INFORMATIKAI RENDSZEREK MEGBÍZHATÓSÁGÁNAK MATEMATIKAI MODELLEZÉSE

Kun István

INFORMATIKAI RENDSZEREK HIBÁINAK TIPIKUS OKAI

Tranzakciós hiba

Az összes tranzakciók (adatváltoztatás, adatmozgatás) mintegy 3 %-ában fordul elő. (Interneten található adat, soknak látszik, de vélhetően van némi valóságalapja.) Szoftverhibának tekintendő.

Rendszer hardverhibája

Processzor, központi memória, stb. menet közbeni meghibásodása, vagy áramkimaradás.

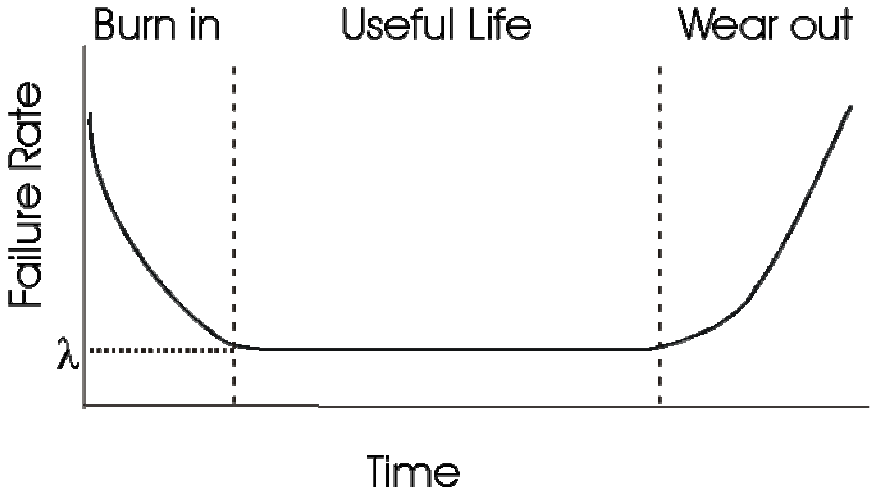
Háttértároló meghibásodása

Író-olvasó-fej, kontroller vagy adattároló közeg meghibásodása.

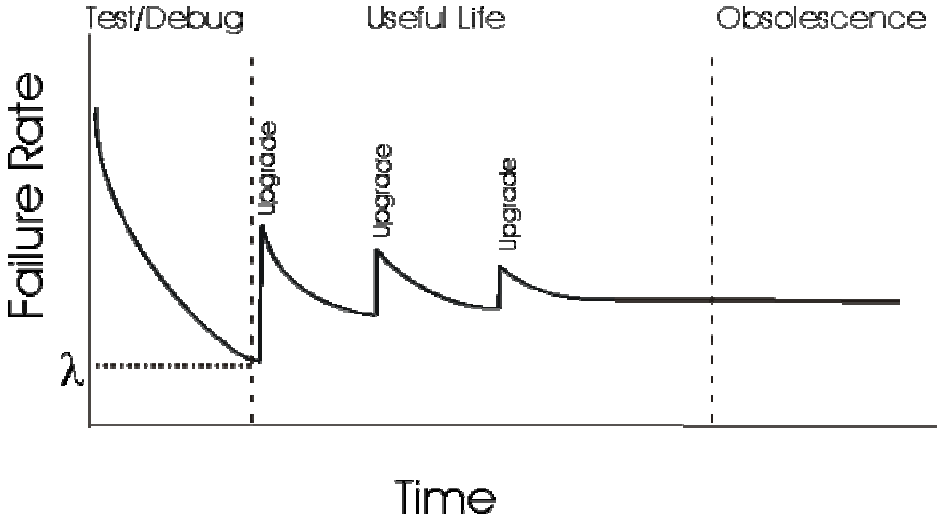
A HARDVER- ÉS SZOFTVER HIBÁK JELLEMZŐ KÜLÖNBSÉGEI

