Pouzdanost softvera

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Nenad Ajvaz, Stefan Kapunac, Filip Jovanović, Aleksandra Radosavljević
nenadajvaz@hotmail.com, stefankapunac@gmail.com,
jovanovic16942@gmail.com, aleksandraradosavljevic.@live.com

2. april 2019

Sažetak

ABSTRAKT ABSTRAKT

Sadržaj

1 Uvod

- Velika uloga softvera u danasnje vreme
- Softver je sve rasprostranjeniji, uskoro ce postati osnovna radna snaga
- Sa povecanjem uloge softvera u drustvu, njegova pouzdanost ima veci znacaj, jer od softvera vec uveliko zavise i ljudski zivoti
- Za razliku od ljudi, softver ne pravi slucajne greske, ali do gresaka ipak dolazi iz raznih razloga (mali promaci i male greske se vremenom ispoljavaju)
- U nastavku cemo predstaviti mere i modele pouzdanosti softvera, metode za poboljsanje pouzdanosti, kao i primere i u kojima su greske u sistemu dovele do ozbiljnih problema

2 Primeri padova softvera

- Izmedju ostalih, dodati primer za Boing (sad ovaj sto je pao, 10. marta)

3 Verifikacija softvera

- Bice okaceno

4 Modeli i metrike pouzdanosti softvera

Kao što je prethodno napomenuto, prilikom izgradnje softvera, veoma je važno imati na umu da se greška može naći u bilo kojoj fazi njegovog razvoja. Zato je bitno da se pronađe način procene broja potencijalnih grešaka u sistemu.
Statistika pokazuje da se krajem 20. veka na 1000 linija koda pojavi oko 8 grešaka, ne računajući prethodno testiranje sistema. Različiti izvori napominju da je danas taj broj veći, nabraja se 15-50 grešaka na 1000 linija. Majkrosoft tvrdi da njihov kod sadrži 0.5 neispravnosti na istom broj linija, dok NASA čak ni na 500,000 linija programskog koda ne sadrži niti jedan softverski problem. [1] Ovi brojevi su rezulat mnogih spoljašnjih faktora i nisu odgovarajuće merilo propusta i grešaka. Zbog toga su napravljeni razni modeli i metrike koji preciznije daju informacije o stabilnosti softvera. U nastavku će biti opisana dva tipa takvih modela.

4.1 Deterministički model

Ovaj model odlikuje preciznim merama i podacima koji se oslanjanju na same karakteristike programskog koda. Prilikom ocenjivanja, ne uzimaju se u obzir slučajni događaji i greške koji oni prouzrokuju, već su performanse računate prema egzaktnim podacima. Oni uključuju broj različitih operatora, operanada, kao i broj mašinskih instrukcija i grešaka. Fokus je na mehanizmu i načinu rada samog softvera, gde se uvek mogu predvideti očekivana stanja.
Dva najpoznatija deterministička modela su Holstedova metrika i Mek-Kejbova ciklomatična složenost.[1]

[1] Moris Hauard Holsted (1977), Tomas Mek-Kejb (1976)

4.1.1 Holstedova metrika

Ova metrika se smatra jednom od najboljih za procenu kompleksnosti softvera, ali i težinu prilikom njegovog testiranja i debagovanja. U obzir ocene implementacije ulazi broj instrukcija i operacija izvornog koda, koji će biti drugačiji za svaki programski jezik ili arhitekturu na kojoj se program izvršava. Uvodi se sledeća notacija:

Obeležje	Opis	Formula
n_1	broj jedinstvenih operatora u programu	
n_2	broj jedinstvenih operanada u programu	
n	vokabular programa	$n_1 + n_2$
N_1	ukupan broj operatora u programu	$n_1 \cdot log_2(n_1)$
N_2	ukupan broj operanada u programu	$n_2 \cdot log_2(n_2)$
N	dužina programa	$N_1 + N_2$
I	broj mašinskih instrukcija u programu	
V	obim programa	$N \cdot log_2(n_1 + n_2)$
D	težina čitanja, razumevanja i debagovanja	$n_1/2 \cdot N_2/2$
M	vreme utrošeno na implementaciju	$V \cdot D$
T	vreme utrošeno na testiranje	M/k
E	broj grešaka i bagova	$V/3000$

Tabela 1: Obeležja u Holstedovoj metrici [2]

- TODO: ubaciti primer iz knjige?

4.1.2 Mek-Kejbova ciklomatična složenost

Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum,Lorem ipsum, Lorem ipsum. Nenad.

4.2 Probabilistički model

Ovaj model pojavu grešaka i kvarova gleda kao na verovatnosne događaje. Softver se posmatra u fazi izvršavanja i pravi se mreža modela koja predstavlja ponašanje programa. Rezultat ovog modeliranja je primenjiv na razne metode iz oblasti statistike, što omogućava dubinsku matematičku analizu, detekciju anomalija, generisanju testova i simulaciju raznih slučajeva.

- Odabrati nekoliko i napisati po recenicu o svakom

5 Budućnost softvera

- TODO

6 Zaključak

- TODO

Literatura

[1] Hacker News, "Number of errors on 1000 lines of code," 2016. On-line at: https://news.ycombinator.com/.

[2] IBM Knowledge Center, "Halstead Metrics." On-line at: https://www.ibm.com/support/knowledgecenter/en/SSSHUF_8.0.2/com.ibm.rational.testrt.studio.doc/topics/csmhalstead.htm.

A Dodatak

Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe. Ovde pišem dodatne stvari, ukoliko za time ima potrebe.