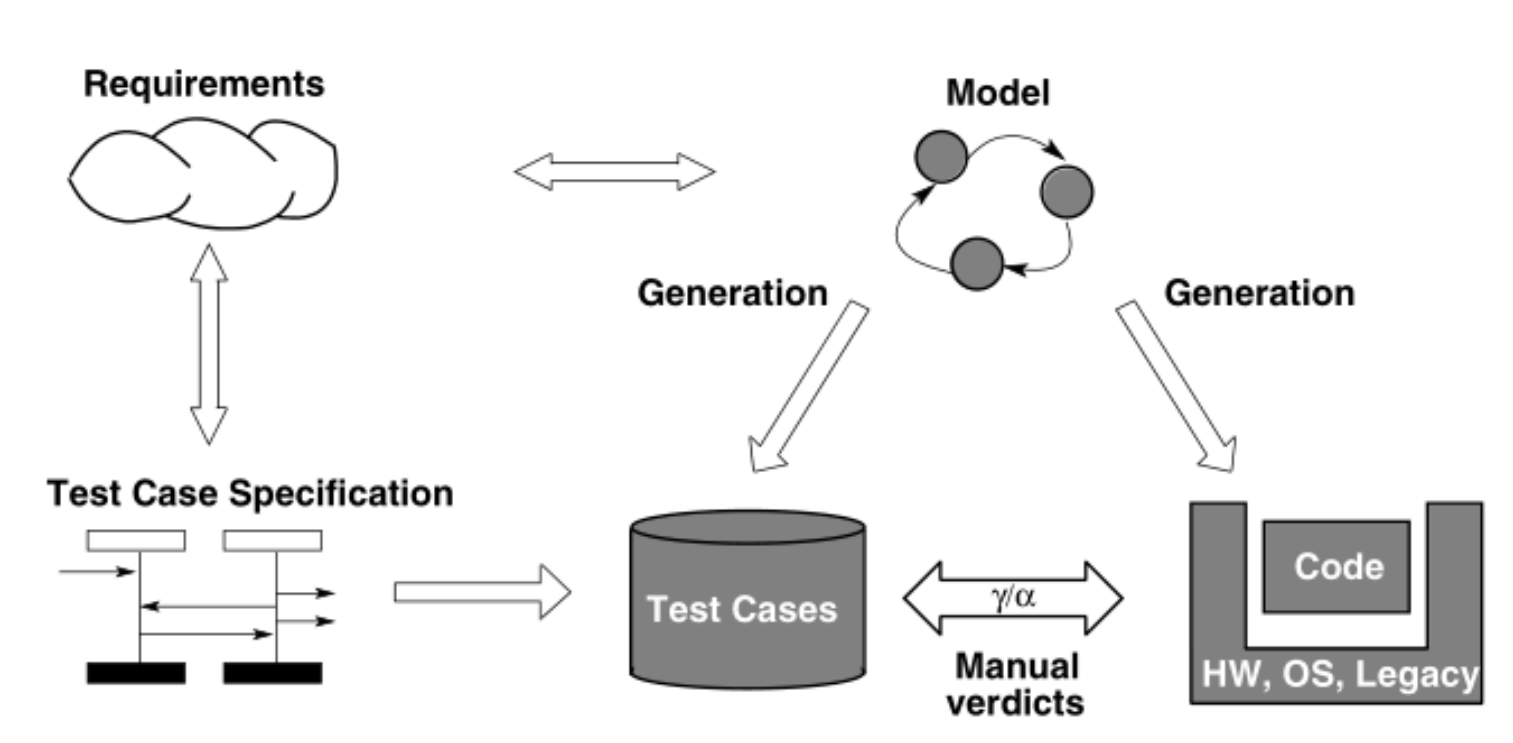
Modell alapú tesztgenerálás

„MBT encompasses the processes and techniques for

- the automatic derivation of abstract test cases from abstract models,
- the generation of concrete tests from abstract tests, and
- the manual or automated execution of the resulting concrete test cases”

Forrás: M. Utting, A. Pretschner, B. Legeard. „A taxonomy of model-based testing approaches”, STVR 2012; 22:297–312

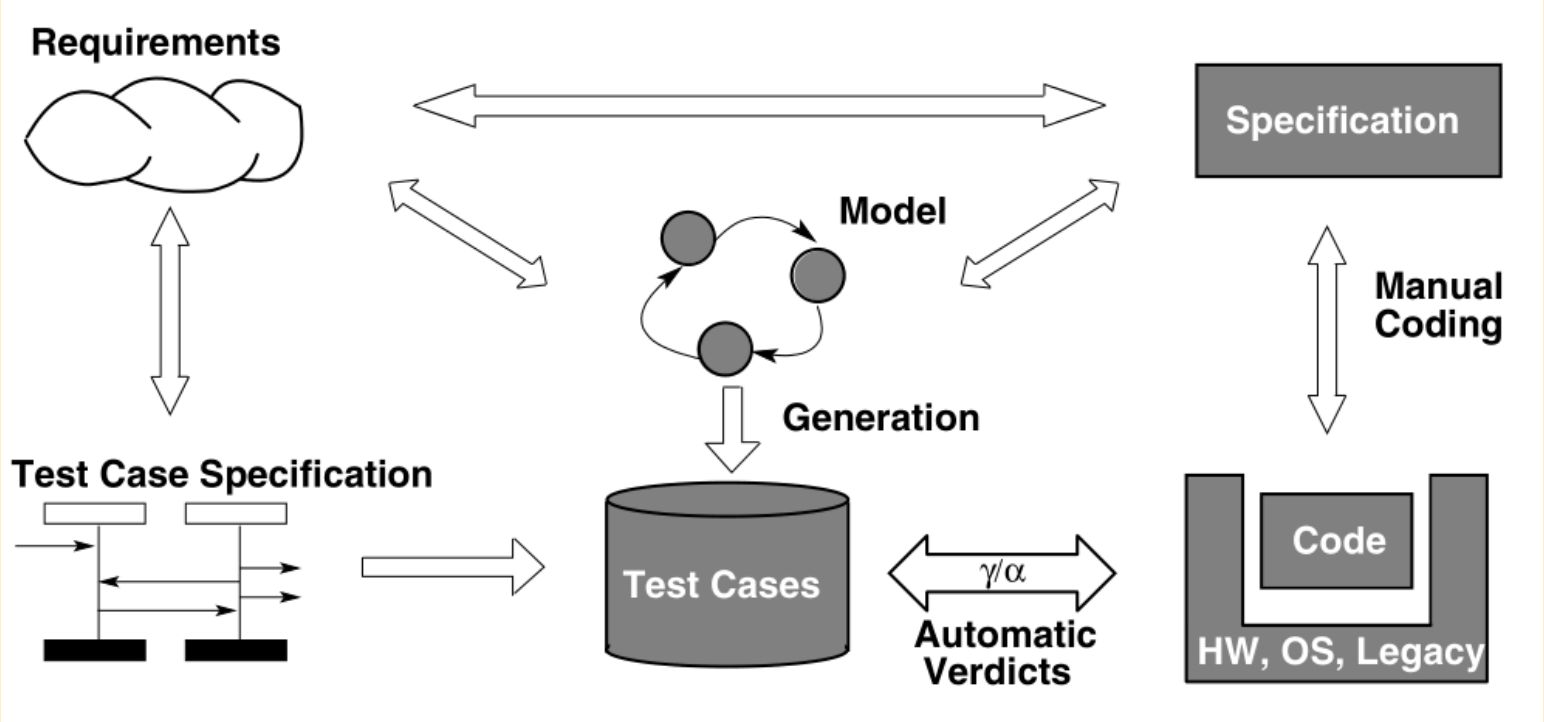
Használati esetek (1)



- Közös fejlesztési és tesztelési modell
- Mit tesztelünk?
  - Kódgenerátor; modell feltételezései