- **A szoftver nem öregszik, a hardver igen.**
- **A hiba nem lép fel, ha a szoftver nincs használatban. A hardver esetében viszont az öregedés ilyenkor is meghibásodást okozhat, ez persze csak az újabb bekapcsoláskor mutatkozik meg.**
- **A szoftverhibákat hibás logika vagy hibás input okozza. A hardverhibákat anyagkifáradás, véletlen külső okok, tervezési hibák, helytelen használat és környezeti tényezők okozhatják.**
- **A szoftverhibákat nem előzik meg figyelmeztető jelek, a hardverhibákat általában igen.**
- **A szoftver tesztelése lényegében egy soha véget nem érő folyamat, a hardvert viszont lehet kimerítően tesztelni.**
- **A működésben levővel azonos részegység hideg vagy meleg tartalékolása a hardver megbízhatóságát növeli, a szoftverét nem.**

A HARDVER ÉS SZOFTVER MEGBÍZHATÓSÁGÁNAK HOSSZÚTÁVÚ ALAKULÁSA



A hardver meghibásodási rátája az élettartam során kádgörbe szerint változik.



A szoftver meghibásodási rátája farkasfog-szerű: az upgrade ciklus elején megnő, a ciklus végére lecsökken.

KRITIKUS SZOFTVERHIBÁK

MI AZ ELMÉLET?

A gondosan tesztelt szoftver soha nem hibásodik meg, mert amíg a számítógép működik, a szoftver nem változik, nem öregedik.

ÉS MI A VALÓSÁG???

Néhány példa:

- **Falklandi háború (1982): a Sheffield rombolót azért süllyesztette el egy Exocet rakéta, mert a védelmi radarrendszer szoftvere a rakétát tévesen „barátinak” minősítette.**
- **Öbölháború (1991): egy Patriot ellenrakéta irányítórendszere 10 másodpercenként 0.000000095 sec-ot tévedett, és az akkumulálódó hibák miatt 100 órás működés után nem találta el a támadó rakétát (28 halott)**
- **Ariane 5 európai űrrakéta: az Ariane 4 jól működő szoftverében egy apró módosítást végeztek, és az Ariane 5-ben indításkor tűz keletkezett.**
- **USA, Kalifornia állam: egy helyi telefonközpontban egy kezelőprogramban 3 sort kicseréltek, és a teljes szolgáltatás megbénult.**

MATEMATIKAI MODELLTÍPUSOK

- **Metrikák**
- **Prediktív statisztikai modellek**

METRIKÁK (NASA SZOFTVERHIBA-ADATBÁZISBÓL)

A metrika valamilyen számszerű mérési eljárással a szoftverhez rendelt függvényértéket jelent, ahol a független változók a szoftver mért adatai.

- **Forráskódhossz-alapú metrikák**
- **Hibaszám-metrikák**
- **Stílus-metrikák (szemantikai elvárásokat – pl. egyértelműség – fogalmaznak meg)**
- **Komplexitási metrikák**

A szoftver tesztelésének, utólagos karbantartásának és módosításának költsége nagyrészben a program strukturális bonyolultságának függvénye. Egy jól használható szoftver-bonyolultsági mérték már a fejlesztési fázisban felfedheti a szoftver kritikus pontjait, elősegítheti az áttekinthetőbb, módosítható és újrafelhasználható kód készítését, és becsléseket adhat a fejlesztési folyamat várható költségeire.

HIBASZÁM-METRIKÁK

- **Működési zavarok száma**
- **Ezer programsorra jutó működési zavarok száma**
- **Ezer nem-megjegyzés programsorra jutó működési zavarok száma**
- **Egy felhasználóra jutó működési zavarok száma**
- **Ezer tesztelési órára jutó működési zavarok száma**
- **Ezer tesztelési órára jutó működési zavarok száma fedettségi mérőszámmal**

KOMPLEXITÁSI METRIKÁK

A komplexitási metrikák egyaránt lehetnek termékjellemzők (az elkészült szoftver statikus tulajdonságai) és folyamatjellemzők (a szoftverkészítési folyamat által felhasznált idő és munkamennyiség).

- **Objektumorientált metrikák**
- **McCabe-féle ciklomatikus komplexitási metrikák (gráfelméleti alapú)**
- **Halstead-féle komplexitási metrikák (információelméleti alapú)**
- **Henry-Kafura-féle információáramlási metrikák (eljárás külső kapcsolatai)**

A Unix operációs rendszeren bizonyították, hogy a McCabe- és a Halstead-komplexitás erősen korrelál egymással és a hibaszámmal is, míg a Henry-Kafura komplexitás statisztikailag független tőlük.

