Predrag Oreški

Fakultet organizacije i informatike
Varaždin

UDK: 681.32.06
Stručni rad

METRIKA SOFTVERA

U ovom radu daje se kratak uvod u disciplinu metrika i modela softverskog inženjerstva. Detaljno se opisuju faktori i kriteriji procjene kvalitete softvera te se prikazuje jedna metoda ocjenjivanja softvera.

U nastavku opisuju se osnovne ideje kod definiranja pojedinih metrika i modela softvera, kao i mišljenje o procjeni vremena i troškova razvoja softvera.

Softversko inženjerstvo; kvaliteta softvera; metrika.

1 UVOD

Razvoj sofvera je skup proces. Uvode se nove tehnike kao što je sotversko inženjerstvo s namjerom da se cijena koštanja softvera smanji.

Početkom '80-ih počeo se prvi put javljati jači zahtjev za utvrđivanjem karakteristika softvera koje bi se mogle mjeriti. Na taj način mogao bi se vrednovati i uspoređivati određeni softver, kvantificirati njegovi atributi, ali i proces njegovog razvoja i rada - životni ciklus softvera. Pomoću ovakvog sustava mjerenja softvera bilo bi moguće utjecati i na njegov razvoj. Razvoj odgovarajućeg sustava mjerenja softvera i njegovog razvoja je bitna pretpostavka za razvoj jeftinog i pouzdanog softvera u točno definiranim rokovima.

Da bi se softver mogao ispravno ocijeniti i vrednovati, potrebni su:određeni sustavi vrijednosti i norme. Zajedno, oni čine sustav mjerenja kompjutorskih programa - metriku softvera.

Da bi se bolje razumio proces razvoja softvera, potrebno je razviti njegov model (Životni ciklus softvera).

Modeli procesa razvoja softvera i metrika softvera zajedno stvaraju novu disciplinu koja se zove metrike i modeli softverskog inženjerstva.

1.1 SOFTVERSKO INŽENJERSTVO

Skup poslova koji su vezani za "inicijalizaciju, projektiranje, realizaciju i prodaju softverskog proizvoda, te (ruko)vođenje svim resursima koji su vezani za taj proizvod" naziva se softverskim inženjeringom (prema Stewardu, W.D.: Software Engineering with Systems Analysis and Design, Brooks-Cole, 1987).

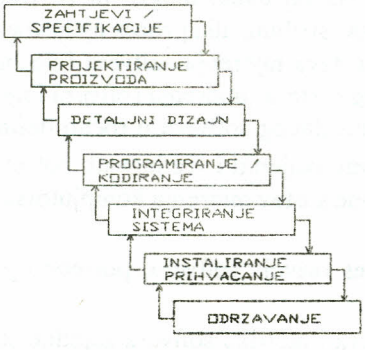
"Sotversko inženjerstvo je sistematski pristup razvoju, eksploataciji, održavanju i zamjeni programskih proizvoda" (ANSI/IEEE 729/1983).

"Softversko inženjerstvo je tehnološka i upravljačka disciplina koja se bavi sistematskom proizvodnjom i održavanjem programskih proizvoda, koji trebaju biti razvijeni na vrijeme i uz predviđene troškove" (Fairley 1984).

Karakteristike softverskog inženjerstva jesu:

* modeliranje kao osnova projektiranja (model objekti-veze, model procesa, model podataka),
* koriste se metode analize, sinteze, identifikacije,
* podjela na nivoe, vezane uz faze životnog ciklusa programskog proizvoda i faze projekta,
* interdisciplinarnost i uključivanje korisnika,
* projektna organizacija,
* validacija i verifikacija rezultata, osiguranje kvalitete, prikaz rezultata i dokumentiranje.

1.2 ŽIVOTNI CIKLUS SOFTVERA



Slika 1. Vodopadni (Waterfall) model životnog ciklusa softvera (izvor:(1), str. 5)

"Životni ciklus programskog proizvoda je vremenski period između iniciranja njegovog razvoja i trenutka kad on više nije na raspolaganju" (ANSI/IEEE 729/1983).

Životni ciklus programskog proizvoda sastoji se od određenih, međusobno povezanih faza. On započinje specificiranjem potreba, a završava procesom održavanja. Model životnog ciklusa softvera može se vidjeti na slici 1.