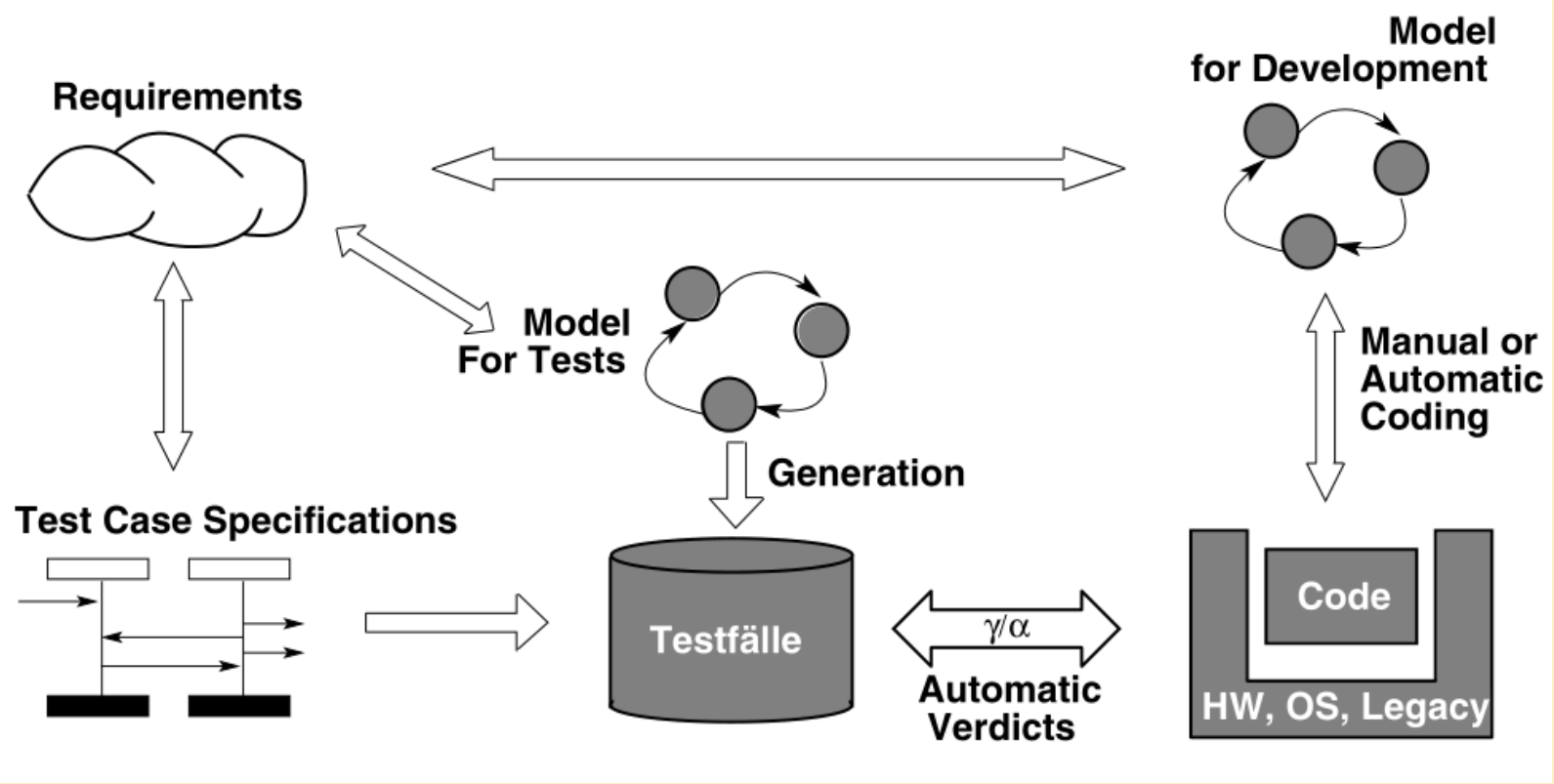
A. Pretschner, J. Philipps. „Methodological Issues in Model-Based Testing”, Model-Based Testing of Reactive Systems, 2005.

Használati esetek (2)



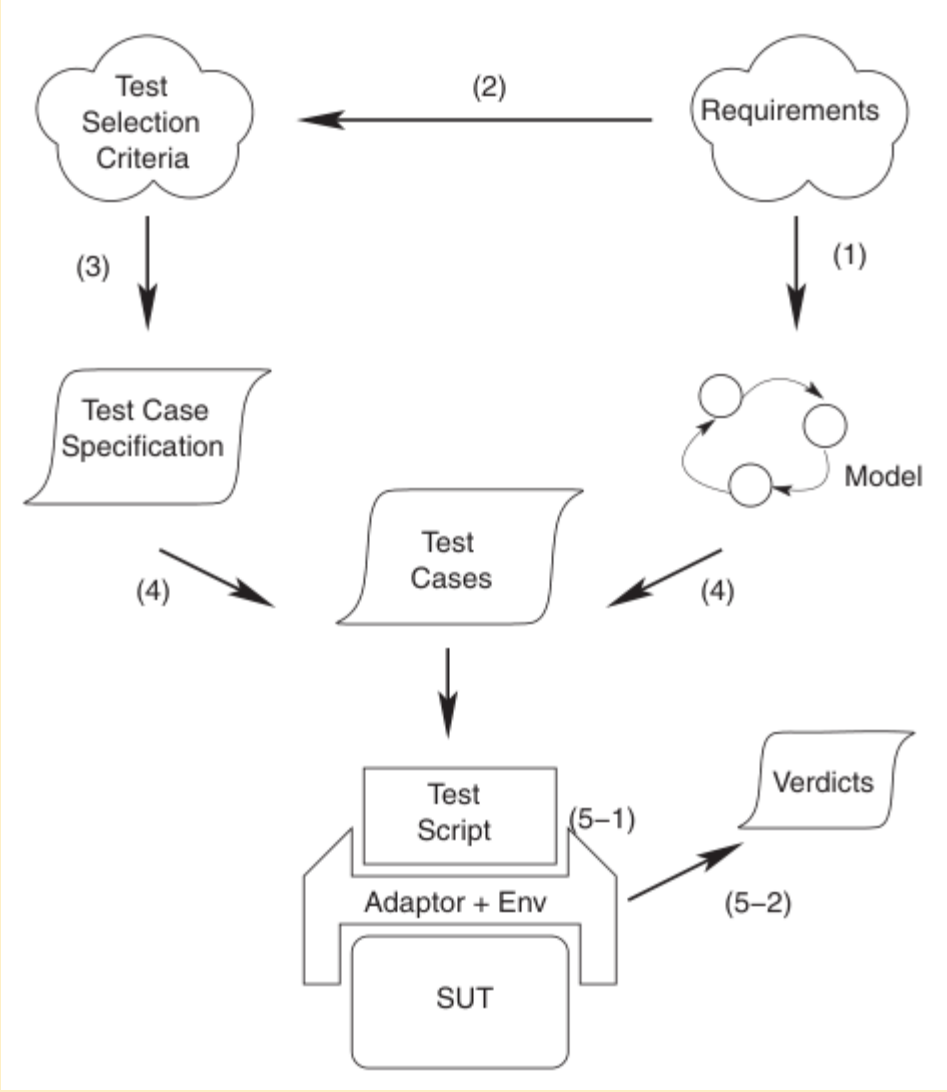
- Kézi implementáció
- Külön tesztmodell

Használati esetek (3)



- Külön fejlesztési és tesztelési modell

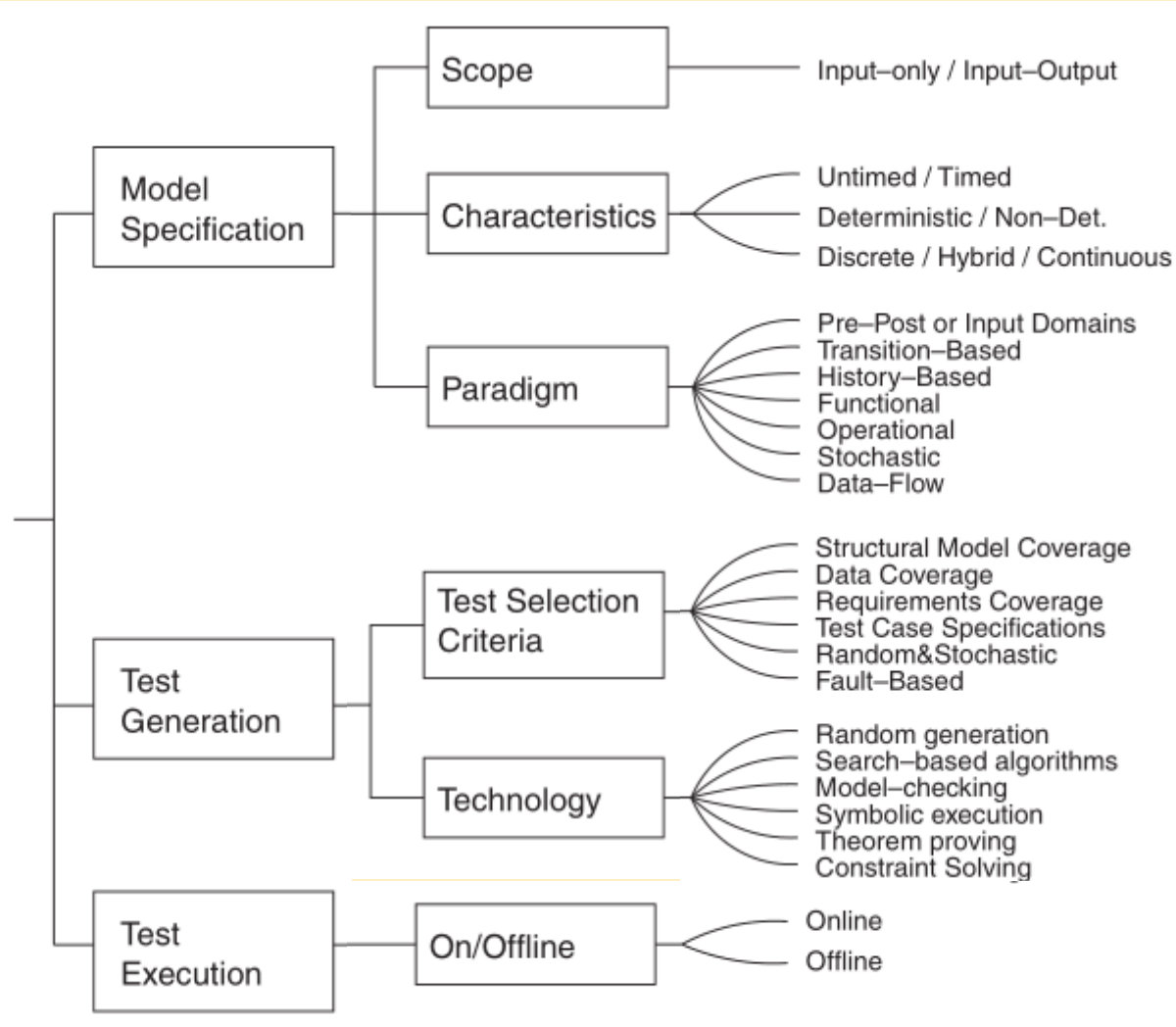
MBT folyamata



- Teszt modell
- Absztrakció
- Absztrakt és konkrét tesztek

Forrás: M. Utting, A. Pretschner, B. Legeard. „A taxonomy of model-based testing approaches”, STVR 2012; 22:297–312

