Predviđanje i osiguravanje kvalitete softverskog proizvoda informacijskom analizom izmjerenih podataka

Gordan Topić
ERP & Business Processes
Ericsson Nikola Tesla d.d.
Krapinska 44, Zagreb, Hrvatska
Telefon: +384 1 364 4618 Fax: +384 1 364 3219 E-mail: gordan.topic@ericsson.com

Sažetak – Za velike količine podataka potrebno je koristiti pomagala kako bi se omogućilo donošenje ispravnih logičkih zaključaka nužnih za održavanje i povećanje kvalitete informacijskog sustava. Nemogućnost odabiranja predikcijskih metoda na temelju neuronskih mreža koje koriste isključivo numeričke veličine, jedna od mogućih metoda predikcije kvalitete softvera je metoda informacijskim stablom odlučivanja. Kako bi se optimiziralo potrošnju razvojnih resursa, a u isto vrijeme održali zahtjevi o sukladnosti proizvoda te postigla planirana kvaliteta proizvoda, razvijena je metoda predviđanja kvalitete temeljena na informacijskoj analizi. Podaci su prikupljeni mjerenjem a analiza je provedena uz pomoć stabla odlučivanja, generiranog pomoću C5.0 algoritma. Algoritam je temeljen na informacijskoj teoriji, prikladan za stvaranjepravila i klasifikacijskog stabla sa sposobnošću korištenja svih tipova ulaznih podataka.

I. UVOD

Osiguravanje kvalitete proizvoda informacijskom analizom izmjerenih podataka, pripada planiranju kvalitete u fazi razvoja proizvoda, koja se temelji na mjerenju, prikupljanju i analizi rezultata mjerenja kvalitete prethodno razvijenih proizvoda. Drugim riječima, nastoji se pridobiti znanje o kvaliteti proizvoda na temelju prethodnih iskustava proizvodnje. U tu svrhu potrebno je izvoditi zahtjevno praćenje kvalitete putem mjerenja i promatranja rada sustava, na temelju kojih se mogu poduzimati korektivne akcije u budućoj proizvodnji. Velika količina podataka, dobivena mjerenjem i promatranjem sustava, ne može biti procesirana ljudskim umom na takav način, da se brzo i lako dođe do ispravnih logičkih zaključaka nužnih za održavanje i povećanje kvalitete sustava. Promatrati i utjecati na kvalitetu i rizike jednog velikog proizvodnog sustava zahtjeva velike analize i prikupljanja podataka. Iz tih podataka je nemoguće izvesti korektivnu radnju, a kamo li izvesti predikciju ponašanja takvog sustava promjenom parametara u cilju poboljšanja kvalitete. Nemogućnost odabiranja predikcijskih metoda na temelju neuronskih mreža koje koriste isključivo numeričke veličine, jedna od mogućih metoda predikcije kvalitete softvera je metoda informacijskim stablom odlučivanja.

Opisana metoda analize i predviđanja kvalitete softvera, temeljena je na informacijskoj analizi prikupljenih podataka mjerenja algoritmom C5.0. Algoritam se temelji na informacijskoj teoriji, gdje je entropija korištena kao mjera koja određuje koliko je koji podatak informativan za postizanje kvalitete proizvoda. Na taj način prikupljeni podaci se brže obrađuju i donose odluke, na temelju kojih se poduzimaju korektivne radnje u sustavu ili korekcije na proizvodu. Pristup omogućuje predikciju kvalitete, prije nego li je otpočeo razvoj softverskog proizvoda, budući je temeljen na analizi podataka mjerenja i ponašanja sustava u prethodnim razvojnim aktivnostima.

