Alati za podršku održavanju softvera

Žagar, Marko

Master's thesis / Diplomski rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:422382

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-26**

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Informatics and Digital Technologies - INFORI Repository

zir.nsk.hr

DIGITALNA KNJIŽNICA

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

Sveučilište u Rijeci – Odjel za informatiku

Diplomski studij informatike - modul Informacijski i komunikacijski sustavi

Marko Žagar

Alati za podršku održavanju softvera

Diplomski rad

Mentorica: izv. prof. dr. sc. Sanja Čandrlić

Komentor: Perino Krneta

Rijeka, Lipanj 2019.

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 7.6.2019.

Zadatak za diplomski rad

Pristupnik: Marko Žagar

Naziv diplomskog rada: Alati za podršku održavanju softvera

Naziv diplomskog rada na eng. jeziku: Tools for software maintenance support

Sadržaj zadatka:

U životnom ciklusu softvera održavanje nosi najveći dio ukupnih troškova. Održavanje uključuje sve izmjene softverskog proizvoda nakon isporuke korisniku s ciljem ispravljanja pogreške, poboljšanja performansi te unaprjeđenja softverskog proizvoda prema novim zahtjevima.

Kako bi se olakšala ova zahtjevna faza, tijekom održavanja mogu se koristiti alati koji olakšavaju taj proces. Student će detaljno analizirati i opisati fazu održavanja te analizirati i opisati jedan odabrani alat za podršku održavanju softvera.

Mentor:

Izv. prof. dr. sc. Sanja Čandrlić

Komentor:

Perino Krneta

Voditeljica za diplomske radove:

Izv. prof. dr. sc. Ana Meštrović

Zadatak preuzet: 10.6.2019.

Marko Žagar

(potpis pristupnika)

Sadržaj

Sažetak

U suvremenom razvoju softvera, održavanje čini najveći dio ukupnih troškova softverskog projekta. Održavanje se odnosi na sve izmjene softverskog proizvoda nakon isporuke korisniku kako bi se ispravile pogreške, poboljšale performanse te unaprijedile karakteristike softverskog proizvoda prema novim pravilima. U radu je detaljno opisan softverski program ESTIMUS koji daje podršku procesima održavanja te je opisan sustav održavanja softvera za poslovne aplikacije. Prikazane su kategorije i modeli održavanja, prikazan je proces održavanja te je prikazano upravljanje procesom održavanja. U okviru rada napravljeno je istraživanje o upotrebi alata za podršku održavanju softvera u informatičkim tvrtkama na području Republike Hrvatske. Također su analizirani i prikazani rezultati navedenog istraživanja.

Ključne riječi

Alati za podršku održavanju softvera, kategorije održavanja softvera, modeli održavanja softvera, proces održavanja softvera, upravljanje održavanjem.

1. UVOD

Proces razvoja softvera rezultira isporukom softverskog proizvoda koji zadovoljava korisničke zahtjeve. Jednom kada se softverski proizvod počinje koristiti, otkrivaju se njegovi nedostaci, mijenjaju se radna okruženja te se pojavljuju novi korisnički zahtjevi.

Prema tome, može se zaključiti da se softver konstantno razvija i dokle god je on u upotrebi, njegov razvoj nikada neće završiti. U današnjem brzo mijenjajućem tehnološkom svijetu, svaki softver zahtijeva neprekidno održavanje kako bi bio u koraku s novim trendovima, tehnologijama, korisničkim zahtjevima te rastućom bazom korisnika. Izmjene postojećih poslovnih procesa te uvođenje novih, kao i promjene očekivanja korisnika stvaraju nove zahtjeve za postojeći softver. Dijelovi softvera se trebaju modificirati kako bi se ispravile pogreške koje se pronalaze u radu softvera, kako bi se softver prilagodio promjenama na hardverskoj i softverskoj strukturi, kako bi se poboljšala njegova učinkovitost te optimizirala potrošnja memorije. Ovo su samo neki primjeri uzroka neprekidnog održavanja softvera, kojima se želi naglasiti važnost održavanja u životnom ciklusu softvera te opravdati trošak održavanja. Održavanje softvera je jako važno jer organizacije ulažu velike količine novca u svoj softver i njihovo poslovanje, a samim tim i njihova profitabilnost i konkurentnost, ovisi o radu tog softvera. Većina velikih tvrtki troši više na održavanje postojećih sustava nego na razvoj novih sustava. Neke studije pokazuju da troškovi održavanja iznose od minimalno 50% do 85% svih troškova tijekom životnog ciklusa softvera [1], [2], [3], [4].

Alati za podršku održavanju softvera su softverski proizvodi koji pojednostavljuju zadatke održavanja softvera, povećavaju učinkovitost i produktivnost, podržavaju evoluciju cijelog softverskog sustava, te u konačnici minimiziraju troškove održavanja softvera.

U prvom poglavlju daju se uvodne informacije o procesu održavanja softvera, prikazuju se temeljne odrednice rada, opisan je problem istraživanja, definirani su ciljevi istraživanja, postavljene su hipoteze istraživanja, navedene su znanstvene metode koje se koriste u istraživanju te je opisan uzorak ispitanika i mjerni instrument.

U drugom poglavlju definirani su i objašnjeni teorijski pojmovi vezani uz održavanje softvera. Kategorizirano je održavanje softvera, prikazani su modeli održavanja softvera, objašnjen je proces održavanja i proces upravljanja održavanjem. Kategorizirani su i alati za podršku

održavanju softvera te je detaljno opisan alat ESTIMUS, koji pruža podršku procesima održavanja softvera.

Treće poglavlje sadrži rezultate istraživanja i analizu dobivenih rezultata. Analiziraju se odgovori na pitanja iz upitnika, provjeravaju se hipoteze istraživanja te razmatraju dobiveni rezultati.

Četvrto poglavlje je posljednje poglavlje ovog rada. U njemu se iznose zaključci iz teorijskog i istraživačkog djela rada.

Na kraju rada nalazi se popis literature i izvora, popis slika i tablica te, kao prilog, anketni upitnik koji se koristio u ovom istraživanju.

1.1. OPIS PROBLEMA

Održavanje softvera je i dalje među najkritičnijim pitanjima s kojima se suočavaju sve informatičke tvrtke koje se bave razvojem softvera. Programeri se suočavaju sa sve većim zahtjevima za novim softverskim aplikacijama, dok u isto vrijeme moraju posvetiti znatne resurse postojećim aplikacijama. U održavanju većinom rade djelatnici koji nisu razvijali softver, pa se puno vremena troši na prikupljanje informacija i razumijevanje onoga što postojeći sustav radi. Proces održavanja otežava i preopterećenost informacijama, gdje djelatnik u održavanju ima poteškoća s identificiranjem ključnih dijelova softvera jer se složenost softvera konstantno povećava i djelatniku se prezentira previše informacija. Također, teško je predvidjeti i razumjeti kako će izmjena jedne značajke utjecati na rad druge značajke softvera.

Jedan od najvećih izazova u održavanju softvera je utvrđivanje učinaka predložene modifikacije na ostatak sustava. Analiza izvodljivosti je aktivnost procjene potencijalnih učinaka promjene s ciljem minimiziranja neočekivanih nuspojava. Zadatak uključuje procjenu prikladnosti predložene izmjene i procjenu rizika povezanih s provedbom izmjene, uključujući procjene učinaka na resurse, napore i raspoređivanje. Uključuje i identifikaciju dijelova sustava koji se trebaju modificirati kao posljedica predložene izmjene [5].

Među najizazovnijim problemima održavanja nalazi se i kreiranje testnih scenarija, regresijsko testiranje te testiranje izmjena. Testiranje je proces izvođenja softvera ili komponente na

umjetno pripravljenim podacima, pod strogo definiranim uvjetima, uz pažljivo analiziranje rezultata [6], [7]. Svrha testiranja može biti verifikacija ili validacija. Verifikacija osigurava da konačni proizvod odgovara originalnom dizajnu, a validacija provjerava zadovoljava li konačni proizvod ono za što bi se trebao koristiti [6], [7]. Regresijskim testiranjem pronalaze se greške koje se pojavljuju nakon velikih promjena koda te se utvrđuje utječu li promjene negativno na postojeće funkcije softvera.

Navedeni problemi stvaraju potrebu za alatima koji će pomoći u održavanju softvera kako bi se smanjilo vrijeme održavanja, povećala efikasnost i poboljšala produktivnost održavanja softvera, te u konačnici, kako bi se unaprijedio proces razvoja novog softvera. Povećanjem efikasnosti procesa održavanja, djelatnicima se omogućava da veći dio svog vremena provode na projektima razvoja novog softvera, dok se u isto vrijeme informacije prikupljene tijekom održavanja softvera koriste kao ulazni podaci za poboljšanje procesa razvoja novog softvera i to na način da se u metodiku razvoja softvera ugrađuju potrebne modifikacije kako bi novoproizvedeni softver bio otporniji na promjene.

1.2. CILJEVI ISTRAŽIVANJA

U informatičkim tvrtkama kod kojih veliki broj djelatnika radi na poslovima održavanja i koji održavaju veliki broj softvera, nije moguće kvalitetno i efikasno upravljati procesom održavanja softvera bez alata koji podupire taj proces. Informatičke tvrtke mogu koristiti komercijalno dostupne alate za podršku održavanju softvera, ali mogu i razviti vlastite za alate za podršku održavanju softvera. Glavni cilj istraživanja je utvrditi u kojoj mjeri informatičke tvrtke na području Republike Hrvatske koriste alate za podršku održavanju softvera. Uz glavni cilj definirani su i sekundarni ciljevi:

- utvrđivanje preferencija tvrtki prema vlastitim ili prema komercijalnim softverskim alatima za podršku održavanju softvera;
- otkrivanje najpopularnijih komercijalnih alata za podršku održavanju softvera u informatičkim tvrtkama na području Republike Hrvatske;
- utvrđivanje za koje se procese i aktivnosti održavanja se najviše koriste alati za podršku održavanju softvera;
- otkrivanje trenda korištenja analitičkih alata;

- utvrđivanje u kojoj mjeri alati za podršku održavanju softvera uključuju i analitičke alate.

1.3. HIPOTEZE

Nakon opisa problema istraživanja i definiranja ciljeva istraživanja postavljene su i istraživačke hipoteze. Potrebno je potvrditi ili opovrgnuti navedene hipoteze u nastavku istraživanja.

H1: Većina informatičkih tvrtki u Republici Hrvatskoj koristi vlastito softversko rješenje za podršku održavanju softvera

H2: Informatičke tvrtke koje koriste vlastito softversko rješenje za podršku održavanju softvera većinom imaju analitički sustav uključen u taj softver

H3: Korištenje analitičkih sustava povećalo se posljednjih deset godina

1.4. ZNANSTVENE METODE

U svrhu ovog istraživanja koristi se sljedeće znanstvene metode [8]:

- Induktivna i deduktivna metoda – postupak gdje se na temelju pojedinačnih činjenica dolazi do općeg zaključka, odnosno gdje se do pojedinačnih činjenica dolazi na temelju općih postavki.
- Metoda analize je postupak raščlanjivanja složenih pojmova, sudova i zaključaka na jednostavnije sastavne dijelove te izučavanje svakog dijela za sebe i u odnosu na druge dijelove, odnosno cjeline.
- Metoda sinteze je postupak znanstvenog istraživanja putem spajanja dijelova ili elementa u cjelinu.
- Metoda deskripcije je postupak jednostavnog opisivanja ili očitovanja činjenica, procesa i predmeta te njihovih potvrđivanja odnosa i veza.
- Metoda klasifikacije je postupak sistematske i potpune podjele općega pojma na posebne, odnosno jednostavnije pojmove, u okviru opsega pojma.

- Metoda kompilacije je postupak preuzimanja tuđih rezultata istraživačkog rada, odnosno tuđih opažanja, stavova, zaključaka i spoznaja.
- Metoda anketiranja - postupak kojim se na temelju anketnog upitnika prikupljaju i istražuju podaci, informacije, mišljenja i stavovi o predmetu istraživanja.
- Statistička metoda je induktivno generalizacijska metoda gdje se na temelju obilježja određenog broja elementa neke skupine ili serije pojava izvode generalni zaključci o prosječnoj vrijednosti obilježja, njihovoj devijaciji od neke sredine u cijeloj masi ili skupini pojava.
- Metoda dokazivanja/opovrgavanja je postupak kojim se utvrđuje, odnosno pobija istinitost pojedinih spoznaja, stavova ili teorija. Traže se pretpostavke koje određenu hipotezu dokazuju, odnosno opovrgavaju.
- Metoda vizualizacije je proces grafičkog prikazivanja statističkih podataka, pomoću čega se omogućuje interpretacija rezultata statističkog istraživanja.

Navedenim metodama u radu su opisani osnovni pojmovi održavanja softvera, navedeni su modeli održavanja softvera, navedene su funkcionalnosti alata za podršku održavanju softvera, napravljeno je istraživanje te su analizirani rezultati istraživanja. Također, opisana je i klasifikacija održavanja softvera, objašnjen je proces održavanja softvera i proces upravljanja održavanjem, te je analiziran alat za podršku održavanja softvera ESTIMUS.

1.5. UZORAK ISPITANIKA

Istraživanje o upotrebi alata za podršku održavanju softvera provedeno je u informatičkim tvrtkama na području Republike Hrvatske. S popisa top 1000 hrvatskih visoko-tehnoloških tvrtki [9], za anketiranje odabrano je 300 tvrtki, koje imaju kao opis šifre djelatnosti navedeno računalno programiranje, upravljanje računalnom opremom i sustavom te obrada podataka, usluge poslužitelja i djelatnosti povezane s njima. E-mail adrese odabranih tvrtki prikupljene su s web stranica tih tvrtki. Anketiranju je pristupilo 45 tvrtki, što znači da je odaziv 15%.

1.6. MJERNI INSTRUMENTI

Za svrhu istraživanja sastavljen je web upitnik koji se sastojao od kratke upute na početku i pet pitanja. Anketni upitnik je sadržavao pitanja otvorenog i zatvorenog tipa, napisan je na hrvatskom jeziku, a distribucija upitnika vršila se elektroničkim putem. Anketa je u potpunosti anonimna. Anketa je napravljena i provedena pomoću online alata „Google obrasci“. Istraživanju se moglo pristupiti od 1. Ožujka 2019. godine do 31. Svibnja 2019. godine.

2. ODRŽAVANJE

U programskom inženjerstvu, održavanje softvera je proces poboljšanja i optimizacije implementiranog softvera, kao i otklanjanja njegovih nedostataka. Održavanje softvera jedna je od faza u procesu razvoja softvera, slijedi nakon uvođenja softvera u svakodnevni rad i traje sve dok je softver u upotrebi. Faza održavanja softvera uključuje promjene softvera kako bi se ispravile greške i nedostaci pronađeni tijekom korištenja te dodale nove funkcionalnosti za poboljšanje upotrebljivosti i primjenjivosti softvera.