MBT taxonómia



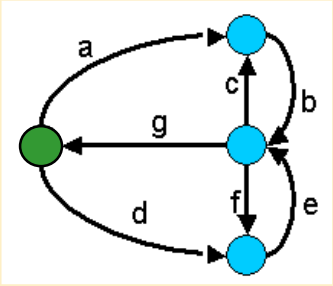
Forrás: M. Utting, A. Pretschner, B. Legeard. „A taxonomy of model-based testing approaches”, STVR 2012; 22:297–312

Tartalomjegyzék

A direkt algoritmusok tipikus alkalmazási területe

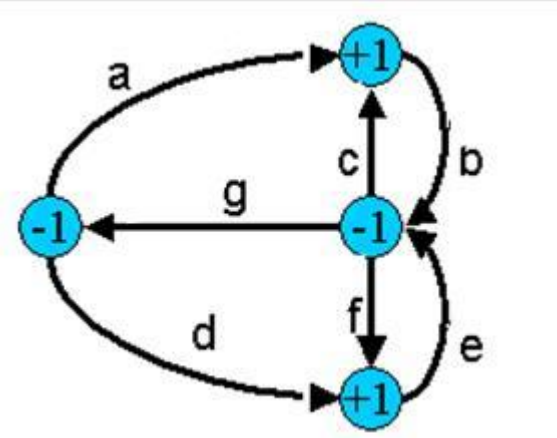
- Állapot alapú, eseményvezérelt működés
  - Eseményre triggerelt állapotátmenetek
  - Akciók (mint válasz jellegű kimenetek)
- Használható modellek
  - Automaták (FSM; Mealy, Moore, Büchi, …)
  - Magasabb szintű formalizmusok leképezhetők
    - UML állapottérkép
    - SCADE Safe Statechart
    - Simulink Stateflow
- Gráfelméleti algoritmusok
  - Algoritmus létezik sokféle tesztelési feladathoz
  - Optimális tesztek: Tipikusan NP-teljes algoritmusok ☹

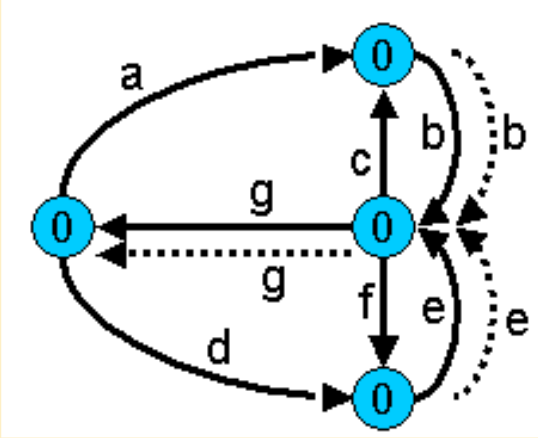
Gráfelméleti algoritmus átmenet fedéshez



- Problémák megfeleltetése
  - Tesztelési probléma: Átmenetek fedése
    - Minden átmenet fedése teszt szekvenciával
    - A teszt szekvencia vigyen vissza a kezdeti állapotba
  - Gráfelméleti probléma: „New York-i utcaseprő” probléma
    - Egy irányított gráfban mi az a (legrövidebb) bejárási szekvencia, ami minden élet bejár és a kezdeti helyre visz vissza?
    - (Ugyanez nem irányított gráfban: „Kínai postás” probléma)
- Megoldás alapötlete:
  - Helyek polaritásainak számítása: Bejövő mínusz kimenő élek száma
  - Olyan élek duplikálása, amelyek pozitívtól negatív polaritású helyekig vezetnek, amíg minden hely nulla polaritású nem lesz
  - Euler-kör keresése az így adódó gráfban (lineáris algoritmus)
    - Euler-kör: Minden élet bejár; ilyen gráfban biztosan képezhető
  - Az Euler-kör bejárása adja a teszt szekvenciát

Egy példa átmenet fedéshez





Eredeti gráf
hely polaritásokkal

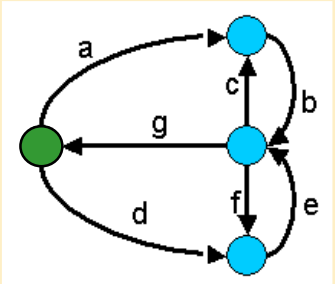
Duplikált élekkel
kiegészített gráf (Euler-gráf)

Bejárási szekvencia (Euler-kör):

a b c b f e g d e g

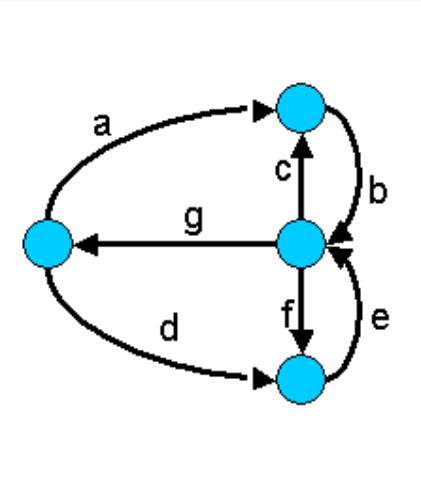
Gráfelméleti algoritmus átmenet kombináció fedéshez

- Problémák megfeleltetése
  - Tesztelési probléma: Átmenet kombinációk fedése
    - Minden lehetséges, egymás után n számú átmenetből álló sorozat fedése a teszt szekvenciával
    - A teszt szekvencia vigyen vissza a kezdeti állapotba
    - Legegyszerűbb eset: Minden lehetséges átmenet-pár fedése
  - Gráfelméleti probléma: „Bankrabló” probléma
    - (Legrövidebb) élszekvencia, amiben minden lehetséges n hosszú élsorozat előfordul (legegyszerűbb eset: n=2)

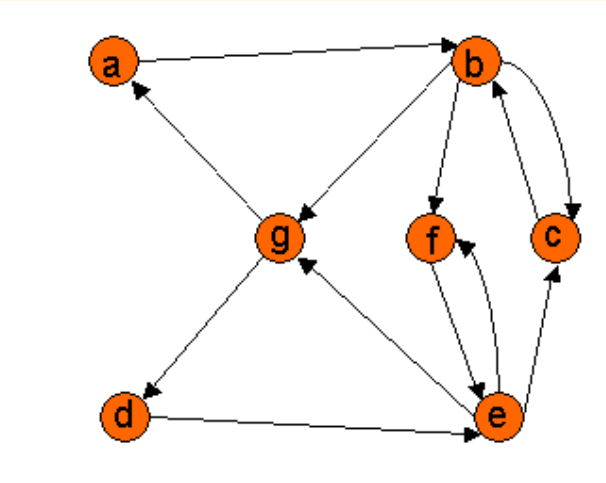


- Megoldás (de Bruijn algoritmus) alapötlete (n=2):
  - Duális gráf megkezdése: Az eredeti gráf éleiből helyek lesznek
  - Az eredeti gráfban létező élpárok esetén él behúzása a duális gráfba az élek által adott helyek közé
  - A duális gráf kiegészítése (élek duplikálásával) Euler-gráffá
  - Az így kapott gráfban az Euler-kör adja a teszt szekvenciát

Egy példa átmenet kombináció fedéshez



Eredeti gráf

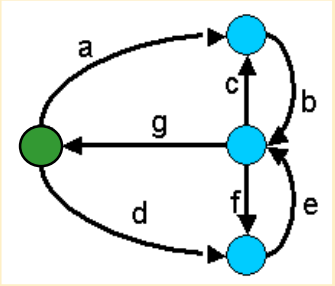


Duális gráf az élpárokkal

Bejárási szekvencia a duális gráf alapján élpárok fedéséhez:

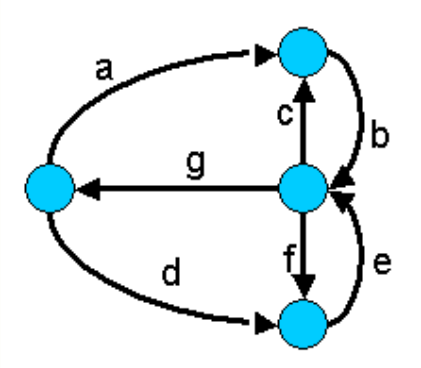
a b c b f e c b g d e f e g

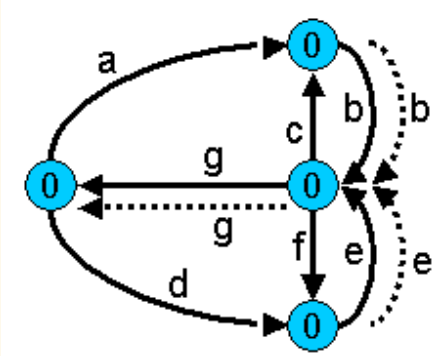
Gráfelméleti algoritmus konkurens átmenet fedéshez



