Remo

Modellezés és metamodellezés

Def: Modell: Egy valós vagy egy hipotetikus világ a rendszer egy részének egyszerűsített képe, amely a rendszert helyettesíti bizonyos megfontolásokban. Egy modell elkészítésének mindig egy kérdés megválaszolása a célja.

A modell előnyei:
↳ Az eredeti résznél kisebb: A problémához elhanyagolható információk nincsenek benne
→ Az eredeti résznél áttekinthetőbb: Csak a problémához releváns infók szerepelnek benne

Def: Diagram: A modell egy nézete, amely a modell bizonyos aspektusait grafikusan ábrázolja

A modellek precíz meghatározásához szükségünk van egy modellezési nyelvre.

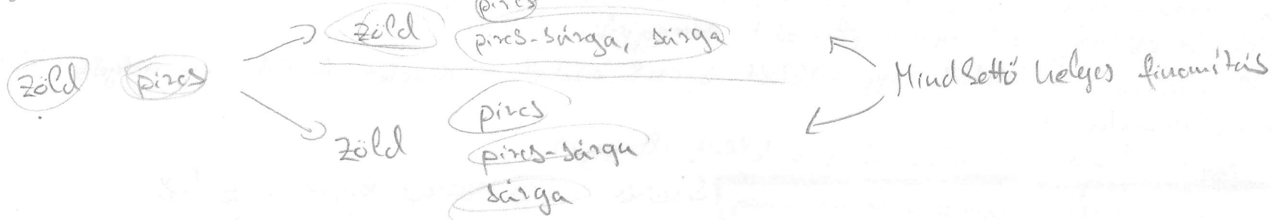
Def: Modellezési nyelv: Négy fő részből áll:
↳ Absztrakt szintaxis (metamodell): Meghatározza hogy a nyelvnek milyen típusú elemei vannak, azok milyen kapcsolatban állnak egymással, és a típusoknak mi a viszonya egymáshoz. Ezt a reprezentációt használják a gépek a modell belső tárolására
→ Konkrét szintaxis: Az absztrakt szintaxis elemtípusokhoz és kapcsolatokhoz szöveges vagy grafikus jelölésrendszert definiál. Ezt a reprezentációt használják az emberek a modell olvasására s szerkesztésére. Egy nyelvhez több konkrét szintaxis is adható.
→ Jólformáltsági kényszer: Megadja, hogy egy érvényes modellnek milyen egyéb követelményeknek kell megfelelnie.
→ Szemantika: Az absztrakt szintaxis által megadott nyelvi elemek jelentését definiáló szabályrendszer. A szemantika megadja, hogy az így kért modell pontosan mit jelent, vagy hogyan működik.

Def: Zárt világ feltételezés: Minden állítás amiről nem ismert, hogy igaz, hamis. ∄ ismeretlen.
Def: Nyílt világ feltételezés: Egy állítás annak ellenére is lehet igaz, hogy ez a tény nem ismert. ∃ ismeretlen

Def: Rendszer és környezete: A környezet a rendszerre ható tényezők összessége. Modellezéskor a modellezett rendszernek mindig egyértelműen definiálni kell a határait, ami ezen belül esik az a rendszer a maradék a környezet. A környezet elemeit szokás két csoportba sorolni: releváns s irreleváns környezeti elemek.

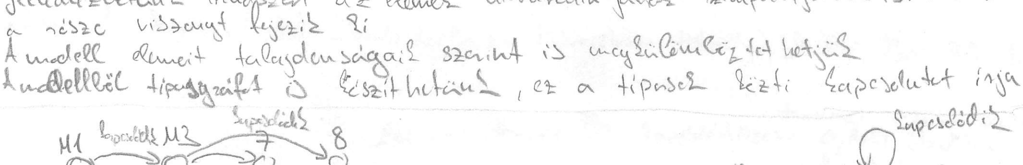
Def: Finomítás: A modell olyan részletezése (pontosítása), hogy a környezet szempontjából nézve a finomított modell (valamilyen tekintetben) helyettesíteni tudja az eredeti modellt.
Def: Absztrakció: A finomítás inverz művelete, vagyis a modell részletességének csökkentése.