A HALSTEAD-FÉLE KOMPLEXITÁSI METRIKA

A következő alapmennyiségekre van szükség:
n_1 — a különböző operátorok (eljárások, függvények, műveletek) száma;
n_2 — a különböző operandusok száma;
N_1 — az operátorok összesített előfordulási száma;
N_2 — az operandusok összesített előfordulási száma.

Az alapmennyiségekből lehet kiszámítani a komplexitási metrikákat:
A program szókincse
Képlete: $n = n_1 + n_2$
Jelentése: a programban található különböző azonosítók száma.
A program hossza
Képlete: $N = N_1 + N_2$
Jelentése: a program azonosítói összes előfordulásainak együttes száma.
A program térfogata
Képlete: $V = N \log_2 n = \mathrm{ld}\, n^N$
Jelentése: a program kódolásához szükséges bitek száma.

A tervezés szakaszában jó előrejelzést adhat a program bonyolultsága miatt várható hibák számáról.

A MCCABE-FÉLE CIKLOMATIKUS KOMPLEXITÁSI METRIKA

Rajzoljuk fel egy program *G* vezérlésfolyam-gráfját a következő módon: a gráf csúcsai az utasítás-szekvenciáknak (elágazás nélküli utasítás-sorozatoknak) felelnek meg, élei pedig a program által a szekvenciák között létrehozott lehetséges közvetlen átmeneteknek. Ekkor felírhatjuk az úgynevezett ciklomatikus mutatószámot:

$$v(G) = e - n + 2$$

ahol
***v* – a ciklomatikus komplexitás**
***e* – a gráf éleinek száma**
***n* – a gráf csúcsainak száma**

Alkalmazás:

Ciklomatikus komplexitás	Kockázati szint
1-10	Egyszerű program, kevés kockázat
11-20	Mérsékelten bonyolult program, mérsékelt kockázat
21-50	Bonyolult program, magas kockázat
Több mint 50	Nem tesztelhető program, nagyon magas kockázat

PREDIKTÍV STATISZTIKAI MODELLEK

Meghibásodási ráta regressziós becslése

- **Duane**
- **Jelinski-Moranda**

Adott jövőbeni időpontig megtalált programhibák számának előrebecslése

- **Musa**
- **Goel-Okumoto**

STATISZTIKAI MODELLEK - DUANE I.

A MODELL JELLEMZŐI:

- **determinisztikus**
- **nem számlálja a hibákat**
- **csak a meghibásodásokkal foglalkozik, a javítással nem**

KONCEPCIÓ

- **a komplex műszaki rendszerek bizonyos mértékig a szoftverhez hasonlóan kezelhetők**
- **az előforduló hibákat teljesen ki lehet javítani**
- **az egyszer már előfordult hibákat a felhasználó azonos formában nem ismétli meg**
- **„tanulógörbe", amely a használat során bővülő ismereteket figyelembe veszi az előrejelzésben**

JELÖLÉSEK:

u **– összes figyelembe vett programfutási idő**
***c*(*u*)– *u* idő elteltéig jelentkezett hibák száma**
***q*(*u*)– *u* idő elteltekor érvényes meghibásodási ráta**
***a*, *b* – skálaparaméterek**

STATISZTIKAI MODELLEK - DUANE II.

MATEMATIKAI MODELL:
a tapasztalatok szerint az (*u*,*q*(*u*)) koordinátájú pontok logaritmikus beosztású papíron egy negatív irányszögű egyenes közelében helyezkednek el, vagyis logaritmusaik kapcsolata lineáris regresszióval jól leírható:

$$\log[\, q(u)] = (b-1)\cdot \log(\, u) + \log(\, a)$$

ahol 0 < *b* < 1, továbbá

$$q(u) \approx \frac{c(u)}{u}$$

ezekből pedig

$$c(u) \approx a \cdot u^{b}$$

A MODELL ÉRTÉKELÉSE:

- **a programfutási idővel és az újonnan előkerülő hibákkal számolva a modell előrejelzési képessége igen jó**
- **rosszul használható abban az esetben, ha a kezdeti adatok erősen eltérnek a feltevésektől, mert ezekre nagyon érzékeny a modell**

STATISZTIKAI MODELLEK – JELINSKI-MORANDA I.

