UDK: 681.3206(083.7)
Stručni rad

O STANDARDIMA ZA RAZVOJ SOFTVERA
ABOUT SOFTWARE DEVELOPMENT STANDARDS

Mr. Predrag Oreški
Fakultet organizacije i informatike, Varaždin

U ovom radu nastoji se odrediti značaj standarda za razvoj softvera. Takvi standardi trebaju doprinijeti razvoju kvalitetnijeg softvera, a posljedica takvog softvera je jeftinije održavanje, odnosno veća produktivnost programera na održavanju softvera. Da bi se razvio kvalitetniji softver, razvijena je disciplina softversko inženjerstvo koje se bavi metodama i tehnikama za razvoj kvalitetnijeg i jeftinijeg softvera.

U radu su prikazane neke od tehnika softverskog inženjerstva, a opisuje se i jedan standard za razvoj softvera.

Klučne riječi: *softver; kvaliteta softvera; standard; održavanje; produktivnost; Def Stan 00-55*

In this paper author discusses about some software development techniques and standards. Software standards are supposed to be used in the development of higher quality software. The maintenance process of such a software is easier, faster id est cheaper. Software development standard Def Stan 00-55 is also presented.

Keywords: *Software, Software Quality, Standard, Maintenance, Productivity, Def Stan 00-55*

1. UVOD

Na osnovi jedne iste specifikacije zahtjeva softvera različiti timovi projektanata, programera i korisnika razvit će takve softvere kod kojih će strukture ulaznih podataka te točnosti izlaznih rezultata biti iste. Tako dobiveni softveri promatrani kao crne kutije vjerojatno će biti isti. Ono, što će se kod softvera promatranih kao bijele kutije razlikovati, bit će unutrašnje logičke strukture, razumljivost, pouzdanost, jednostavnost, veličina koda, mogućnost održavanja, ...

Zato se uvodi niz metoda, tehnika i pravila pomoću kojih bi različiti timovi projektanata, programera i korisnika razvijali i po unutrašnjim strukturama sličan softver. Pored

više tehnika (CM, QA, V&V, strukturni dizajn, strukturno programiranje) postoje i standardi[1] koji bi trebali u tom smislu utjecati na razvoj softvera informacijskih sustava.

Životni ciklus softvera je vremenski period između trenutka inicijalizacije njegovog razvoja pa do trenutka kada on više nije na raspolaganju. Jedna od faza u životnom ciklusu softvera je održavanje. U ovoj fazi nastoji se postići da u svakom trenutku postojeći softver maksimalno zadovoljava potrebama korisnika, odnosno informacijskog sustava. Postoji više različitih razloga za održavanje, pa zato i postoji slijedeća klasifikacija održavanja (GORLA 91):

- korektivno održavanje služi za uklanjanje grešaka u softveru koje su otkrivene od strane korisnika ili projektanata i programera. Ovdje su uključene i aktivnosti kao što su testiranje softvera, testiranje cijelog softverskog sustava te ažuriranje programske i korisničke dokumentacije;
- unapređujuće održavanje obavlja se uglavnom zbog promijenjenih informacijskih potreba korisnika (bilo zbog organizacijskih promjena ili zbog zahtjeva korisnika za drugim tipom informacija koje mu više odgovaraju);
- održavanje zbog djelotvornosti služi za povećanje djelotvornosti softvera u izvršavanju i za povećanje čitljivosti softvera (ostvaruje se restrukturiranjem softvera, poboljšanjem dokumentacije, itd.);
- preventivno održavanje očituje se u pisanju dodatnih programa koji služe za zaštitu podataka i za zaštitu od grešaka u izvršavanju softvera;
- adaptivno održavanje obavlja se kako bi se prilagodio postojeći softver novoj, naprednijoj informatičkoj tehnologiji (bolja strojna podrška, novi operativni sustav, novi alati, itd.).