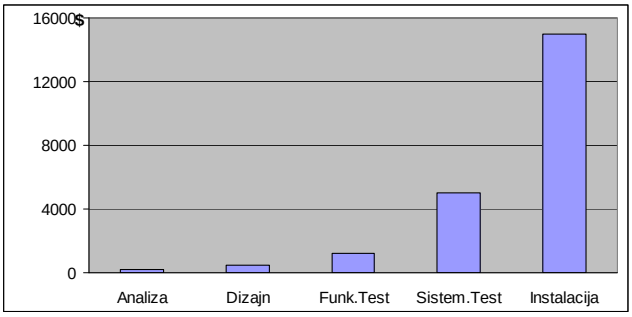
II. KLASIČNI PRISTUP OSIGURAVANJA I MJERENJA KVALITETE SOFTVERA

Proizvodnja softvera zahtijeva stalna mjerenja tijekom svih faza procesa proizvodnje. Nezamislivo je proizvesti kvalitetan softverski proizvod bez testiranja. Svaki softverski proizvod u svom proizvodnom ciklusu mora proći kroz osnovni test (*engl. Basic Test*) i funkcijski test (*engl. Function Test*), a projektna dokumentacija na temelju koje se softver proizvodi, mora proći inspekcije softverskih dokumenata. Inspekcije softverskih dokumenata (*engl. Software Document Inspections*) nužne su kako bi pogreške softverskog koda bile otkrivene u što ranijim fazama, što bitno smanjuje troškove proizvodnje, testiranja i održavanja softvera. Pod pretpostavkom da se inspekcije dokumenata pravilno obavljaju, bilježe rezultati, te vodi statistika inspekcija i testiranja, moguće je statistički predvidjeti koliko će pogrešaka ostati neotkriveno u isporučenom softveru. Tako je moguće proračunati i približni iznos troškova održavanja tog softvera. Nakon određenog broja projekata u sličnim uvjetima rada, moguće je u svakom nadolazećem projektu, grubo predvidjeti kvalitetu budućeg softvera u određenim proizvodnim okolnostima i dobiti elemente za analizu rizika [10]. Dosadašnja predviđanja kvalitete softvera temeljila su se na statističkim parametrima numeričke prirode poput rezultata inspekcija softverskih dokumenata, gustoće pogrešaka po volumenu koda, količine utrošenih radnih sati, troškova razvoja i sl.

A. Inspekcija dokumenata

Inspekcija softverskih dokumenata je formalna metoda za provjeru dokumenata, čitanjem i analiziranjem. Dokumenti koji se provjeravaju su implementacijski dokumenti koji prethode samom programiranju, kao npr. funkcijske specifikacije, prijedlog i razrada implementacije, test plan i sl. Inspekcijski proces razvijen je u IBM-u, gdje su inspekcije najrigorozniji formalni tip

pregleda i osvrta na dokumente. Osnovna svrha inspekcija dokumenata je uštedjeti i sačuvati vrijeme i novac dostizanjem ispravne kvalitete dokumenta. Oko 60% defekata na proizvodu pronađeni su prije početka kodiranja. Defekti pronađeni u ranim fazama lakše su i jeftinije ispravljivi, stoga se propušteni defekti nasljeđuju u slijedećim fazama projekta [2, 5]. Cijena defekata eksponencijalno raste s obzirom na razvojne faze proizvoda "Sl. 1." [9].



Sl. 1. Eksponencijalno rastući troškovi ispravljanja pogrešaka u različitim fazama razvoja softverskog proizvoda [9].

Svrha inspekcija nije samo traženje defekata u ranoj fazi razvoja softvera, već i smanjenje ovisnosti o testiranju, reduciranje održavanja softverskog proizvoda nakon isporuke, provjera kvalitete dokumenata, unapređivanje procesa, širenje tehničkog znanja i edukacija u cilju inspekcijske prakse. Svrha nije traženje novih tehničkih alternativa ili tehnička diskusija. U inspekcijama se prikupljaju podaci na temelju kojih može biti predviđeno koliko preostaje prikrivenih pogrešaka unutar kôda, te u kojim granicama bi se mogli kretati troškovi održavanja isporučenog proizvoda. Bilježi se: trajanje pripreme (*engl. preparation rate*), trajanje sastanka i brzina inspekcije (broj stranica/satu inspekcije) koja ovisi o sposobnosti apsorpcije informacija, tipu dokumenta, kompleksnosti inspektirane materije i broju ulaznih dokumenata. Također, bilježe se i prebrojavaju defekti u dokumentu koji mogu biti veliki (*engl. major*) i mali defekti (*engl. minor*). Neotkrivanje velikih defekata uzrokuje implementacijsku grešku ili grešku funkcionalnosti čija ispravka će koštati puno više u kasnijim fazama, dok mali defekt može biti razriješen kasnije, jer ne uzrokuje implementacijsku pogrešku ili povećava troškove projekta. Primjer malih defekata su npr. gramatičke pogrešaka. Za praćenje i predviđanje kvalitete softverskog proizvoda, promatraju se količine pronađenih velikih defekata, količina utrošenog vremena i brzina inspektiranja [2].

