POENOSTAVLJENA NAPOVED NASTOPA NAPAK PRI TESTIRANJU PROGRAMSKE OPREME

Tomaž Dogša, dipl. ing., dr. Ivo Rozman
TEHNIŠKA FAKULTETA MARIBOR
JUGOSLAVIJA

UDK: 681.3.06

POVZETEK - V članku je prikazana praktična in enostavna metoda za predikacijo nastopa napak v zadnjem obdobju testiranja. Ocenjen potrebni čas testiranja za odkritje naslednjih napak nam omogoča objektivno odločitev o prenehanju testiranja. Prikazan je tudi konkreten primer predikcije, ki se je uporabil pri testiranju simulatorja teleinformacijskih sistemov v železniškem prometu.

SIMPLE METHOD FOR SOFTWARE FAILURE PREDICTION - A practical and simple method for software failure prediction in the last period of software testing in presented. Objectiv decision when to stop with testing, is possible with results of prediction. A practical example of prediction that was used by software testing of computer railway control systems is also included.

UVOD

Eden izmed glavnih kazalcev kvalitete programske opreme je njena zanesljivost. S testiranjem programov skušamo dobiti odgovore na naslednja vprašanja:

a) Koliko je vseh napak v programu?
b) Kolikšna je zanesljivost delovanja programa v realnem okolju?
c) Kako dolgo je potrebno še testirati, da bo odkritih npr. 90 % vseh napak?
d) Kolikšni bodo stroški testiranja?

Danes so že razviti določeni matematični modeli /2/, /5/, ki bolj ali manj uspešno dajejo odgovore na ta vprašanja. Predvsem v praksi pogosto nerealne predpostavke, na katerih bazirajo ti modeli in pa vpletenost čisto subjektivnega faktorja otežuje deterministični pristop. (Pogoste predpostavke: število vseh napak v programu je konstantno, velikost programa se ne spreminja, večina napak je odpravljenih itd.)

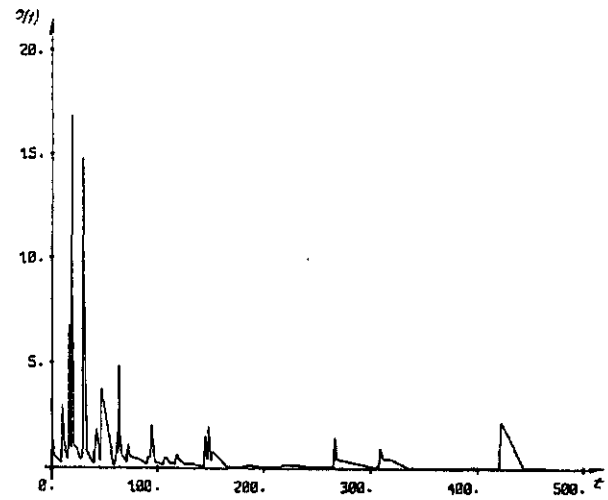
V nadaljevanju članka je prikazano, kako se da brez uporabe zapletenih modelov odgovoriti na vprašanje d). Ta metoda bo ilustrirala še s konkretnim primerom testiranja.

POENOSTAVLJENA PREDIKCIJA

Pri uporabi vsakega modela potrebujemo podatke o rezultatih testiranja. V ta namen je bila razvita posebna procedura /4/, ki je omogočala zbiranje najosnovnejših podatkov:

1. CPU čas potrebnega odkritja napake
2. število odkritih napak (v primeru, da jih je več kot 1)
3. koledarski čas potreben za odkritje napake
4. čas potreben za odpravo napake
5. velikost programa, ki ga testiramo.

Iz teh podatkov se dobijo osnovni pokazatelji, ki jih potrebujemo pri zasledovanju uspešnosti testiranja. V konkretnem primeru smo opazovali intenzivnost pojavljanja napak v odvisnosti od skupnega časa testiranja (slika 1) in pa kumulativno število napak v odvisnosti od CPU časa testiranja (slika 2) /2/.



Slika 1: Intenzivnost pojavljanja napak $\varphi(t)$ v odvisnosti od skupnega CPU časa testiranja t.

Z interpolacijo grafa na sliki 2 dobimo potreben čas za odkritje nekaj naslednjih napak (slika 3). Ta pristop temelji na naslednjih predpostavkah:

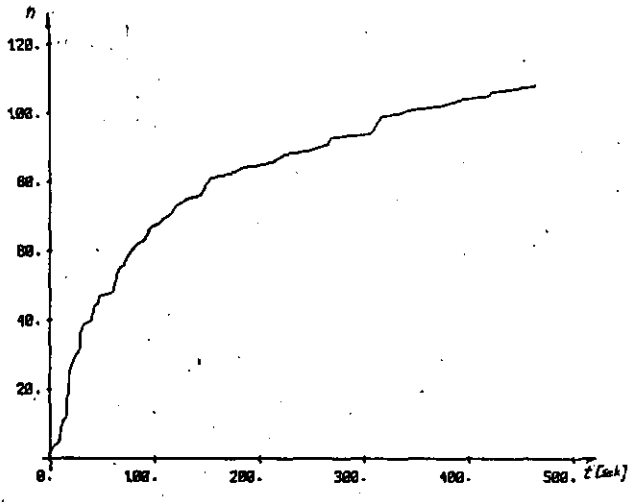
1. večina napak je že odkritih
2. intenzivnost pojavljanja napak ne bo naraščala ampak padala.