Faze jesu:

a) Zahtjevi i specifikacije. Definiranje zahtjeva, specifikacija zahtjeva koje treba riješiti određenim programskim proizvodom.
b) Projektiranje (dizajn) programskog proizvoda. U ovoj fazi treba se specificirati cjelokupna konfiguracija sustava, programski jezik, glavni moduli i njihove veze, strukture podataka, plan testiranja.
c) Detaljni dizajn. Detaljnije se specificiraju moduli, njihova neophodna međusobna komunikacija, algoritmi, strukture podataka i interne kontrolne strukture.
d) Programiranje/kodiranje. U ovoj fazi izgrađuju se moduli. Istovremeno se testiraju pojedine jedinice modula kao i testiranje pojedinih podsistema.
e) Integriranje sustava. Testiraju se pojedini moduli u međusobnom interaktivnom radu. Testira se cjelokupni sustav.
f) Instaliranje/prihvaćanje sistema. Programski proizvod isporučuje se korisniku na testiranje. Isporučuje se korisnička dokumentacija, izvodi se obučavanje korisnika. Ispravljaju se eventualne greške prije prihvaćanja programskog proizvoda.
g) Održavanje. Ovo je dugotrajna faza ispravljanja naknadno uočenih grešaka, izmjena koda, priručnika, ...

Uobičajeno je da se ovih 7 faza životnog ciklusa programskog proizvoda grupira u 4 kategorije (prema (1), str.6):

" a) Dizajn. Ova grupa sadrži prve tri faze životnog ciklusa.
b) Kodiranje. Praktički, ovdje se razvija softver.
c) Testiranje. Integriranje sistema i Instaliranje i prihvaćanje sistema obuhvaćeno je ovom grupom.
d) Održavanje. Programski proizvod je ovdje u upotrebi. Ova grupa odgovara fazi Održavanja životnog ciklusa."

Za razmatranje u ovom radu posebno su važne slijedeće 4 faze životnog ciklusa s odnosom prosječno uloženog utroška vremena/napora (prema (1), str.6):

" a) Projektiranje (dizajn) - 15 %
b) Detaljni dizajn - 25 %
c) Programiranje/kodiranje - 40 %
d) Integriranje sistema - 20 % "

Potrebno je napomenuti da se u računskim centrima po raznim institucijama od ukupno raspoloživog vremena najviše vremena i napora troši na održavanju softvera. Na taj način održavanje predstavlja i najveće troškove za računski centar.

Na proces održavanja i njegovu složenost bitno utječe kvaliteta softvera.

2 FAKTORI KVALITETE I KRITERIJI PROCJENE SOFTVERA

U svakodnevnom radu s računalom dolazimo u kontakt s raznim sofverom. To su procesori teksta, programi za tabelarna izračunavanja, programi za upravljanje bazama podataka, vlastite aplikacije, itd. Za neko područje primjene računala postoji više programskih proizvoda različitih proizvođača. Mi se odlučujemo za jedan od njih i kažemo svom kolegi ili prijatelju da je taj najbolji.

Zašto ?

Za neki kompjutorski program možemo reći da je kvalitetan ako je bez greške. Međutim, što znači krajnjem korisniku da je program bez greške ako je on nepraktičan za korištenje, težak za razumijevanje, itd. Da li je i to kvalitetan softver ?

Softver ima atribute koji ga opisuju. Njih uzimamo kao kriterije procjene te pomoću njih definiramo (kvantificiramo ?) faktore kvalitete. Možemo nabrojati slijedeće faktore kvalitete s pripadajućim kriterijima procjene (atributima), prema (4):

" - CORRECTNESS - (korektnost)	traceability, completness, consistency,
- MAINTAINABILITY - (mogućnost održavanja)	traceability, consistency, conciseness, modularity,

- RELIABILITY - (pouzdanost)	conciseness, accuracy, error tolerance, simplicity,
- EFFICIENCY - (efikasnost)	execution efficiency, storage efficiency,
- INTEGRITY - (integralnost)	access control, access audit,
- USABILITY - (primjenljivost)	operability, training, communicativeness,
- TESTABILITY - (mogućnost testiranja)	simplicity, instrumentation, self-descriptiveness,
- FLEXIBILITY - (prilagodljivost)	modularity, generality, expandability,
- PORTABILITY - (prenosivost)	modularity, self-descriptiveness, software system independence, machine independence,
- REUSABILITY - (obnovljivost)	modularity, generality, sofware system independence,
- INTEROPERABILITY - (kooperativnost)	modularity, communications commonality, data commonality."