U vezi s održavanjem softvera su troškovi održavanja. U nekim slučajevima (GORLA 91) čak 90% troškova životnog ciklusa softvera otpada na održavanje. Najveći utjecaj na troškove održavanja imaju slijedeći faktori (GORLA 91):

- nedostatak znanja korisnika o sustavu za koji se izrađuje softver. Zbog toga stalno nastaju dodatni zahtjevi korisnika za unapređujućim održavanjem. Također, ako korisnik nije obrazovan za rad sa softverom, ako ga ne poznaje dovoljno, tada će imati zahtjeve za dodatnim mogućnostima koje su inače ugrađene, ali ih on ne zna iskoristiti;

---

1 Standard se definira kao "1. (kao imenica) predmet, kvaliteta ili mjera koja služi kao osnova ili primjer ili princip kojeg se drugi trebaju pridržavati ili na osnovi kojeg se procjenjuju; zahtijevani ili specificirani stupanj odličnosti-vrsnoće; 2. 'Standardni' (kao pridjev) imati priznatu i stalnu (autoritativnu vrijednost)" (OXFORD 87); imenica "norma, obrazac, pravilo, prosjek, razina, osnovna mjera ili težina po kojoj se određuju druge mjere ili težine"; (pridjev) " koji se odnosi na standard, koji odgovara njegovim uvjetima; tipičan, jednotipan, normalan, klasičan, uzoran, temeljni, osnovni, uobičajen, općenito primljen, mjerodavan" (KLAIĆ 90).

- neefektivnost programera na održavanju softvera također ima velik utjecaj na troškove. Ukoliko programer nema dovoljno iskustva ili znanja, tada će održavanje zahtijevati puno vremena i troškova;
- loše tehnike programiranja i dokumentiranja imaju značajan utjecaj na proces održavanja. Ukoliko softver nije razumljiv, jednostavno napravljen, dovoljno opisan, odgovarajuće dokumentiran, tada će to značajno usporiti proces održavanja;
- programer ograničen rokovima pokušat će izvesti zadatak na održavanju brzo, što će možda i biti od koristi na kratki rok, međutim ovakav način rada može na dugi rok zahtijevati više korektivnog i unapređujućeg održavanja;
- neodgovarajući resursi za obradu podataka mogu zahtijevati od programera da segmentira neku veću obradu na više manjih obrada. Ukoliko se nabavi nova strojna podrška tada će svakako biti potreban ili novi softver ili adaptivno održavanje postojećeg. Postoji jasna veza između kvalitete [1]softvera, produktivnosti rada na održavanju i troškova održavanja:
- troškovi održavanja će biti mali ako produktivnost programera koji radi na održavanju softvera bude visoka.

Produktivnost programera koji radi na održavanju bit će visoka ako on u kratkom vremenu bude izvršio zadatak na održavanju. Programer će izvršiti zadatak održavanja u kratkom vremenu ako je softver kvalitetan. Softver je kvalitetan ukoliko zadovoljava zahtjevima, kao što su npr.: modularnost, kompletnost, razumljivost, jednostavnost, uspješnost, kompletna i ažurna korisnička i programska dokumentacija, itd[2].

Jedan od načina da se osigura određena kvaliteta softvera je taj da se definira kako treba, sa stajališta projektanata i programera (bijela kutija), izgledati softver i njegov razvoj[3].

Na slici 1., osim što je to ilustracija osnovne ideje Znanosti o softveru (Software Science, prof. Halstead, Purduec University), a koja govori o tome da se program od N1 operatora i N2 operanada konstruira izborom između m1 različitih operatora i m2