Opće prihvaćena definicija održavanja, koja proizlazi iz IEEE standarda, glasi:

> „*Održavanje softvera podrazumijeva izmjenu softvera nakon isporuke kako bi se ispravili nedostaci (pogreške), poboljšale performanse ili drugi atributi ili prilagodba softvera izmjenama u okolini*“ [10].

Ova definicija odražava uobičajeno stajalište da je održavanje softvera aktivnost nakon isporuke. Održavanje počinje kada se sustav izdaje klijentu ili korisniku i obuhvaća sve aktivnosti koje održavaju sustav operativnim i zadovoljavaju potrebe korisnika. Ovo gledište može se i uočiti u klasičnim modelima vodopada životnog ciklusa softvera, koji općenito obuhvaćaju završnu fazu primjene i održavanja.

Nekoliko se autora ne slaže s tim stajalištem i smatraju da bi održavanje softvera trebalo početi koristiti prije nego sustav postane operativan [11], [12], [13], [14]. Pa tako, Pigoski smatra da se aktivnosti održavanja izvode prije i nakon isporuke. Aktivnosti prije isporuke uključuju planiranje operacija nakon isporuke, mogućnost podrške i određivanje logistike. Aktivnosti nakon isporuke uključuju modifikaciju softvera, obuku i rad s korisničkom podrškom [14].

U literaturi se nalazi više različitih definicija održavanja softvera, a ovdje je istaknuta još jedna koja na specifičan način opisuje održavanje.

Održavanje informacijskog sustava (i softvera koji mu daje podršku) podrazumijeva izvođenje sljedećih aktivnosti:

- postavljanje korisničkih zahtjeva za izmjenama
- odobravanje zahtjeva od nadležnog menadžera
- projektiranje izmjena modela prema zahtjevima
- proizvodnja novih programskih modula i izmjena aplikacije

- alfa-testiranje i beta-testiranje
- uvođenje novih funkcionalnosti

radi prilagodbe programskih proizvoda i baze podataka potrebama korisnika [13].

2.1. Kategorije održavanja softvera

Modifikacije, koje se provode tijekom održavanja softvera, mogu biti inicirane zbog više razloga. Različiti su autori pokušali klasificirati te modifikacije, što je rezultiralo taksonomijom koja se sastoji od korektivnih, adaptivnih, perfektivnih i preventivnih modifikacija [3], [6], [7], [16], [17], [18].

2.1.1. Korektivno održavanje

Označava modifikaciju softvera nakon isporuke, kojom se uklanjaju otkrivene greške (engl. bugs) u softveru. Greške mogu biti posljedica pogrešaka u dizajnu (engl. design errors), pogrešaka u programskom kodu (engl. coding errors) i logičkih pogrešaka (engl. logic errors). Greške u dizajnu nastaju kada su, na primjer, promjene u softveru netočne, nepotpune, pogrešno priopćene ili kada je zahtjev za promjenom pogrešno shvaćen. Pogreške kodiranja su uzrokovane nepravilnom implementacijom detaljnog logičkog dizajna i pogrešnom uporabom logike izvornog koda. Logičke pogreške proizlaze iz neispravnih testova i zaključaka, nepravilne implementacije projektnih specifikacija, neispravnog logičkog toka ili nepotpunog testiranja podataka. Greške su također uzrokovane pogreškama u obradi podataka i pogreškama u izvedbi sustava. Sve te pogreške, koje se ponekad nazivaju "preostale pogreške" (engl. residual errors), sprječavaju da softver zadovoljava ugovorenu specifikaciju. Potrebu za korektivnim održavanjem obično iniciraju izvještaji o greškama koje sastavljaju krajnji korisnici. Primjeri korektivnog održavanja uključuju ispravljanje neuspjeha testiranja svih mogućih uvjeta ili neuspjeh obrade posljednjeg zapisa u datoteci.

2.1.2. Adaptivno održavanje

Označava modifikaciju softvera nakon isporuke, kojom se postojeći softver prilagođava novom, promjenjivom okruženju. Pojam okruženje u ovom kontekstu se odnosi na ukupnost svih utjecaja i uvjeta koji izvana djeluju na sustav (poslovna pravila, propisi Vlade RH i Europske Unije, novi operativni sustav, nova baza podataka, napredniji hardver i softver itd.). Potreba za adaptivnim održavanjem može se prepoznati samo praćenjem okoliša. Primjer vladine politike koja može imati utjecaja na softverski sustav je ulazak zemlje članice Europske unije u eurozonu, gdje banke, ali i ostale institucije, moraju napraviti značajne promjene u svojim softverskim sustavima kako bi se prilagodile toj valuti. Drugi primjer je implementacija novog sustava za upravljanje bazom podataka i prilagodba više programa kako bi se omogućilo korištenje istih zapisa.

2.1.3. Perfektivno održavanje

Označava modifikaciju softvera nakon isporuke, kojom se postojeći softver ažurira prema novim ili izmijenjenim korisničkim zahtjevima. Perfektivno održavanje se bavi funkcionalnim obogaćivanjem softvera kako bi se poboljšale njegove performanse ili obogatilo korisničko sučelje. Primjeri perfektivnog održavanja uključuju modificiranje programa za obračun plaća kako bi se uključila nova sindikalna nagodba, dodavanje novog izvješća u sustav analize prodaje, poboljšanje terminološkog dijaloga kako bi ga se učinio jednostavnijim za korištenje.

2.1.4. Preventivno održavanje

Označava preventivnu modifikaciju softvera te se odnosi na aktivnosti usmjerene na povećanje održivosti (engl. maintainability) sustava, kao što su ažuriranje dokumentacije, dodavanje komentara i poboljšanje strukture programskih modula. Također, služi za zaštitu podataka i za zaštitu od grešaka u izvršavanju softvera [18]. Korektivne, adaptivne i perfektivne modifikacije uvelike povećavaju kompleksnost sustava. Preventivno održavanje se obično inicira unutar organizacije s namjerom da se nastala kompleksnost sustava smanji, da se programi lakše shvate i time olakšaju budući radovi na održavanju. Primjeri preventivnih promjena uključuju restrukturiranje i optimizaciju koda i ažuriranje dokumentacije.

Postoji nekoliko istraživanja održavanja softvera koja su razmatrala odnos između održavanja i razvoja te odnos između različitih aktivnosti održavanja [19], [20], [21]. Zbog razlika u korištenoj terminologiji, detalji istraživanja teško su usporedivi, no unatoč tome mogu se izvući zaključci koji su zajednički većini istraživanja. Može se zaključiti da održavanje softvera zauzima veći dio (oko 70%) IT budžeta te da se veći dio budžeta za održavanje (oko 70%) troši na provedbu novih zahtjeva, nego na novi razvoj. Ako se promatra vrijeme potrošeno na određene vrste održavanja, dolazi se do zaključka da perfektivno održavanje oduzima najviše (preko 50%) vremena, adaptivno 25%, korektivno nešto više od 20%, a samo 5% oduzima preventivno održavanje [3].

Između ove četiri kategorije održavanja, samo korektivno održavanje je održavanje u tradicionalnom smislu riječi. Druge kategorije mogu se smatrati „evolucijom softvera“. Pojam se koristi od ranih 60-ih godina prošlog stoljeća za karakterizaciju dinamike rasta softvera. Evolucija softvera sada se uvelike koristi, pa je tako npr. Journal of Software Maintenance u naziv dodao evolucija, kako bi se odrazio taj prijelaz korištenja terminologije [17].

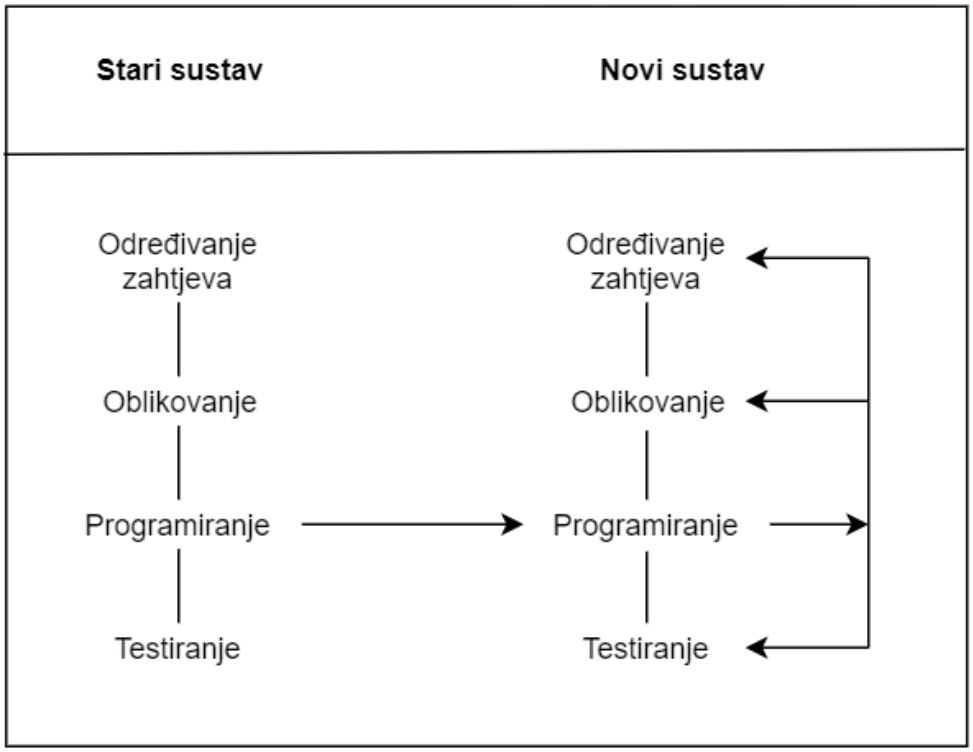
2.2. MODELI ODRŽAVANJA SOFTVERA

U nastavku navedena su i opisana tri modela održavanja softvera, koja se najviše koriste u industriji.

2.2.1. Model brzog popravka

Tipičan pristup održavanju softvera je prvo raditi na kodu, a zatim napraviti potrebne izmjene u pratećoj dokumentaciji. Ovaj pristup je obuhvaćen modelom brzog popravka (engl. quick-fix model), prikazanim na slici 1., koji pokazuje tijek promjena od stare do nove verzije sustava. U idealnom slučaju, nakon što je kod promijenjen, ažurirani su dostupni dokumenti na koje utječe ta izmjena. Međutim, korisnici često očekuju da će se softver brzo i jeftino modificirati, pa se promjene često provode na brzinu, bez pravilnog planiranja, dizajna, analize utjecaja i regresijskog testiranja. Dokumenti se mogu ili ne moraju ažurirati kako se kod mijenja. Zbog vremenskih rokova i ograničenog budžeta promjene u kodu nisu dokumentirane i to vrlo brzo umanjuje vrijednost dokumentacije. Osim toga, velikom količinom modifikacija može se uništiti izvorni dizajn softvera, čime se cijena i vremenski rokovi buduće modifikacije postupno

povećavaju. Prednost ovog modela je što se modifikacija obavi brzo, bez prevelikih troškova [16], [22], [23].



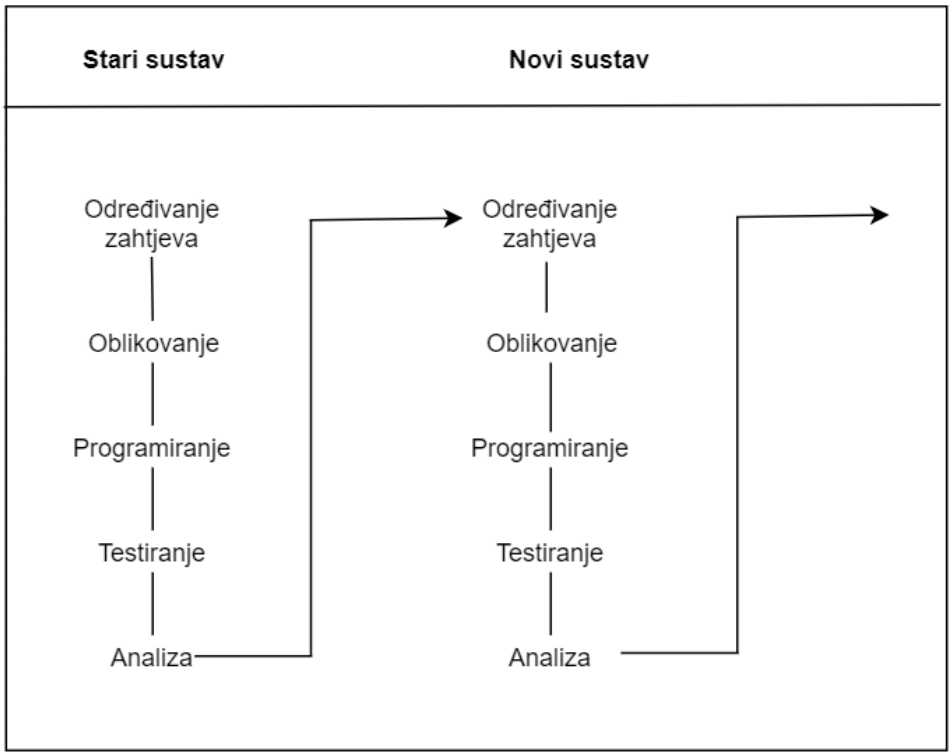
Slika 1. Model brzog popravka [24]

2.2.2. Iterativni model poboljšanja

U iterativnom modelu poboljšanja (engl. iterative enhancement model) promjene, izvršene tijekom životnog ciklusa softvera, čine iterativni proces. Prikaz modela vidljiv je na slici 2. Ovaj model uključuje promjene u softveru na temelju analize postojećeg sustava te uzima u obzir promjene koje su napravljene u sustavu. Pretpostavlja se da je u početku dostupna cjelovita dokumentacija softvera. Štoviše, ovim modelom se pokušava kontrolirati kompleksnost i održati dobar dizajn. Model je podijeljen u tri faze: analiza softvera, klasifikacija traženih izmjena i implementacija izmjena [16], [22], [23].

Ključna prednost modela iterativnog modela poboljšanja je što se dokumentacija ažurira kako se kod mijenja. Visaggio je proveo kontrolirane eksperimente kako bi usporedio modele brzog popravka i iterativnog poboljšanja [24]. Istraživanje pokazuje da se održivost sustava brže

degradira, ako se koristi model brzog popravka. Također, pokazuje da organizacije, koje koriste iterativni model poboljšanja, brže rade izmjene u održavanju nego one koje koriste model brzog popravka.