Kriteriji procjene softvera pomoću kojih su opisani faktori kvalitete jesu:

- traceability - mogućnost rekonstrukcije slijeda promjena i utjecajnih faktora iz okoline koji su na promjene utjecali, od analize do implementacije,
- completness - osiguranje potpune implementacije specificiranih svojstava, pri čemu je svaka faza razvojnog ciklusa zapravo implementacija prethodne,
- consistency - primjena jedinstvenih projektantskih tehnika i notacija,

- accuracy - osiguranje zahtjevane točnosti i ponovljivosti rezultata i točnosti notacije,
- error tolerance - osiguranje kontinuiteta rada pod uvjetima koji nisu predviđeni,
- simplicity - implementacija specificiranih svojstava na najjednostavniji i najrazumljiviji način,
- modularity - tvorba strukture od nezavisnih modula,
- generality - osiguranje općenitosti primjene rješenja,
- expandability - omogućavanje povećanja funkcionalnosti softvera ili sposobnosti pohrane podataka i informacija,
- instrumentation - osiguranje mjerenja svojstava i pronalaženje grešaka,
- self-descriptiveness - sposobnost razumljivog, jednoznačnog i točnog opisa implementacije specificiranih svojstava,
- execution efficiency - osiguranje minimalnog utroška resursa,
- storage efficiency - minimiziranje potreba za vanjskim memorijama,
- access control & access audit - osiguranje kontrole pristupa softveru i podacima,
- operability - određivanje formalnih postupaka i procedura za rad programskog proizvoda,
- training - osiguranje prijelaza na normalno korištenje softvera kroz uvođenje korisnika,
- communicativeness - omogućavanje prikladnih načina komuniciranja s korisnikom (sučelje, interface),
- softvare system independence - određivanje (ne)zavisnosti softvera od okoline (operativni sistem, uslužni programi, ulazno- izlazni programi),
- machine independence - određivanje (ne)zavisnosti softvera od hardvera,
- communications commonality - upotreba standardnih protokola i rutina za povezivanje,
- data commonality - upotreba standardne reprezentacije podataka.

Na taj način možemo kvantificirati svaki faktor kvalitete preko kriterija procjene softvera. Neka je $H_i = (1, ..., n)$ faktor kvalitete softvera. Neka je m_i broj kriterija procjene softvera za svaki faktor. Neka je s_{ij} rezultat za i-ti faktor kvalitete i j-ti kriterij procjene. S_{ij} može poprimiti vrijednost od 0 do 5.

Tada je rezultat za pojedini faktor kvalitete SC_i (score)

$$SC_i = \sum_{j=1}^{m_i} s_{ij}$$

a ukupni rezultat TSC (total score)

$$TSC = \sum_{i=1}^{n} \sum_{j=1}^{m_i} s_{ij}$$

Možemo uvesti ponder w (weights) za koji vrijedi da je

$$\sum w_i = n$$

i tada možemo dobiti izraz TWSC (total weighted score)

$$TWSC = \sum_{i=1}^{n} w_i \sum_{j=1}^{m_i} s_{ij}$$

Na ovakav način korisnik može sam povećati "težinu" (ponder) faktora kvalitete za koje smatra da su najvažniji za aplikaciju koju promatra.

TSC_{max} je maksimalni mogući rezultat za određen broj faktora kvalitete n i broj kriterija procjene softvera m

$$TSC_{max} = 5 \sum_{i=1}^{m} m_i$$

Konačno, izračunajmo i vrijednost koja označava kvalitetu softvera NWSC (normalized weighted score)

$$NWSC = \frac{TWSC}{TSC_{max}}$$

Vrijednost NWSC ima karakteristiku

$$\mathbf{0 \leq NWSC \leq 1}$$

i što je bliža 1, to je softver kvalitetniji. Da bismo došli do ranga vrijednosti NWSC koji je za nas prihvatljiv, potrebno je da izračunamo vrijednost NWSC za softver za koji znamo da je za nas dovoljno kvalitetan i da nas zadovoljava.