B. Izvještaji pogrešaka testiranja i gustoća pogrešaka

Jedna od metoda procjene kvalitete softverskog proizvoda temelji se na praćenju i prebrojavanju pogrešaka (*engl. Trouble Reports*) u procesima testiranja. Svaki pojedini softverski modul nakon uspješnog funkcijskog testa ima određeni broj prijavljenih pogrešaka. Iskustvo govori da se u funkcijskom testiranju otkriva 60% pogrešaka, u sistemskom testiranju 30%, a 10% pogrešaka pronalazi se u 6 mjeseci rada softvera nakon isporuke. Stoga se nastoji što više pogrešaka pronaći u funkcijskom testu, ali s druge strane teži se da takvih pogrešaka bude što manje. To se postiže većim angažmanom u inspektiranju projektne dokumentacije. Mjera koja opisuje kvalitetu svakog pojedinog modula naziva se gustoća pogrešaka (*engl. Fault Density-FD*). Gustoća pogrešaka je omjer pogrešaka izvještavanja nekog testa i volumena kôda izraženog u tisućama nekomentiranih linija. Određivanje razine kvalitete nekog softverskog proizvoda čini prosječna vrijednost svih gustoća pogrešaka po modulima, i ona je definirana tržištem ili planom kvalitete koji se temelji na analizama i strategiji poslovanja. Uz gustoću pogrešaka, u svrhu praćenja kvalitete, koriste se i mjere preciznosti troškova (*engl. Cost Planning Precision-CPP*), te mjera odstupanja u vremenu razvoja proizvoda (*engl. Lead Time Precision-LTP*). Mjere CPP i LTP, promatraju od početka razvoja softverskog proizvoda do završetka testiranja. CPP je relativno odstupanje u planiranoj i realiziranoj cijeni izraženoj kao čovjek-sat postocima, dok je LTP relativno odstupanje u realiziranim i planiranim danima također izraženo u postocima. CPP i LTP nisu korišteni u ovom radu za procjenu i analizu kvalitete softvera, dok je gustoća pogrešaka u funkcijskom testu korištena kao glavni kriterij kvalitete u informacijskoj analizi [2].

III. NADOGRADNJA KLASIČNOG PRISTUPA OSIGURANJA KVALITETE

Obzirom da softver zadire duboko u područje misli i ideje čovjeka, kvaliteta je usko povezana s cijelim nizom psiho-sociooloških faktora radne grupe i pojedinca. Ograničavanje na parametre isključivo tehničke prirode, bez da se uzmu u obzir psiho-sociološki faktori, vode nepouzdanom zaključivanju, a time i poduzimanju neadekvatnih korektivnih akcija [10].

A. Nenumerička priroda humanističkih parametara kvalitete softvera

Klasični pristup osiguranja kvalitete softvera nužno je nadopuniti parametrima poput razine stresa, kompetencije programera, radnog opterećenja, radnih uvjeta i okoline, motivacije i zadovoljstva, učestalih promjena zahtjeva na funkcionalnost, te mnogih drugih parametara koje se može proizvoljno odabrati prema intuiciji ili preuzeti iz dijagrama uzročnih veza. Takvi parametri čak ne moraju imati vidljivu uzročno-posljedičnu vezu s kvalitetom, međutim moguća važnost i uzročnost biva dokazana tek informacijskom analizom nad ispravno prikupljenim podacima. Ljudsko ponašanje, radne sposobnosti, intelektualni doseg, zadovoljstvo ili motivacija programera u različitim uvjetima rada jednostavnije je opisati riječima, nego brojevima. Podaci ovakve vrste mogu biti stvarani na temelju promatranja, mogu biti preuzeti iz dokumenata upravljanja ljudskim resursima ili lokalnih anketa napravljenih u svrhu osiguravanja kvalitete. Tako se dobivaju nenumerički parametri poput riječi ili čak simbola koji moraju biti ugrađeni u model analize i predikcije kvalitete softvera u fazi razvoja proizvoda. Sistemski pristup počiva na nužnosti obrade velikog opsega različitih informacija, na temelju kojih se donose odluke o akcijama u sustavu [6]. Stoga je problem naći primjeren način predviđanja kvalitete softverskog proizvoda ili metodu koja bi mogla obuhvatiti i parametre nenumeričke prirode, posebno kad je u promatranju

prisutan velik broj parametara. U tu svrhu potrebno je koristiti sustave potpore odlučivanju.

B. Sustavi potpore odlučivanju i informacijska analiza

Sustavi potpore odlučivanju (*engl. Decision Support System, DSS*) su računalni sustavi koji podupiru procese odlučivanja, tako što pomažu rukovodstvima organizacija u analizi i identifikaciji informacija potrebnih za donošenje odluka. Sustavi za potporu odlučivanju najkorisniji su u situacijama u kojima nije očito koje su informacije potrebne za odluku, te koje bi kriterije trebalo upotrijebiti pri odlučivanju. Ozbiljne organizacije razvoja softvera nastoje poduzimati što opsežnija mjerenja i praćenja kvalitete softverskog proizvoda i procesa. Pri tom se mogu pojaviti složene baze podataka iz kojih je nemoguće zaključivati s ciljem poboljšanja kvalitete proizvoda. Klasične statističke metode koje se koriste pri analizi ne daju odgovore na uzroke rasta ili pada kvalitete, niti daju mogućnost predviđanja kvalitete nekog budućeg proizvoda s obzirom na neke početne uvjete. Nova stepenica razvoja kod osiguranja kvalitete softverskog proizvoda teži korištenju informacijske analize. Pri tom nas metode rudarenja podataka mogu dovesti do znanja o njima. Rudarenje podataka (*engl. data mining*) je jedan od načina koji se koristi kao potpora odlučivanju. Rudarenje podataka obuhvaća metode ispitivanja povezanosti među podacima, klasificiranje podataka u srodne skupine, ispitivanje trendova i sl. Rudarenje podataka može koristiti klasifikaciju objekta u skupove izrazite i neizrazite logike.