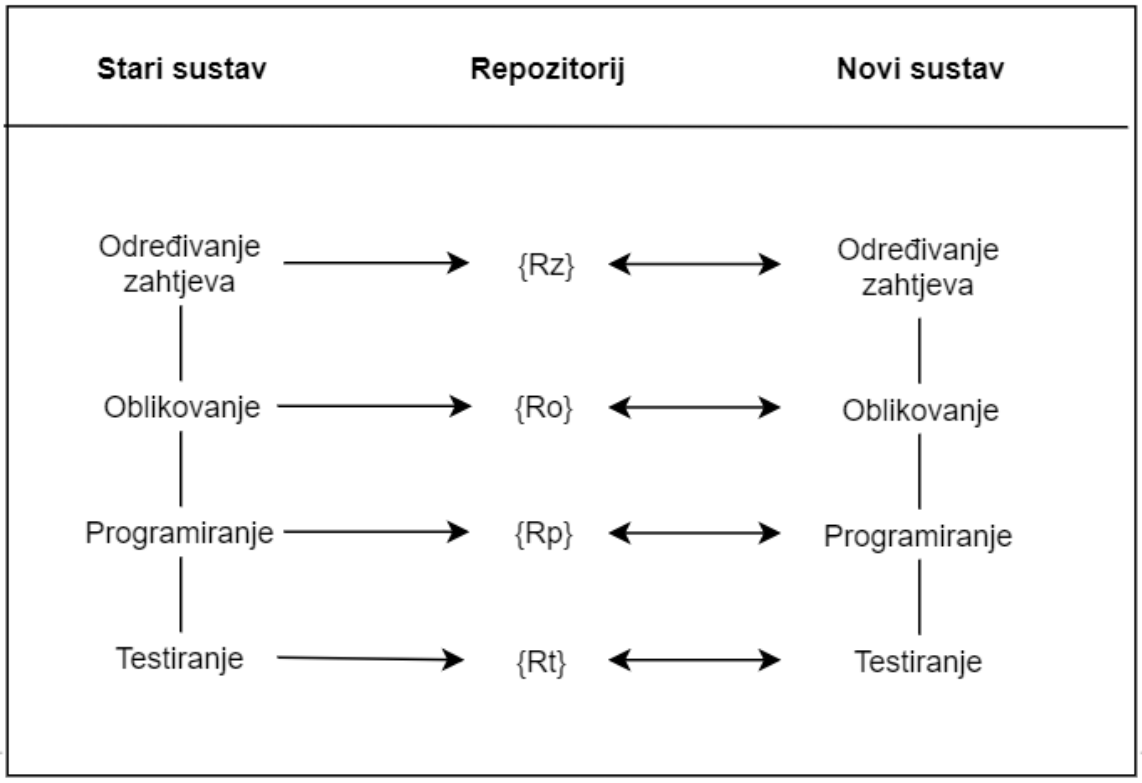


Slika 2. Iterativni model poboljšanja [24]

2.2.3. Model ponovne upotrebe

U modelu ponovne upotrebe (engl. reuse model) održavanje se smatra posebnim slučajem razvoja softvera orijentiranog na ponovnu uporabu. Okosnica ovog modela je repozitorij dokumenta i komponenti ranijih verzija trenutnog sustava, kao i ostalih sustava u istoj aplikacijskoj domeni. Model je prikazan na slici 3. Započinje analizom zahtjeva i dizajnom (oblikovanjem) novog sustava, zatim se identificiraju dijelovi iz repozitorija (zahtjevi, kodovi, testovi itd.) koji se mogu ponovno upotrijebiti. Ti dijelovi se modificiraju i poboljšavaju, na temelju novih zahtjeva. Posljednji korak ovog modela je integracija modificiranih dijelova u novi sustav. Ovaj model promiče razvoj komponenti koje se mogu ponovno koristiti [16], [22], [23].

Iterativni model poboljšanja dobro je prilagođen sustavima koji imaju dugi vijek trajanja i koji se razvijaju tijekom vremena, podržava evoluciju sustava na način da olakša buduće modifikacije. Model ponovne upotrebe prikladniji je za razvoj sličnih sustava. Organizacije koje primjenjuju model ponovne upotrebe akumuliraju komponente za višekratnu upotrebu svih vrsta i različitih razina apstrakcija, a to čini buduće razvoje softvera isplativijima [16], [22], [23].



Slika 3. Model ponovne upotrebe [24]

2.3. PROCES ODRŽAVANJA

IEEE standard za održavanje softvera organizira održavanje u sedam faza. Osim redoslijeda izvršavanja faze, definiraju se i ulazne i izlazne isporuke (engl. deliverables), grupirane aktivnosti, povezani i prateći procesi, skup metrika i kontrola. Također uključuje aktivnosti za prihvaćanje i dodjeljivanje zahtjeva skupu izmjena predviđenih za implementaciju. U nastavku su navedene i opisane faze održavanja softvera IEEE Standards [10].

Identifikacija problema/promjene, klasifikacija i prioretizacija (engl. Problem/modification identification, classification, prioritization)

- U ovoj se fazi zahtjev za promjenom (engl. modification request - MR) dodjeljuje kategoriji održavanja te mu se odredi prioritet i jedinstven identifikator.

Analiza (engl. Analysis)

- Ova faza osmišljava preliminarni plan za izradu, provedbu, testiranje i isporuku. Analiza se provodi na dvije razine: analiza izvodljivosti i detaljna analiza korisničkog zahtjeva. Analiza izvodljivosti identificira alternativna rješenja, procjenjuje njihove učinke i troškove. Također obuhvaća, utjecaj promjene na ostatak sustava, analizu zahtjeva na konverzije podataka, sigurnosni utjecaj, troškove, korisnost promjene… Detaljna analiza definira zahtjeve za izmjenu, identificira sve elemente koje treba mijenjati, osmišljava strategiju testiranja i razvija plan implementacije izmjene.

Dizajn (engl. Design)

- Modifikacija sustava je zapravo dizajnirana u ovoj fazi. To podrazumijeva korištenje svih postojećih sustava i projektne dokumentacije, postojećih softverskih baza podataka i izlazne isporuke faze analize. Aktivnosti uključuju identifikaciju softverskih modula za izmjenu, modifikaciju dokumentacije softverskih modula, kreiranje testnih slučajeva za novi dizajn i identifikaciju i kreiranje regresijskih testova.

Implementacija (engl. Implementation)

- Ova faza uključuje aktivnosti kodiranja i testiranja izmjena, integraciju modificiranog koda, integracijsko i regresijsko testiranje, analizu rizika i izvještaj o spremnosti za testiranje. U ovoj fazi testiranje uglavnom provodi programer na neintegriranoj verziji softvera.

Regresijsko i sistemsko testiranje (engl. Regression/system testing)

- Cijeli se sustav testira kako bi se osigurala usklađenost s prvobitnim zahtjevima i izmjenama. Osim funkcionalnog testiranja i testiranja korisničkog sučelja, faza uključuje regresijsko testiranje kojim se potvrđuje da nisu dodane nove pogreške. Konačno, ova je faza odgovorna za provjeru spremnosti za testiranje prihvatljivosti izmjene.

Testiranje prihvatljivosti (engl. Acceptance testing)

- Provodi se testiranje na potpuno integriranom sustavu i uključuje korisnike, kupce ili treću stranu koju je odredio klijent. Testiranje prihvatljivosti obuhvaća funkcionalna ispitivanja i testove utjecaja na ostale sustave.

Isporuka (engl. Delivery)

- Modificirani softver pušta se u rad. Ova faza uključuje aktivnost kao što je obavještavanje korisnika o izmjenama, instalacija modificiranog softvera, obuka korisnika te izrada sigurnosnih kopija.

2.4. UPRAVLJANJE ODRŽAVANJEM

Upravljanje je proces koji zahtijeva učinkovito korištenje resursa u kombinaciji s usmjeravanjem ljudi kako bi se postigao određeni organizacijski cilj. Cilj upravljanja održavanjem je pružanje troškovno prihvatljive podrške softverskim sustavima tijekom cijelog životnog ciklusa softvera. Organizacije mogu povećati kvalitetu, produktivnost, efektivnost i efikasnost održavanja, odabirom učinkovite strategije upravljanja te dobrom organizacijom i vođenjem procesa upravljanja. Mnogi autori razvili su različite klasifikacije funkcija upravljanja, ali postoji konsenzus da se upravljanje sastoji od pet funkcija. Te funkcije su planiranje, organiziranje, osoblje, vođenje i kontroliranje [5], [16].

Planiranje je najosnovnija i najbitnija funkcija upravljanja. Uključuje izradu detaljnog plana za postizanje određenog cilja te odgovara na pitanja „što“, „kako“ i „kada“. Također, uključuje utvrđivanje dugoročnih i kratkoročnih ciljeva, razvoj strategija i pravaca djelovanja koje treba slijediti za postizanje tih ciljeva te formuliranje procedura, postupaka i pravila za provedbu strategija i planova. Planiranje osigurava pravilno korištenje i raspodjelu dostupnih resursa kako bi se postigao cilj.

Organiziranje uključuje identifikaciju aktivnosti potrebnih za postizanje ciljeva i provedbu planova, grupiranje aktivnosti u poslove, dodjeljivanje tih poslova odjelima i pojedincima, utvrđivanje interne strukture uloga koje će ljudi obavljati unutar organizacije te definiranje odgovornosti za pojedinu ulogu. Obuhvaća organizacijske pozicije, prateće zadaće i odgovornosti te mrežu odnosa ovlasti i odgovornosti. Organiziranje je stoga osnovni proces

kombiniranja i integriranja ljudskih, fizičkih i financijskih resursa u produktivne odnose za postizanje ciljeva.

Osoblje je jako važna funkcija upravljanja budući da učinkovitost i djelotvornost organizacije uvelike ovisi o kvaliteti njezinog osoblja. Funkcija ima za cilj osigurati da organizacija uvijek ima prave ljude na pravim pozicijama i da organizacijska struktura nije otežana nedostatkom ili prekomjernim osobljem. Uključuje popunjavanje pozicija u organizaciji s ljudima te njihovu obuku za obavljanje pojedinih uloga. Dvije glavne aktivnosti su evaluacija osoblja i poboljšavanje znanja, stavova i vještina.

Vođenje se odnosi na vođenje zaposlenika da djeluju učinkovito i da doprinose ostvarenju ciljeva organizacije. Radna mjesta koja su dodijeljena podređenima moraju biti objašnjena i razjašnjena te ih potrebno je voditi i motivirati u obavljanju posla.

Posljednja funkcija upravljanja je **kontroliranje**. Funkcija osigurava da se ostale četiri funkcije ispravno prate, mjeri performanse i uspoređuje ih sa planiranim ciljevima. Odstupanja od ciljeva i planova moraju se identificirati i istražiti te poduzeti korektivne mjere, što podrazumijeva nagrade i kazne djelatnika.

Najčešći problem koji se javlja u upravljanju održavanjem je neiskustvo djelatnika. Beath i Swanson navode da 61% novo zaposlenih djelatnika te 25% studenata radi na održavanju [25]. Pigoski također navodi da čak do 80% novo zaposlenih djelatnika radi na održavanju [14]. Kod mnogih organizacija održavanje se još uvijek ne doživljava kao strateško pitanje, a to je problem i na mnogim sveučilištima gdje se održavanje softvera većinom samo spominje kao faza u životnom ciklusu softvera. Postoji nekoliko razloga iz kojih organizacije na poslovima održavanja softvera zapošljavaju neiskusne djelatnike: iskusni djelatnici su angažirani na poslovima razvoja, održavanje softvera tehnički je manje zahtjevno nego razvoj softvera, na poslovima održavanja mladi djelatnici lakše stječu iskustvo te se održavanje vidi kao prilika za brzu i jeftinu obuku djelatnika. Djelatnici koji su zaposleni na poslovima održavanja često to doživljavaju kao kaznu ili stagnaciju u svom razvoju karijere, zbog toga im opada motivacija, što rezultira opadanjem produktivnosti. Tom problemu treba pristupiti ozbiljno i svaka informatička tvrtka bi trebala organizirati održavanje softvera na način da umanji taj problem.

2.5. ALATI

Zadatak održavanja softvera postao je toliko vitalan i složen proces da je potrebna automatizirana podrška kako bi se taj proces obavio što učinkovitije. Alati za održavanje softvera su softverski proizvodi koji pomažu djelatnicima u održavanju u obavljanju njihovih zadataka [3]. Omogućuju bolje strukturiranje i organiziranje procesa. Povratne informacije o korištenju automatiziranih procesa održavanja i alata za podršku pomažu u razvoju područja i usmjeravaju budući napredak. Alati za održavanje softvera su i sami softverski sustavi koji se razvijaju i moraju se održavati.

Postoji nekoliko dobavljača koji razvijaju i prodaju širok raspon alata za koje tvrde da podržavaju održavanje softvera. Neki od alata to rade bolje, neki lošije. Imajući to u vidu, potrebno je uzeti u obzir nekoliko čimbenika za odabir pravog alata za određeni zadatak. Ti čimbenici su mogućnost (engl. capability), značajke, odnos troška i koristi, platforma, programski jezik, jednostavnost korištenja, otvorenost arhitekture, stabilnost dobavljača i organizacijska kultura [17].

Mogućnost pokazuje ima li alat mogućnosti za obaviti određeni zadatak i jedan je od najvažnijih kriterija koje treba uzeti u obzir pri ocjenjivanju alata. Nakon što je odlučeno da neki zadatak može imati koristi od automatizacije, treba uzeti u obzir značajke koje se očekuju od bilo kojeg potencijalnog alata. Također, treba napraviti analizu alata koja prikazuje koje prednosti on donosi u odnosu na troškove. Prednosti koje alat donosi moraju se procijeniti u smislu pokazatelja kao što su kvaliteta, produktivnost, brzina reagiranja i smanjenje troškova. Okruženje na kojem se alat pokreće naziva se platforma. Jezik izvornog koda naziva se programski jezik. Važno je odabrati alat koji podržava jezik koji je industrijski standard. Alat bi trebao biti jednostavan za korištenje, odnosno trebao bi trebao imati slične funkcionalnosti, kao i oni s kojima su korisnici već upoznati. Alat bi trebao imati mogućnost integriranja s alatima različitih dobavljača, što će biti od velike pomoći kada alat treba pokrenuti s drugim alatima. Otvorenost arhitekture igra važnu ulogu kada je problem održavanja složen, kada postoji potreba za više alata koji rade zajedno. Također je važno uzeti u obzir vjerodostojnost i ugled dobavljača. Zbog velike konkurencije u računalnoj industriji, tvrtka koja ne može pratiti konkurenciju može nestati s poslovne scene i ostaviti svoje korisnike bez podrške. Dobavljač bi trebao biti sposoban podržati alat u budućnosti i omogućiti dugotrajnu korisničku podršku.

Drugi važan čimbenik je kultura organizacije. Svaka kultura ima svoj radni obrazac, stoga je važno uzeti u obzir hoće li ciljani korisnici prihvatiti alat [3], [17].

Odabrani alati moraju podržavati razumijevanje programa (engl. program understanding) i obrnuti inženjering (engl. reverse engineering), testiranje, upravljanje konfiguracijom i dokumentacijom [3]. Na temelju toga razlikuju se četiri kategorije alata za održavanje, koje su klasificirane na temelju specifičnih zadataka koje alati podržavaju [3].

