SVEUČILIŠTE/UNIVERZITET „VITEZ“

FAKULTET INFORMACIJSKIH TEHNOLOGIJA

ZULKA MUSIĆ

PISANJE DOKUMENTA SA SOFTVERSKIM ZAHTJEVIMA

SEMINARSKI RAD

Predmet: Prikupljanje softverskih zahtjeva

Mentor: prof.dr.sc. Tihomir Latinović

Asistent: mr. Denijal Imamović

Travnik, 2026. godina

SVEUČILIŠTE/UNIVERZITET „VITEZ“

FAKULTET INFORMACIONIH TEHNOLOGIJA

PISANJE DOKUMENTA SA SOFTVERSKIM ZAHTJEVIMA

SEMINARSKI RAD

IZJAVA: Ja **Zulka Musić**, studentica Sveučilišta/Univerziteta „VITEZ“, Indeks broj: **390-24/RIIT** odgovorno i uz moralnu i akademsku odgovornost izjavljujem da sam ovaj rad izradila potpuno samostalno uz korištenje citirane literature i pomoć predmetnog profesora.

Student: Zulka Musić

Predmet: Prikupljanje softverskih alata

Mentor: prof.dr.sc. Tihomir Latinović

Asistent: mr. Denijal Imamović

SADRŽAJ

1. UVOD

1.1. PROBLEM, PREDMET I OBJEKT ISTRAŽIVANJA

U razvoju savremenih softverskih sistema jedan od ključnih problema predstavlja neadekvatno definisanje i dokumentovanje softverskih zahtjeva. U praksi se često dešava da korisnici ne mogu precizno izraziti svoje potrebe, dok razvojni timovi zahtjeve tumače na različite načine. Takva situacija dovodi do nesporazuma, pogrešne implementacije funkcionalnosti, dodatnih izmjena tokom razvoja, povećanja troškova i kašnjenja u isporuci softverskog proizvoda.

Nedostatak kvalitetnog dokumenta sa softverskim zahtjevima može rezultirati razvojem sistema koji ne zadovoljava stvarne potrebe korisnika. Upravo zbog toga se u softverskom inženjerstvu poseban naglasak stavlja na pravilno pisanje i strukturiranje dokumenta sa softverskim zahtjevima, koji služi kao osnova za sve naredne faze razvoja softvera.

Predmet istraživanja ovog seminarskog rada jeste proces pisanja dokumenta sa softverskim zahtjevima, kao i njegova struktura i sadržaj.

Objekt istraživanja predstavlja proces prikupljanja, analize i dokumentovanja softverskih zahtjeva u okviru razvoja softverskih sistema, s ciljem osiguranja kvaliteta i uspješnosti softverskog projekta.

1.2. SVRHA I CILJEVI ISTRAŽIVANJA

Svrha ovog rada je da se prikaže značaj pravilnog i sistematičnog pisanja dokumenta sa softverskim zahtjevima, te da se ukaže na njegovu ulogu u uspješnom razvoju softverskih sistema.

Ciljevi istraživanja su:

- definisati pojam softverskih zahtjeva
- objasniti vrste softverskih zahtjeva
- analizirati strukturu dokumenta sa softverskim zahtjevima
- prikazati metode prikupljanja zahtjeva
- kroz praktičan primjer demonstrirati način pisanja SRS dokumenta

1.3. RADNA I POMOĆNE HIPOTEZE

Radna hipoteza rada glasi:

Kvalitetno napisan dokument sa softverskim zahtjevima ima presudan uticaj na uspješnost razvoja softverskog sistema.

Pomoćne hipoteze su:

- jasno definisani softverski zahtjevi smanjuju mogućnost grešaka u razvoju softvera
- kvalitetna dokumentacija poboljšava komunikaciju između korisnika i razvojnog tima
- primjena standarda doprinosi većoj pouzdanosti softverskog sistema

1.4. NAUČNE METODE

U izradi ovog seminarskog rada korištene su sljedeće naučne metode:

- metoda analize stručne i naučne literature
- metoda deskripcije
- metoda komparacije
- induktivno-deduktivna metoda

1.5. STRUKTURA RADA

Rad je strukturiran u devet poglavlja.