C. Pristup rudarenju podataka i organizacija znanja izrazitom logikom.

Klasični pristup osiguranja kvalitete softvera koristi teoriju vjerojatnosti koja govori o vjerojatnosti pojave nekog fenomena u pripadnosti određenoj populaciji. Na primjer, analizom kvalitete programskih modula dobivena je vjerojatnost kvalitete određenog modula u nekom rasponu kvalitete. Međutim, ako se proširi sustav definicije kvalitete i iskoristi se izrazita logika, tada se ne govori o populaciji već o određenom individualnom objektu populacije. Izrazita logika opisuje karakteristike objekta kvalitete. Na primjer, modul je programirao ekspert u dovoljnom vremenskom roku, pa se za modul smatra da zadovoljava zadane kriterije kvalitete. Sve prikupljene informacije dobivene mjerenjem i promatranjem, nužno je prilagoditi informacijskoj analizi i rudarenju podataka na jedinstven način. Kao prvo, potrebno je definirati kriterij odluke, u ovom slučaju kategorije kvalitete softverskog proizvoda [1, 10].

IV. INFORMACIJSKA ANALIZA C5.0 ALGORITMOM

U svrhu informacijske analize nad prikupljenim podacima o kvaliteti, korišten je algoritam C5.0. Algoritam C5.0 je poboljšana i komercijalna verzija osnovnog ID3/C4.5 algoritma kojeg je razvio Quinlan u svrhu stvaranja klasifikacijskih modela iz podataka, zvanih također i stabla odlučivanja. Algoritam ID3/C4.5 služi za klasifikaciju podataka iz velikih i zašumljenih baza podataka, koristeći Shannonovu i Weaverovu teoriju informacije, te računanje informativnosti pomoću entropije. Podaci su organizirani u instance ili objekte, koji su predstavljeni kao fiksni skup atributa s pripadajućim vrijednostima, kojima je pridružena diskretna kategorija ili klasa. Zadatak algoritma je izgraditi stablo odlučivanja na temelju pitanja i odgovora koje predstavljaju atributi s vrijednostima. Stablo odluke se sastoji od čvorova, grana i završnih stanja ili listova stabla. Čvorove predstavljaju atributi, grane su njihove vrijednosti, a lišće su klase, kategorije odluke, tj. vrijednosti završnog, klasifikacijskog atributa na temelju kojeg se vrši odluka, npr. da li je modul kvalitetno programiran ili ne. Objekt sastavljen od atributa i vrijednosti predstavlja jednu od putanja po stablu, od korijenskog atributa do klase odluke. Da bi se stvorila odluka, kreće se od korijena stabla kojeg čini najinformativniji atribut , tj. atribut s najmanjom izračunatom entropijom. Zatim se granama, koje predstavljaju vrijednosti tog atributa, stiže do slijedećeg najinformativnijeg atributa, također s najmanjom izračunatom entropijom između preostalih atributa. Postupak se iterativno ponavlja sve dok ne preostanu čvorovi čije grane sadrže isključivo listove iste kategorije ili klase [3, 4].

Entropiju, kao mjeru nesređenosti ili kolebljivosti, uveo je u komunikacijske sustave Shannon. Entropija je mjera nečistoće ili homogenosti nekog odabranog podskupa unutar nekog skupa.

Kako je C5.0 usavršena inačica ID3/C4.5, postoji cijeli niz funkcionalnosti koje on obavlja:

- može koristiti atribute s kontinuiranim veličinama, datume, funkcije, itd.
- radi sa "zašumljenim" podacima i vrijednostima atributa koje nedostaju ili su neodređene,
- posjeduje algoritme skraćivanja stabala u svrhu dobivanja pouzdanijeg stabla,
- procjenjuje iznose pogrešaka kod donošenja odluka,
- izgrađuje pravila s mogućnošću spajanja više vrijednosti u jednu,
- koristi metode validacije i testiranja stabla odlučivanja,
- mogućnost korištenja više klasa ili kategorija odluke, kao i mogućnost fuzzy-evih skupova itd.