- Problémák megfeleltetése
  - Tesztelési probléma:
    Konkurens tesztelés átmenetek fedéséhez
    - Teljes átmenet fedés a cél, de több tesztelő van
    - Célszerű egyenletesen megosztani a problémát, hogy a legrövidebb idő alatt végezzenek; mindegyik a kezdőállapotból kezd
    - Feltétel: Egy bemenettel bárhonnan kezdőállapotba vihető a rendszer
  - Gráfelméleti probléma: „Utcaseprő brigád” probléma
- Megoldás: Heurisztika (nem optimális megoldás)
  - Egy-egy bejáráshoz k felső határ megadása
  - Olyan élszekvencia keresése, amely a legtöbb eddig nem érintett élet tartalmazza, de legfeljebb k hosszú; a végén kezdőállapotba vezérelve a bejárást
  - Ezután újabb élszekvenciák felvétele, amíg van be nem járt él
  - A k felső határral lehet próbálkozni

Egy példa konkurens átmenet fedéshez





Eredeti bejárási szekvencia (Euler-kör, 1 tesztelő):

a b c b f e g d e g

Egy lehetséges megosztás 2 tesztelőre:

- Tesztelő 1: a b c b f e g (7 időegység kell)
- Tesztelő 2: d e g

Egy jobb megosztás (heurisztikával) 2 tesztelőre:

- Tesztelő 1: a b c b g (5 időegység kell)
- Tesztelő 2: d e f e g

Tartalomjegyzék

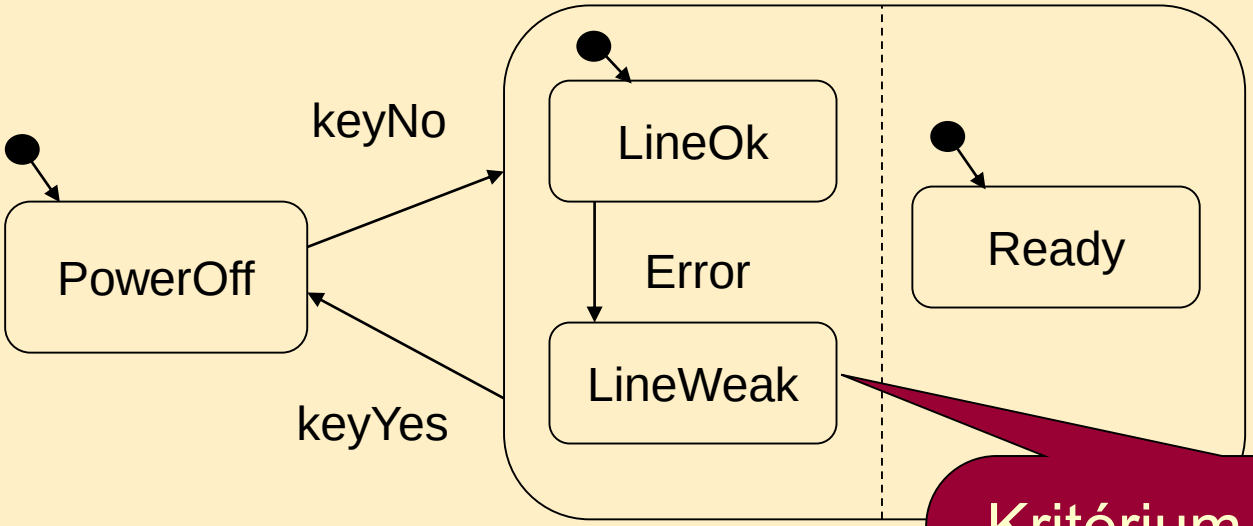
- Motiváció
  - Modellek szerepe a tesztelésben
  - Modell alapú tesztgenerálás
- Tesztgenerálás fedettségi kritériumokhoz
  - Direkt algoritmusok
  - Modellellenőrzők használata
  - Tesztgenerálás korlátos modellellenőrzéssel
- Tesztgenerálás hibamodellek alapján
  - Modell mutációk
  - Ekvivalencia relációk tesztgeneráláshoz
- Eszközök a tesztgeneráláshoz

Alapötlet

- Tipikus tesztelési kritériumok:
  - Vezérlés alapú: Állapotok, átmenetek lefedése, be- és kimenő átmenet-párok lefedése egy-egy állapothoz
  - Adatfolyam alapú: Változó definiálások és felhasználások lefedése
- Tesztgeneráláshoz szükséges:
  - Állapottér bejárása → Modellellenőrző is ezt csinálja
- Alapötlet:
  - Járja be a modellellenőrző az állapotteret!
  - Irányítsuk úgy, hogy az általa adott ellenpélda legyen a teszteset!

A modellellenőrző használata tesztgenerálásra

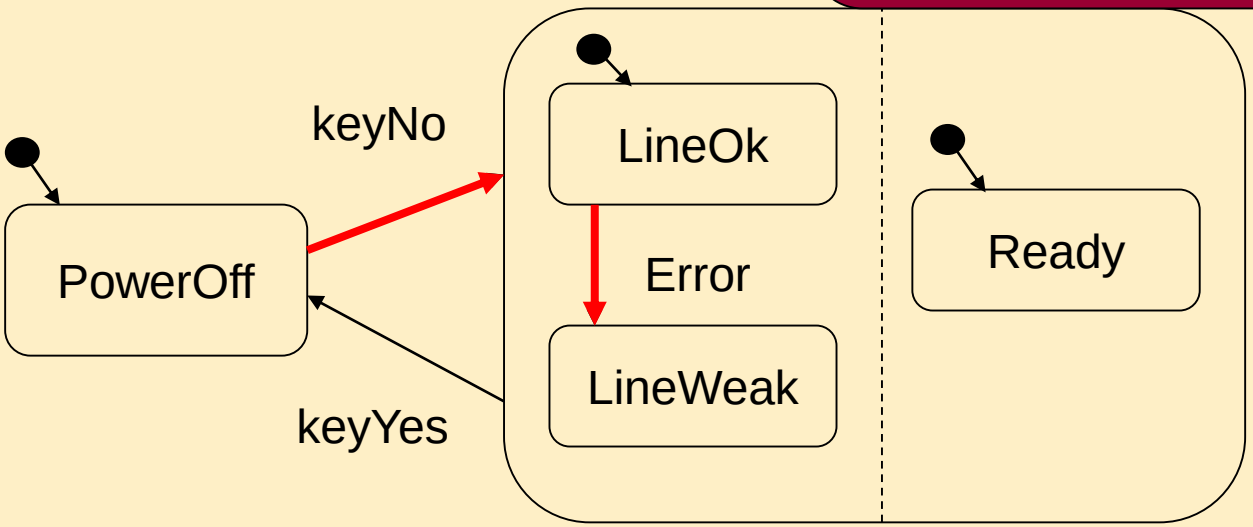
LineWeak állapotot elérő (lefedő) teszteset?



Kritérium megadása:
A LineWeak állapotot soha sem lehet elérni:
¬ (EF LineWeak)

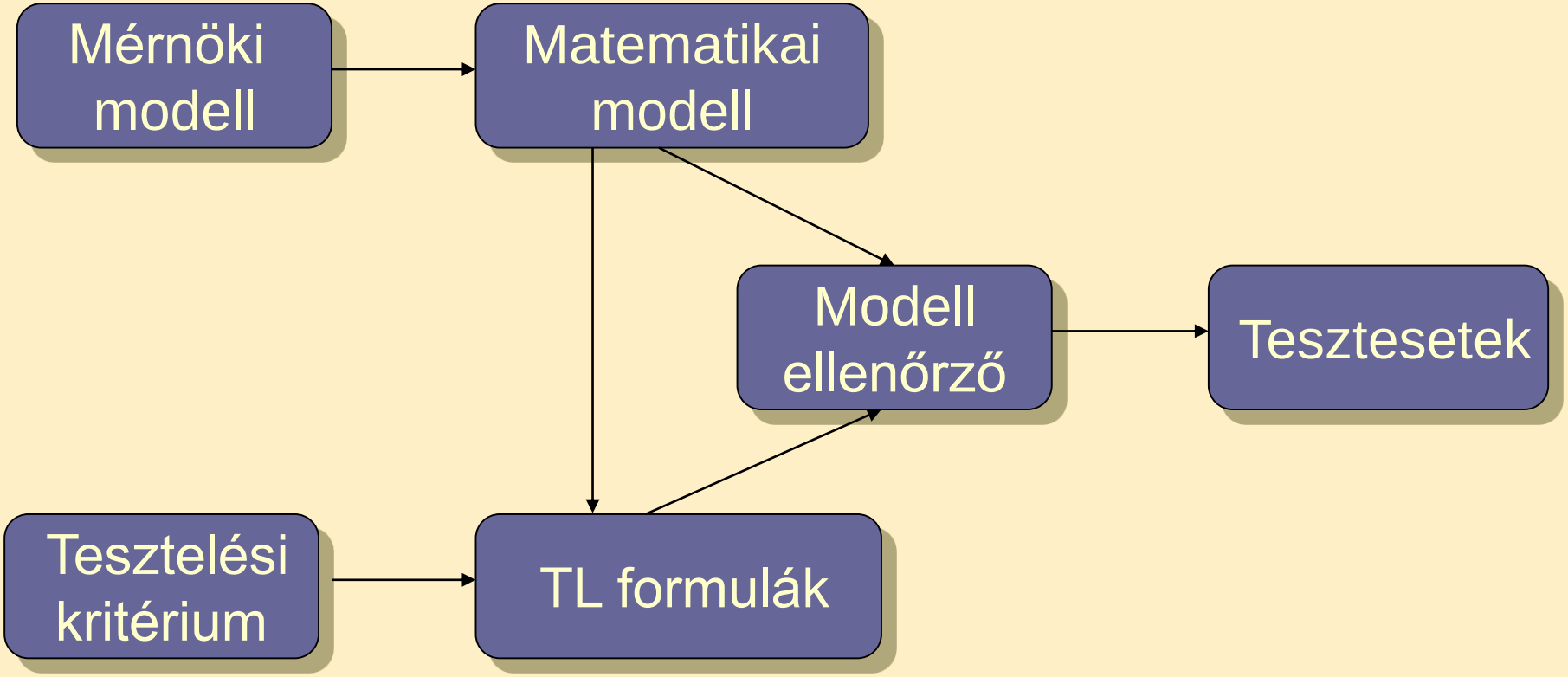
A modellellenőrző használata tesztgenerálásra

Modellellenőrző: ellenpélda tulajdonság nem teljesül, az állapot elérhető.