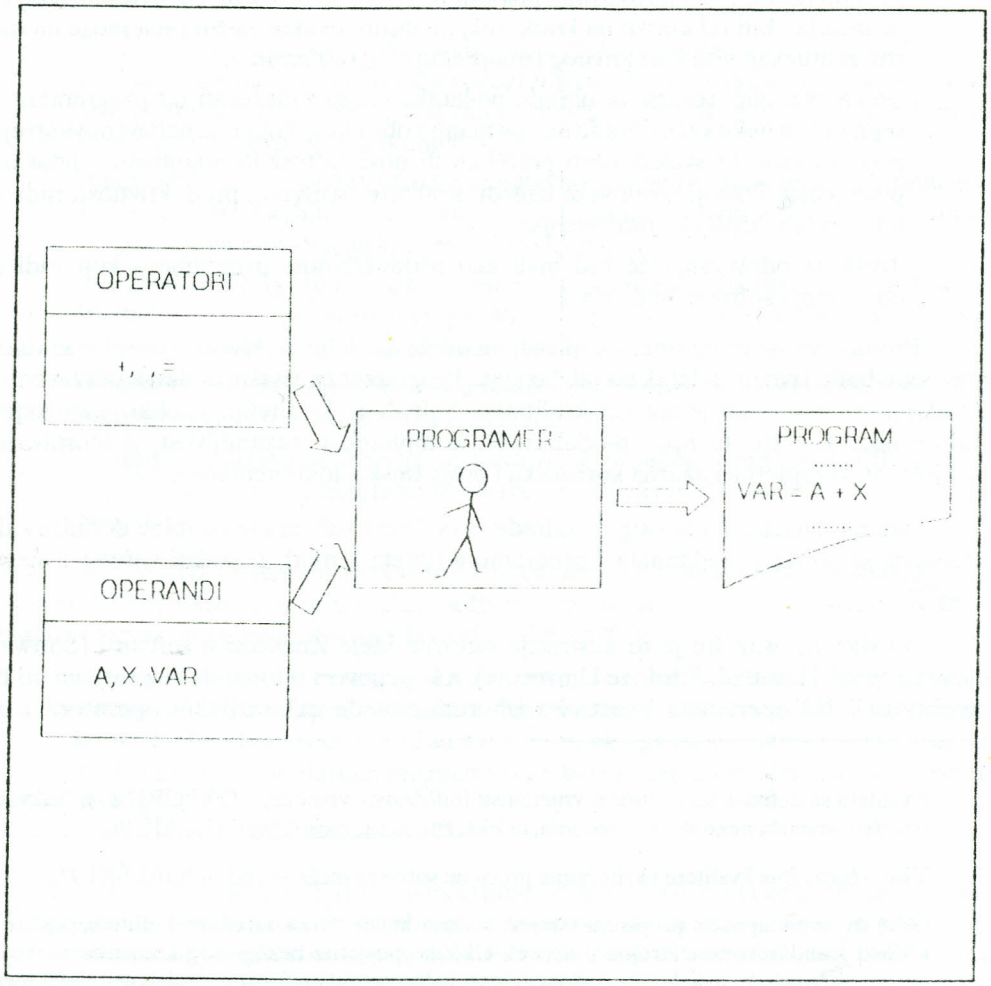
1 Kvaliteta se definira kao "stupanj vrijednosti (odličnosti, vrsnoće)" (OXFORD 87); "kakvoća, svojstvo; vrsnoća neke stvari; vrednota, odlika, značajka, sposobnost" (KLAIĆ 90).

2 Više o faktorima kvalitete i kriterijima procjene softvera može se naći u (OREŠKI 92).

3 Osim što se na taj način propisuju zahtjevi za izgradnju softvera određene kvalitete, postiže se i "efekt standardizirane strujne utičnice". Ukoliko posjetite nekog svog korisnika sa svojim lap-top računalom, uvijek ćete kod njega naći točno određeni format strujne utičnice i točno određeni napon. Samo su uređaji koji se priključuju različiti. Zašto tako ne bi bilo i sa softverom? Zašto aplikativni softver u bilo kojoj organizaciji ne bi imao sasvim određenu strukturu i način razvoja?

različitih operanada, jasno se vidi da karakteristike softvera ovise u velikoj mjeri o osobi koja izrađuje softver. Dakle, programiranje je dosta individualna stvar (bar prema toj ideji).

Vjerojatno je da će dva različita programera na različit način (gledano sa stajališta bijele kutije) riješiti isti programski zadatak.



Slika 1. Model procesa programiranja (Izvor: (CONTE 86), str. 80)

Ovaj problem pokušava se riješiti metodama i tehnikama softverskog inženjerstva[1].