2.5.1. Alati za razumijevanje programa i obrnuti inženjering

Razumijevanje programa provodi se s ciljem razumijevanja izvornog koda, dokumentacije i dizajna, odnosno uključuje znanje o tome što i kako program radi, gdje će se u sustavu vršiti izmjene te kako te izmjene utjecati na rad programa. Obrnuti inženjering je proces analize postojećeg sustava u svrhu identificiranja sistemskih komponenti, njihovih međusobnih odnosa te prikaza sustava u drugom obliku ili na višem nivou apstrakcije [3], [6], [25].

Odabir alata, koji podržava razumijevanje programa, vrlo je važan u provedbi izmjena jer se mnogo vremena potroši za proučavanje i razumijevanje programa. Alati za obrnuti inženjering također postižu isti cilj. Alati se uglavnom sastoje od alata za vizualizaciju, koji pomažu programeru u crtanju modela sustava. Primjeri alata za razumijevanja programa i za obrnuti inženjering uključuju programski rezač (engl. program slicer), statički analizator (engl. static analyzer), dinamički analizator (engl. dynamic analyzer), unakrsni analizator (engl. cross-referencer) i analizator ovisnosti (engl. dependency analyzer) [3], [17], [26].

Jedan od glavnih problema u održavanju softvera je suočavanje s veličinom izvornog koda programa. Važno je da programeri mogu odabrati i pregledati samo one dijelove programa na koje utječe predložena izmjena, a da ih ne ometaju irelevantni dijelovi koda. Jedna tehnika koja pomaže s ovim problemom poznata je kao rezanje - mehanički proces označavanja svih dijelova teksta programa koji mogu utjecati na vrijednost varijable. Alat koji se koristi za potporu rezanja poznat je kao programski rezač. Programski rezač također prikazuje podatkovne veze i povezane značajke kako bi programer mogao pratiti učinak promjena [3], [17].

Statički analizator koristi se za analizu različitih aspekata programa, kao što su moduli, procedure, varijable, podatkovni elementi, objekti i klase. Statički analizator omogućuje

generiranje sažetaka sadržaja i korištenje odabranih elemenata programu, kao što su varijable ili objekti [3], [17], [26].

Pomoću dinamičkog analizatora prati se put izvršenja (engl. execution path) sustava dok se program izvršava. Omogućuje djelatniku održavanja da odredi puteve na koje će utjecati izmjena [3].

Analizator protoka podataka je alat za statičku analizu koji omogućuje praćenje svih mogućih tokova podataka u programu što uvelike pomaže u opisu temeljne logike programa te u prikazu odnosa komponenti sustava. Praćenjem toka podataka može se, na primjer, odrediti gdje je varijabla poprimila svoju vrijednost i na koje sve dijelove programa utječe modifikacija varijable [3].

Unakrsnim analizatorom dobivaju se informacije o korištenju programa. Ovaj alat, također, pomaže korisniku da se usredotoči na dijelove programa na koje izmjena utječe [3], [17], [26].

Analizator ovisnosti pomaže u analiziranju i razumijevanju međusobnih odnosa između entiteta u programu. Takav alat pruža mogućnost pretraživanja baze podataka o ovisnostima u programu. Također pruža grafičke prikaze ovisnosti, gdje čvor predstavlja entitet programa, a luk ovisnosti među entitetima [3], [17], [26].

2.5.2. Alati za podršku testiranja

Testiranje je najdugotrajniji i najzahtjevniji zadatak u održavanju softvera, stoga bi ta aktivnost mogla imati najviše koristi od alata.

Alat za simuliranje testova (engl. test simulator tool) testira učinke izmjena u kontroliranom okruženju prije implementacije izmjene na stvarnom sustavu. Ključna prednost ovog pristupa je da djelatnik održavanja može isprobati učinak izmjene prije uvođenja izmjene na stvarni operativni sustav. Nedostatak je što rezultati i opažanja mogu biti pogrešni jer se neka ograničenja u stvarnom okruženju možda ne odražavaju u kontroliranom okruženju [3], [17], [26].

Generator testnog slučaja (engl. test case generator) proizvodi testne podatke koji se koriste za testiranje funkcionalnosti modificiranog sustava, dok generator testnih puteva (engl. test path

generator) pronalazi sve tokove podataka i puteve kontrole toka koji su pod utjecajem promjena [3], [17], [26].

2.5.3. Alati za upravljanje konfiguracijom

Upravljanje konfiguracijom (engl. configuration management) je organizirani skup pravila, postupaka i alata kojima se kontrolira mijenjanje softvera i evidentiraju se njegove različite verzije. Upravljanje konfiguracijom potrebno je kod velikih sustava gdje postoji velik broj različitih verzija koje su istovremeno u razvoju ili u upotrebi [6]. Učinkovito upravljanje konfiguracijom nije moguće bez pomoći neke vrste automatiziranih alata. Alati za upravljanje konfiguracijom i upravljanje verzijama pomažu pri pohranjivanju objekata koji čine softverski sustav. Sustav kontrole izvora koristi se za čuvanje povijesti datoteka, kako bi se mogle pratiti verzije programa i kako bi programer mogao pratiti promjene datoteka [3], [17], [6].

2.6. ESTIMUS

Informatička tvrtka Ris d.o.o. razvila je alat ESTIMUS koji podupire proces održavanja. ESTIMUS je softver za upravljanje održavanjem softvera koji omogućuje brzu i jednostavnu organizaciju vremena i kadrova uključenih u projekte. Pomoću aplikacije može se jednostavno nadzirati rad svakog djelatnika te se mogu vidjeti detalji o projektima na kojima oni rade. Korisnik može u svakom trenutku dobiti pregledan presjek aktivnosti i grupa aktivnosti te faza u kojima se oni trenutno nalaze. Prednost alata je i to što se nove i nepredvidive aktivnosti mogu jednostavno dodavati, prilikom čega im se dodijeli stupanj prioriteta te djelatnik koji je zadužen za njihovo izvršavanje. Ukoliko se korisniku približava zadani rok aktivnosti ili je on prekršen, alat ga na to podsjeća. Svaki korisnik ima onakav pregled kakav odgovara njegovoj projektnoj ulozi, a sve komponente alata su međusobno povezane. Pošto aplikacija radi po principu „softver kao usluga“, smanjuju se inicijalna ulaganja u računalnu opremu kao i troškovi njenog održavanja te se pruža visok stupanj sigurnosti podataka [27].

Sustav je implementiran kao SaaS aplikacija nad Oracle bazom podataka, a za modeliranje sustava korištena je specijalizirana metodologija MIRIS [13]. Analiza poslovnih procesa upravljanja održavanjem napravljena je pomoću tradicionalne analize potreba kupca i analize

stanja gotovih rješenja na tržištu. Za analizu potreba kupca korištena je metoda dekompozicije i dijagrama toka podataka u skladu sa specijaliziranom metodologijom MIRIS, a za analizu podataka i oblikovanje relacijske baze podataka korištena je proširena Chen-ova metoda Entity-Relationship [13].

2.6.1. Funkcionalnosti

Alat ESTIMUS ima mnogo funkcionalnosti koje se koriste za upravljanje održavanjem, a u ovom radu su izdvojene neke najvažnije [27]:

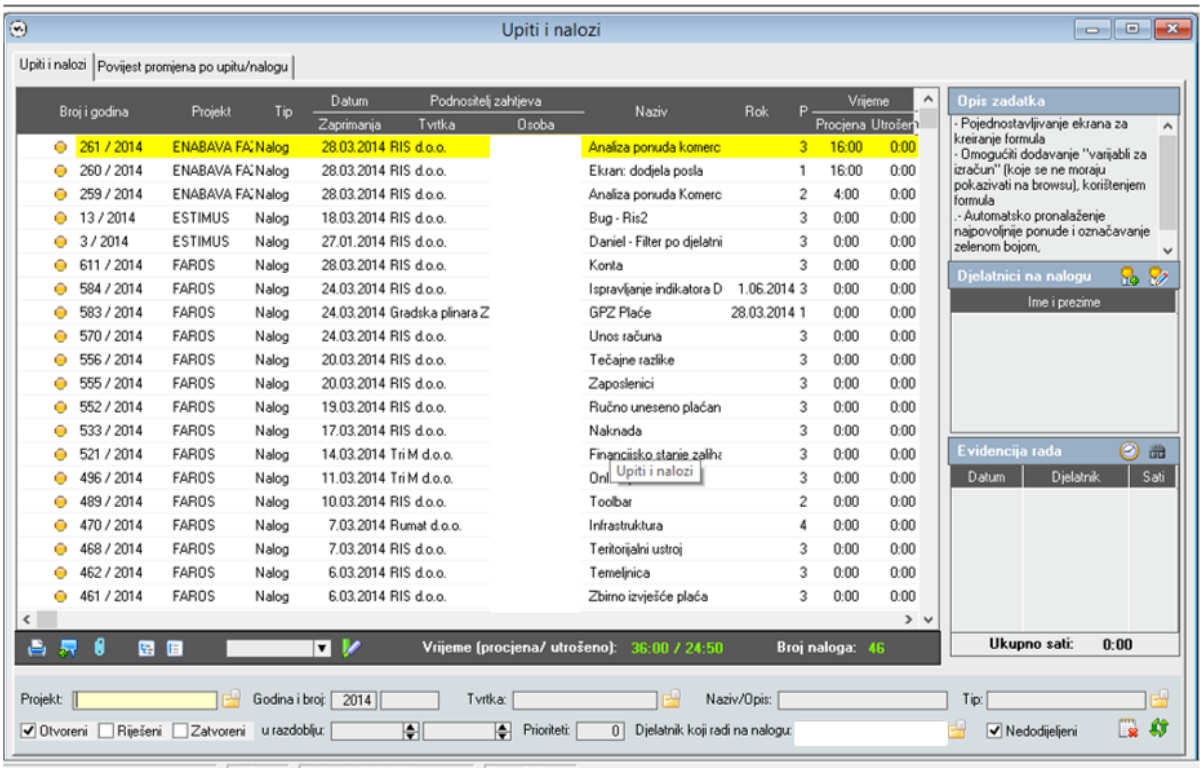
- evidencija projekata i aktivnosti na projektima
- pregled i ažuriranje tvrtki, djelatnika, uloga na projektu, programskih modula, ovlasti u aplikaciji, dokumentacije…
- pridruživanje resursa aktivnostima i praćenje njihova izvršavanja
- pregled rokova, kontrolnih točaka, aktivnosti, kalendara, pohranjenih izvještaja…
- praćenje radnih naloga od njihova podnošenja do zatvaranja
- praćenje pojedinih verzija proizvoda
- evidencija rada zaposlenika (prema projektima, radnim nalozima…)
- evidencija obveza prikazanih u kalendaru
- praćenje rokova i obveza na projektima i radnim nalozima
- generiranje poslovnih izvještaja (engl. Business Inteligence, BI)

2.6.2. Korisničko sučelje

Korisničko sučelje dizajnirano je za tri glavne grupe korisnika, a to su: programeri koji evidentiraju rad, voditelji programerskih timova koji upravljaju zahtjevima od preuzimanja do isporuke verzije i menadžment koji nadgledava i analizira opseg i sadržaj poslova i zahtjeva. U nastavku prikazano je nekoliko glavnih ekrana korisničkog sučelja s kratkim opisom svakog od ekrana.

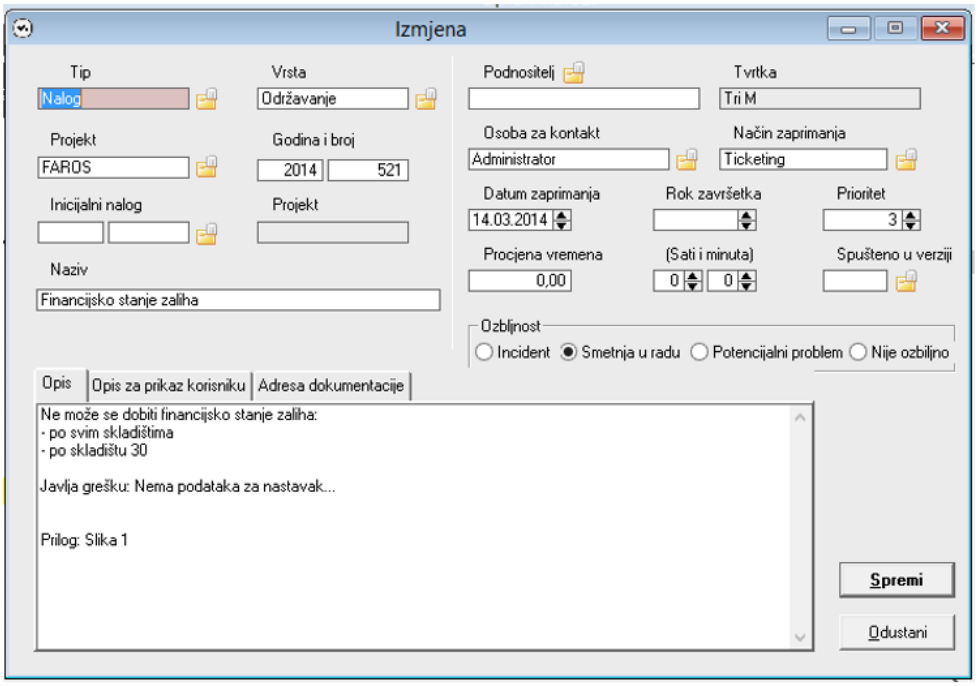
Na slici 4. prikazan je ekran za pregled i pretraživanje radnih naloga. Otvaranjem ekrana prikazuju se otvoreni nalozi na kojima radi korisnik, koji je prijavljen u aplikaciju, kao i otvoreni nalozi koji još nisu dodijeljeni niti jednom korisniku. Označavanjem odgovarajuće

potvrdne kućice (engl. checkbox) mogu se prikazati riješeni i zatvoreni radni nalozi. Pretraživanje se može vršiti prema više parametara, tako da se željeni parametri unesu u odgovarajuće polje u donjoj alatnoj traci. Međusobno povezani nalozi i upiti prikazuju se kao hijerarhijska struktura u obliku stabla. Ukoliko je potrebno prikazati sve naloge, treba odabrati alat u podnožju tablice čime se prikazuje cijela hijerarhija.