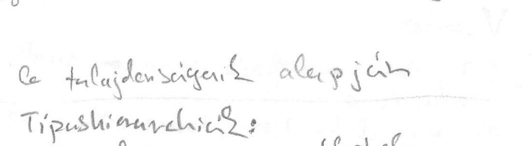
Egy finomítási lépésnek sokféle eredménye lehet, de egy absztrakciós lépésnek csak egy. Az absztrakció egyértelmű, a finomítás nem az.



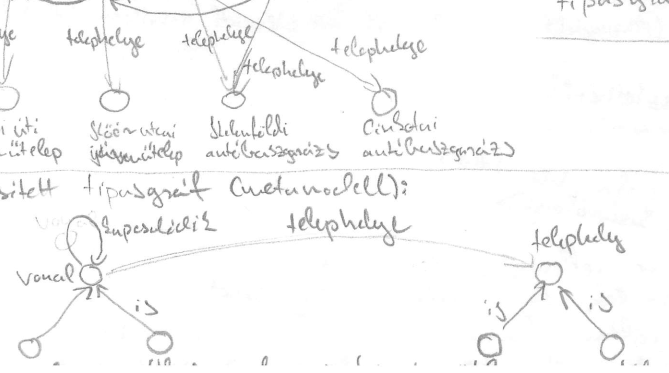
Struktúra alapú modellezés

Egy rendszert gyakran úgy jellemezhetünk legjobban, ha bizonyos elemeit megkülönböztetjük és leírjuk az ezek közötti kapcsolatokat. Ezt gráffal tehetjük meg
Jellemezhetjük rendszert az elemek hierarchiájának szempontjából is. Ehhez egy fát használhatunk. Ekkor a gráf élei a rész viszonyt fejezik ki
A modell elemeit tulajdonságaik szerint is megkülönböztethetjük
A modellből típusgráfot is készíthetünk, ez a típusok közti kapcsolatot írja le tulajdonságaik alapján





összesített típusgráf (metamodell):



A strukturális modellezés célja hogy a rendszer felépítését jellemezze, beleértve egyes elemek típusát a köztük lévő kapcsolatokat s hierarchiát, illetve az elemek tulajdonságait.

Def: A strukturális modell: A rendszer felépítésére vonatkozó tudás. A strukturális modell a rendszer alkotórészeire, ezek tulajdonságaira és egymással való viszonyaira vonatkozó statikus (pillanatnyi) tudást reprezentál.

Def: A jellemző: Egy a modell által megadott parciális függvény amelyet a modellelemeken értelmezünk.

jellemzők	azonosító	funkció	kapacitás	vágányhossz	max. üzemanyag
jellemzők értéke	T1	metró járműtelep	24	8500 m	—
	T2	metró járműtelep	60	16512 m	—
	T3	autóbusz garázs	265	—	250000 liter
	T4	autóbusz garázs	322	—	200 000 liter

Az elemeket az azonosító jellemzőjükkel különböztetjük meg.
Egy adott f jellemzőhöz tartozó függvény $f(id) \rightarrow v$ ahol id egy elem azonosítója, v az értékkészlet egy eleme.
Típustól függően lehetnek olyan jellemzők melyeket csak bizonyos típusokra értelmezünk.

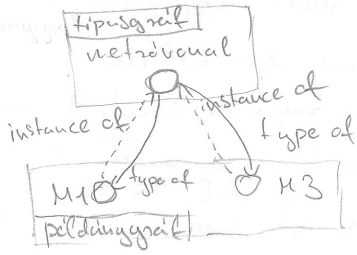
Def: Típus: Egy kitüntetett jellemző, amely meghatározza, hogy milyen más jellemzők lehetnek értelmezettek az adott modellelemre, illetve milyen más modellelemekkel lehet kapcsolatban. A többi jellemzőt tulajdonságnak hívjuk.

Def: Egy adott t típus példányainak nevezzük azokat a modellelemeket, amelyek típusa t.

A típusok meghatározzák, hogy az egyes csomópontok milyen élekkel köthetők össze.

Def: Típusgráf: Egy olyan gráf amelyben minden csomóponttípushoz egy típus csomópont, minden éltípushoz egy típusél tartozik.

Def: Egy adott típusgráf példánygráfja: Egy olyan gráfmodell melynek elemei a típusgráf csomópont- és éltípusainak példányai, valamint minden él forrása és célja rendre az éltípus forrásának és céljának példánya.