3 METRIKE I MODELI SOFTVERA

Mnoge od metrika softvera mjere objektivno (algoritamski), no neke samo subjektivno. Metrike koje su ovisne o ocjenjivaču (subjektivne metrike) prepoznat će se i na taj način što će se ocjene pojedinih ocjenjivača razlikovati. Vjerojatno je da će se i ocjena istog ocjenjivača dana sada i ona ocjena dobivena za godinu dana razlikovati.

Zato kažemo da je objektivno, algoritamsko mjerenje ono mjerenje gdje se vrijednost rezultata može precizno izračunati po algoritmu. Vrijednost rezultata ne mijenja se ni u vremenu, ni u prostoru, niti je ovisna o ocjenjivaču (osobi koja koristi algoritam).

Metrika softvera obično se dijeli na procesnu metriku i metriku proizvoda. Procesna metrika kvantificira atribute procesa razvoja i razvojne okoline. Ona mjeri karakteristike kao što su iskustvo programera ili cijena razvoja i održavanja. (Npr. iskustvo programera: kako dugo (u godinama) razvojni tim koristi određeni programski jezik, kako dugo je programer u organizaciji ili timu, kako dugo programer radi na srodnim programima razvoja. Ostali faktori mogli bi biti: da li se koristi top-down ili bottom-up razvojna tehnika, strukturirano programiranje, prisustvo pomoćnih alata, itd.)

Metrika proizvoda mjeri softver. Ona uključuje veličinu sofvera (broj linija koda, broj ključnih riječi), logičku strukturu (kontrola toka, složenost izraza odlučivanja (IF-THEN, DO CASE)), itd.

Međutim, ni jedna od metrika nije samo procesna ili pak metrika proizvoda. Neku od njih možemo smatrati takvima samo onda ako istaknemo samo ona svojstva koja su karakteristika pojedine vrste metrika.

Model razvoja softvera može biti prikazan kao

$$y = f(x_1, x_2, ..., x_n)$$

gdje je y zavisna varijabla, obično procesna metrika koja nas zanima. To može biti ukupno potrebno vrijeme/napor, cijena razvoja, ili kvaliteta softvera u vidu pronalaženja naknadnih grešaka u softveru. Ona je funkcija nezavisnih varijabli $x_1,...,x_n$ koje mogu biti orijentirane procesu razvoja ili sofveru.

Modeli mogu biti vođeni podacima i teorijom. Modeli vođeni teorijom su bazirani na pretpostavljenim odnosima između pojedinih faktora. Oni su razvijeni nezavisno od podataka.

Modeli vođeni podacima obično su rezultat statističke analize. Većina modela je kompleksnog tipa: sadrži dijelove koji su rezultat statističke analize i dijelove koji predstavljaju našu intuiciju.

Konstrukcija modela razvoja predstavlja lakši dio našeg istraživanja. Naime, potrebno je empirijski dokazati vrijednost konstruiranog modela. Da bismo to uopće mogli i pokušati, potrebno je prvo prikupiti podatke o postojećim procesima razvoja softvera. Ovo se obično radi preko upitnika ili intervjuom.

Nakon što su podaci prikupljeni, potrebno ih je statistički obraditi i interpretirati. Istraživač treba iz njih izvući što je moguće više pretpostavljenih zavisnosti te na osnovi njih konstruirati adekvatniji model.

4 METRIKE SOFTVERA

Puno je karakteristika softvera koje možemo mjeriti: veličina u linijama koda, cijena razvoja u dinarima, vrijeme potrebno za razvoj u radnim danima, potrebna veličina memorije u bajtovima, broj pritužbi korisnika, itd.

Potrebno je napomenuti da, kada uspoređujemo rezultate dobivene metrikom, trebamo do rezultata doći uspoređujući "jabuke s jabukama". Tako, npr. ne bismo smjeli uspoređivati dva programa koji rješavaju isti problem po linijama koda ako je jedan program pisan u COBOL-u, a drugi u ZIM-u. COBOL je jezik III generacije, a ZIM IV. generacije.