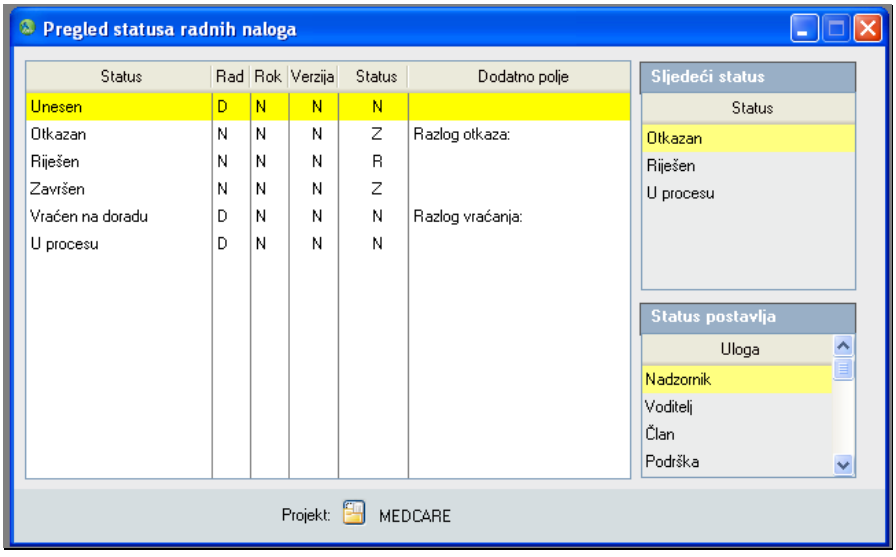
Slika 4. ESTIMUS - ekran za pregled naloga i upita

Naloge je moguće ažurirati korištenjem padajućeg izbornika do kojeg se dolazi pritiskom desne tipke miša unutar pregleda naloga i upita. Prilikom unosa (slika 5.) broj naloga se automatski postavlja na prvi sljedeći broj unutar aktualne godine i odabranog projekta. Prema potrebi taj se broj može mijenjati, ali mora biti jedinstven. Kod unosa naloga niže razine (podnaloga) polje „Inicijalni nalog“ automatski će se popuniti brojem naloga koji je bio odabran prije unosa. Korištenje tipke F5 ili ekrana za odabir podatka moguće je na svim poljima uz koja postoji alat za odabir. Prilikom odabira verzije u polju „Spušteno u verziji“ moguće je odabrati samo verziju (za odabrani projekt) koja nije zaključana.



Slika 5. ESTIMUS - ekran za unos i izmjenu naloga i upita

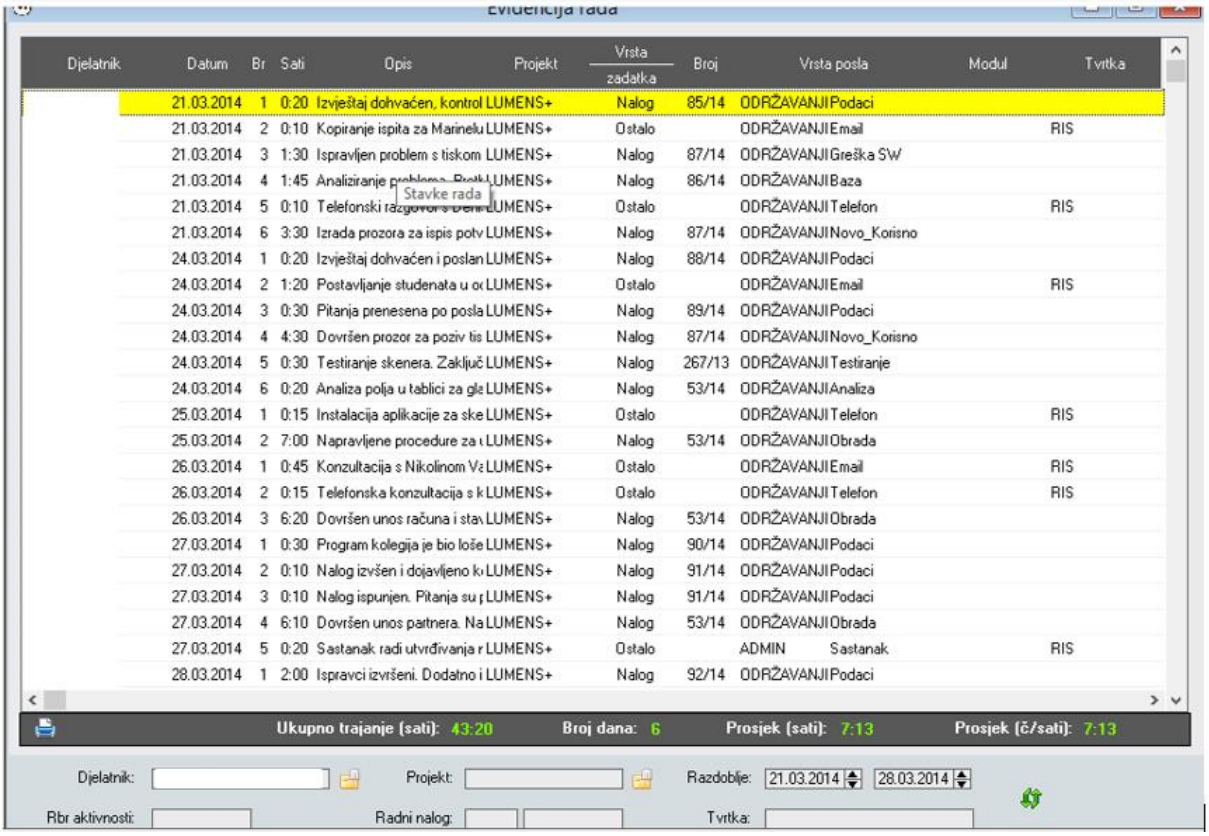
Na slici 6. prikazan je ekran za pregled statusa radnih naloga, gdje se definiraju statusi koji se će koristiti za pojedini nalog ili upit na projektu, njihov redoslijed korištenja te uloga korisnika koji može postaviti pojedini status.



Slika 6. ESTIMUS - ekran za pregled statusa radnih naloga

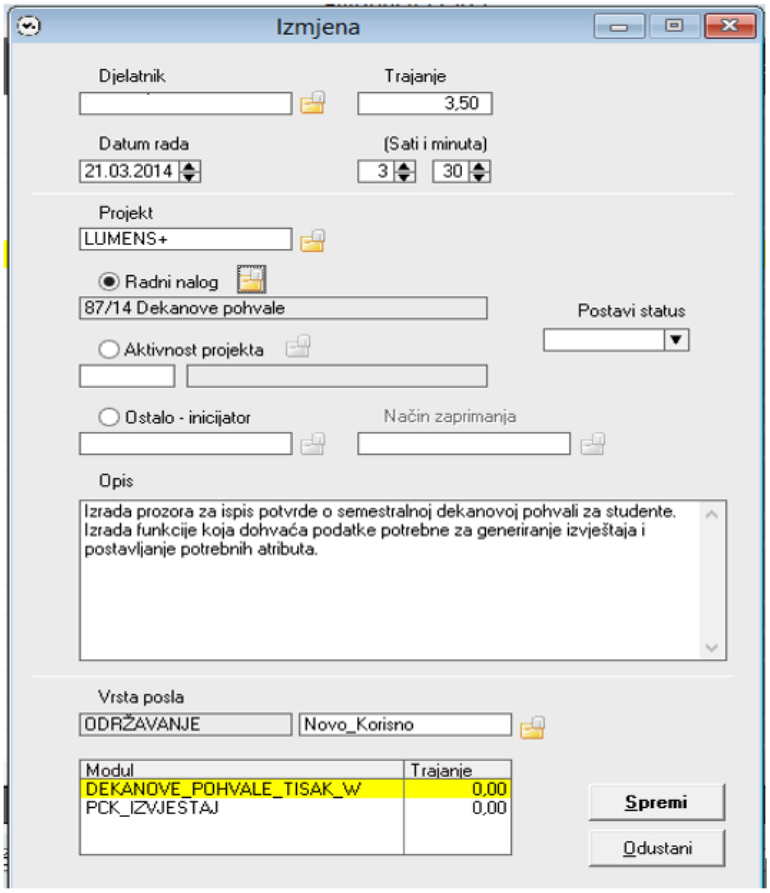
Slika 7. prikazuje evidenciju rada. Kako bi se prikazali podaci, vrijednosti pojedinih polja za pretragu popunjene su unaprijed definiranim vrijednostima prilikom otvaranja ekrana. Polje „Djelatnik“ izjednačeno je s korisnikom koji je prijavljen u aplikaciju, a polje „Razdoblje“ s

tekućim datumom. Unosom željenih parametara i odabirom gumba vrši se pretraživanje samo onih podataka koji odgovaraju parametrima.



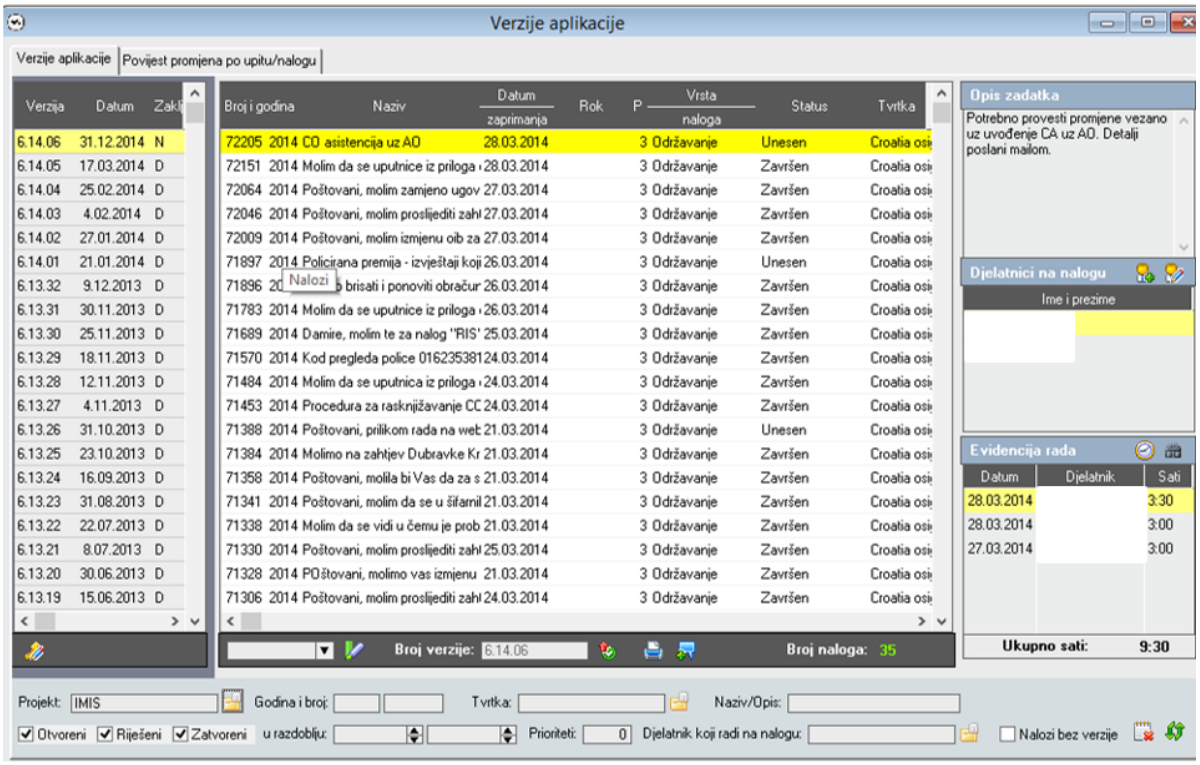
Slika 7. ESTIMUS - ekran za evidenciju rada

Na slici 8. prikazan je unos nove stavke evidencije rada. Do unosa nove stavke evidencije rada dolazi se desnim klikom miša unutar pregleda te odabirom odgovarajuće opcije na padajućem izborniku. Također, evidencija rada može se ažurirati na svakom ekranu koji je vezan uz rad nekog djelatnika. Ako se prilikom unosa evidencije rada na nalogu ili upitu u polje „Vrsta zadatka“ unese „Nalog“, tada se otvara obvezno polje „Radni nalog“ gdje je moguće odabrati samo naloge koji su dodijeljeni korisniku upisanom u polju „Djelatnik“. Moguće je u polje „Vrsta zadatka“ unijeti i vrijednost „Projekt“, prilikom čega se otvara obvezno polje „Rbr aktivnosti“. U tom slučaju mogu se odabrati samo one stavke rada na projektu u kojima je kao resurs evidentiran korisnik upisan u polje „Djelatnik“. Polje „Vrsta posla“ je obvezno, a u ovisnosti odabrane vrste posla može se aktivirati i polje „Programski modul“ koje također postaje obvezno. Promjene se spremaju pritiskom na gumb „Spremi“.



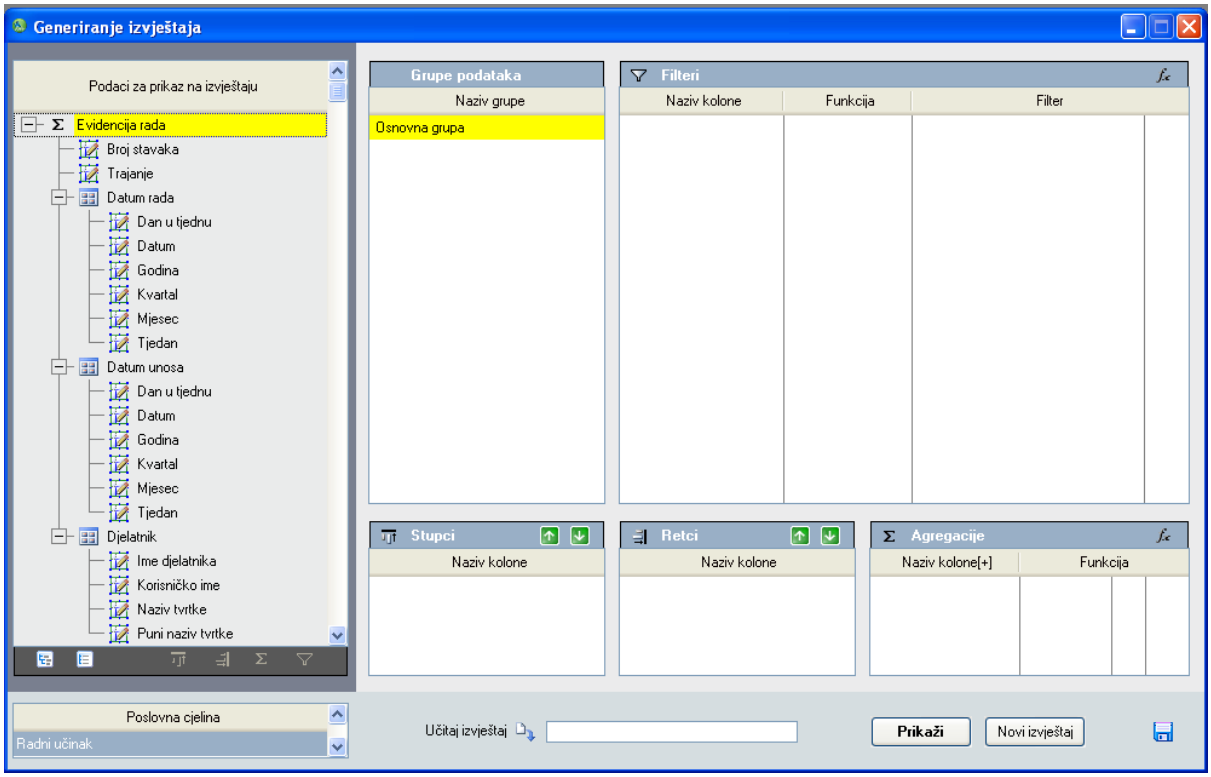
Slika 8. ESTIMUS - ekran za unos i izmjenu evidencije rada

Slika 9. prikazuje ekran za pregled i pretraživanje naloga prema verzijama aplikacije u kojoj su isporučeni i za grupiranje naloga u nove isporuke. Odabirom projekta, u polju „Projekt“, prikažu se sve verzije tog projekta na lijevoj strani ekrana. U sredini ekrana prikazani su svi nalozi koji su vezani za odabranu verziju projekta, a na desnoj strani nalaze se pregledi koji se odnose na odabrani nalog. Unos nove verzije vrši se pritiskom desne tipke miša te odabirom odgovarajuće opcije u padajućem izborniku. Verziju može zaključati (ili otključati) samo voditelj projekta i to odabirom alata u podnožju pregleda verzija. Naloge je moguće dodijeliti samo otključanim verzijama korištenjem ekrana za unos naloga (slika 8.) ili označavanjem opcije „Nalozi bez verzije“, nakon čega se prikazuju svi nalozi odabranog projekta koji nisu dodijeljeni niti jednoj verziji te se odabire željeni nalog iz tog popisa. Pretraživanje naloga odabrane verzije vrši se unosom željenih parametara u podnožju ekrana.