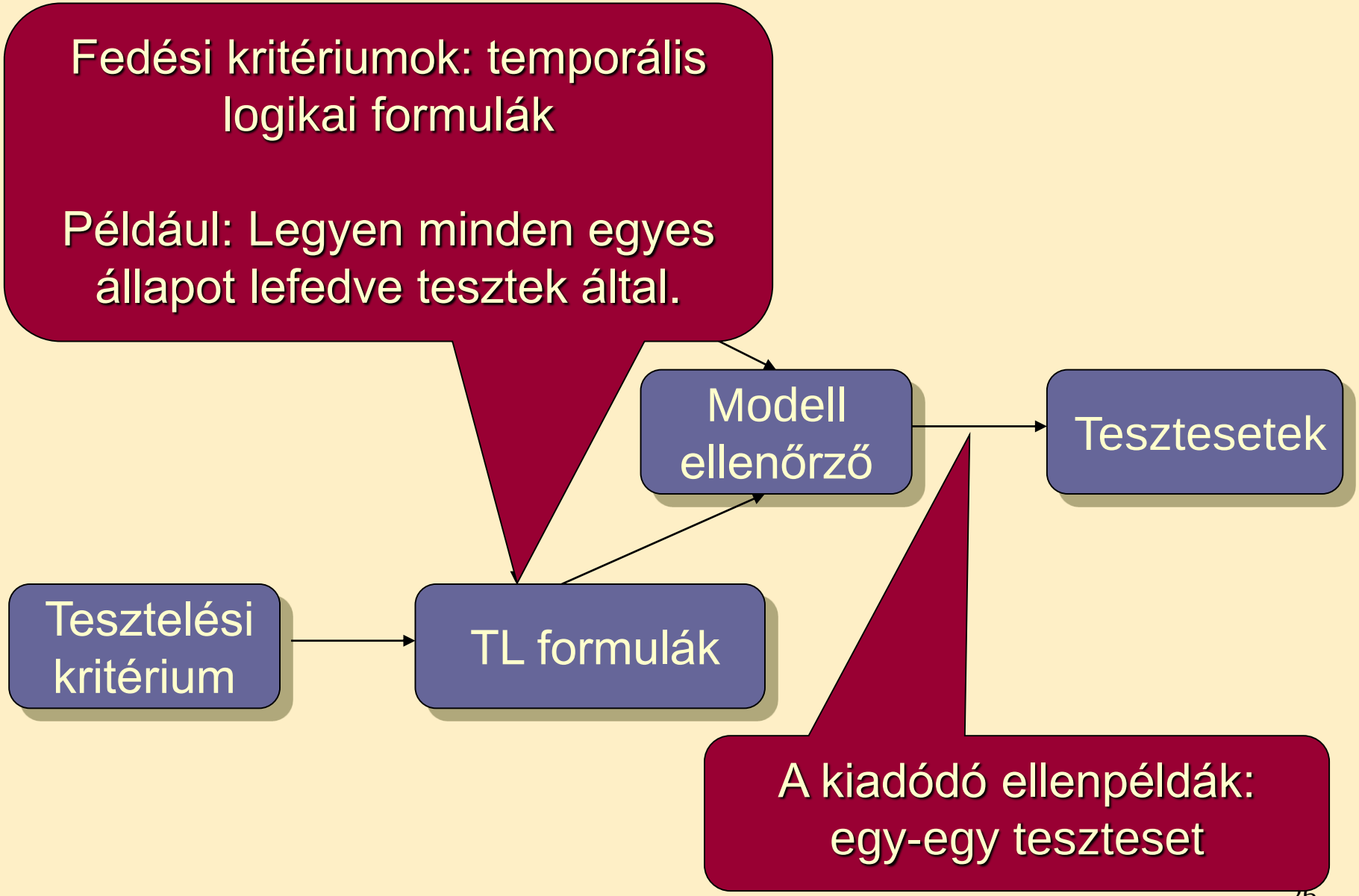


Ez a LineWeak állapotot lefedő teszteset!

Automatikus tesztgenerálás



Automatikus tesztgenerálás



Vezérlés alapú fedettségi kritériumok

- Állapotfedés:
  {¬EF s | s alapszintű állapot}

Kritériumhalmaz!

Ha megfelelő stabil állapot is kell újabb teszthez:
{¬EF (s ∧ EF exit) | s alapszintű állapot}

A további képletekben itt EF exit nem szerepel.

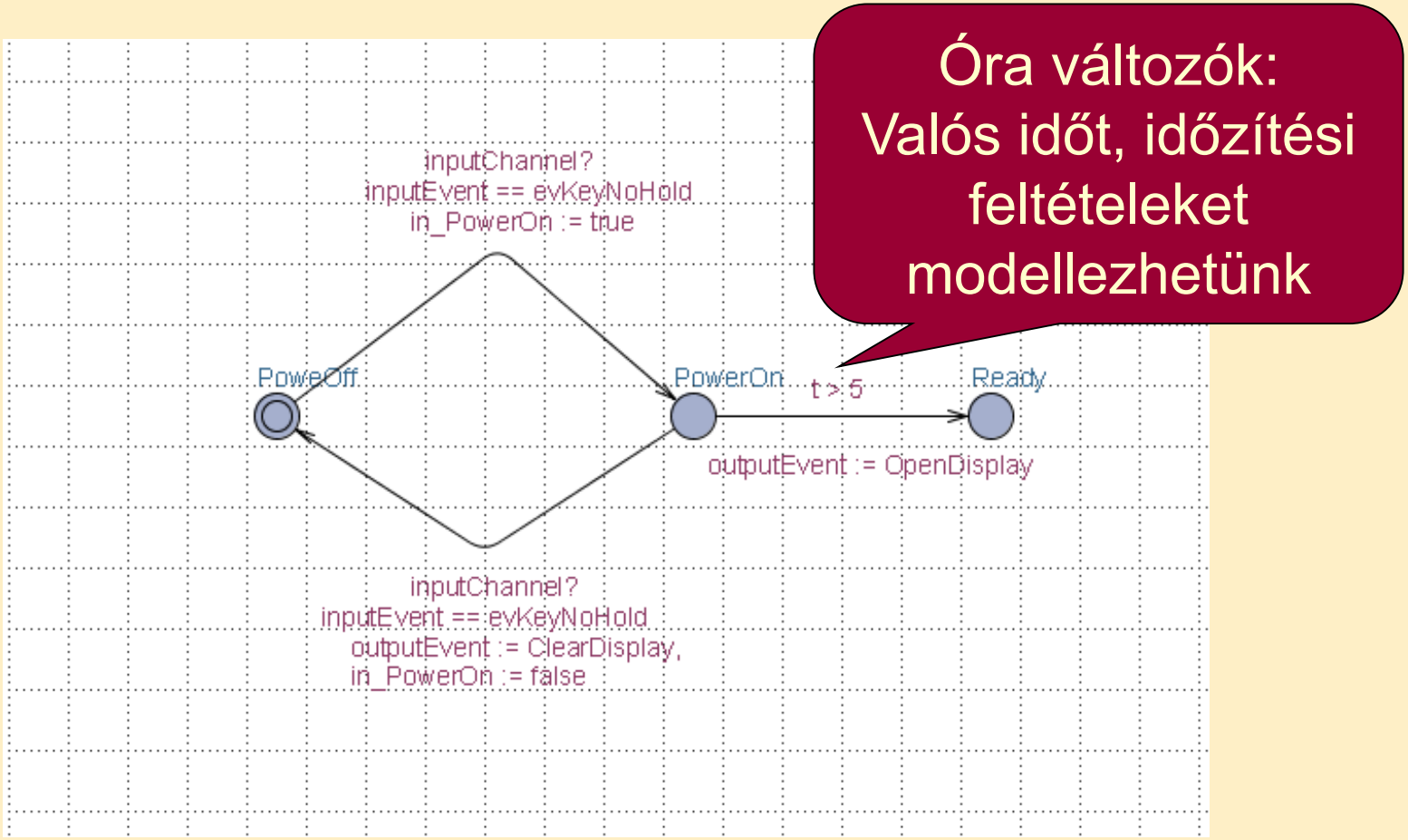
- Gyenge átmenet fedés:
  {¬EF t | t átmenet}
- Erős átmenet fedés:
  {¬EF t | t átmenet} ∪ {¬EF it | it implicit átmenet}