A. Definicija strukture objekta kvalitete i klasa odluke

Kvaliteta softvera u ovom primjeru ima tri kategorije ili klase odluke: kategoriju kvalitete, relativne kvalitete i nekvalitete, koje procijenjene na temelju pronađenog broja pogrešaka pojedinačnog modula nakon implementacije. Svaki objekt, tj. modul koji je uključen u informacijsku analizu striktno će pripadati jednoj od tih kategorija. Granicu među kategorijama ili definiciju klasa, određuje rukovodstvo za osiguravanje kvalitete i određena je temeljem broja pogrešaka nastalih u radu pojedinog softverskog modula. Redovita je projektna praksa u kojoj rukovodstvo unaprijed definira razinu kvalitete pojedinih programskih modula, prije nego li je započet njihov razvoj, a odnosi se na veličine vezane uz sistemsko testiranje. To je obično izraženo gustoćom pogrešaka po volumenu efektivnog kôda, iznad koje se programski modul smatra nekvalitetnim. Ako se uzme za primjer kvaliteta softverskog proizvoda, promatra se 12 parametara: iskustvo programera, njegova motiviranost za posao,

opterećenje izraženo u vremenskim rokovima i korištenje inspekcijskih praksi kod pisanja dokumentacije i ost. Ti parametri su atributi kojima opisujemo objekt promatranja kroz definirane klase, u ovom slučaju se promatra kvaliteta svakog softverskog modula, s obzirom na pronađeni broj pogrešaka u funkcionalnosti nakon njegove implementacije. Stoga, parametri kvalitete kojima opisujemo modul su atributi s njihovim vrijednostima, koji su pridruženi svakom modulu, a na temelju kojih će klasifikacija algoritmom biti izvršena. Nadalje svaki atribut može biti numeričke ili nenumeričke prirode. Atribut nenumeričke prirode definiran je određenim skupom opisnih vrijednosti koje može poprimiti i koje su unaprijed definirane (npr. opis nekog svojstva, dan u tjednu i sl.), dok numerički atributi koriste kontinuirane veličine, tj. poprimaju bilo koju numeričku vrijednost (npr. realni brojevi, temperatura i sl.). Za nenumeričke atribute, definiraju se opisne vrijednosti kao svojstva ili kriteriji tog atributa, na temelju kojeg odlučujemo da li je atribut, npr. programer , neiskusan, iskusan ili ekspert. Tako se definiraju atributi, uključujući i kategoriju kvalitete s klasama odluke, pa se dobiva Tablica 1 transformacije.

TABLICA I
TABLICA ATRIBUTA I NJIHOVIH VRIJEDNOSTI

ATRIBUT	VRIJEDNOSTI
INS_VRIJEME	kontinuirane
BR_STRANICA	kontinuirane
INS_DEFEKTI	kontinuirane
BRZINA_INS	kontinuirane
VOLUMEN	kontinuirane
IMPAKTIRANO	kontinuirane
RADNI_SATI	kontinuirane
F_TEST	kontinuirane
FD_STARI	kontinuirane
PROGRAMER	neiskusan, iskusan, ekspert
ROKOVI	dovoljni, kratki, stresno
MOTIVACIJA	motiviran, slaba
KVALITETA	**Kvalitetno, Relativno, Nekvalitetno**

Ovaj primjer nastoji koristiti sve moguće podatke koji su dostupni jednom koordinatoru kvalitete u odjelu razvoja softvera. I dalje je korištena metoda praćenja kvalitete svakog pojedinog modula, tj. softverskog bloka. Analiza je rađena na 20 objektnih zapisa koji predstavljaju pojedini softverski modul. Svaki modul temelji se na podacima preuzetim iz inspekcija softverskih dokumenata, podacima o samom bloku, te podacima o osobi koja je radila na dizajniranju modula. Podaci inspekcija dokumenata čine kontinuirane veličine i tu pripadaju: ukupno vrijeme potrošeno na inspekcije dokumenata potrebnih za budući dizajn funkcionalnosti modula, ukupni broj stranica dokumenata, broj pogrešaka, te brzina inspektiranja. Radi se o inspekciji dokumenata isključivo vezanih uz izgrađivanje novih funkcionalnosti. Podaci o modulu sastoje se od ključnih značajki svakog modula, također strukturiranih kao kontinuirane veličine: prvobitni volumen kôda prije nadogradnje, volumen ugradbenog kôda, broj utrošenih radnih sati na novu funkcionalnost, broj pogrešaka funkcijskog testa, te naslijeđena gustoća pogrešaka starog modula.

Dakle, svaki modul kao objekt sustava kvalitete, bit će opisan s 12 parametara kvalitete ili atributa, kojima će biti pridijeljena jedna od klasa kategorije kvalitete: kvalitetno, relativno kvalitetno ili nekvalitetno, u odnosu na definiranu granicu kvalitete. Korištene su i diskretne, tj. opisne veličine u posljednjoj grupi podataka o stresu, motivaciji i iskustvu programera, dok je na temelju završne gustoće pogrešaka modula rađena klasifikacija. Od ukupno 13 atributa, 9 ih koristi kontinuirane veličine, 3 diskretne, dok se klasifikacijski atribut sastoji od 3 klase. Klasifikacija kvalitete u ovom primjeru rađena je s pomoću 3 klasifikacijska skupa: kvalitetnim modulom se smatra modul s manje od 1 pogreške po 1000 nekomentiranih linija koda, relativno kvalitetnim se smatraju svi oni moduli čija je gustoća pogrešaka između 1 i 2 pogreške po 1000 linija nekomentiranog koda, a nekvalitetnim modulima smatraju se ostali moduli, tj. s gustoćom pogreške većom od 2 pogreške po 1000 linija nekomentiranog koda.

