SOFTVERSKE METRIKE ZA POUZDANOST SOFTVERA

Ratko Dejanović, *Elektrotehnički fakultet u Banjaluci*

Sadržaj - *Ovaj rad se bavi siftverskim metrikama i pouzdanošću softvera. Svaki isporučeni kod je vezan za kvalitet kroz sve faze procesa i razvoja produkta u životnom ciklusu softvera.Da bi se povećao kvalitet i pouzdanost softverskog produkta noephodno je upotrebiti softverske metrike. Mnoge softverske metrike su pogrešne ili su loše definisane ili imaju nepostojeće ciljeve. GQM (ciljevi, pitanja, metrike) paradigma je upotrebljena za generisanje nekih metrika u cilju poboljšanja pouzdanosti. Ove metrike mogu biti upotrebljene za mjerenje pouzdanosti u tri životne faze softvera: zahtjevi, kodiranje i testiranje.*

1. UVOD

Pouzdanost se definiše kao niz atributa koji predstavljaju mogućnost softvera da održi svoj nivo performansi pod određenim uslovima u određenom periodu vremena i ona je jedan od ključnih faktora kvaliteta softvera.(ISO 9126). Kod softvera ne postoji trošenje ili starenje i ograničenja u nepouzdanosti su posledica grešaka u zahtjevima, projektovanju i implementaciji.

Podkarakteristike pouzdanosti su zrelost (maturity) tj. svojstvo softvera koje odražava frekvenciju grešaka uslijed nedostataka u softveru, otpornost prema greškama (fault tolerance) tj. svojstvo softvera da održi određeni nivo performansi u slučaju greške i oporavljivost (recoverability) tj. svojstvo softvera da ponovo uspostavi svoj nivo performansi i povrati podatke na koje je direktno uticao u slučaju greške.

Izgradnja visoko pouzdanog softvera zavisi od učešća atributa kvaliteta u svakoj fazi životnog ciklusa razvoja sa naglaskom na prevenciju grešaka, specijalno u ranim fazama. Da bi se ovi atributi mjerili sa ciljem poboljšanja kvaliteta i pouzdanosti potrebno je definisati softverske metrike za svaku razvojnu fazu (zahtjevana dokumentacija, kod,planovi testiranja i testiranje).

IEEE 982.1-1988 definiše upravljanje pouzdanošću softvera kao proces optimizacije softvera kroz tri aktivnosti u kontekstu ograničenja u projektu kao što su resursi, vremena i performanse:

- Prevencija greške (error)
- Detekcija i otklanjanje fault-ova
- Mjerenje da bi se maksimizirala pouzdanost za prve dvije aktivnosti

Kada su u pitanju greške onda dolazi do zbrke u terminima error, fault, failure koji se upotrebljavaju za istu stvar premda imaju različito značenje. U slučaju softvera error je obično programerska greška koja koja rezultira u fault-u. Fault je softverski defekt koji prouzrokuje failure, failure je neprihvatljiva programska operacija u odnosu na programske zahtjeve.

2.GQM-CILJEVI/PITANJA/METRIKE PARADIGMA

Bilo kakve aktivnosti mjerenja moraju imati jasan cilj. U početku treba biti jasno da li se mjerenje vrši zbog procjene ili zbog predikcije. Sledeće, moraju se znati ineresantni entiteti za mjerenje. Zatim treba odlučiti koji su atributi izabranih entiteta značajni. Dugo godina su se softverski praktičari pitali : " Za šta softverske metrike mogu biti upotrebljene ?" zbog ignorancije teorije mjerenja u softverskom inženjerstvu. Revolucionaran korak naprijed su napravili Basili i Rombach 1988 sa Goal/Question/Metric (cilj/pitanje/metrika) paradigmom. GQM postavlja ciljeve prije bilo koje aktivnosti mjerenja.Mjerenje da bi bilo efektivno mora biti :

- fokusirano na specifične ciljeve
- primjenjeno u svim životnim ciklusima produkta, procesa i resursa
- interpretirano na razumjevanju organizacionog konteksta, okruženja i ciljeva

To znači da mjerenje mora biti definisano u top-down (od vrha prema dnu) stilu i mora biti fokusirano i bazirano na ciljevima i modelima. Drugi pristup bottom-up (od dna ka vrhu) nije dobar jer postoje mnoge opservabilne karakteristike softvera (npr. vrijeme, broj gršaka, kompleksnost, linije koda, napor, produktivnost itd.) ali nije sasvim jasno koje metrike upotrebiti i na koji način ih interpretirati bez odgovarajućeg modela i ciljeva. GQM model ima tri nivoa:

1. Konceptualni nivo (GOAL) **:** Cilj se definiše za objekt iz različitih razloga, vodeći računa o različitim modelima kvaliteta i različitim tačkama gledišta. Objekti mjerenja su:

- Produkti koji nastaju u živtnom ciklusu softvera npr. : specifikacije, projektovanje, programi, testni podaci.