Erős fedés: Implicit átmenetek (helyben maradás) is tesztelve

Optimalizáció

- Modellellenőrző feladata:
  - Állapottér hatékony bejárása: Gyorsan, kis tárigénnyel
  - Ellenpélda generálása: Minél hamarabb álljon elő
- A tesztgenerálási feladat:
  Gyorsan minél rövidebb ellenpéldát találni
  → Speciális beállítások szükségesek a modellellenőrzőben
  - Legrövidebb/legkisebb tesztkészlet kiválasztása: NP-teljes probléma!
- Lehetőségek (pl. SPIN modellellenőrző esetén):
  - Szélességi keresés az állapottérben (BFS)
  - Mélységi keresés, de mélységkorláttal (limited DFS)
  - Rövidebb ellenpélda iteratív keresése
  - Közelítő modell ellenőrzés (hash függvény az állapottároláshoz)
    - Bizonyos állapotokat nem jár be a keresés során
    - Ha talál ellenpéldát, az valós teszt lesz

Kiterjesztés valósidejű rendszerekre



Időzített automaták használata

Speciális modellellenőrző: UPPAAL

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 tesztesetben

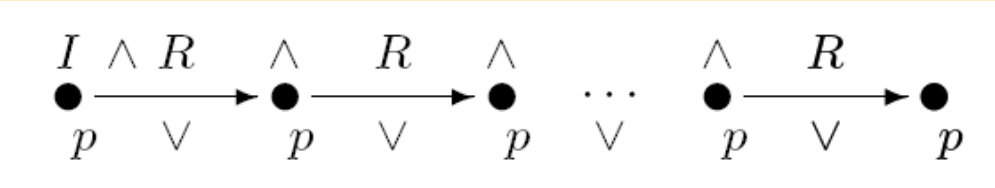
State:
(input.sending mobile.PowerOn mobile1.LineOK mobile2.CallWait)
t=6 inputEvent=28 outputEvent=14 in_PowerOn=1 #depth=5

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

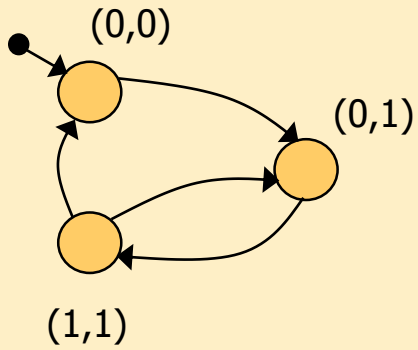
Tartalomjegyzék

Alapötlet: Korlátos modellellenőrzés alkalmazása

- SAT probléma megoldóinak használata
  - SAT megoldó: Boole függvényekhez keres helyettesítési értéket, ami a függvény értékét igazzá teszi
- A modell elemeinek leképzése logikai függvénybe:
  - Kezdőállapotokra vonatkozó predikátum: I(s)
  - Elérendő állapotokra vonatkozó predikátum: p(s)
  - Állapotátmeneti reláció: R(s, s')
    - Lépésenkénti „kibontás”: $R(s_i, s_{i+1})$
- A logikai függvény felírása
  - Kezdőállapotból indul: Az I(s) predikátum az első állapotra
  - Kibontott átmenetek: Az $R(s_i, s_{i+1})$ reláció alkalmazása
  - Elérendő állapot: A p(s) predikátum valahol fennáll



Példa: A modell leképzése logikai függvénybe

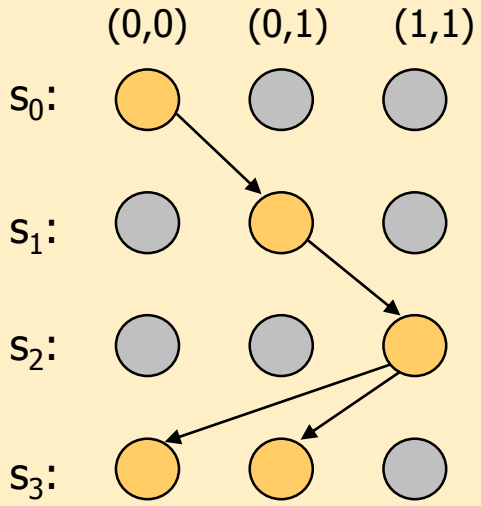


Kezdőállapot predikátum:

$I(x,y) = (\neg x \wedge \neg y)$

Állapotátmeneti reláció:

$$R(x,y,x',y') = (\neg x \wedge \neg y \wedge \neg x' \wedge y') \vee (\neg x \wedge y \wedge x' \wedge y') \vee (x \wedge y \wedge \neg x' \wedge y') \vee (x \wedge y \wedge \neg x' \wedge \neg y')$$



3 lépéses kihajtogatás a kezdőállapotból:

$$I(x_0,y_0) \wedge R(x_0,y_0,x_1,y_1) \wedge R(x_1,y_1,x_2,y_2) \wedge R(x_2,y_2,x_3,y_3)$$

SAT alapú tesztgenerálás fedési kritériumokhoz

- Formula konstruálás tesztgeneráláshoz:
  - Kibontás k lépésben a kezdőállapotból
  - Teszt kritérium megadása: TG formula, pl.:
    - Adott s állapot elérése,
    - Adott t állapotátmenet végrehajtása,
    - Adott modellrészlet bejárása, …

$$\exists s_0, s_1, \ldots, s_k : I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge TG$$

Állapot-szekvencia | Modell széthajtogatás | Teszt cél

- Ha ez a formula kielégíthető, akkor az egy tesztet ad:
  - A teszt teljesíti a TG kritériumot
  - Ha nem kielégíthető a formula, akkor nincs teszt a kritériumhoz

Használhatóság

- A tesztgenerálás korlátai
  - Legfeljebb adott hosszúságú teszt generálható
    - Iteratívan növelhető a kibontás korlátja
  - Így részleges megoldás adódik
    - Amit megtalál, az biztosan teszt eset lesz
    - Nem garantált, hogy megtalálja a teszt esetet (ha az hosszabb lenne, mint amit figyelembe veszünk)
- A modellből SAT probléma leképzése automatikus
- A TG teszt célok megadása egyszerűsíthető
  - C programokhoz: FQL nyelv teszt célokhoz (FSHELL)

    in /code.c/ cover @line(6),@call(f1) passing @file(code.c) \ @call(f2)
  - Elő- és utófeltételek megadása:
    - Előfeltétel szerepe: Érvényes teszt adatok
    - Utófeltétel szerepe: Van-e olyan teszt eset, amikor nem teljesül?
    - Ellenpélda generálás

Tartalomjegyzék

- Motiváció
  - Modellek szerepe a tesztelésben
  - Modell alapú tesztgenerálás
- Tesztgenerálás fedettségi kritériumokhoz
  - Direkt algoritmusok
  - Modellellenőrzők használata
  - Tesztgenerálás korlátos modellellenőrzéssel
- Tesztgenerálás hibamodellek alapján
  - Modell mutációk
  - Ekvivalencia relációk tesztgeneráláshoz
- Eszközök a tesztgeneráláshoz

Hibamodellek használata

- Tapasztalatok a szoftver tesztelés során
  - Csatolási effektus (coupling effect): Azok a teszt esetek, amik egyszerű hibákat megtalálnak, bonyolultabbakra is hatékonyak
  - Kompetens programozó hipotézis: A programok általában jók, a hibák nagy része gyakran előforduló tipikus hiba
- Alapötlet:
  - Állítsunk elő olyan „mutáns” modelleket, amik tipikus hibákat tartalmaznak, és generáljunk ezek kimutatására teszteket
  - Ezek várhatóan bonyolultabb hibákhoz is jobbak a véletlen teszteknél
- Tipikus mutációk:
  - Aritmetikai operátorok felcserélése feltételekben
  - Akciók (műveletek, üzenetek) sorrendjének megváltoztatása
  - Akciók kihagyása
  - …

Tesztgenerálás hibamodell alapján

- A tesztgenerálási feladat:
  - Olyan tesztek előállítása, amelyek különbséget tesznek az eredeti (hibamentes) és a mutáns (hibás) viselkedés között
  - Ezek ún. negatív tesztek (sikertelen lesz a teszt: nincs hiba!)
- Hogyan definiáljuk a „különbséget” két viselkedés között?
  - Specifikált – implementált, hibamentes – mutáns viselkedés között

  Milyen különbség megengedett?
  - Más viselkedés megengedett-e a specifikált mellett?
    - Pl. több kimenet, más reakció, …
  - Kihagyás (elmaradt kimenet) megengedett-e?
- Szokásos megoldások
  - Biztonságkritikus rendszer a fejlesztés végén:
    - Szigorúan a specifikáció szerint
    - Teljes specifikáció szükséges
  - „Hétköznapi rendszer” fejlesztés közben:
    - Beférjen a tesztelt implementáció specifikáció keretei közé

k-ekvivalencia a teszteléshez

- Alkalmazás: Fekete doboz teszteléshez
  - Bemenetek egy s állapotban: in(s) – vezérelhetők
  - Kimenetek egy s állapotban: out(s) – megfigyelhetők
  - Kimeneti akció hiánya is formalizálható: Speciális δ akció
- A k-ekvivalencia definíciója:

  Azonos bemeneti sorozat mellett azonos kimenetek az első k lépésre

- Jelölések:

	Eredeti M modell:	Mutáns M' modell:
Kezdeti állapot predikátum:	$I(s_0)$	$I'(s'_0)$
Állapotátmeneti reláció:	$R(s_i, s_{i+1})$	$R'(s'_i, s'_{i+1})$

A modell kibontása k lépésre: $I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1})$