4.1 METRIKE VELIČINE

Svi kompjutorski programi imaju veličinu. Gotovo svi modeli razvoja softvera uzimaju u obzir ovu karakteristiku i zbog toga što je ona normalno neposredno vezana na produktivnost programera.

Tako metrika veličine uzima u obzir broj linija koda (linija koda je red u izvornoj (source) verziji softvera koji nije komentar ili prazni red), broj rezerviranih riječi, broj upotrijebljenih funkcija. To su nezavisne varijable metrike veličine softvera.

4.2 METRIKE STRUKTURE PODATAKA

Svrha postojanja softvera je obrada podataka. Podaci mogu biti ulazni, interni i izlazni. Kod metrika strukture podataka mjeri se količina tih podataka. Vjerojatno je da softver, koji ima više podataka za unos i/ili više internih varijabli, ima veću vjerojatnost za greške u radu, troši više memorije i vremena za svoj rad.

4.3 METRIKE LOGIČKE STRUKTURE

Ove metrike mjere, odnosno broje grananja u programu, broj i nivoe IF i DO CASE struktura. Ako postoje dva programa koji rješavaju isti problem i prvi program ima manje grananja u programu i niži nivo IF i DO CASE struktura, tada je on svakako logički jednostavniji te na taj način i kvalitetniji što se tiče logičke strukture.

5 PROCJENA VREMENA I TROŠKOVA RAZVOJA SOFTVERA

Za svakog rukovodioca računskog centra trebali bi biti važni podaci koji se tiču predviđanja potrebnog vremena i troškova razvoja određene vrste softvera. Da bi se takvo predviđanje moglo uspješno napraviti, potrebno je imati ili veliko iskustvo u radu na razvoju softvera ili pak podatke o tome kako je izgledao razvoj softvera prije, u pogledu vremena i troškova.

Projekti razvoja softvera mogu se podijeliti na tzv. projekte na mikro nivou i projekte na makro nivou.

Projekti razvoja na mikro nivou su jednostavni i zahtijevaju rad obično jednog programera nekoliko sati ili nekoliko dana. Na produktivnost programera u takvim projektima puno utječu čak i telefonski pozivi ili pauza za kavu. Potrebno je napomenuti da su iskustvo programera i njegovo znanje ovdje od presudnog značenja za projekt. Npr. jednom neiskusnom programeru s npr. nedovoljnim znanjem matematike ili logike bit će potrebni sati da kodira jednostavan algoritam npr. sortiranja, dok će drugome trebati tek nekoliko minuta ili pola sata. Dakle, ovdje procjena troškova i vremena ovisi praktično o programeru, te to treba imati svakako u vidu.

Projekti razvoja na makro nivou su složeniji i obično zahtijevaju rad tima programera u toku nekoliko mjeseci ili godina. Za razliku od projekata na mikro nivou, gdje je mjerna jedinica programer/sat, ovdje su mjerne jedinice troškova i vremena programer/mjesec i programer/godina. Ovdje razvoj nije ovisan samo o jednom čovjeku u kratkom vremenu te se podaci mogu jednostavno prikupiti pri kraju projekta razvoja iz računovodstva. Kod analize ovakvih podataka treba imati u vidu da programer ne može raditi cijelo radno vrijeme maksimalnim tempom više mjeseci ili godina kao što bi to mogao kod projekata na mikro nivou. Vjerojatno je da će produktivnost kod makro projekata razvoja softvera biti manja.

Da bi se izračunala cijena koštanja pojedinog projekta razvoja softvera, potrebno je znati samo jediničnu cijenu programera/sat, programera/mjesec i programera/godina.

6 GREŠKE I POUZDANOST SOFTVERA

Program će samo slučajno "proraditi od prve". S ovom konstatacijom će se vjerojatno složiti svi programeri. Također, vjerojatna je konstatacija da nema savršenog softvera - softvera bez greške.

Za softver možemo reći da nije ispravan ukoliko ne radi ono što korisnik od njega očekuje, odnosno kada postoji greška u programu koja uzrokuje netočne rezultate na osnovi ispravnih ulaznih podataka. Ove greške mogu varirati od prividno točnih rezultata pa do 'pada' programa (nenormalnog završetka izvođenja).