Def: Egy rendszer metamodellje: Tartalmazza a típusgráfot, az egyes típusok közti kapcsolatokat, s további megkötéseket is.

Egy rendszer hierarchiája a rendszer dekompozíciójával állítható elő.

Def: Dekompozíció (factoring): Egy rendszer kisebb komponensekre bontása, melyek könnyebben érthetők, fejleszthetők és karbantarthatók.

Def: Egy dekompozíció helyes, ha a dekompozícióval kapott rendszer minden elemének megfeleltethető az eredeti rendszer valamelyik eleme, és az eredeti rendszer minden eleméhez hozzárendelhető a dekompozícióval kapott rendszer egy vagy több eleme.

A struktúra modellekből különböző nézeteket állíthatunk elő.
A tulajdonságmodellben leggyakrabban használt műveletek a szűrés s vetítés. Ezek során úgy absztraháljuk a modellt, hogy bizonyos modellelemeket és/vagy azok egyes jellemzőit elhagyjuk.

Def: Szűrés: A modell elemein kiértékelünk egy feltételt s csak azokat az elemeket tartjuk meg melyekre teljesül.
A gráfmodell szűrése egy részgráfot állít elő.

Def: Vetítés: A modell egyes jellemzőit kiválasztjuk s a többit elhagyjuk.

azonosító	funkció	kapacitás	vágányhossz	max. üzemanyag
T1	metró járműtelep	24	8500 m	
T2	metró járműtelep	60	16512 m	
T3	autóbusz garázs	265		250000 liter
T4	autóbusz garázs	322		200000 liter

S szűrés

V vetítés

S szűrés: kapacitás ≤ 100
V vetítés: azonosító, funkció, kapacitás

Def: Top down modellezés során a rendszert felülről lefelé (összetett rendszertől az alkotóelemek felé) építjük. A modellezés alaplépése a dekompozíció.

Jellemzői: ⊕ Rendszer funkciójának szerepe már ismert
⊖ „Félidőben" még nincsenek működő (teljes részletességgel elkészített) részek.
⊖ Részek problémái, igényei későn derülnek ki

Def: Bottom up modellezés során a rendszert alulról felfelé (elszigetelt alkotóelemektől az összetett rendszer felé) építjük. A modellezés alaplépése a kompozíció.

Jellemzői: ⊕ A rendszer részei önmagukban kipróbálhatóak, tesztelhetők
⊕ "Félidőben" könnyebben előállítható a rendszer prototípusa
⊖ Nem látszik előre a rész szerepe az egészben

Állapotalapú modellezés

Elsőként meg kell határozni a rendszer állapotterét -> csak akkor ha teljes s kizárólagos

Def: Teljesség: Minden időpontban az állapottér legalább egy eleme jellemzi a rendszert

Def: Kizárólagosság: Minden időpontban az állapottér legfeljebb egy eleme jellemzi a rendszert.

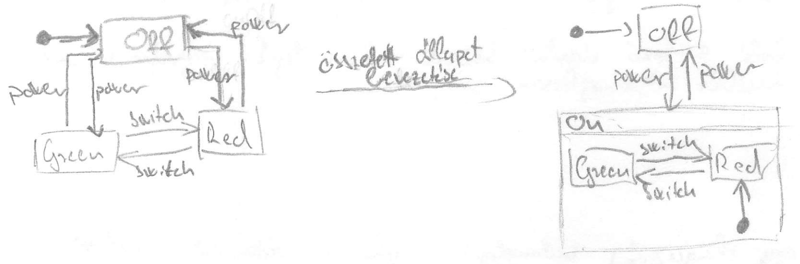
Egy állapotátmenet segítségével tud a rendszer állapotot váltani forrás s célállapot közt.

Def: Determinisztikus: Az állapotgépnek legfeljebb egy kezdőállapota van, valamint bármely állapotban bármely bemeneti eseményre max 1 állapotátmenetet vált ki.

Def: Teljesen specifikált: Az állapotgépnek legalább egy kezdőállapota van, valamint bármely állapotban bármely bemeneti eseményre min 1 állapotátmenetet vált ki.

Egy rendszer viselkedését annak végrehajtási szekvenciái jellemzik. Ez állapotok s események alternáló sorozata. Egy állapot elérhető, ha a rendszernek létezik véges végrehajtási szekvenciája az állapotba.