U prvom poglavlju dat je uvod u temu rada, u okviru kojeg su definisani problem, predmet i objekt istraživanja, svrha i ciljevi rada, radna i pomoćne hipoteze, kao i korištene naučne metode. Na kraju ovog poglavlja prikazana je struktura rada.

Drugo poglavlje bavi se primjenom pisanja dokumenta sa softverskim zahtjevima. U ovom poglavlju obrađena je uloga dokumenta sa softverskim zahtjevima u razvoju softvera, njegova primjena u različitim vrstama softverskih projekata, učesnici koji koriste ovaj dokument, kao i značaj dokumenta sa softverskim zahtjevima za kvalitet softverskog proizvoda i njegova povezanost sa softverskim standardima.

U trećem poglavlju prikazan je historijat pisanja softverskih zahtjeva. Posebna pažnja posvećena je ranom razvoju softvera i početnim pristupima definisanja zahtjeva, razvoju softverskog inženjerstva i institucionalizaciji zahtjeva, standardizaciji pisanja softverskih zahtjeva, razvoju discipline Requirements Engineering, savremenim pristupima pisanju softverskih zahtjeva, kao i značaju historijskog razvoja za savremenu praksu.

Četvrto poglavlje obrađuje osnovne pojmove pisanja softverskih zahtjeva. U ovom dijelu rada definisan je pojam softverskog zahtjeva, objašnjeni su funkcionalni i nefunkcionalni zahtjevi, razlika između korisničkih i sistemskih zahtjeva, dokument sa softverskim zahtjevima (SRS), uloga zainteresovanih strana, kao i kvalitet softverskih zahtjeva.

Peto poglavlje odnosi se na primjenu pisanja softverskih zahtjeva u praksi. U okviru ovog poglavlja analizirana je uloga softverskih zahtjeva u razvoju softverskih sistema, njihova primjena u različitim fazama životnog ciklusa softvera, primjena zahtjeva u tradicionalnim i agilnim metodologijama, kao i njihova primjena u različitim oblastima. Takođe su razmotreni problemi u praksi i prednosti primjene dokumentovanih softverskih zahtjeva.

U šestom poglavlju obrađena je upotreba računara, metoda i alata u pisanju softverskih zahtjeva. Ovo poglavlje obuhvata ulogu računara u procesu pisanja zahtjeva, metode prikupljanja softverskih zahtjeva, modelovanje zahtjeva, upotrebu statistike u njihovoj analizi, kao i softverske alate za upravljanje zahtjevima.

Sedmo poglavlje predstavlja vlastiti primjer i eksperiment u oblasti pisanja softverskih zahtjeva. U ovom dijelu rada definisani su cilj i svrha eksperimenta, opisan je sistem koji je predmet eksperimenta, prikazana je metodologija, analizirani su neformalno i formalno definisani zahtjevi, izvršeno je poređenje rezultata, te data analiza i diskusija rezultata, kao i ograničenja eksperimenta.

U osmom poglavlju dat je zaključak rada, u kojem su sumirani glavni rezultati istraživanja i izvedeni konačni zaključci.

Deveto poglavlje sadrži spisak korištene literature.

2. PRIMJENA PISANJA DOKUMENTA SA SOFTVERSKIM ZAHTJEVIMA

2.1. ULOGA DOKUMENATA SA SOFTVERSKIM ZAHTJEVIMA U RAZVOJU SOFTVERA

Dokument sa softverskim zahtjevima, poznat kao SRS (Software Requirements Specification), predstavlja osnovni dokument u fazi analize i planiranja softverskog sistema. Njegova primarna uloga jeste da precizno definiše šta softverski sistem treba da radi, koje funkcionalnosti mora imati, kao i pod kojim ograničenjima treba funkcionisati. Ovaj dokument služi kao formalni dogovor između korisnika i razvojnog tima.