Slika 9. ESTIMUS - ekran za pregled verzija aplikacije

Alat ESTIMUS sadrži i Business Inteligence modul za generiranje izvještaja (slika 10.), koji korisniku omogućuje kreiranje izvještaja prema raznim kriterijima, spremanje istih za buduće korištenje te izvoz podataka u razne formate (xls, pdf, csv, itd.). Na lijevoj strani ekrana nalazi se popis podataka koji se mogu smjestiti u stupac izvještaja ili redak izvještaja, a mogu se koristiti kao filter ili agregacija u izvještaju. Kako bi se odabrani podatak koristio u stupcu ili retku izvještaja, potrebno ga je dodati u pregled podataka za stupac, odnosno redak i to odabirom određenog alata u podnožju pregleda podataka. Redoslijed podataka moguće je mijenjati upotrebom strelica koje se nalaze na vrhu pregleda stupaca, odnosno redaka. Odabir podataka za filter ili agregaciju vrši se na isti način kao i odabir podataka za stupac ili redak. Uvjeti filtra postavljaju se odabirom alata koji se nalazi u gornjem desnom kutu pregleda filtra ili desnim klikom miša i odabirom odgovarajuće opcije unutar padajućeg izbornika. Postavke agregacije postavljaju se odabirom alata koji se nalazi u gornjem desnom kutu pregleda agregacija. Napravljeni izvještaj moguće je pohraniti za buduće korištenje pritiskom na ikonu za spremanje koja se nalazi u donjem desnom kutu ekrana. Izvještaji se pokreću odabirom gumba „Prikaži“ nakon čega se otvara ekran s pregledom podataka za izvještaj, gdje je moguće odabrati bilo koji ćeliju na presjeku retka i stupca i za nju pregledati podatke na kartici Analitički podaci odabrane ćelije.



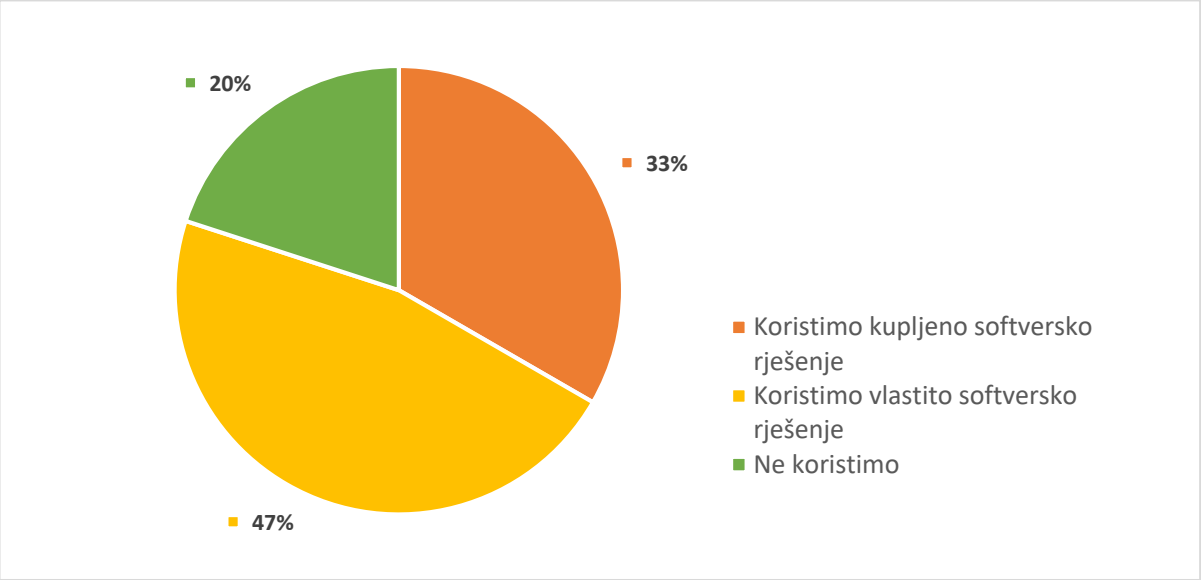
Slika 10. ESTIMUS - ekran za generiranje izvještaja

3. REZULTATI ISTRAŽIVANJA I ANALIZA DOBIVENIH REZULTATA

Ovaj rad obuhvatio je i istraživanje o zastupljenosti alata za podršku održavanju softvera. U nastavku rada prikazani su rezultati istraživanja koji su prikupljeni putem upitnika čiji je sadržaj dan u prilogu rada. Obrađeni su i analizirani svi odgovori na pitanja iz online upitnika. Za obradu jednostavnih pitanja iz upitnika nije bila potrebna napredna statistička obrada, pa se za obradu i analizu odgovora koristio samo alat Microsoft Excel.

S ciljem provjere prve hipoteze (Većina informatičkih tvrtki u Republici Hrvatskoj koristi vlastito softversko rješenje za podršku održavanju softvera), ispitanicima je postavljeno odgovarajuće pitanje, a rezultati su pokazali da 47% informatičkih tvrtki u Republici Hrvatskoj, koje su pristupile istraživanju, koristi vlastito softversko rješenje za podršku održavanju softvera (slika 11.). Ovime se može i potvrditi prva hipoteza ovog istraživačkog rada. Tvrtke preferiraju vlastita softverska rješenja zbog prilagođenih postavki, jednostavnog sučelja, specifičnih zahtjeva koje komercijalni alati ne ispunjavaju, bolje sigurnosti, manje kompleksnosti…

Trećina informatičkih tvrtki koristi kupljeno softversko rješenje, dok 20% tvrtki ne koristi alate za podršku održavanju softvera, što je iznenađujuće visok postotak. Budući da se neke tvrtke kojima je upitnik upućen ne bave razvojem, odnosno održavanjem softvera, nego prodajom računalnih komponenti te servisiranjem računala, može se pretpostaviti da takva rješenja njima ne trebaju, ali zbog anonimnosti ankete nemoguće je potvrditi tu pretpostavku.

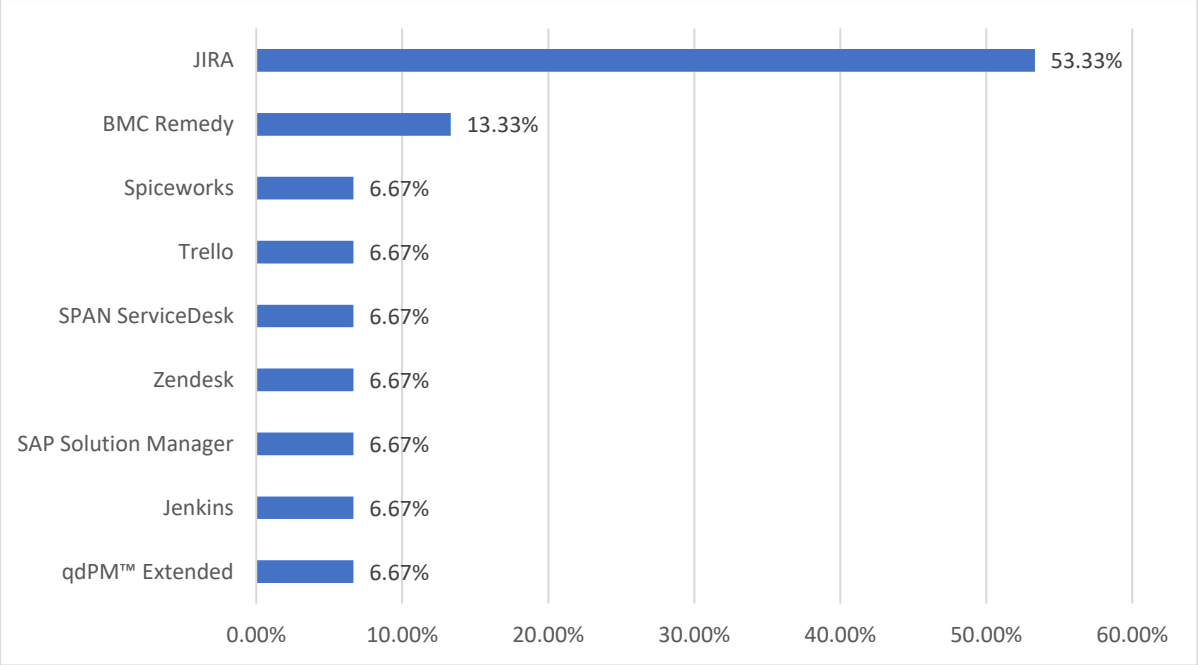


Slika 11. Prikaz odgovora na pitanje: "Koristite li neku vrstu softverskog alata za podršku održavanja softvera u Vašoj organizaciji/sektoru/odjelu?"

Petnaest ispitanika, koji su u prethodnom pitanju označili da koriste kupljeno softversko rješenje, prilikom odgovaranja na drugo pitanje, trebalo je navesti koji komercijalni alat koriste za podršku održavanju softvera.

Kao što je vidljivo na slici 12., najviše ispitanika, njih osam (53.33%), navelo je da koriste alat JIRA. JIRA je alat za upravljanje projektima koji omogućuje jednostavno praćenje zadataka, problema, grešaka i projekta, integraciju koda te izvještavanje i analizu [28]. Alat koristi preko 75.000 korisnika u 122 zemlje. Neke od organizacija, koje su u nekom trenutku koristile Jiru za praćenje bugova i upravljanje projektima, uključuju Fedora Commons, Hibernate, NASA, Skype Technologies, Twitter te Ministarstvo obrane Sjedinjenih Američkih Država [29].

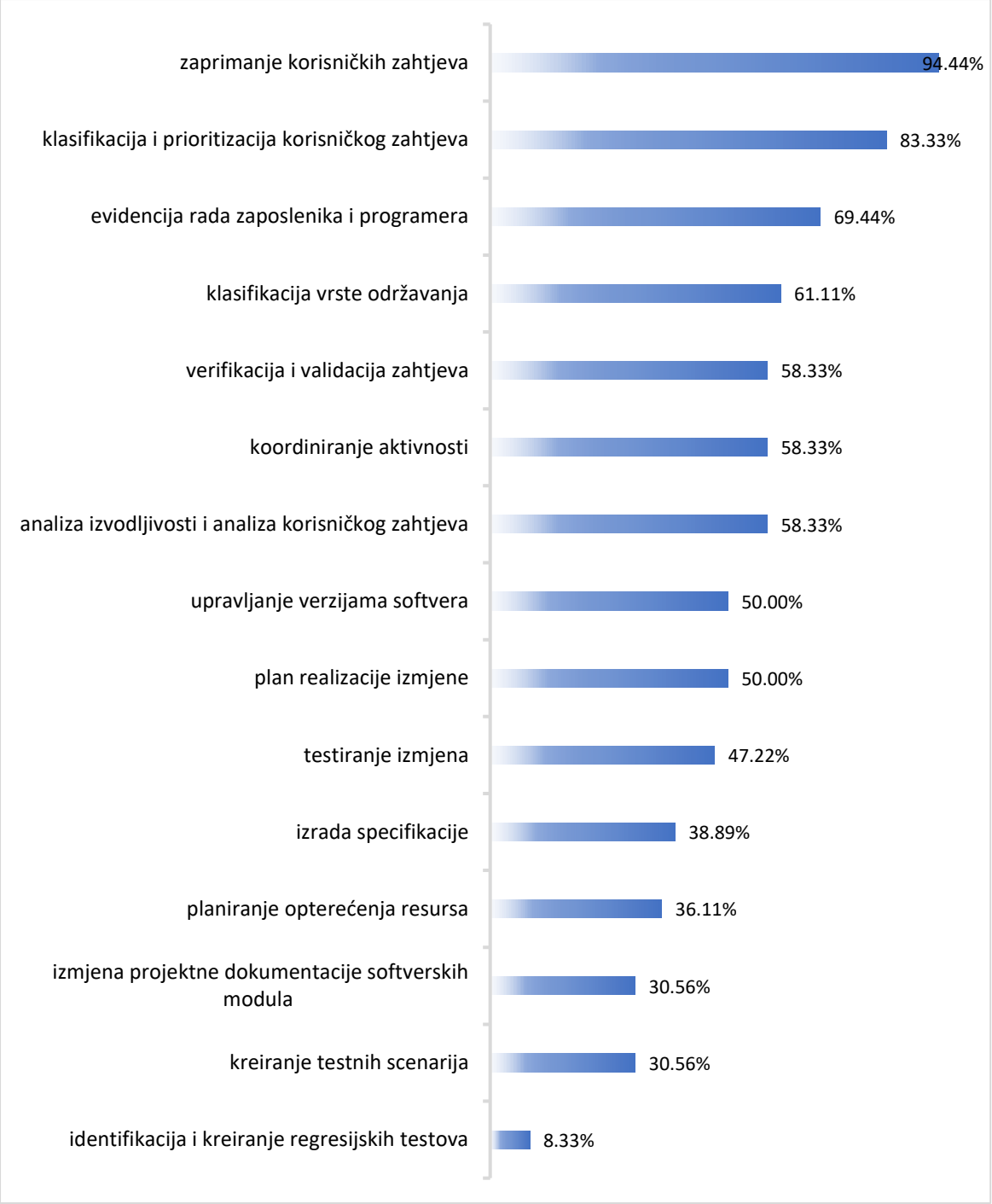
Alat BMC Remedy koristi 13.33% ispitanika, a po 6.67% ispitanika koristi sljedeće alate za obavljanje određenih procesa i aktivnosti održavanja softvera: Spiceworks, Trello, SPAN ServiceDes, SAP Solution Manager i qdPM™ Extended. Važno je napomenuti da je dvoje ispitanika napomenulo da uz alat JIRA, koriste i druge alate za podršku održavanju softvera. Jedan ispitanik koristi Zendesk, a drugi Jenkins.