Az állapotgépek is szervezhetőek hierarchikusan összetett állapotok bevezetésével



Az összetett állapot szintaktikailag megfelel egy egyszerű állapotnak, de saját régióval rendelkezik.
Minden régió tartalmaz egy belső állapotmodellt.
Az állapotkonfiguráció az állapotok egy olyan nem bővíthető halmaza, melyek egyszerre lehetnek aktívak a rendszerben
Jelen példában: {Off}
{On, Green}
{On, Red}

Tartalmazó s tartalmazott állapotok közt nem érvényesül a kizárólagosság. Hierarchikus állapot bevezetésével többféle érvényes állapottér adódik

Mindig egyszerre lépés a komponensek / régiók

Mindig 1 komponens régió lép

Potenciális állapottér olyan állapottér, mely a valódi állapottereket alkalmazza

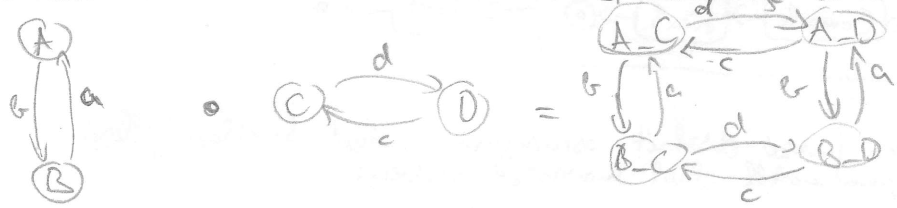
Állapotgépeknél:
Aszinkron szorzat: Mindent mindennel
Szinkron: Csak az értelmezett
Vegyes: Both

Komponensekre való értés: Egy absztrakciós művelet mely a szorzat állapotteréből egy, vagy több komponenst elhagy (legalább egy kivételével)

Jelen példában: {Off, On} ①
{Off, Green, Red} ②
{{Off}, {On, Green}, {On, Red}} ③

A ② állapottér az ①-beli On-t finomítja Greennel s Reddel.
Az ① állapottér a ②-beli Greent s Redet absztrahálja az Onnal

Állapotmodell rendelkezhet belső változóval is, ekkor egy pillanatnyi rendszerállapotot nem csak az adott állapot csomópont határozza meg hanem az éppen érvényes változóértékek is.
Őrfeltétellel biztosíthatjuk, hogy egy állapotátmenet csak a megfelelő változó értékkel történhessen meg.
Példa: A —x<3→ B
Változó értékét csak utasítással lehet változtatni, mely egy tranzícióhoz kapcsolható akció
Az interfészváltozó olyan változó amellyel az állapotgép kommunikálhat a külvilággal.
Állapotgépeket tudunk szorozni, ekkor minden lehetséges közös állapotot fel kell venni s a köztük lévő kapcsolatokat is. Ezen művelet neve az aszinkron szorzás, eredménye szorzatautomata



Predikátum absztrakció:
Folytonos változó értékészletek absztrakciója predikátumok szerint
Pl.: Gát vízszintje 0-40: vízszint rendben
40-60: kritikus
60+: túlcsordulás

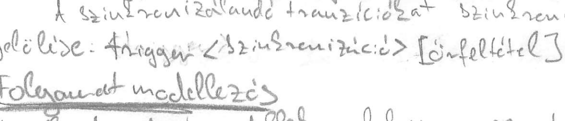
Ha állapottér szorzás után túl sok állapot keletkezne (pl 4 állapotú • 5 állapotú modell = $4^5 = 1024$ állapotú modell, ezt állapottér robbanásnak nevezzük) akkor alkalmazhatunk ortogonális állapotot is.
Az ortogonális állapot olyan összetett állapot mely több régióval rendelkezik.
Az ortogonális régiók aszinkron működnek: a tranzíciók külön-külön vezethetőek be.
Ortogonális dekompozíció során az állapot konfiguráció fogalma is összetettebbé válik: minden aktív régiónak egy állapota lesz aktív

Def: Állapotgépek szinkronizációja: két kooperáló állapotgépben bizonyos állapotátmenetek csak egyszerre történhetnek meg.
A szinkronizálandó tranzíciókat szinkronizációs címkével jelöljük meg.
Jelölése: trigger <szinkronizáció> [őrfeltétel] / akció