2. O METODAMA I TEHNIKAMA SOFTVERSKOG INŽENJERSTVA

Zadatak softverskog inženjerstva je proizvesti kvalitetniji i jeftiniji softver. Da bi se to postiglo, razvijeno je više metoda, tehnika i postupaka.

Neki od njih su: ER analiza, metodologija strukturnog dizajna, top-down dizajn, HIPO dijagrami, strukturno programiranje, tehnika timovi s glavnim programerom, code walk-throughs, inspekcija koda, kontrolne točke, korištenje biblioteka programa, Configuration Management (CM), Quality Assurance (QA), Verification and Validation (V&V), standardi za razvoj softvera, ...

U nastavku su pobliže objašnjene neke od tehnika.

Verificiranje i vrednovanje (V & V) označavaju aktivnosti provjere da li softver u pojedinim fazama razvoja stvarno zadovoljava u odnosu na zadane ciljeve:

- verificiranje treba osigurati da softver bude u skladu s njegovom specifikacijom zahtjeva,
- vrednovanjem se treba ustanoviti vrijednost softvera za njegovo područje rada[2].

Zadatak aktivnosti V & V je (CHAR 86):

1 - pregled zahtjeva s obzirom na kompletnost, konzistentnost, izvodljivost i mogućnost testiranja,

2 - ako kriteriji kompletnosti, konzistentnosti, izvodljivosti i mogućnosti testiranja ne mogu biti izvedeni na nivou zahtjeva, tada se trebaju izvesti na osnovi specifikacije,

3 - odrediti da li specifikacije ispravno opisuju zahtjeve,

4 - odrediti da li dizajn korektno odgovara specifikaciji,

5 - odrediti da li dekompozicija dizajna ne utječe negativno na dizajn proizvoda,

6 - odrediti da li je svaka komponenta implementiranog proizvoda određena detaljnim dizajnom; odrediti da li proizvod korektno predstavlja implementirane zahtjeve i specifikacije.

---

1 "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.)" (STRAH 89)

2 Neformalno, definicije V&V mogu se prevesti u pitanja: Verificiranje: "Da li ispravno gradimo proizvod?", Vrednovanje: "Da li gradimo pravi proizvod?" (BOEHM 81).

Upravljanje konfiguracijom (CM) treba osigurati da se na projektu razvoja softvera u svakom trenutku zna što je potrebno dobiti (gotov proizvod ili neki od kontroliranih međuproizvoda). Ovo pomaže rukovodstvu da zna što je to što oni trebaju izgraditi, što oni u stvari izgrađuju i što je do sada izgrađeno. CM služi za povezivanje upravljanja i kontrole procesa razvoja softvera i upravljanja i kontrole softvera kao proizvoda. Kod toga se koristi tehnika kontrolnih točaka (milestones) i međuproizvoda (baselines).

Osiguranje kvalitete (QA) "uključuje razvoj i nadgledanje projektnih standarda i tehničke provjere softverskog proizvoda i procesa" (BOEHM 81). To je aktivnost koja pokušava osigurati da proizvod zadovoljava ili da bude bolji od specifikacije (CHAR 86). QA je definirana IEEE Standardom za kvalitetu softvera (IEEE Standard for Software Quality) 1981. godine kao primjer akcija koje su potrebne da se osigura odgovarajuće povjerenje da dijelovi ili cijeli projekt odgovaraju tehničkim zahtjevima.

Standardi za razvoj softvera i standardi za kvalitetu softvera još uvijek su slabo poznati ili se nedovoljno upotrebljavaju u procesu razvoja softvera.

3. PRIMJER STANDARDA ZA RAZVOJ SOFTVERA - Def Stan 00-55

Navedeni standard razvijen je za upotrebu u Ministarstvu obrane Velike Britanije za potrebe razvoja sigurnosno-kritičnog softvera. U vrijeme kada je objavljen članak (OULD 90), ovaj standard bio je predstavljen kao privremen, i to u tom smislu što se još uvijek radilo na njegovom razvoju, a predstavljena verzija bila je određena za "javnu raspravu". Sigurnosno-kritični (safety critical) softver koji je predmet razmatranja ovog standarda definira se u istom standardu. Sigurnosno-kritični softver je takav softver koji "mora biti bez opasnosti (grešaka) koje bi mogle nanijeti veliku štetu zdravlju (blagostanju) i sigurnosti krajnjem korisniku ili trećoj strani" (OULD 90).