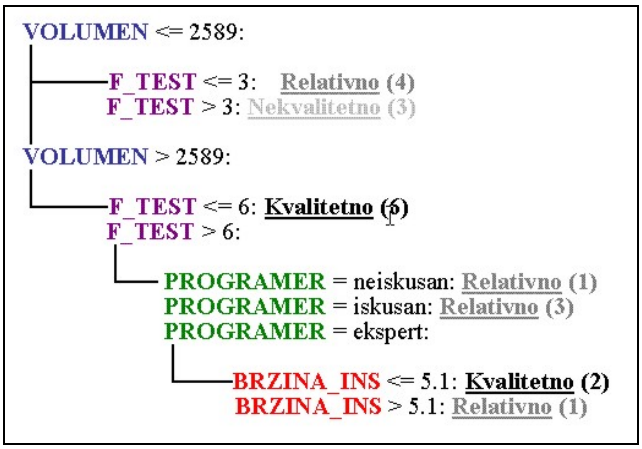
B. Stvaranje informacijske baze podataka

Nakon definicije klasa odluke, objektnih atributa i njihovih vrijednosti, primjenom tablice atributa i pravila definicije koja pridružuju vrijednosti atributima, moguće je iz baza podataka mjerenja i nadziranja sustava, načiniti novu bazu, tzv. bazu učenja (*engl. training set*), koja će biti pogodna za informacijsku analizu algoritmima izrazite ili neizrazite logike ID3/C4.5 ili C5.0, drugim riječima, rudarenju podataka. U ovom slučaju, sačinjena je baza od 20 objekata, tj. softverskih modula koji su opisani putem vrijednosti spomenutih 13 atributa i kojima je pridružena klasa, tj. kategorija kvalitete. Dakle, baza učenja nastala je na temelju prikupljanja podataka o kvaliteti 20 modula tokom inspekcija softverskih dokumenata, dizajna i testiranja. Sastoji se od slijedećih atributa: inspekcijskog vremena, broja stranica, broja velikih pogrešaka, brzine inspektiranja; naslijeđenog volumena koda, impaktiranog volumena koda, utrošenih radnih sati, broja pogrešaka u funkcijskom testu, naslijeđene gustoće pogrešaka, iskustva programera, rokova, motivacije programera i nove gustoće pogrešaka. Svaki opisani modul tvori jedan objekt zapisa (*engl. record*) i poželjno je da baza učenja sadrži što veći broj objektnih zapisa kako bi dobiveno znanje o sustavu bilo što točnije izvedeno. Primjenom informacijsko klasifikacijskog algoritma C5.0 nad kreiranom bazom, trenutačno se dobiva znanje nad podacima u obliku informacijskog stabla ili statističkih pravila.

C. Predikcija kvalitete izvedena na temelju generiranog stabla odlučivanja

Nakon analize algoritmom C5.0 dobiveno je stablo odlučivanja koje ne koristi sve atribute, već samo one atribute koji su bitni za donošenje odluke, dok ostali atributi ne igraju veliku ulogu u odlučivanju. Kao što se vidi, stablo se sastoji od 4 razine, 4 čvorova i 7 završnih skupova odluke. U zagradama završnih skupova naznačena je količina objekata ili zapisa čija putanja vodi do tog klasifikacijskog skupa, tj. lista stabla. Ovako generirano stablo odlučivanja može izvrsno poslužiti za

predikciju kvalitete softvera već kod samog planiranja dizajna modula "Sl. 2.".



Sl. 2. Klasifikacijsko stablo generirano C5.0 algoritmom

Za primjer je odabran softverski modul od 3000 linija nekomentiranog kôda: postavlja se pitanje, kako pristupiti dizajniranju takvog modula, a da on statistički zadovoljava kvalitetu? Kako algoritam C5.0 unutar sebe ima i mogućnost testiranja stabla odlučivanja, unoseći poznate vrijednosti budućeg modula dobivaju se informacije o vjerojatnoj razini kvalitete nakon isporuke modula Tablica II. U prvom slučaju predikcije vidi se da modul od 3000 linija (softverska dokumentacija je prošla inspekciju brzine 4 stranice/satu), kojeg programira ekspert, postiže