Folyamat modellezés

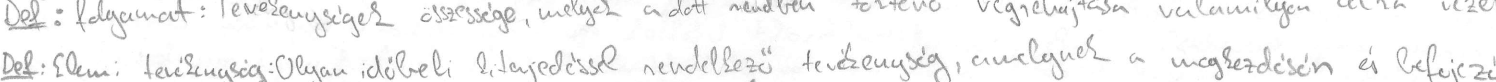
Az állapot alapú modellek leírja az hogy miként változhat a rendszer.
A folyamatmodellek arra fókuszálnak hogy mit csinál egy rendszer. A folyamatmodellezés célja, hogy a rendszer folyamatát leírja.
Def: folyamat: Tevékenységek összessége, melyek adott rendben történő végrehajtása valamilyen célra vezet.

Def: Elemi tevékenység: Olyan időbeli kiterjedéssel rendelkező tevékenység, amelynek a megkezdésén és befejezésén túl további részleteit nem modellezzük



Minden elemi tevékenység leírható egy három állapotú állapottérrel:
{nem kezdődött el, folyamatban van, már befejeződött}

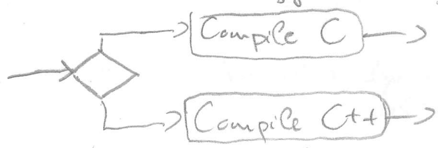
Def: Szekvencia: A tevékenységek szigorú végrehajtási sorrendjét definiálja



{még nem kezdődött el egyik tevékenység sem, Compile folyamatban van, Compile befejeződött, Link még nem kezdődött el, Link folyamatban van, befejeződött mindkét tevékenység}

A folyamat egy konkrét lefutását folyamatpéldánynak nevezzük

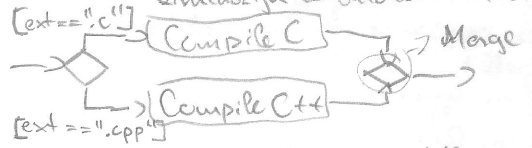
Def: Vezérlési elem: Olyan csomópont a folyamatban, mely a folyamatmodell tevékenységei közül választ ki egyet vagy többet végrehajtásra



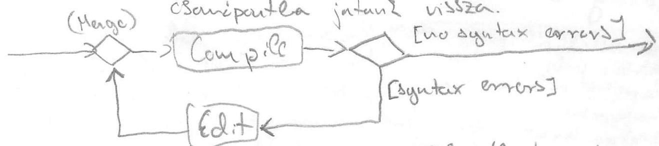
Bármely ágat is választjuk a folyamat modell egy érvényes lefutását kapjuk.

Def: Döntési csomópont: Olyan csomópont a folyamatban amely a belé érkező egyetlen vezérlési él hatására a belőle kiinduló ágak közül pontosan egyet választ ki végrehajtásra, az esetleges őrfeltételekkel összhangban.

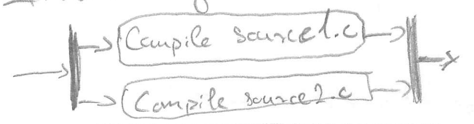
Def: Merge: Olyan csomópont a folyamatban mely a belé érkező ágak közül akármelyik végrehajtásakor kiválasztja a belőle kiinduló vezérlési élet további végrehajtásra



Def: Ciklus: Olyan folyamatmodell részlet, amelyben egy elágazás valamelyik ágain az elágazást megelőző merge csomóponthoz jutunk vissza.

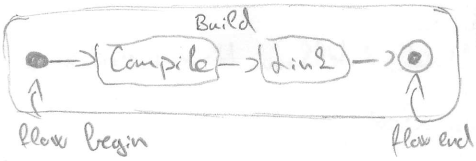


Def: Két esemény konkurens, ha bekövetkezési sorrendjükre nézve nincsen megkötés



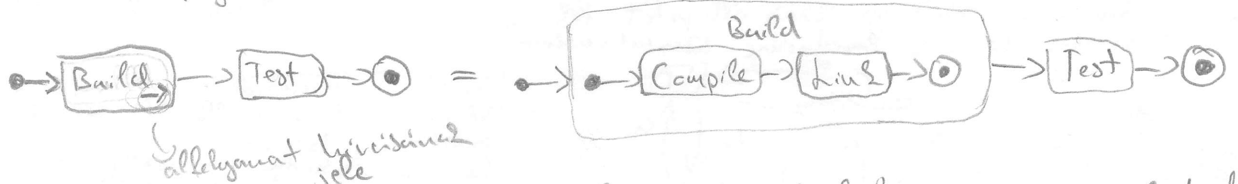
Def: Fork: Olyan csomópont a folyamatban, mely a belé érkező egyetlen vezérlési él hatására a belőle kiinduló összes párhuzamos folyamatot kiválasztja