Za interpolacijsko funkcijo je bila zbrana

$$f(x) = k_2\sqrt{x} + k_1.$$

Za napoved potrebnega časa Δt, v katerem bomo našli še ΔN napak, moramo določiti enačbo, ki ponazarja

graf na sliki 2. S pomočjo metode povprečnih vrednosti



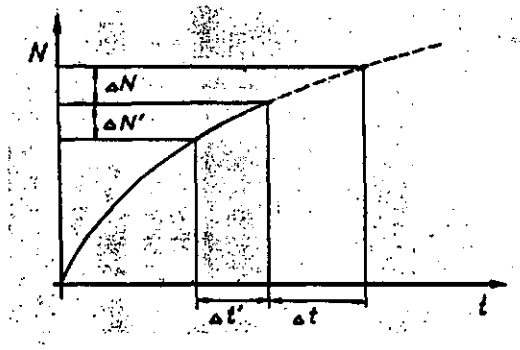
Slika 2: Kumulativno število napak n v odvisnosti od CPU časa testiranja t.

določimo koeficiente interpolacijske funkcije za interval $\Delta N'$. Za $\Delta N'$ naj velja:

$$\Delta N' \geq \Delta N.$$

Ko je funkcija f(x) določena, lahko izračunamo nastop napak v intervalu Δt.

Tabela na sliki 4 prikazuje del numeričnih rezultatov, ki smo jih dobili pri testiranju simulatorja procesov v že-



Slika 3: Preprosta predikcija.

lezniškem prometu. Ko smo odkrili 85 napako, smo napravili predikcijo za nastop naslednjih 5 napak. Napovedan potreben čas testiranja je bil 48,5 sek.(CPU), kasneje izmerjen pa 40,9 sek. Razlika je znašala 18,6 %.

Iz primerjave med napovedanimi podatki in dejanskimi je razvidno, da izbrana interpolacijska funkcija f(x) dobro ustreza. Maksimalno odstopanje na celotnem intervalu je bilo 4 %. Napoved za daljši interval je manj zanesljiva. Če bi v prejšnjem primeru povečali interval ΔN in $\Delta N'$ na 10, bi maksimalno odstopanje naraslo na 10,4 %.

Ker izbrana interpolacijska funkcija nima limitne vrednosti, se analitično ne da določiti število vseh napak. Slednje bi omogočila izbira funkcije

$$f(x) = k_2 (1 - e^{k_1/x}),$$

ki je podlaga znanemu Musovemu modelu /6/. V tem primeru bi postalo določanje koeficientov k_1 in k_2 precej zapleteno, saj bi morali reševati transcendentne enačbe.

t	t^*	N		odstopanje	
150,9	146,49	80		4,41	
152,2	156,25	81		-4,05	
165,7	166,33	82	$\Delta N'$	-0,63	
177,5	176,72	83		0,78	
183,1	187,43	84		-4,33	
201,8	198,45	85		3,35	
213,9	207,79	86		4,11	
219,0	221,44	87		-2,44	
224,3	233,41	88	ΔN	-9,11	(-4 %)
244,7	245,69	89		-0,99	
254,8	258,29	90		-3,49	

$\Delta t = 40{,}9$ $\Delta t' = 48{,}5$ $\Delta N = 5$

Slika 4: Primerjava med predikacijo t^* in dejanskimi rezultati t za napake od 86 do 90. Za kontrolo je izračunano odstopanje interpolacijske krivulje v področju že dobljenih rezultatov $\Delta N'$.

ZAKLJUČEK

Z izbiro preprostega matematičnega modela se da dokaj natančno napovedati nastop napak v zadnjem obdobju testiranja. S temi podatki lahko zelo dobro ocenimo nadaljnje stroške testiranja. Z opazovanjem intenzivnosti pojavljanja napak lahko sklepamo o umirjanju pojavljanja napak. Velika nihanja pomenijo, da s testiranjem vsekakor ne smemo končati.

LITERATURA

/1/ Myers J. G.: The Art of Software Testing, John Wiley and Sons, Inc., New York 1979

/2/ J. Virant: Zanesljivost računalniških sistemov, Zal. Fakulteta za elektrotehniko v Ljubljani, Ljubljana 1981

/3/ M. L. Schooman: Software Reliability: A Historical Perspective, IEEE Trausactions on Raliability, 1984 april, Vol. R-33, No 1, str. 48-55

/4/ T. Dogša: Testiranje programske opreme teleinformacijskih sistemov za vodenje železniškega prometa, Magistrska naloga, Tehniška fakulteta v Mariboru, Maribor 1985

/5/ Alan N. Sukert: Empirical Validation of Three Software Error Prediction Models, IEEE Traus. on Reliability, Vol. R - 28, No. 3, August 1979, str. 199-204

/6/ J. D. Musa: Validity of Execution - Time Theoriy of Software Reliability, IEEE Traus. on Reliability, Vol. R - 28, No. 3, August 1979, str. 181-191