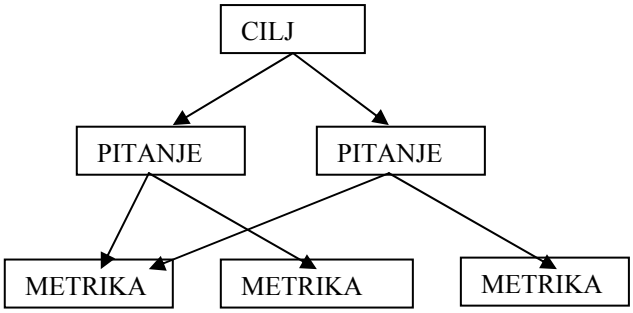
- Procesi koji obuvataju aktivnosti vezane za vrijeme npr. : specificiranje, projektovanje, testiranje, intervijujisanje.

- Resursi upotrebljeni u procesu da bi se dobio produkt npr. :personal, hardver, softver, radni prostor.

2. Operativni nivo (QUESTION) : Niz pitanja koja se upotrebljavaju da specificiraju put za postizanje željenog cilja. Pitanja pokušavaju da karakterišu objekte mjerenja

(produkti, procesi, resursi) sa respektom na selektirani nivo izlazećeg kvaliteta sa različitih tačaka gledišta.
3. Kvantitativni nivo (METRIC) : Niz podataka pridruženih svakom pitanju da bi se dobio odgovor u kvantitativnoj formi. Podaci mogu biti objektivni i subjektivni.

Objektivni podaci zavise samo od objekta mjerenja ali ne i od tačke gledišta za razliku od subjektivnih koji zavise i od jednog i od drugog. GQM model ima hijerarhijsku strukturu prikazanu na Sl. 1.



Sl 1. *Hijerahijska struktura GQM*

GQM prilaz je u stvari mehanizam za definisanje i interpretiranje mjerljivog softvera i može se upotrebiti zasebno ili u kombinaciji sa nekim drugim pristupima u smislu poboljšanja kvaliteta softvera. Međutim postoje situacije kada se koristi mjerenje softvera a da ciljevi nisu jasno definisani. Ove situacije se obično dešavavaju kada se mali broj metrika odnosi na različite ciljeve. Sledeći problem je ko postavlja ciljeve. Najčešće rukovodstvo može identifikovati loše ciljeve ili ciljeve za koje se ne mogu praktično izračunati metrike na inženjerijskom nivou koji zahtijeva korisne i praktične metrike. Pored toga ovaj model je jedostavan i intuitivan.

3. GQM ANALIZA

U Tabeli 1. date su dobro poznate metrike koje treba da odgovore na tri kompleksna pitanja vezana za zahtjeve, kod i testiranje a u cilju poboljšanja pouzdanosti i pronalaženje modula sklonim greškama.

Tabela 1. *GQM za poboljšanje pouzdanosti*

GOAL (ciljevi)	**QUESTIONS (pitanja)**	**METRICS (metrike)**
Poboljšanje pouzdanosti softvera i identifikacija zahtjeva i koda koji su potencijalni uzročnici grešaka	Kakav je kvalitet zahtjeva (dokumenta)?	Broj linija teksta Imperativi-komandne riječi i fraze Fraze koje slijede imperative (zahtjevi na nižem nivou) Direktive-reference na slike, tabele i napomene Slabe fraze-neizvjesnost i višestruka interpretacija Nekompletnost iskaza Opcije – opsne riječi zbog širine značrnja
	Kakav je kvalitet dizajna i koda?	Veličina koda (KLOC) Kompleksnost koda (McCabe) FP (funkcionalne tačke) Modularnost (coupling, kohezija) OO Metrike – WMC, DIT, NOC, CBO, RFC, LCOM Ponovna upoteba koda (reuse(c))
	Koliki je broj pronađenih grešaka, Vrijeme između grešaka, Vrijeme da nastane greške Vrijeme otklanjanja greške, Frekvencija događanja grešaka, Predviđanje broja preostalih grešaka ?	Broj otkrivenih defekata MTBF (Mean Time Between Failure) MTTF (Mean Time To Failure MTTR (Mean time to Repair) RCOF (Rate of Occurence of failures) Vrijeme testiranja Vrijeme upotrebe Broj testnih slučajeva KLOC, McCabe, FP (za razne normalizacije)

Svakako da je broj metrika prevelik i da bi ga trebalo redukovati na manji broj jednostavnih i prihvatljivih metrika zavisno od okruženja i aspekta gledanja.