Def: Join: Olyan csomópont a folyamatban, mely a belé érkező összes párhuzamos folyam végrehajtása után kiválasztja a belőle kiinduló egyetlen vezérlési élet további végrehajtásra



Def: Minden folyamat egy flow begin csomópont vezérlési elemmel indul, és egy flow end csomópont elemmel fejeződik be. A flow begin a folyamat kiindulását jelző elem pontosan egy kimenettel. A flow end a folyamat befejezését jelző elem, melynek pontosan egy bemenete van.

Hierarchikus folyamat modell:



Def: Folyamat példány: Olyan adott folyamatmodellhez tartozó diszkrét eseménysor amelyet következő jellegű események alkotják a folyamatmodell által megszabott időben:
- → folyamat kezdete
- → folyamatot alkotó egyik tevékenység kezdete
- → folyamatot alkotó egyik tevékenység vége
- → folyamat vége

Def: Egy program által meghatározott folyamatmodellt, amely az elvégzendő lépéseket s sorrendjüket írja le, a program vezérlési folyamának nevezzük

Def: Azon programokat melyek vezérlési folyamat határoznak meg imperatív programoknak nevezzük.

Def: A vezérlési folyamhoz tartozó ciklomatikus komplexitás: $M=E-N+2$ ahol E a vezérlési élek, N a vezérlési csomópontok és tevékenységek száma

Def: Azon (egy be s kilépési ponttal) részfolyamatokat tekintjük jólstrukturált blokknak/részfolyamatnak melyek:
- egyetlen elemi tevékenység önmagában
- egyetlen hívás
- üres vezérlési szakasz
- „szekvencia kapcsolás": a $P_1, P_2 \dots P_n$ jólstrukturált blokkok szekvenciája.
- „fork-join kapcsolás": a $P_1, P_2 \dots P_n$ jólstrukturált blokkok egy n-ágú fork és egy n-ágú join közé zárva.
- „decision-merge kapcsolás": a $P_1, P_2 \dots P_n$ jólstrukturált blokkok egy n-ágú decision és egy n-ágú merge közé zárva.
- „Ciklus": egy kétágú merge-l kezdődik amely után egy jólstrukturált P_1 blokk következik, majd egy kétágú decision melynek egyik ága a részfolyamat vége, a másik a P_2 jólstrukturált blokkon keresztül az előbbi mergebe tér vissza.

A teljes folyamatmodell jólstrukturált, ha egyetlen kilépési s belépési pontja egy jólstrukturált blokkot zár közre.

Teljesítmény modellezés

A teljes rendszer belé érkező felhasználói kéréseket tranzakcióknak hívjuk, a rendszer által alrendszereknek továbbított feladatrészeket pedig kéréseknek.

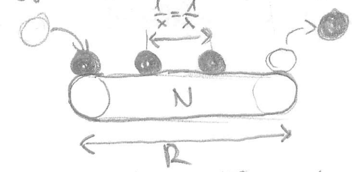
Def: Érkezési ráta: A vizsgált rendszer határához egységnyi idő alatt érkező felhasználói kérések átlagos száma. Mértékegysége $\frac{db}{s}$. Jele: λ.

Def: Átbocsátás: A vizsgált rendszert egységnyi idő alatt elhagyó feldolgozott felhasználói kérések átlagos száma. Mértékegysége: $\frac{db}{s}$. Jele: X.

Def: Válaszidő: A felhasználói kérések által a rendszer határain belül töltött átlagos idő. Mértékegysége: s. Jele: R.

Def: A Rendszerben lévő kérések átlagos száma: A Rendszerben lévő kérések átlagos száma. Mértékegysége: db. Jele: N.

Egyensúlyi állapotban igaz a Little törvény: $N = X \cdot R$



Def: Látogatások átlagos száma: Megadja, hogy egy tranzakció átlagosan hány kérést generál az i. alrendszer (erőforrás) felé. Mértékegysége: $\frac{k}{tr}$ $\left(\frac{\text{kérés}}{\text{tranzakció}}\right)$. Jele: V_i.