Def Stan 00-55 ima tri glavna područja:

1 - Sigurnosno-upravljački postupci koji se moraju usvojiti,

2 - Postupci softverskog inženjerstva koji se moraju koristiti,

3 -Procedure u životnom ciklusu projekta koje se moraju pratiti.

U radu (OULD 90) razmatra se samo 2. područje. Jedan od ciljeva tog područja je razvoj softvera čije će se ponašanje moći u potpunosti analizirati i uspoređivati sa specifikacijom zahtjeva.

ZAHTJEVI STANDARDA Def Stan 00-55

Navedeno područje standarda prati životni ciklus razvoja softvera koji se može opisati slijedećim fazama:

1 - specificiranje sigurnosno-kritičnog softvera,

2 - dizajniranje,

3 - kodiranje,

4 - testiranje i

5 - vrednovanje sigurnosno-kritičnog softvera.

U prvoj fazi zahtijeva se da se napišu dvije specifikacije: jedna na "čistom engleskom" jeziku, te druga u kojoj se koriste formalne matematičke tehnike. Već je u ovoj fazi potrebno vrednovanje: da li formalno matematički prikazana specifikacija odgovara neformalno, prirodnim jezikom opisanoj specifikaciji. Ono što se zahtijeva u ovoj fazi vjerojatno još uvijek nije ostvarivo: zahtijeva se animacija formalno napisanih specifikacija i izrada prototipa, i to na potpuno automatizirani način.

Za drugu fazu, DIZAJNIRANJE, propisuju se odgovarajući postupci, a zabranjuju oni drugi. Zanimljivo je da se odgovarajuće (poželjne) tehnike i postupci tek površno nabrajaju, ali se zato one nepoželjne detaljnije navode. Dakle, ne smije se koristiti slijedeće:

- aritmetika s kliznim zarezom,
- rekurzija,
- korištenje jezika Assembler,
- multi-tasking (višezadaćni rad), itd.

Jednostavno objašnjenje zašto ove tehnike nisu zadovoljavajuće je u tome što softveri nastali njihovom primjenom nisu dovoljno "predvidljivog ponašanja". Cilj je standarda da se poveća vjerojatnost da će softver biti predvidljivog ponašanja.

Dokument Opis dizajna (Design Description) mora sadržavati slijedeća tri opisa:

- opis na nivou aplikacije (cijelog softverskog sustava),
- opis na nivou modula,
- argumente i dokaze za korektnost dizajna.

Za svaki modul mora uz formalnu specifikaciju biti i specifikacija na čistom engleskom jeziku. Za konstrukciju modula koristit će se formalna matematička specifikacija, a za provjeru služiti će specifikacija na engleskom jeziku.

U argumente i dokaze za korektnost dizajna uključeno je slijedeće: - svi dokazi i argumenti za korektnost koji su dobiveni u toku razvoja,

- izvještaj o provjeri korištenih algoritama,
- analiza veličine i utrošenog vremena,
- izvještaj o svakoj neodređenosti ili nekonzistentnosti koja je došla do izražaja u originalnoj specifikaciji.

U fazi KODIRANJE propisuju se čak i jezici koji se mogu, a koji se ne smiju koristiti. Zabranjen je Assembler, a dozvoljavaju se jezici visokog nivoa, kao što je Pascal. Jezik koji se koristi mora imati formalno definiranu sintaksu i dobro definiranu semantiku. Softver u odabranom jeziku mora biti podložan statičkoj analizi. To znači da se može razviti softver koji može provjeriti drugi softver i utvrditi i pronaći slijedeće:

- nepotrebne petlje, nedohvatljiv kod i kod s više ulaza,
- varijable koje se koriste prije nego što su definirane, ili varijable koje su definirane, ali se ne koriste,
- suvišne ili nedostajuće dijelove softvera koji se pozivaju između ulaza i izlaza softvera,
- nepodudaranje akcije koda i specificirane akcije, itd.