U praksi, dokument sa softverskim zahtjevima koristi se kao polazna tačka za sve naredne faze razvoja softvera, uključujući dizajn sistema, implementaciju, testiranje i održavanje. Prema literaturi iz oblasti softverskog inženjerstva, jasno definisani zahtjevi direktno utiču na smanjenje broja grešaka u kasnijim fazama razvoja (Sommerville, 2011). Također, IEEE standardi naglašavaju da SRS dokument mora biti jasan, konzistentan i provjerljiv kako bi bio efikasan u praksi (IEEE Std 830, 1998) (ISO/IEC/IEEE 29148, 2011).

2.2. PRIMJENA SRS DOKUMENTA U RAZLIČITIM VRSTAMA SOFTVERSKIH PROJEKATA

Dokument sa softverskim zahtjevima ima široku primjenu u gotovo svim vrstama softverskih projekata. Koristi se prilikom razvoja informacionih sistema u obrazovanju, zdravstvu, bankarstvu i državnoj upravi, ali i kod razvoja web i mobilnih aplikacija, poslovnih softverskih rješenja i informacionih platformi.

U velikim softverskim projektima SRS dokument omogućava koordinaciju između više razvojnih timova, dok u manjim projektima pomaže da se jasno definiše obim

sistema. Prema stručnim izvorima iz oblasti softverskog inženjerstva, čak i jednostavni softverski sistemi zahtijevaju osnovni oblik dokumenta sa softverskim zahtjevima kako bi se izbjegle nejasnoće i pogrešna tumačenja zahtjeva (Pressman, 2010).

2.3. UČESNICI KOJI KORISTE DOKUMENT SA SOFTVERSKIM ZAHTJEVIMA

Primjena dokumenta sa softverskim zahtjevima nije ograničena samo na analitičare i programere. Ovaj dokument koriste različite zainteresovane strane tokom cijelog životnog ciklusa razvoja softvera.

Analitičari koriste SRS dokument za definisanje i strukturiranje zahtjeva korisnika. Programeri se oslanjaju na ovaj dokument prilikom implementacije funkcionalnosti sistema. Testeri koriste zahtjeve kao osnovu za izradu testnih slučajeva i provjeru ispravnosti rada sistema. Projektni menadžeri koriste dokument sa zahtjevima za planiranje resursa, praćenje napretka i kontrolu obima projekta. Ovakva višestruka primjena SRS dokumenta potvrđuje njegovu centralnu ulogu u razvoju softvera (Sommerville, 2011).

2.4. ZNAČAJ DOKUMENTA SA SOFTVERSKIM ZAHTJEVIMA ZA KVALITET SOFTVERA

Jedan od osnovnih ciljeva pisanja dokumenta sa softverskim zahtjevima jeste osiguranje kvaliteta softverskog proizvoda. Kvalitetan SRS dokument omogućava jasnu specifikaciju funkcionalnosti sistema, čime se smanjuje vjerovatnoća pogrešne implementacije.

Prema stručnim izvorima, veliki broj grešaka u softverskim projektima potiče upravo iz faze analize zahtjeva. Ukoliko su zahtjevi nejasni ili nepotpuni, greške se prenose u kasnije faze razvoja, gdje ih je znatno teže i skuplje ispraviti (Pressman, 2010). Zbog toga

se pravilno pisanje dokumenta sa softverskim zahtjevima smatra jednim od ključnih faktora uspješnog softverskog projekta.

2.5. POVEZANOST DOKUMENTA SA SOFTVERSKIM ZAHTJEVIMA I SOFTVERSKIH STANDARDA

Primjena standarda u pisanju dokumenta sa softverskim zahtjevima doprinosi njegovoj kvaliteti i jednoobraznosti. IEEE standardi, kao što su IEEE 830 i IEEE 29148, definišu strukturu, sadržaj i karakteristike koje SRS dokument treba da posjeduje (IEEE Std 830, 1998) (ISO/IEC/IEEE 29148, 2011).