A MODELL JELLEMZŐI:

- **naptári idő**
- **sztochasztikus**
- **a folyamatosan érkező adatokat is fel tudja dolgozni**
- **a hibák száma véges**
- **a hibák csak akkor számítanak, ha megjelennek**
- **a javítás tökéletes**
- **a meghibásodási ráta csak a hibák megjelenésével változik**

JELÖLÉSEK:

T – az első programfutástól számított naptári idő
t – a legutóbbi hibától számított naptári idő
t_i – az i-1 és i sorszámú meghibásodások közti naptári idő
n – a programban található összes hibák száma
c – az adott időpontig megtalált és kijavított hibák száma
$q(c)$ – c számú hiba után érvényes meghibásodási ráta
z – skálaparaméter
Σ – összegzés 1-től c-ig

STATISZTIKAI MODELLEK – JELINSKI-MORANDA II.

MATEMATIKAI MODELL:

- **Feltételezzük, hogy**

$$q(c) = z \cdot (n - c)$$

- **vagyis a következő időegység alatt felbukkanó hibák számának várható értéke a programban maradt hibák számával arányos.**
- **Bizonyítható, hogy *n* optimális becslését kapjuk a következő egyenletből:**

$$\sum \frac{1}{n-i} = \frac{c \cdot T}{n \cdot T - \sum i \cdot t_i} \qquad \text{és} \qquad z = \frac{c}{n \cdot T - \sum i \cdot t_i}$$

A MODELL ÉRTÉKELÉSE:

- **eredeti formájában kevéssé vált be a gyakorlatban**
- **kiindulópontja több későbbi, sikeres modellnek**

STATISZTIKAI MODELLEK - MUSA I.

A MODELL JELLEMZŐI:

- **programfutási idő**
- **sztochasztikus**
- **a folyamatosan érkező adatokat is fel tudja dolgozni**
- **a hibák száma véges**
- **a hibák csak akkor számítanak, ha megjelennek**
- **a javítás tökéletes**
- **a meghibásodási ráta csak a hibák megjelenésével változik**

JELÖLÉSEK:

t	**a legutóbbi hibától számított programfutási idő**
n	**a programban található összes hibák száma**
$c(t)$	**az adott időpontig megtalált és kijavított hibák száma** $y(t)$ **–** t **futási idő elteltekor a következő meghibásodásig hátralevő idő várható értéke (a megbízhatóságelméletben szokásos jelölése: MTTF)**
$m(t)$	$c(t)$ **várható értéke**
C	**tesztkompressziós faktor; azt fejezi ki, hogy a tesztelés során hányszor gyorsabban akadunk hibára, mint normál programfutás során**

STATISZTIKAI MODELLEK - MUSA II.

MATEMATIKAI MODELL:

- **A *t* futási összidőig megtalált hibák számának várható értéke:**

$$m(t) = n \cdot \left\{ 1 - \exp\left(- \frac{C}{y(0) \cdot n} \cdot t \right) \right\}$$

- **Látható, hogy a jobboldalon a kapcsos zárójelen belüli rész *t* növekedésével 1-hez tart, így *m*(*t*) is tart *n*-hez.**
- **A legközelebbi meghibásodásig eltelő idő várható értéke (MTTF):**

$$y(t) = y(0) \cdot \exp\left(\frac{C}{y(0) \cdot n} \cdot t \right)$$

- **A jobboldalon az exponenciális részben *t* együtthatója pozitív, ezért a jobboldal végtelenhez tart. Tehát sok programfutás után a legközelebbi meghibásodásig eltelő idő minden határon túl nő.**

STATISZTIKAI MODELLEK - MUSA III.

- **A megbízhatósági függvény:**

$$R(t) = \exp\left(-\frac{t}{y(t)}\right)$$

- **Ennek alapján meghatározható, hogy ha a legközelebbi meghibásodásig várhatóan eltelő időt y_1-ről y_2-re akarjuk növelni, ehhez hány hibát kell megtalálnunk és kijavítanunk:**

$$\Delta_c = (y(0) \cdot n) \cdot \left\{\frac{1}{y_1} - \frac{1}{y_2}\right\}$$

STATISZTIKAI MODELLEK - MUSA IV.