Slika 12. Prikaz odgovora na pitanje: "Ako je Vaš odgovor bio „Koristimo kupljeno softversko rješenje“, možete li navesti koji komercijalni softverski alat koristite za podršku održavanja softvera?"

U trećem pitanju, trideset i šestero ispitanika, koji su naveli da koriste neki oblik alata za podršku održavanju softvera, trebalo je označiti za koje procese i aktivnosti održavanja koriste taj softver. Njihovi odgovori su grafički prikazani na slici 13. Najzastupljeniji odgovor (94.44% ispitanika) je za zaprimanje korisničkih zahtjeva, zatim (83.33%) za klasifikaciju i prioritizaciju korisničkog zahtjeva, za evidenciju rada zaposlenika i programera (69.44%), te za klasifikaciju vrste održavanja (61.11%). Odgovori koji su jednako zastupljeni (58.33%) su za analizu izvodljivosti i analizu korisničkog zahtjeva, koordiniranje aktivnosti te za verifikaciju i validaciju zahtjeva. Niže postotke dobili su planiranje realizacije izmjene i upravljanje verzijama softvera (50%), testiranje izmjena (47.22%), izrada specifikacije (38.89%), planiranje opterećenja resursa (36.11%) i izmjene projektne dokumentacije softverskih modula i kreiranja testnih scenarija (30.56%). Najmanje ispitanika, njih 8.33%, označilo je da alat koristi za identifikacije i kreiranje regresijskih testova.

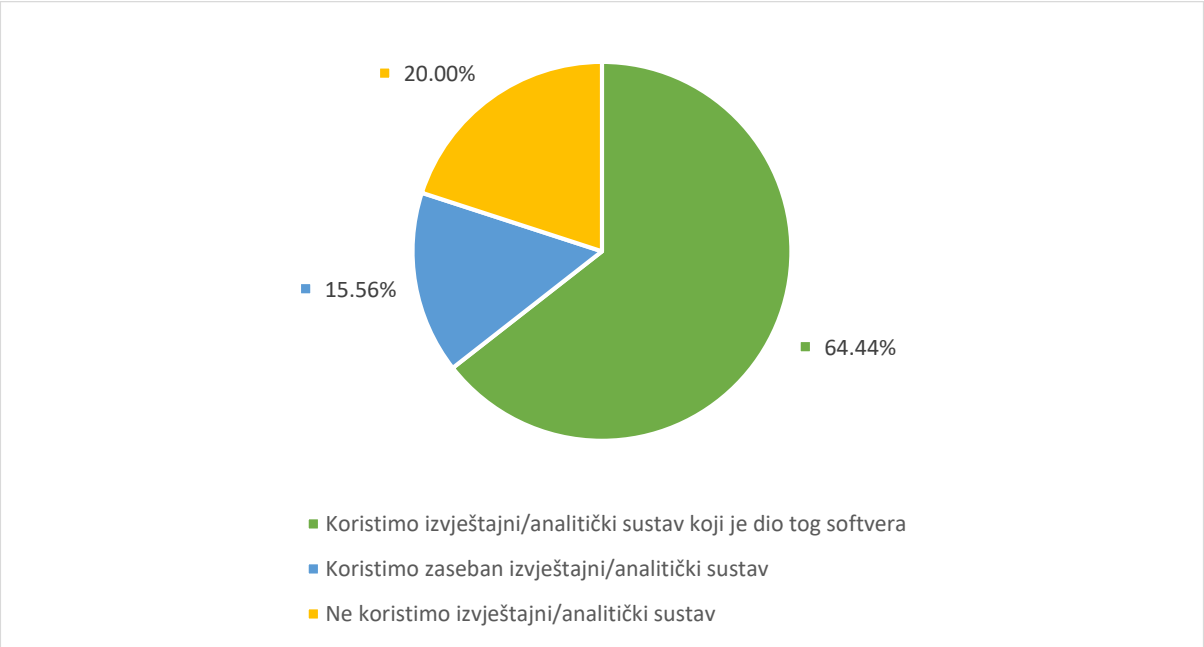
Ispitanici su imali i mogućnost upisivanja vlastitog odgovora na pitanje, pa su tako neki naveli da alate, uz navedene aktivnosti, koriste i za upravljanje serverima na puno nivoa, upravljanje promjenama, praćenje SLA (engl. Service Level Agreement - SLA), hosting firmi sa specifičnim zahtjevima i potrebama, rješavanje incidenata i problema te za naplatu.



Slika 13. Prikaz odgovora na pitanje: "Molim Vas označite za koje procese i aktivnosti, vezane za operativno rješavanje korisničkih zahtjeva, koristite taj softver "

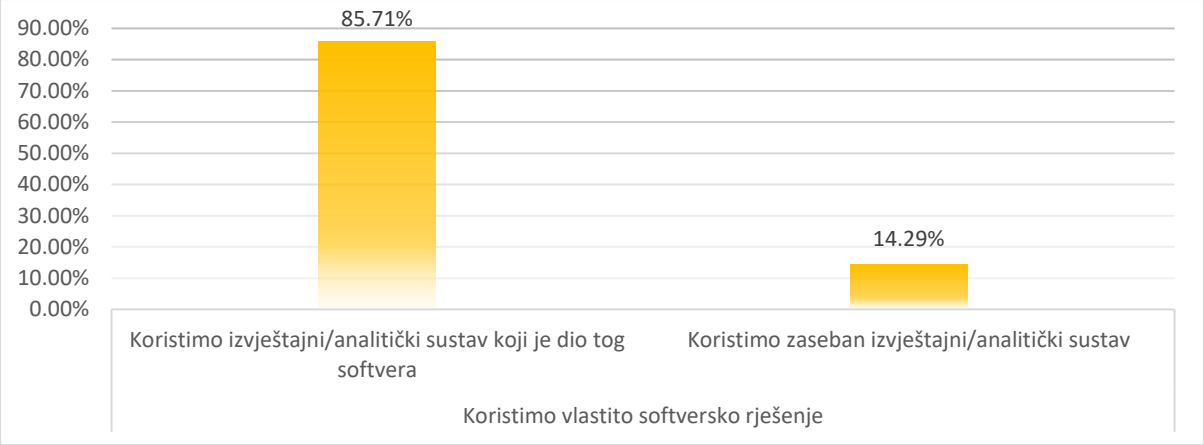
U četvrtom pitanju svi ispitanici su trebali označiti koriste li izvještajni, odnosno analitički sustav (slika 14.) i je li taj sustav dio alata za podršku održavanju softvera ili je zaseban sustav. Analitički sustav mora imati mogućnost kreiranja korisničkog izvještaja po raznim kriterijima i parametrima (npr. utrošak vremena djelatnika po pojedinom zahtjevu ili grupi zahtjeva,

postotak riješenih zahtjeva prema tipu ili kategoriji zahtjeva, itd.). Grafički prikaz odgovora na pitanje prikazan je na slici 15. Najviše ispitanika (64.44%) ispitanika označilo da je koriste analitički sustav koji je dio alata za podršku održavanju softvera. 20% ispitanika koristi zaseban analitički sustav, dok 15.56% ispitanika uopće ne koristi analitički sustav.



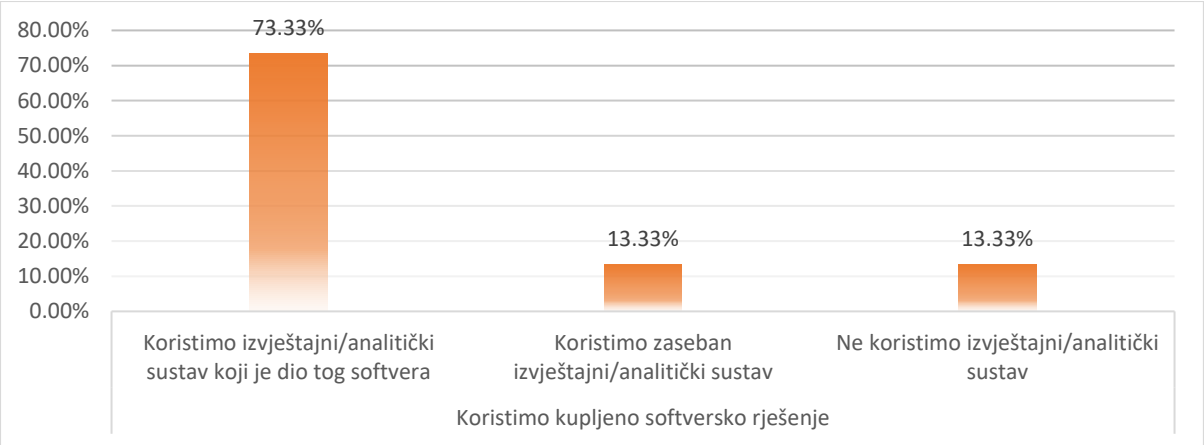
Slika 14. Prikaz odgovora na pitanje: "Koristite li izvještajni, odnosno analitički sustav koji je dio tog softvera ili zaseban sustav?"

Druga hipoteza (Informatičke tvrtke koje koriste vlastito softversko rješenje za podršku održavanju softvera većinom imaju analitički sustav uključen u taj softver) može se potvrditi slikom 15., gdje je prikazano da 85.71% informatičkih tvrtki, koje imaju vlastiti alat za podršku održavanju softvera, koristi analitički sustav koji je dio tog alata, dok 14.29% takvih tvrtki koristi zaseban analitički sustav.



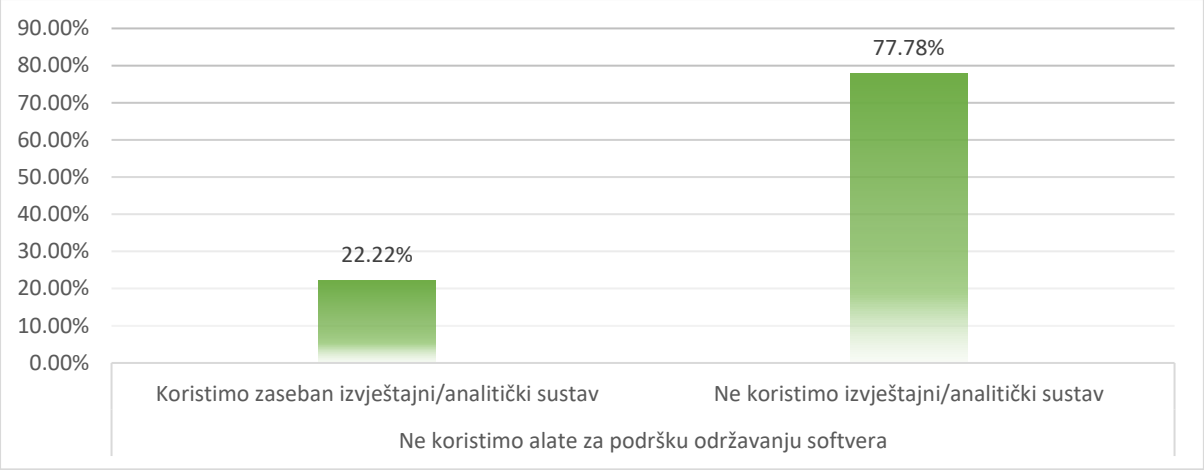
Slika 15. Udio korištenja određene vrste analitičkog sustav kod tvrtki koje koriste vlastite alate za podršku održavanju softvera

Slične rezultate može se vidjeti i kod informatičkih tvrtki koje koriste komercijalne alate za podršku održavanju softvera, pa tako 73.33% tvrtki koristi analitički sustav koji je dio tog komercijalnog alata, 13.33% koristi zaseban analitički sustav, dok 13.33% ne koristi analitički sustav (slika 16.).



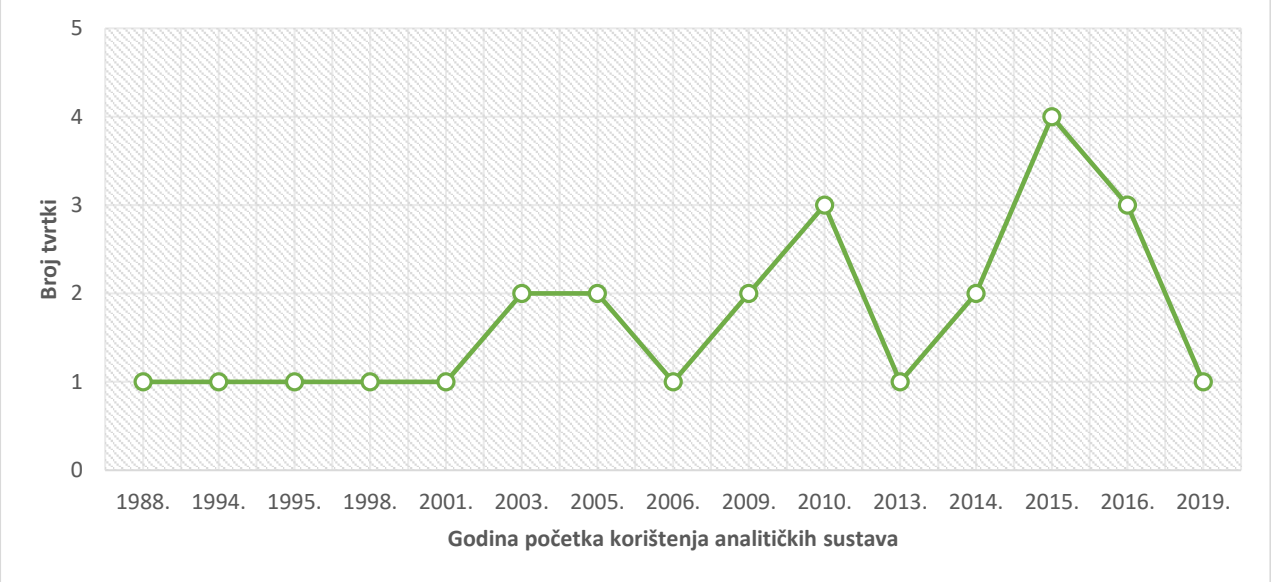
Slika 16. Udio korištenja određene vrste analitičkog sustav kod tvrtki koje koriste komercijalne alate za podršku održavanju softvera

Kod informatičkih tvrtki koje ne koriste alate za podršku održavanju softvera, 77.78% njih, ne koristi ni analitičke alate, dok 22.22% njih koristi analitičke alate (slika 17.).