Def: Szolgáltatásigény: Megadja, hogy egy tranzakció átlagosan mennyi ideig használja az adott alrendszert (erőforrást). Mértékegysége: $\frac{s}{tr}$ Jele: D_i

Def: Erőforrásigény: Megmondja, hogy egy kérés átlagosan mennyi ideig használja az adott alrendszert (erőforrást). Mértékegysége $\frac{s}{k}$ Jele: S_i

A két fenti fogalom közti kapcsolat: $D_i = V_i \cdot S_i$

Az erőforrások szempontjából fontos tulajdonság az átlagos kihasználtság. Jele: U

Az erőforrás felé érkező kérések számát a Forced Flow törvénnyel tudjuk kiszámolni: $X_i = V_i \cdot X_0$ (a rendszer felé érkező össz kérések)

Így az erőforrásra felírt Little törvény: $N_i = S_i \cdot X_i$

Kihasználtság törvénye: $U_i = \frac{N_i}{n_i} = \frac{S_i \cdot X_i}{n_i}$ ahol n_i az erőforrásban max tartózkodható kérések száma $\quad U = \frac{X}{X^{max}}$

Általában egy erőforrásban egy kérés tartózkodhat, ekkor n_i az adott típusú erőforrások száma.

Az erőforrásrészlet felső határt szab az egységnyi idő alatt kiszolgálható kéréseknek, ezt a felső határt hívjuk átbocsájtóképességnek

Ha feltesszük, hogy $U_i^{max} = 1 \Rightarrow X_i^{max} = n_i \cdot \frac{U_i^{max}}{S_i} = n_i \cdot \frac{1}{S_i}$

Azt az erőforrást amely a teljes rendszer átbocsájtóképességét korlátozza szűk keresztmetszetnek hívjuk.

Ekkor $X_0^{max} = \min_i \left(\frac{X_i^{max}}{V_i} \right)$

A szolgáltatásigény törvénye egy adott erőforrásra vonatkozó szolgáltatásigény meghatározását teszi lehetővé $n_i = 1$ esetben.

$$D_i = V_i \cdot S_i = \frac{X_i}{X_0} \cdot n_i \frac{U_i}{X_i} = \frac{U_i}{X_0}$$

Az alábbi fogalmakat mérések/szimulációk során használjuk, a mérések/szimulációkban egyensúlyi helyzetet vizsgálunk tehát $\lambda = X_0$

Def: Mérési idő: Mérési idő. Mértékegysége s. Jele: T

Def: Tranzakciók száma: A mérési idő alatt elvégzett tranzakciók száma. Mértékegység: tr. Jele: C_0.

Def: Foglaltsági idő: Az egyes erőforrások foglaltsági ideje a mért idő tartamon belül. Mértékegység: s. Jele: B_i

Ekkor: $X_0 = \frac{C_0}{T}$

$D_i = \frac{B_i}{C_0}$

$U_i = \frac{B_i}{T}$ (ha $n_i = 1$)

$U_i = \frac{B_i}{T} = \frac{B_i}{\frac{C_0}{X_0}} = \frac{B_i}{C_0} \cdot X_0 = D_i \cdot X_0$ (ha $n_i = 1$)

Modellek ellenőrzése

Def: Funkcionális követelmény: Egy rendszer által ellátott funkciót definiál

Def: Nemfunkcionális követelmény: A funkcionális követelményeken kívül eső, rendszer minőségére vonatkozó követelmények, pl.: megbízhatóság, teljesítmény.

Def: Biztonsági követelmény: Megadja, hogy milyen viselkedés engedélyezett, s mely tiltott.

Def: Élőségi követelmény: Elvárt viselkedést definiál. A rendszer előbb-utóbb teljesíteni képes bizonyos elvárásokat.

Egy fontos rendszertől független követelmény a holtpontmentesség

Def: Holtpont: Olyan állapota a rendszernek ahol a végrehajtás megáll, a rendszer nem képes többé állapotot váltani.

Holtpontra gyakori példa amikor két alfolyamat egymásra vár.

Holtponthoz hasonló a livelock: amikor a rendszer egy hasztalan végtelen ciklusba ragad.