Zahtjevi specificiraju funkcionalnost koja mora biti uključena u finalni softver i moraju biti struktuirani, kompletni i jasni za komunikaciju između projektanta i korisnika.Dvije bitne metrike za vrednovanje dvosmislenosti termina su broj slabih fraza (npr. adekvatan, odgovarajući, normalan itd.) i broj opcionalnih fraza (npr. može, smije, opcionalno itd.). Takođe treba izbjegavati i nekompletne termine kao što su "treba da bude dodano i treba da bude determinisano". Važnost korektno dokumentovanih zahtjeva je prouzrokovala da se na tržištu pojave specijalizovani alati za mjerenje kvaliteta specifikacije zahtjeva npr. ARM (Automated Requirements Measurement) [1].

Metrike za pouzdanost dizajna i koda se uglavnom odnose na veličinu produkta izraženu u broju linija koda (LOC-Lines Of Code), kompleksnost koda izraženu preko ciklomatskog broja (McCabe) i modularnost produkta. Što je modul kompleksniji veće su poteškoće u negovom razumjevanju i veća je vjerovatnoća defekata nego u manje kompleksnim modulima Iako su veličina i kompleksnost korisne metrike često se koristi i njihova korelacija da bi se dobile dodatne informacije. Za kvalitet objektno orijentisanih struktura koriste se OO metrike.

Kod testnih metrika postoje dva pristupa za vrednovanje puzdanosti [1]. Prvi se odnosi na vrednovanje test plana da se obezbjedi da sistem ima funkcionalnost predviđenu specifikacijom zahtjeva. Drugi pristup vezan za pouzdanost se sastoji u vrednovanju broja grešaka u kodu i ratama njihovog fiksiranja i otklanjanja. Podaci o greškama se dobijaju za vrijeme testiranja i stvarne upotrebe softverskog produkta. Ovi podaci se koriste za izgradnju modela pouzdanosti na osnovu koga se može izvršiti predikcija vremena slijedeće greške na osnovu istorijata prethodnih grešaka.

4. ZAKLJUČAK

Pouzdanost je jedan od ključnih faktora kvaliteta i zato pouzdanost isporučenog koda zavisi od kvaliteta svih procesa i produkta u svim životnim fazama razvoja softvera.

Podaci o pouzdanosti softvera pomažu u povećanju produktivnosti, redukciji isporučenih defekata, redukciji frekvecija prekida i redukciji padova sistema.

Da bi se dobili podaci u tim fazama treba upotrebiti softverske metrike za pomoć u vrednovanju pouzdanosti. Na taj način se povećava kvalitet a sa samim tim i pouzdanost. Upotreba efektivnih softverskih metrika stabilizira proces razvoja softvera i osigurava konzistenciju u njegovom kvalitetu i pomaže u detekciji i otklananju defekata u ranim fazama razvoja.

U ovom slučaju za izbor metrika upotrebljena je GQM paradigma jer ona predpostavlja da je mjerenje fokusirano na specifične ciljeve, primjenjeno u svim životnim ciklusima produkta, procesa i resursa i interpretirano na razumjevanju organizacionog konteksta, okruženja i ciljeva.

LITERATURA

[1] L. Rosenberg, T. Hammer, J. Show, "Software Metrics and Reliability," *9th International Symposium "BEST PAPER", 1988, Germany*

[2] N. Fenton, S. Pleeger, *Software Metrics: A Rigorous & Practical Approach,*ITC PRESS, 1998.

[3] L. Westfall, "Seven Steps To Design A software metrics," *Software measurement Service*, 2003.

[4] J. Marciniak, *Encyclopedia of Software Engineering,* John Willey&Sons, Vol I i II, 1994.

[5] R. Dejanović, "Dizajniranje softverskih metrika,", IX *Simpozij Informacione tehnologije IT ,*Žabljak, 2004.

Abstract - This paper deals with software metrics and reliability. Evry delivered code is related to the quality through all of the stages of processes and products in the software life cycle. It is necessary to apply software metrics to improve quality and reliability of software products. Many software metrics have failed because they had poorly defined, or even non-existent objectives.GQM (Goal-Question-Metrics) paradigm is used to generated several metrics for single goal for improving reliability.These metrics can be applied to impact the reliability in three software life cycle phases: requirements, coding and testing.

SOFTWARE METRICS FOR SOFTWARE RELIABILITY

Ratko Dejanović