Mutáció alapú tesztgenerálás k-ekvivalencia alapján

- SAT formula konstruálás a k-ekvivalenciához:
  - Azonos bemeneti szekvencia mindkét modellre
  - Kibontás k lépésben az eredeti modellre
  - Kibontás k lépésben a mutáns modellre
  - Legalább egy különböző kimenet lesz a kimeneti szekvenciában

$$\bigwedge_{i=0}^{k}(\mathrm{in}(s_i)=\mathrm{in}(s_i^{'})) \wedge I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i,s_{i+1}) \wedge I^{'}(s_0^{'}) \wedge \bigwedge_{i=0}^{k-1} R^{'}(s_i^{'},s_{i+1}^{'}) \wedge \bigvee_{i=0}^{k}(\mathrm{out}(s_i)\neq \mathrm{out}(s_i^{'}))$$

Egyező bemenetek | Eredeti modell | Mutáns modell | Eltérő kimenet

- Ha ez a formula kielégíthető, akkor az egy tesztet ad
  - A teszt különbséget tesz a modellek között:
    Kimutatja a mutációt, tehát a hiba felderítésére használható
  - Ha a formula nem kielégíthető, akkor ekvivalens a két modell

Mutáció alapú tesztgenerálás az IOCO reláció alapján

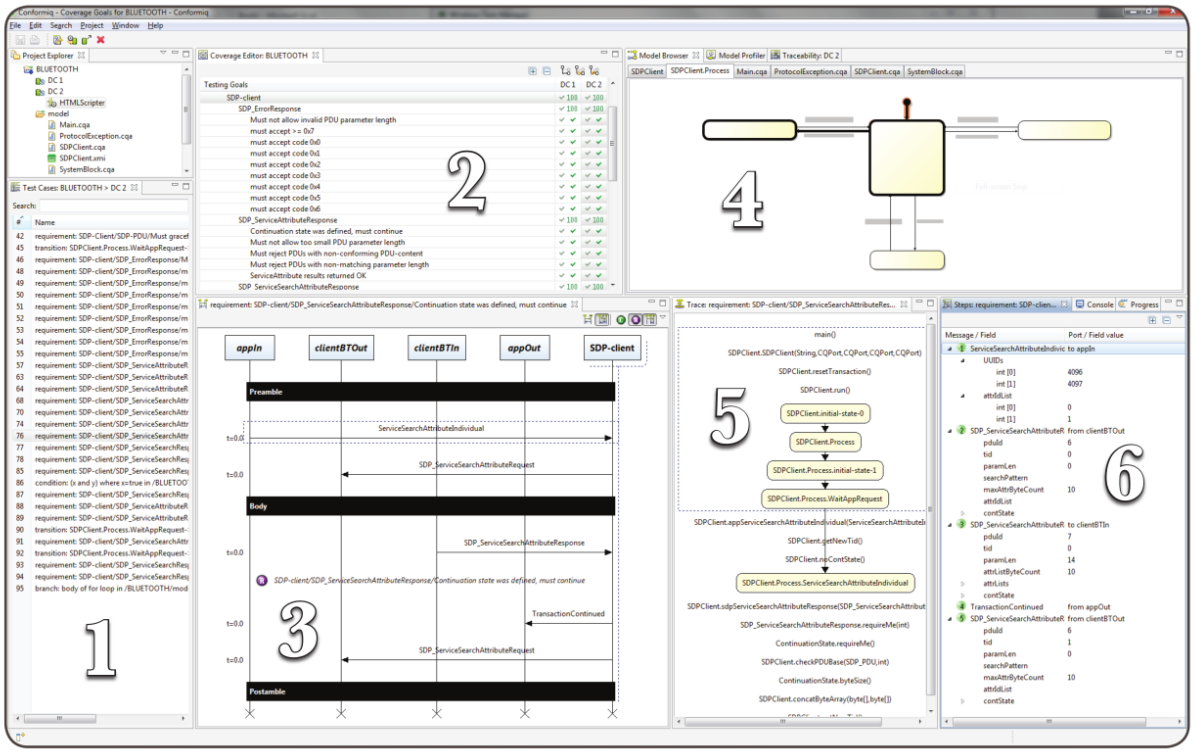
- Az IOCO reláció informálisan:
  - Megengedett, hogy azonos bemeneti szekvenciára a mutáns modell kimenetei részhalmazát képezik az eredeti modellben rögzített kimeneteknek (azaz „beleférnek” az eredeti modellbe)
    - Részleges viselkedés megengedett, de eltérő viselkedés nem
    - Többlet funkció megengedett az eredeti modellben nem rögzített bemeneti szekvenciára
  - A k-ekvivalenciánál megengedőbb konformancia reláció
- Definíció:

  Minden, az eredeti modellben felvehető akciószekvenciára igaz:
  Az így elérhető állapotokban a mutáns által nyújtott kimeneti akciók részhalmazát képezik az eredeti modell által nyújtott kimeneti akcióknak
- Tesztek generálhatók SAT megoldóval
  - Bonyolultabb a részhalmaz reláció vizsgálata miatt (itt nem írjuk fel)

Tartalomjegyzék

- Motiváció
  - Modellek szerepe a tesztelésben
  - Modell alapú tesztgenerálás
- Tesztgenerálás fedettségi kritériumokhoz
  - Direkt algoritmusok
  - Modellellenőrzők használata
  - Tesztgenerálás korlátos modellellenőrzéssel
- Tesztgenerálás hibamodellek alapján
  - Modell mutációk
  - Ekvivalencia relációk tesztgeneráláshoz
- Eszközök a tesztgeneráláshoz

Ipari MBT eszközök – Conformiq

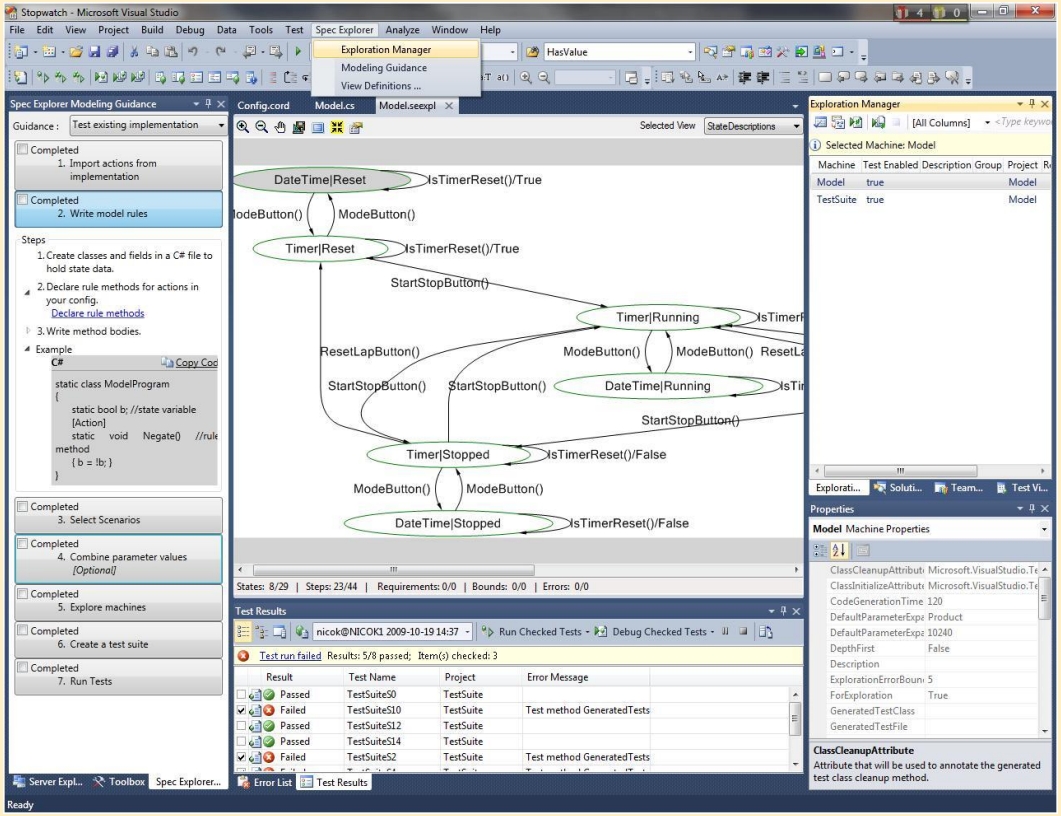


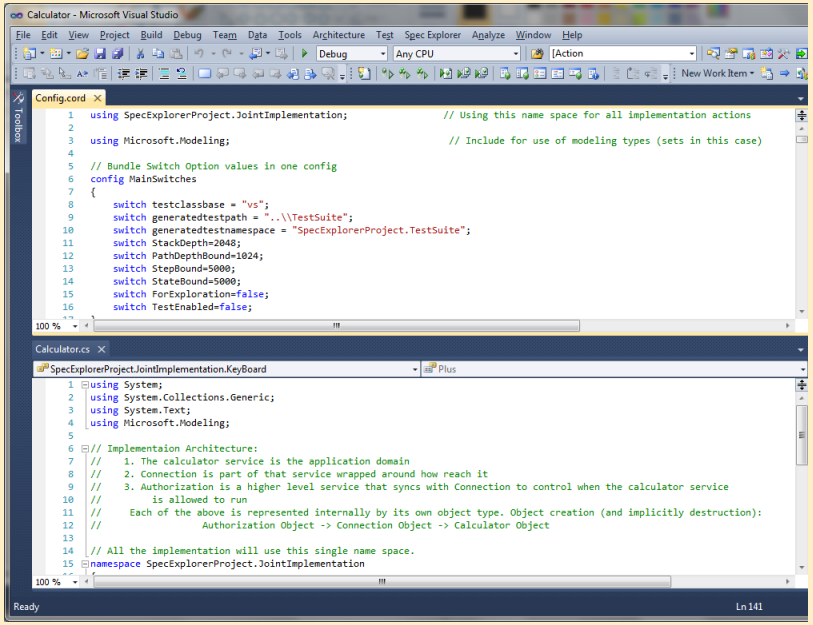
Conformiq Designer IDE for automatic test case generation