Slika 17. Udio korištenja određene vrste analitičkog sustav kod tvrtki koje ne koriste alate za podršku održavanju softvera

Na peto pitanje, koje je glasilo „Ako koristite analitički sustav, od koje godine ga koristite?“, odgovorilo je dvadeset i šest ispitanika. Deset ispitanika, koji koriste analitički sustav, nije odgovorilo na ovo pitanje. Na slici 18. može se vidjeti koliko se tvrtki u kojoj godini počelo koristiti analitičkim sustavom. U razdoblju od 1988. godine do 2009. godine 10 tvrtki je počelo koristiti analitičke alate, dok je u razdoblju od 2009. godine do 2019. godine 16 tvrtki počelo koristiti analitičke alate, što je povećanje od 24%. Ovime se može potvrditi treća i posljednja hipoteza (Korištenje analitičkih sustava povećalo se posljednjih deset godina) ovoga istraživanja.



Slika 18. Broj tvrtki koje su počele koristiti analitičke alate određene godine

4. ZAKLJUČAK

Održavanje igra vrlo važnu ulogu u životnom ciklusu softverskog proizvoda. Kao što je ranije spomenuto, troškovi održavanja iznose do 85% ukupnih troškova razvoja softvera. Dok se "tradicionalno održavanje" odnosi samo na korektivne popravke grešaka u kodu, faza održavanja, također uključuje tri druga aspekta koji se smatraju elementima evolucije softvera. Adaptivno održavanje služi za modificiranje softvera za korištenje u novim okruženjima, kao što je nova hardverska platforma ili sučelje s drugim sustavom baze podataka. Perfektivno održavanje odnosi se na dodavanje nove funkcionalnosti proizvodu, obično kao rezultat zahtjeva kupca. Konačno, preventivno održavanje povećava održivost sustava, ažurirajući dokumentaciju ili mijenjajući strukturu softvera.

Postoje razni modeli održavanja koji koriste različite pristupe za organiziranje aktivnosti i procesa sljedećih faza održavanja softvera: identifikacija i klasifikacija problema, detaljna analiza i analiza izvodivosti, dizajniranje modifikacije, implementacija te modifikacije, testiranje te isporuka modificiranog sustava.

Brojni alati, kao što su alati za razumijevanje programa i obrnuti inženjering, alati za upravljanje konfiguracijom te alati za podršku testiranja pomažu u ispunjavanju zadataka održavanja softvera. Postojeći alati mogu se grupirati u dvije kategorije. Prva kategorija sadrži alate koji pomažu u zadacima održavanja ili razvoja pružanjem informacija koje se generiraju dinamički, koristeći tehnike analize koda. Druga kategorija sadrži alate koji generiraju dokumentaciju iz izvornog koda te komentara i napomena koje su unijeli razvojni programeri o svakom segmentu programa.

U radu je detaljno opisan softverski proizvod ESTIMUS, koji podupire proces održavanja brzom i jednostavnom organizacijom vremena i kadrova uključenih u projekte. Navedene su njegove funkcionalnosti te su prikazani neki važniji ekrani korisničkog sučelja.

U okviru ovog diplomskog rada napravljeno je istraživanje o upotrebi alata za podršku održavanju softvera u informatičkim tvrtkama na području Republike Hrvatske. Opisan je problem istraživanja, navedeni su ciljevi istraživanja te su postavljene hipoteze istraživanja. Istraživanju je pristupilo 45 informatičkih tvrtki, kojima je anketni upitnik poslan elektroničkim putem. Odgovori na upitnik su analizirani i najvažniji zaključci su grafički prikazani. Istraživanjem se dolazi do zaključka da gotovo polovica informatičkih tvrtki u

Republici Hrvatskoj, koje su pristupile istraživanju, koristi vlastito softversko rješenje za podršku održavanju softvera, nešto manje njih koristi kupljeno softversko rješenje, dok čak 20% tvrtki ne koristi alate za podršku održavanju softvera. Od komercijalnih alata, više od polovice ispitanika koristi alat JIRA. Alati za podršku održavanju softvera najviše se koriste za razne aktivnosti tijekom održavanja softvera, a uglavnom za one koje se direktno tiču samih korisničkih zahtjeva te evidenciju rada zaposlenika i programera. Alati se nešto manje koriste za izmjenu projektne dokumentacije softverskih modula i kreiranje testnih scenarija, a najmanje se koriste za identifikaciju i kreiranje regresijskih testova. Analitičke sustave koji su dio alata za podršku održavanju softvera, koristi najviše ispitanika (64.44%), zaseban analitički sustav koristi 20% ispitanika, dok 15.56% ispitanika ne koristi analitički sustav. 85.71% informatičkih tvrtki, koje imaju vlastiti alat za podršku održavanju softvera, imaju i analitički sustav koji je dio tog alata. Također, može se vidjeti trend povećanja korištenja analitičkih alata tijekom godina. Ovi rezultati istraživanja potvrdili su sve tri hipoteze istraživanja.

5. LITERATURA

[1] L. Erlikh, »Leveraging legacy system dollars for e-business« Lipanj 2000. Dostupno na: https://www.researchgate.net/publication/3426422_Leveraging_legacy_system_dollars_for_e-business. [Pokušaj pristupa 2. Svibanj 2019.].

[2] B. Lientz, E. B. Swanson i G. E. Tompkins, »Characteristics of Application Software Maintenance« Lipanj 1978. Dostupno na: https://www.researchgate.net/publication/225070557_Characteristics_of_Application_Software_Maintenance. [Pokušaj pristupa 5. Svibanj 2019.].

[3] P. Grubb i A. A. Takang, Software maintenance: concepts and practice (2nd Edition), Singapore: World Scientific Publishing Co. Pte. Ltd., 2003.

[4] F. Niessnik i V. H. Vliet, »Software Maintenance from a Service Perspective« 2000. Dostupno na: https://www.cs.vu.nl/~hans/publications/y2000/SMRP00-2.pdf. [Pokušaj pristupa 10. Lipanj 2019.].

[5] P. K. Tiwari i A. Chauhan , »Categories of Sofware Maintenance« Dostupno na: https://www.scribd.com/doc/31214177/Categories-of-Software-Maintenance. [Pokušaj pristupa 10. Svibanj 2019.].

[6] R. Manger, Softversko inženjerstvo, Zagreb: Sveučilište u Zagrebu, 2012.

[7] I. Sommerville, Software engineering, Pearson Education, Inc., 2011.

[8] A. Morožin , »Rudarenje podataka u projektnom menadžmentu u ovkiru razvoja softvera« Rujan 2017. Dostupno na: https://repozitorij.efst.unist.hr/islandora/object/efst%3A1673/datastream/PDF/view. [Pokušaj pristupa 15. Svibanj 2019.].

[9] WHOISWHOINIT, »TOP 1000 hrvatskih visoko-tehnoloških tvrtki« 2017. Dostupno na: https://www.whoiswhoinit.com/novosti/26-top-1000-hrvatskih-visoko-tehnoloskih-tvrtki. [Pokušaj pristupa 20. Veljača 2019.].

[10] »IEEE Standard for Software Maintenance« IEEE, 21. Listopad 1998. Dostupno na: https://pdfs.semanticscholar.org/9cf0/081cd97d8d1f782a3380e56c57201487a786.pdf. [Pokušaj pristupa 20. Ožujak 2019].

[11] W. M. Osborne i E. J. Chikofsky, »Fitting pieces to the maintenance puzzle« Siječanj 1990. Dostupno na: https://www.researchgate.net/publication/297805608_FITTING_PIECES_TO_THE_MAINTENANCE_PUZZLE. [Pokušaj pristupa 21. Svibanj 2019.].

[12] N. F. Schneidewind, »The state of software maintenance« 3. Ožujak 1987. Dostupno na: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.821.9980&rep=rep1&type=pdf. [Pokušaj pristupa 23. Svibanj 2019.].

[13] T. Pigoski, »Software maintenance« Svibanj 2001. Dostupno na: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.582.5421&rep=rep1&type=pdf. [Pokušaj pristupa 20. Ožujak 2019.].

[14] T. M. Pigoski, »Practical software maintenance: best practices for managing your software investment« 1997. Dostupno na: https://www.scribd.com/document/76129139/Pigoski-Practical-Software-Maintenance-Best-Practices-for-Managing-Your-Software-Investment. [Pokušaj pristupa 15. Svibanj 2019.].

[15] M. Pavlić, Informacijski sustavi, Rijeka: Odjel za informatiku Sveučilišta u Rijeci, 2009.

[16] G. Canfora i A. Cimitile, »Software Maintenance« 29. Studeni 2000. Dostupno na: https://pdfs.semanticscholar.org/4393/e3d118269f374df7f9828a7be034b645336b.pdf. [Pokušaj pristupa 25. Svibanj 2019.].

[17] K. Erdil, E. Finn, K. Keating, J. Mieattle, S. Park i D. Yoon, »Software Maintenance As Part of the Software Life Cycle« 2003.

[18] P. Oreški, »O standardima za razvoj softvera« Journal of Information and Organizational Sciences, pp. 159-167, 1992.

[19] B. P. Lientz i E. B. Swanson, Software Maintenance Management: A Study of the Maintenance of Computer Application Software in 487 Data Processing Organizations, Addison-Wesley, 1980.

[20] J. T. Nosek i P. Palvia, »Software maintenance management: Changes in the last decade« Rujan 1990. Dostupno na: https://www.researchgate.net/publication/229666091_Software_maintenance_management_Changes_in_the_last_decade. [Pokušaj pristupa 25. Svibanj 2019.].

[21] Sousa i Moreira, »A Survey on the Software Maintenance Process« Prosinac 1998. Dostupno na: https://www.researchgate.net/publication/3784749_A_Survey_on_the_Software_Maintenance_Process. [Pokušaj pristupa 14. Svibanj 2019.].

[22] ProfessionalQA, »Software Maintenance Models« 24. Kolovoz 2018. Dostupno na: http://www.professionalqa.com/software-maintenance-models. [Pokušaj pristupa 02. Lipanj 2019.].

[23] Shivani, »Detailed Study of Software Maintenance Models« Srpanj 2015. Dostupno na: https://www.airo.co.in/paper/admin/upload/international_volume/6968Shivani%20International_%20vol%205.pdf. [Pokušaj pristupa 03. Lipanj 2019.].

[24] M. Tamimi, »Software Evolution and Maintenance Models,« 10. Ožujak 2018.. [Mrežno]. Available: https://www.slideshare.net/moutasmtamimi/software-evolution-and-maintenance-models. [Pokušaj pristupa 15. Svibanj 2019.].

[25] G. Visaggio, »Assessing the maintenance process through replicated, controlled experiments« Siječanj 1999. Dostupno na: https://www.researchgate.net/publication/222481617_Assessing_the_maintenance_process_through_replicated_controlled_experiments. [Pokušaj pristupa 04. Lipanj 2019.].

[26] C. M. Beath i B. E. Swanson, »Maintaining Information Systems in Organization« Siječanj 1990. Dostupno na: https://www.researchgate.net/publication/234814138_Maintaining_Information_Systems_in_Organization. [Pokušaj pristupa 05. Lipanj 2019.].

[27] T. Hung., » Software Maintenance« Dostupno na: https://cnx.org/contents/YlrpVYtq@1/Software-Maintenance. [Pokušaj pristupa 06. Lipanj 2019.].

[28] Ris, »Estimus« Dostupno na: https://ris.hr/proizvodi/estimus/. [Pokušaj pristupa 10. Lipanj 2019.].

[29] Info novitas, »Jira Project Trackin,« Dostupno na: https://www.info-novitas.hr/rjesenja/atlassian-rjesenja/jira-project-tracking/. [Pokušaj pristupa 09. Lipanj 2019.].

[30] Wikipedia, »Jira (software)« Dostupno na: https://en.wikipedia.org/wiki/Jira_(software)#cite_ref-9. [Pokušaj pristupa 05. Lipanj 2019.].

6. POPIS SLIKA I TABLICA

7. PRILOZI

Anketni upitnik

Upotreba alata za podršku upravljanja održavanjem softvera i korisničkih zahtjeva

Poštovani,
zamolio bih Vas da odvojite 2 minute Vašeg vremena i odgovorite na sljedećih 5 pitanja o upotrebi alata za podršku održavanja softvera (alati kao što su alati za primanje korisničkih zahtjeva, za praćenje razvoja softvera, za koordiniranje aktivnosti itd.) u Vašoj organizaciji/sektoru/odjelu. Istraživanje se provodi na području Republike Hrvatske u svrhu izrade diplomskog rada na Odjelu za informatiku, Sveučilišta u Rijeci. Anketa je anonimna i rezultati služe isključivo u svrhu znanstvenog istraživanja, te se neće upotrijebiti u niti jednu drugu svrhu.

1. Koristite li neku vrstu softverskog alata za podršku održavanja softvera u Vašoj

- ○ Ne koristimo
- ○ Koristimo vlastito softversko rješenje
- ○ Koristimo kupljeno softversko rješenje

2. Ako je Vaš odgovor bio „Koristimo kupljeno softversko rješenje“, možete li navesti koji komercijalni softverski alat koristite za podršku održavanja softvera? (npr. Corrigo , UpKeep, Dude Solutions, Fiix, ManagerPlus, Maintenance Connection, Proteus, eMaint, eWorkOrders, Tikkit, Estimus ...)

Tekst kratkog odgovora

3. Molim Vas označite za koje procese i aktivnosti, vezane za operativno rješavanje korisničkih zahtjeva, koristite taj softver

- [ ] zaprimanje korisničkih zahtjeva
- [ ] analiza izvodljivosti i analiza korisničkog zahtjeva
- [ ] klasifikacija i prioritizacija korisničkog zahtjeva
- [ ] plan realizacije izmjene
- [ ] izrada specifikacije
- [ ] izmjena projektne dokumentacije softverskih modula
- [ ] kreiranje testnih scenarija
- [ ] identifikacija i kreiranje regresijskih testova
- [ ] testiranje izmjena
- [ ] verifikacija i validacija zahtjeva
- [ ] upravljanje verzijama softvera
- [ ] planiranje opterećenja resursa
- [ ] evidencija rada zaposlenika i programera
- [ ] klasifikacija vrste održavanja
- [ ] koordiniranje aktivnosti
- [ ] Ostalo...

4. Koristite li izvještajni, odnosno analitički sustav koji je dio tog softvera ili zaseban sustav? Analitički sustav mora imati mogućnost kreiranja korisničkog izvještaja po raznim kriterijima i parametrima (npr. utrošak vremena djelatnika po pojedinom zahtjevu ili grupi zahtjeva, postotak riješenih zahtjeva prema tipu ili kategoriji zahtjeva itd.)

- ○ Ne koristimo izvještajni/analitički sustav
- ○ Koristimo izvještajni/analitički sustav koji je dio tog softvera
- ○ Koristimo zaseban izvještajni/analitički sustav

5. Ako koristite analitički sustav, od koje godine ga koristite?

Tekst kratkog odgovora