Pošto greške u programu bitno utječu na produktivnost programera, na ukupno utrošeno vrijeme i troškove, kao i na proces održavanja, i ovdje možemo uključiti metrike. Tako, npr. postoje metrike:

a) broj promjena u projektiranju - uslijed krivog razumijevanja zahtjeva i specifikacija može doći do grešaka u dizajnu softvera;
b) broj grešaka - obično se može detektirati u toku faze testiranja softvera (obično su to sintaksne greške);
c) broj promjena u programu - ovo su zapravo logičke greške, greške u kodiranju algoritama.

Obično korištena metrika za detektiranje grešaka je gustoća grešaka:

$$\text{gustoća\_grešaka} = \text{broj\_grešaka} / S$$

gdje je S dužina koda u tisućama linija.

Također, bitno je razlikovati takve greške koje nenormalno završavaju (prekidaju) rad programa i koje trebaju svega par minuta za ispravak, i one koje daju prividno točne rezultate te zahtijevaju sate i sate analize koda.

POUZDANOST SOFTVERA

Za program kažemo da je pouzdan ako uvijek daje isti odgovor na isti upit. Ako je još taj odgovor i ispravan tada je sve u redu. U pravilu, pouzdan program će 'pasti' uslijed greške u strojnoj komponenti.

Matematički, pouzdanost softvera $R(\tau)$ je vjerojatnost da će program raditi bez greške u vremenskom intervalu τ. Ukoliko pretpostavimo da je otkrivanje grešaka stohastički proces, tada se vrijeme između dvije detekcije greški može prikazati funkcijom $F(\tau)$ za 0. Tada je pouzdanost

$$R(\tau) = 1 - F(\tau)$$

Interesantno je spomenuti i prosječno vrijeme za grešku (Mean time to failure) MTTF

$$MTTF = \frac{1}{n} \sum_{i=1}^{n} t_i$$

gdje je t_i vremenski interval između (i-1)-te i i-te greške.

7 ZAKLJUČAK

Cilj softverskog inženjerstva je proizvesti softver više kvalitete uz manje troškove. Koji su praktični doprinosi metrike softvera tom cilju ?

To su:

- mogućnost kvantitativnog opisivanja softverskih atributa kao što su kvaliteta, utrošak resursa, produktivnost,
- mogućnost predviđanja (prognoziranja) cijene koštanja softvera, vremena isporuke i pouzdanosti,
- mogućnost kvantitativnog izražavanja zahtjeva s obzirom na softver te kriterije prihvaćanja,
- mogućnost nadgledavanja procesa razvoja softvera, predviđanja problema u razvoju, i
- veza s osobljem na razvoju softvera u vezi sa sadašnjim i mogućim problemima.

Potrebno je napomenuti da se, pogotovo zadnjih godina, puno truda ulaže u poboljšanje postojećih metrika koje, iako sadrže velik dio objektivnog, zahtijevaju kalibraciju (praktično određivanje) određenih konstanti. Posebno su važni podaci na osnovi kojih se provjeravaju metrike.

Velik utjecaj na produktivnost programera i projektanata softvera imaju CASE (Computer Aided System/Software Engineering) alati i tzv. generatori aplikacija. Oni omogućuju jednostavno specificiranje zahtjeva te dobivanje (skoro) gotovih aplikacija. Međutim i ovi alati imaju svoje nedostatke te su programeri i projektanti još uvijek oni koji proizvode, ako ne jednostavniji, onda manji po obimu i efikasniji softver.

LITERATURA:

1. Conte, Dunsmore, Shen: Software Engineering Metrics and Models, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, 1986.
2. Srića, V.: Uvod u sistemski inženjering, Informator Zagreb, 1988.
3. Radovan, M.: Projektiranje informacijskih sistema, Informator Zagreb, 1990.
4. Strahonja, Novak, Antonović: Razvoj informacijskih sistema, skripta Centra za permanentno obrazovanje Fakulteta organizacije i informatike Varaždin, 1989.

Primljeno: 1990-05-07

Oreški P. Software Metrics

SUMMARY

This paper is an introduction to software metrics. It discusses some factors and criteria of software valuation. A method for software valuation is explained. Basic ideas of some software metrics are also included.