Forrás: Conformiq, „Testing Bluetooth Protocol Stacks with Computer-Generated Tests”. Technology brief. 2010

- Állapotgép modellek + Java akció kód
- Fedés: követelmény, állapot, átmenet…
- Integráció sok eszközzel

Ipari MBT eszközök – SpecExplorer





Forrás: https://visualstudiogallery.msdn.microsoft.com/271d0904-f178-4ce9-956b-d9bfa4902745

- C# modell program + adapter kód
- Slicing: forgatókönyvek, akció minták

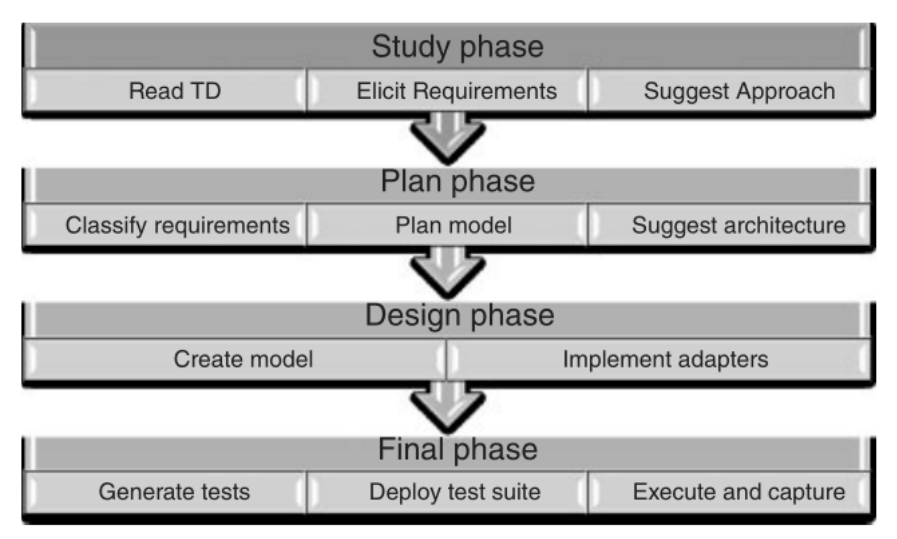
További eszközök

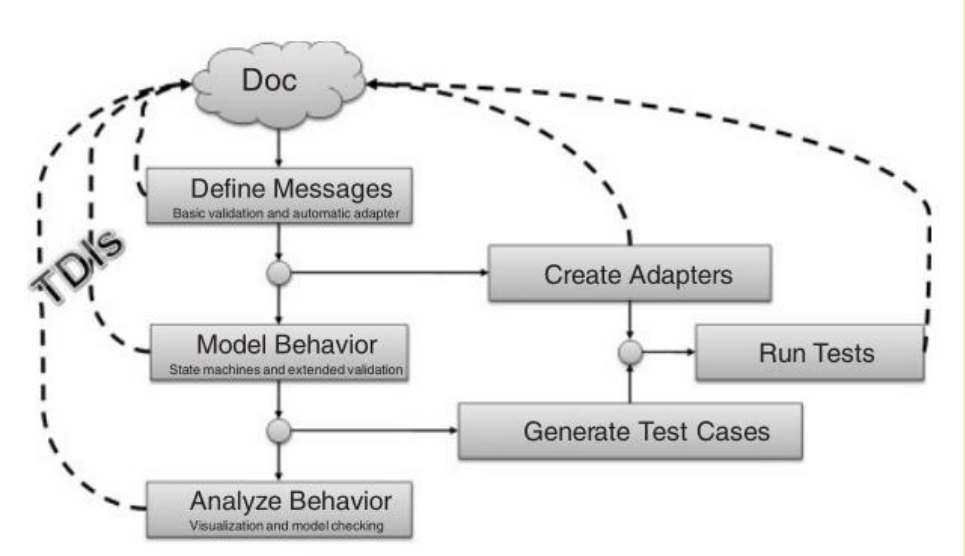
- CertifyIt (Smartesting)
  - UML + OCL modellek
- GraphWalker (OSS)
  - FSM modell
  - Gráf bejárása: véletlen, A*, legrövidebb út
- MoMuT::UML (akadémiai)
  - UML állapotgépek, mutációs tesztelés

További eszközök: http://mit.bme.hu/~micskeiz/pages/modelbased_testing.html

Esettanulmány: MS protokoll dokumentáció

- 250+ protokoll, 25ezer+ oldal dokumentáció
- 250+ mérnökév, 350+ mérnök
- Eszköz: Spec Explorer

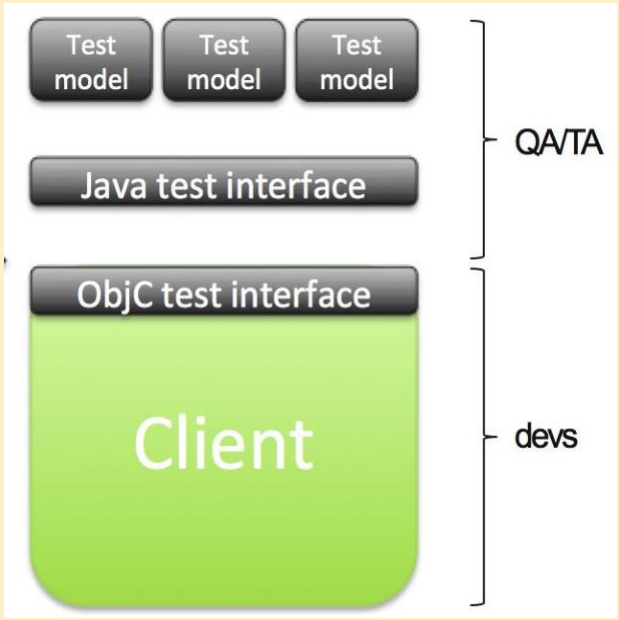




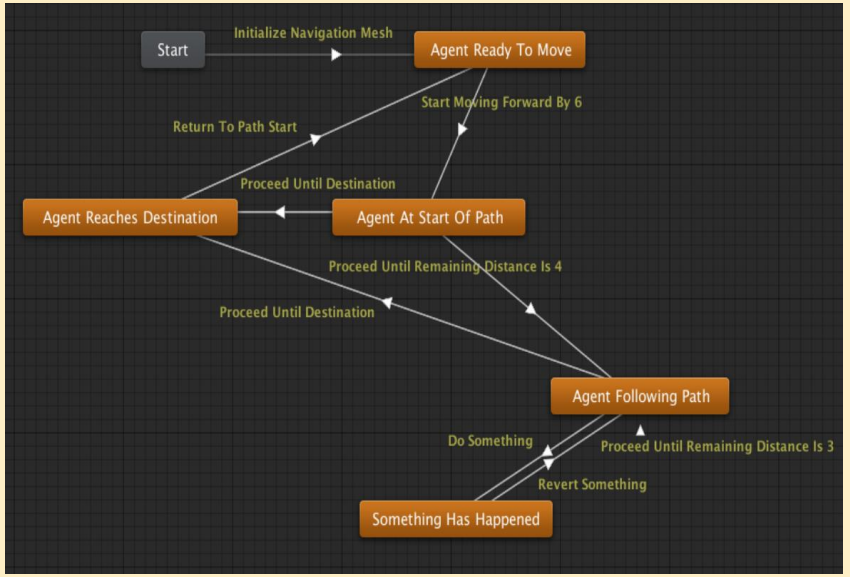
Bővebben: W. Grieskamp et al. „Model-based quality assurance of protocol documentation: tools and methodology," STVR, 21:55-71, 2011

Esettanulmányok: Spotify és Unity

- GraphWalker



- Spec Explorer és Graph Walker



„Cheat sheet” MBT bevezetéshez

Forrás: Robert V. Binder (http://robertvbinder.com/)

Javasolt	Nem javasolt
Komplex SUT működés	Egyszerű funkcionalitás
Absztrahálható követelmények	Szubjektív kiértékelés
Tesztelhető interfészek	Monolitikus GUI
Szükséges regressziós tesztelés és a tesztek karbantartása költséges	SUT nem túl értékes, már nem fejlesztett
Képzett tesztmérnökök	Kevés/nincs meglévő tesztelés
	Nem technikai tesztelő csapat

Összefoglalás

- Modell alapú tesztgenerálás
  - Fedési kritériumok teljesítéséhez
    - Vezérlés-orientált: állapotok, átmenetek fedése
    - Adatfolyam-orientált: értékadás és használat fedése
  - Specifikációval nem konform viselkedés kimutatásához
    - k-ekvivalencia reláció szerint
    - IOCO reláció szerint
- Algoritmusok és eszközök
  - Direkt (gráfelméleti) algoritmusok: Bejárás generálása
  - Modellellenőrzők: Bejárás mint ellenpélda generálása
  - SAT megoldók: Helyettesítési értékek generálása
  - Tervkészítő algoritmusok: Cél-orientált bejárás
  - Kényszer-kielégítők: Bejárási feltételek teljesítése
  - Evolúciós algoritmusok: Intelligens keresés