TESTIRANJE softvera obavlja se tehnikama crne kutije i slučajnim testiranjem. Testiranjem se mora postići slijedeće:

- svaka naredba mora se izvršiti,
- u svakoj IF-THEN-ELSE sekvenci mora se probno izvršiti i THEN i ELSE slijed naredbi,
- svaka petlja mora biti izvedena nijednom, jednom i mnogo puta (tamo gdje je to razumljivo).

VREDNOVANJE softvera kao proizvoda ostvaruje se u odnosu na specifikaciju na engleskom jeziku (formalna matematička specifikacija vrednovana je u odnosu na specifikaciju na engleskom jeziku).

4. ZAKLJUČAK

Niz tehnika i aktivnosti softverskog inženjerstva omogućuju izgradnju kvalitetnijeg softvera uz jedan multidisciplinarni i discipliniraniji pristup. Razvoj softvera prati se po fazama te se tako bolje nadgleda proces razvoja i utrošak resursa.

Standardi za razvoj softvera trebali bi propisati takve općenito priznate i prihvatljive zahtjeve u odnosu na proces razvoja softvera i karakteristike softvera koji bi osigurali kvalitetan proizvod. Standard za razvoj softvera Def Stan 00-55 čini jednu zaokruženu cjelinu zahtjeva i propisanih postupaka čijim bi se ispunjenjem trebalo doći do kvalitetne, pouzdane i predvidive sigurnosno-kritične aplikacije.

Ono što u tom standardu nije prihvatljivo je to što se predviđaju određeni postupci i alati koji još nisu na raspolaganju ili su još u eksperimentalnoj fazi. Možda je to i razlog zašto se standard naziva privremenim i što je na "javnoj raspravi".

Pitanju standarda za razvoj softvera trebalo bi posvetiti više pažnje. Naime, u literaturi s područja softverskog inženjerstva o standardima vrlo se malo govori[1], a trebalo bi puno više.

1 Primjer: u tri cjelokupna godišta časopisa IEEE Transactions on Software Engineering (1989., 1990., 1991.) doslovno nema ni jedne riječi o standardima za razvoj softvera (!).

LITERATURA:

1. **(BOEHM 81) Boehm, B.W.:** *Software Engineering Economics*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981.

2. **(CHAR 86) Charette, R.N.:** *Software Engineering Environments - Concepts and Technology*, Intertext Publications, Inc., McGraw- Hill, Inc., New York, 1986.

3. **(CONTE 86) Conte, S.D., Dunsmore, H.E., Shen, V.Y.:** *Software Engineering Metrics and Models*, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, 1986.

4. **(GORLA 91) Gorla, N.:** *Techniques for application software maintenance*, Information & Software Technology, Butterworth- Heinemann Ltd, volumen 33, broj 1, siječanj/veljača 1991.

5. **(KLAIĆ 90) Klaić, B.:** *Rječnik stranih riječi*, Nakladni zavod Matice Hrvatske, Zagreb 1990.

6. **(OREŠKI 92) Oreški, P.:** *Ocjenjivanje kvalitete aplikativne programske komponente informacijskih sistema*, magistarski rad, obranjen na FOI Varaždin, 1992.

7. **(OULD 90) Ould, M.A.:** *Software development under Def Stan 00-55: a guide*, Information & Software Technology, Butterworth- Heinemann Ltd, volumen 32, broj 3, travanj 1990.

8. **(OXFORD 87) ...** *The Oxford Dictionary of Current English*, Oxford University Press, 1987.

9. **(STRAH 89) Strahonja, V., Novak, I., Antonović, V.:** *Razvoj informacijskih sistema*, skripta Centra za permanentno obrazovanje Fakulteta organizacije i informatike Varaždin, 1989.