Primjenom ovih standarda osigurava se da dokument bude jasan, konzistentan, potpun i provjerljiv. Standardizacija omogućava lakše razumijevanje zahtjeva među različitim timovima i organizacijama, što je posebno važno u velikim i kompleksnim softverskim projektima. Prema preporukama iz literature, korištenje standarda predstavlja dobru praksu u profesionalnom razvoju softvera (Sommerville, 2011).

3. HISTORIJAT PISANJA SOFTVERSKIH ZAHTJEVA

3.1. RANI RAZVOJ SOFTVERA I POČETNI PRISTUPI DEFINISANJA ZAHTJEVA

U ranim fazama razvoja softvera, tokom pedesetih i šezdesetih godina dvadesetog vijeka, softverski sistemi su bili razvijani prvenstveno za vojne, akademske i istraživačke potrebe. U tom periodu razvoj softvera bio je usko povezan sa hardverom, a zahtjevi su se rijetko formalno dokumentovali. Programeri su često istovremeno obavljali ulogu analitičara, dizajnera i implementatora, zbog čega nije postojala jasna potreba za posebnim dokumentima sa softverskim zahtjevima (Kotonya & Sommerville, 1998).

Zahtjevi su se uglavnom prenosili usmenim putem ili kroz neformalne zapise, što je bilo prihvatljivo zbog malog broja korisnika i ograničene složenosti sistema. Međutim, kako navode autori iz oblasti zahtjevnog inženjerstva, ovakav pristup nije omogućavao dugoročnu održivost softverskih sistema niti njihovu lakšu nadogradnju (Young, 2004).

3.2. RAZVOJ SOFTVERSKOG INŽENJERSTVA I INSTITUCIONALIZACIJA ZAHTJEVA

Tokom sedamdesetih godina dolazi do naglog povećanja složenosti softverskih sistema, što je dovelo do čestih problema u razvoju, kao što su prekoračenje budžeta, kašnjenje projekata i neispunjavanje očekivanja korisnika. Ovi problemi su doveli do formiranja softverskog inženjerstva kao posebne discipline, čiji je cilj bio uvođenje sistematičnih metoda u razvoj softvera.

U ovom periodu zahtjevi se počinju posmatrati kao ključni element uspješnog softverskog projekta. Uvodi se praksa njihove formalne analize i dokumentovanja, a dokument sa softverskim zahtjevima postaje centralni artefakt koji definiše obim sistema i služi kao osnova za dalji razvoj (Kotonya & Sommerville, 1998).

3.3. STANDARDIZACIJA PISANJA SOFTVERSKIH ZAHTJEVA

Sa daljim razvojem softverskih sistema i globalizacijom IT industrije javlja se potreba za standardizacijom načina pisanja dokumentacije. Standardi omogućavaju jednoobraznost u strukturi i terminologiji, čime se olakšava saradnja između različitih organizacija i razvojnih timova. Prema međunarodnim preporukama, dokument sa softverskim zahtjevima mora biti jasan, potpun i nedvosmislen kako bi se izbjegle pogrešne interpretacije (ISO/IEC/IEEE 29148, 2011).

Standardizacija pisanja zahtjeva doprinijela je razvoju profesionalnih praksi u softverskom inženjerstvu i omogućila lakše upravljanje velikim i kompleksnim projektima. Na taj način dokumentovanje softverskih zahtjeva postalo je sastavni dio savremenih softverskih metodologija.

3.4. RAZVOJ DISCIPLINE REQUIREMENTS ENGINEERING

Tokom devedesetih godina razvija se posebna disciplina poznata kao Requirements Engineering, koja se fokusira isključivo na prikupljanje, analizu, validaciju i dokumentovanje zahtjeva. Ova disciplina naglašava važnost kontinuirane saradnje sa korisnicima i zainteresovanim stranama tokom cijelog životnog ciklusa softvera (Wiegers & Beatty, 2013).

Razvoj Requirements Engineering discipline doveo je do jasnijeg razgraničenja između korisničkih i sistemskih zahtjeva, kao i do uvođenja različitih tehnika za njihovu validaciju. Prema savremenim izvorima, pravilno upravljanje zahtjevima predstavlja jedan od ključnih faktora uspješnosti softverskih projekata (Young, 2004).