TABLICA II
PRIMJER PREDIKCIJE KVALITETE GENERIRANIM STABLOM ODLUČIVANJA

#	Vol.	Funk. test	Kompet. programera	Brzina inspekc.	Kvaliteta
1	3000	?	ekspert	4 str/satu	Kvalitetno [68%]
2	3000	7 pogr.	ekspert	4 str/satu	Kvalitetno [60%]
3	3000	?	iskusan	–	Kvalitetno [36%] Relativno [36%]
4	3000	?	neiskusan	–	Kvalitetno [36%] Relativno [27%]
5	3000	?	?	4 str/satu	Kvalitetno [40%] Relativno [19%]

vjerojatnost od 68%. U slučaju da se pretpostavi količina od 7 pogrešaka pronađenih u funkcijskom testu, vjerojatnost kvalitete će pasti za 8%. U slučaju da se zadatak dizajniranja modula pridijeli programeru s manje iskustva, vjerojatnost kvalitete i dalje će padati. Budući da su nepoznate vrijednosti nekih atributa, niti vjerojatnost kvalitete neće biti pouzdana. Na primjer u slučaju da se ne poznaje iskustvo i kompetencija programera, kao u slučaju 4, uz nešto formalnije inspekcije od 4 str/satu, vjerojatnost isporuke zadovoljavajuće kvalitete modula bit će 40% s vjerojatnošću relativne kvalitete od 19%. Na sličan način može se vrlo brzo izvesti predikcija bilo kakvog modula, s poznatim i nepoznatim vrijednostima atributa, što je jako povoljno za upravljanje rizicima, planiranje resursa u projektu, kao i kvalitete, prije nego li je i otpočela faza dizajna.

D. Derivacija klasifikacijskih pravila iz stabla

Klasifikacijska pravila (*engl. Classification rules*) predstavljaju alternativu stablu odlučivanja. Jednom kad je stablo odlučivanja stvoreno, vrlo ga je jednostavno pretvoriti u sustav pravila. Pretvaranje stabla u sustav pravila ima više prednosti, a kao prva je preglednost i čitljivost, budući su pravila puno lakša za percepciju i shvaćanje, nego li je to samo stablo, naročito kad se radi o velikim i kompliciranim stablima. Tehnika se sastoji u trasiranju svakog mogućeg puta u stablu odlučivanja, gdje se serije testova bilježe kao preduvjeti, a završni čvorovi kao zaključci koji daju klasu koja zadovoljava pravilo. Nepotrebni preduvjeti se eliminiraju u svrhu eliminiranja pravila korištenjem raznih statističkim metodama; Hi-kvadrat test, Fisherov egzaktni test, Yatesova korekcija kontinuiteta, konstruiranje Tablica kontigencije ili jednostavno, eliminacija preduvjeta koji nemaju efekta ili se pojavljuju samo jednom. Stablo odluke je jednostavno transformirati u sustav pravila, kao i pridodati novo pravilo skupu pravila, ali je gotovo nemoguće iz klasifikacijskih pravila u stablo odluke. O odabranoj tehnici generiranja pravila ovisi kolika će biti pouzdanost generiranih pravila. Iz navedenog primjera stabla odlučivanja izvedeno je 5 klasifikacijskih pravila navedenih u Tablici III.

TABLICA III
KLASIFIKACIJSKA PRAVILA DERIVIRANA IZ STABLA

#	Klasifikacijska pravila	Kvaliteta	Vjerojatnost
1.	VOLUMEN > 2489 F_TEST <= 6	Kvalitetno	87.4 %
2.	BRZINA_INS <= 4.1 PROGRAMER = ekspert	Kvalitetno	80 %
3.	VOLUMEN <= 2489 F_TEST <= 3	Relativno	83.3 %
4.	F_TEST > 6	Relativno	66.7 %
5.	VOLUMEN <= 2489 F_TEST > 3	Nekvalitetno	80 %

Na temelju prva dva pravila može se zaključiti da kvaliteta softverskog proizvoda u opisanom projektnom timu ovisi o broju pogrešaka pronađenih u funkcijskom testu, koji ne smije za velike module prelaziti broj 6. Inspekcije bi se trebale pažljivo pripremati tako da se pokuša osigurati brzina inspekcija manja od 4 stranica po

satu, za eksperte. Za ostale programere, potrebno je raditi detaljnije inspekcije s još manjom brzinom inspektiranja. Iz stabla je vidljivo da relativno mali moduli mogu biti kvalitetni, jedino ako broj pogrešaka funkcijskog testa ne prelazi broj 3, dok za veće vrijednosti pada kvaliteta. Veliki moduli ovise također o pogreškama funkcijskog testa, ali i o iskustvu programera, te pažljivim softverskim inspekcijama dokumenata nad opširnom dokumentacijom [3, 4, 8, 11].