Def: Verifikáció: Amikor azt vizsgáljuk, hogy az implementáció megfelel-e a specifikációnak.

Def: Validáció: Amikor a rendszert a felhasználói elvárásokhoz hasonlítjuk

Def: Statikus ellenőrzés: Olyankor a vizsgált rendszert végrehajtása/szimulációja nélkül elemezzük

Kétféle statikus hibát lehet különböztetni: szintaktikai és szemantikai. A szintaktikai hiba során a modell nem felel meg a metamodelljének. Szemantikai hiba során a rendszer valószínűleg nem értelmes, vagy nem elvárt módon fog viselkedni, pl.: C-ben: $x = y/0$;

A jól konstruált folyamatmodellek sok szemantikai hiba elkerülését elősegítik

Def: Tesztelés: Olyan tevékenység amely során a rendszert bizonyos meghatározott körülmények között futtatjuk majd az eredményeket összehasonlítjuk az elvárásainkkal. A tesztelés célja a hibák azonosítása s ezáltal a rendszer javítása/minőségének felmérése.

A tesztelés néhány alapfogalma

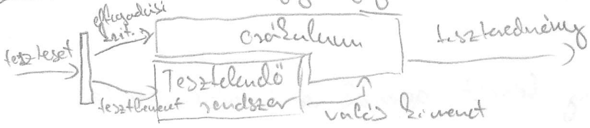
Def: Tesztelendő rendszer: A rendszer amit a teszt során vizsgálat céljából futtatni fogunk

Def: Tesztbemenetek: A tesztelendő rendszer számára biztosítandó bemeneti adatok.

Def: Teszteset: Azon adatok összessége melyek egy adott teszt futtatásához s értékeléséhez kellenek. Bemeneti értékek, végrehajtási előfeltételek, elvárt eredmények (elfogadási kritérium) és végrehajtási utófeltételek halmaza

Def: Tesztkészlet: Tesztesetek egy adott halmaza

Def: Tesztfuttatás: Egy vagy több teszteset végrehajtása.



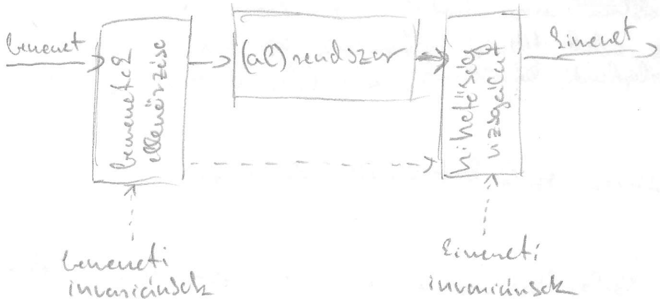
A tesztkészlet egy fontos metrikája, hogy az adott modellt mennyire fedi:

- Állapotgépben: - Állapot fedettség egy adott tesztkészlet által érintett állapotok s az összes állapot aránya
  - Átmenet fedettség egy adott tesztkészlet által érintett átmenetek s az összes átmenet aránya
- Vezérlési folyamban: - Utasításfedettség egy adott tesztkészlet által érintett utasítások s az összes utasítás aránya

Egy magas fedettségi arány szükséges de nem elégséges feltétele egy jó minőségű rendszer fejlesztésének

Ha rendszerünk külső komponensektől kapja a bemenetét, akkor ezt is ellenőriznünk kell, futásidőben. Azokat a kritériumokat melyeknek teljesülését minden esetben elvárjuk invariánsoknak nevezzük.

(Ezt monitorozásnak hívjuk)



A monitorozás két fő lépésből áll:

- bemenetek ellenőrzéséből, amely során a bemeneti adatok megfelelőségét vizsgáljuk a definiált bemeneti invariánsok alapján. Pl.: alrendszer: másodfokú egyenlet megoldó fgv
  bemeneti invariáns: $D \geq 0$
- hihetőségvizsgálat, amely során a kimeneti adatok megfelelőségét vizsgáljuk a bemeneti adatok s a kimeneti invariánsok alapján. Pl.: alrendszer: négyzetre emelő fgv
  kimeneti invariáns: eredmény ≥ 0

Ha a bemenet ellenőrzése során van hiba akkor annak oka a rendszer helytelen használata

Ha a hihetőség vizsgálat során van hiba akkor annak oka vagy hibás implementáció vagy futásidejű hiba