- **Ugyancsak meghatározható, hogy ehhez mekkora programfutási időre van szükség:**

$$\Delta_t = \frac{y(0) \cdot n}{C} \cdot \ln\left(\frac{y_2}{y_1}\right)$$

A MODELL ÉRTÉKELÉSE:

- **figyelembe veszi, hogy különbség van a tesztelési és felhasználási célú programfuttatás között**
- **lehetőséget ad a továbbfejlesztésre, új paraméterek bevezetésével jobb illeszkedés elérésére**
- **hátránya, hogy nem tesz különbséget az egyes hibák veszélyessége között**

PÉLDA - I.

- **Egy nagy program feltételezhetően kb. 300 hibát tartalmaz. A program indításától az első hiba megjelenéséig átlagosan eltelő idő (MTTF) 1.5 óra. A tesztkompressziós faktor becsült értéke 4.**
- **Mennyi ideig kell tesztelni a programot ahhoz, hogy a megmaradó hibák számát 300-ról 10-re csökkenthessük?**

- **Megoldás:**

$$\Delta_c = (y(0) \cdot n) \cdot \left\{ \frac{1}{y_1} - \frac{1}{y_2} \right\}$$

- **Behelyettesítve:**

$$300 - 10 = (300 \cdot 1{,}5) \cdot \left\{ \frac{1}{1{,}5} - \frac{1}{y_2} \right\}$$

- **Innen**

$$y_2 = \textbf{ 45 óra}$$

PÉLDA - II.

- **Továbbá**

$$\Delta_t = \frac{y(0) \cdot n}{C} \cdot \ln\left(\frac{y_2}{y_1}\right)$$

- **Behelyettesítve:**

$$\Delta_t = \frac{300 \cdot 1{,}5}{4} \cdot \ln\left(\frac{45}{1{,}5}\right) = 382{,}6$$

- **Innen**

$$\Delta_t = \textbf{382.6 óra}$$

- **Tehát ennyi tesztelési időre van szükség, hogy a programhibák száma 10-re csökkenjen.**

PÉLDA - III.

- **Végül**

$$R(t) = \exp\left(-\frac{t}{y(t)}\right)$$

- **Behelyettesítve:**

$$R(50) = \exp\left(-\frac{50}{45}\right) = 0{,}33$$

- **Vagyis a 382.6 órányi tesztelés elvégzése után legalább 50 óra hibamentes működés valószínűsége 0.33 lesz.**

STATISZTIKAI MODELLEK – GOEL-OKUMOTO I.

A MODELL JELLEMZŐI:

- **programfutási időt alkalmaz**
- **sztochasztikus**
- **gyakoriságokkal számol**
- **fekete doboz**
- **a hibák száma véges, de véletlen**
- **a hibák csak akkor számítanak, ha megjelennek**
- **a javítás nem feltétlenül tökéletes**
- **a meghibásodási ráta csak a hibák megjelenésével változik**

JELÖLÉSEK:

t **– a legutóbbi hibától számított programfutási idő**
n **– a programban található összes hibák számának várható értéke**
c**(*****t*****) – az adott időpontig megtalált hibák száma**
m**(*****t*****) –** ***c*****(*****t*****) várható értéke**
z **– arányossági tényező a meglevő és megmutatkozó hibák között**

STATISZTIKAI MODELLEK – GOEL-OKUMOTO II.

MATEMATIKAI MODELL:

- **Az időegység alatt megmutatkozó hibák átlagos száma egyenesen arányos a programban még benn található hibák átlagos számával**
- **A *t* futási összidőig megtalált hibák számának várható értéke:**

$$m(t) = n \cdot \left\{1 - e^{-zt}\right\}$$

- **Látható, hogy a jobboldalon a kapcsos zárójelen belüli rész *t* növekedésével 1-hez tart, így *m*(*t*) is tart *n*-hez.**
- **Feltételezzük, hogy *c*(*t*) Poisson-eloszlást követ, paramétere és várható értéke *m*(*t*).**

A MODELL ÉRTÉKELÉSE:

- **Goel és Okumoto modellje egyaránt közel áll Musa, valamint Jelinski és Moranda modelljéhez.**
- **Egyetlen új paraméter bevezetésével modellezhető a nem tökéletes javítás. Legyen *p* annak a valószínűsége, hogy sikeres a javítás. *z* helyett *pz* paramétert használhatunk.**