V. ZAKLJUČAK

Dobar sustav sam sebe održava. Da bi se sustav dovelo na takvu organizacijsku razinu, potrebno je znanje o njegovim životnim ciklusima i znanja koja nam omogućuju da takve sustave mjerimo, simuliramo i predviđamo. Praćenje životnih ciklusa softvera i njihovo umjeravanje je nezahvalno, jer ciklusi mogu trajati mjesecima i godinama, često se u pojedinim fazama mijenjaju, iskaču iz pravila ponašanja, prilagođavaju se novim uvjetima, a sve to otežava dobivanje prave slike o tim ciklusima. Za predviđanje je potrebna statistika, što znači dovoljna količina podataka o ciklusima. Velika količina podataka statistički se relativno lako obradi i može dati dobre pokazatelje sustava, ali ne daje smjernice što činiti da bi se nešto promijenilo, jer previše je parametara koji se mogu mijenjati i utjecati na kvalitetu.

Obrađen je primjer pripreme podataka i njihove informacijske analize putem C5.0 algoritma, koji pripada metodama klasifikacije u izrazite skupove. Algoritmom se dobiva klasifikacijsko stablo odlučivanja i pravila. Prednosti metode analize i predikcije stablom odlučivanja su:

- sposobnost generiranja razumljivih modela,
- relativno mali zahtjevi na računalne resurse,
- sposobnost korištenja svih tipova atributa,
- stabla odlučivanja jasno odražavaju važnost pojedinih atributa za konkretni klasifikacijski, odnosno predikcijski problem,
- laka priprema i manipulacija podacima preuzetim iz poslovnih sustava.

Međutim, postoje i neka ograničenja i mane ovakve metode, stoga kod pripreme podataka i analize sustava treba imati na umu da su stabla odlučivanja:

- manje prikladna za probleme kod kojih se traži predikcija kontinuiranih vrijednosti ciljnog atributa,
- sklona greškama u višeklasnim problemima s relativno malim brojem primjera za učenje modela, tj. objekata,
- mogu biti računalno zahtjevan problem, npr: gdje je potrebno generirati velik broj stabala za izbor najpovoljnijeg rješenja,
- nisu dobro rješenje za probleme kod kojih su regije određenih klasa omeđene nelinearnim krivuljama u višedimenzionalnom prostoru.

Skup podataka za učenje, odnosno tablica atributa i njihovih vrijednosti trebala bi sadržavati puno više elemenata, bili to atributi, njihove vrijednosti ili objekti koji tvore zapise tablice o kojima ovisi pouzdanost generiranog stabla odlučivanja. Tim osiguranja kvalitete mora imati visoke i široke kompetencije kako bi što bolje mogao obuhvatiti uzroke koji utječu na promjenu kvalitete i izgraditi relativno pouzdani model pri donošenju odluka i predviđanju buduće kvalitete proizvoda [3, 4, 6].

LITERATURA

[1] Enterprise Storage Group, june 2003, Gordan Topić and Dragan Jevtić, *Software Quality Prediction Based on Information Analysis – A Decision Tree Approach*, Proceedings of the 11th International Conference on Software, Telecommunications and Computer Networks - SoftCOM 2003, pp. 277-281, Split, Venice, Ancona, Dubrovnik, Croatia, 2003.

[2] Tom Gilb and Dorothy Graham, *Software Inspections*, Addison Wesley Longman Limited, 1993.

[3] UGAI Lectures, Workshop: *Building Classification Models: ID3 and C4.5*, Providing and Integrating Educational Resources for Faculty Teaching Artificial Intelligence, Temple University in Philadelphia, June 20 - June 24, 1994.

[4] Tutorial: *Decision Trees: ID3*, Monash University, Faculty of Information Technology, CSE4230 Data Mining, Semester 2, 2002.

[5] Measurement: 3-day training, Q-Labs Presentation Template: *Managing Software Risks with World -Class Customers*, Conducted by Erik Johansson, Johan Brantestam.

[6] Lucas Ballard, "Topics in Machine Learning – Decision Tree Learning", Web Project September 18, 2002, http://www.cs.brandeis.edu/~cs113/

[7] Osmar R. Zaiane, *Principles of Knowledge Discovery in Databases*, Chapter 7: "Data Classification", University of Alberta, 1999.

[8] Siniša Sovilj, *Otkrivanje znanja u skupovima podataka – seminarski rad*, FER, Sveučilište u Zagrebu, rujan 2003.

[9] William E.Perry, *Effective Methods for Software Testing, Second Edition*, Published by John Wiley & Sons, Inc.

[10] Gordan Topić, *Metode planiranja i nadzora proizvodnje softvera i procesa s ciljem povećanja kvalitete*, XXVII Međunarodni skup MIPRO, Opatija 2004, Zbornik radova savjetovanja: Telekomunikacije i informacije, ISBN 953-233-002-X str. 106 - 111.

[11] Rulequest Research, Effective data mining tools, Data Mining Tools See4 and C5.0 – Help, http://www.rulequest.com/see5-info.html