3.5. SAVREMENI PRISTUPI PISANJU SOFTVERSKIH ZAHTJEVA

U savremenom razvoju softvera pisanje dokumenta sa softverskim zahtjevima prilagođeno je potrebama fleksibilnih razvojnih metodologija. Iako se danas često koriste agilni pristupi koji favorizuju kraću i iterativnu dokumentaciju, osnovni principi zahtjevnog inženjerstva ostaju nepromijenjeni. Zahtjevi moraju biti jasno definisani, razumljivi i provjerljivi kako bi sistem ispunio očekivanja korisnika.

Savremeni pristupi kombinuju formalne dokumente sa praktičnim tehnikama, kao što su korisničke priče i iterativno unapređivanje zahtjeva. Na ovaj način postiže se ravnoteža između fleksibilnosti i potrebe za pouzdanom dokumentacijom, što potvrđuje trajni značaj dokumenta sa softverskim zahtjevima u razvoju softvera (Wiegers & Beatty, 2013).

3.6. ZNAČAJ HISTORIJSKOG RAZVOJA ZA SAVREMENU PRAKSU

Razumijevanje historijskog razvoja pisanja softverskih zahtjeva omogućava bolje sagledavanje savremenih praksi u softverskom inženjerstvu. Evolucija od neformalnih zapisa ka standardizovanim dokumentima pokazuje da kvalitetno dokumentovanje zahtjeva predstavlja odgovor na probleme koji su se javljali u prošlosti.

Savremena praksa pisanja dokumenta sa softverskim zahtjevima zasniva se na dugogodišnjem iskustvu i istraživanjima, čime se osigurava stabilnost, kvalitet i uspješnost softverskih sistema (ISO/IEC/IEEE 29148, 2011).

4. OSNOVNI POJMOVI PISANJA SOFTVERSKIH ZAHTJEVA

4.1. SOFTVERSKI ZAHTJEV

Softverski zahtjev predstavlja formalno definisanu potrebu ili očekivanje koje softverski sistem mora ispuniti kako bi zadovoljio korisnika ili drugu zainteresovanu stranu. Zahtjevi opisuju funkcionalnosti sistema, ali i ograničenja pod kojima sistem mora raditi, uključujući performanse, sigurnost i pouzdanost (Robertson & Robertson, 2013).

Jasno definisani softverski zahtjevi omogućavaju razumijevanje onoga što sistem treba da radi, bez ulaska u detalje implementacije. Na taj način zahtjevi služe kao komunikacioni most između korisnika i razvojnog tima.

Primjer softverskog zahtjeva:

„Softverski sistem mora omogućiti registrovanim korisnicima prijavu na sistem korištenjem korisničkog imena i lozinke.“

Ovaj primjer jasno definiše šta sistem mora omogućiti, bez ulaska u način implementacije.

4.2. FUNKCIONALNI ZAHTJEVI

Funkcionalni zahtjevi opisuju konkretne funkcije koje sistem mora izvršavati. Oni definišu ponašanje sistema u određenim situacijama i odgovaraju na pitanje šta sistem treba da radi. Funkcionalni zahtjevi često uključuju opis ulaza, obrade i izlaza sistema.

Ova vrsta zahtjeva je od ključnog značaja jer direktno utiče na način na koji korisnici komuniciraju sa sistemom. Nejasno ili nepotpuno definisani funkcionalni zahtjevi mogu dovesti do pogrešne implementacije i nezadovoljstva korisnika (Robertson & Robertson, 2013).

Primjer funkcionalnog zahtjeva:

„Sistem mora omogućiti korisniku dodavanje, izmjenu i brisanje podataka o klijentima.“

Ovaj zahtjev precizira koje operacije sistem mora podržavati i šta korisnik može raditi unutar sistema.

4.3. NEFUNKCIONALNI ZAHTJEVI

Nefunkcionalni zahtjevi definišu kvalitativne karakteristike softverskog sistema i ograničenja njegovog rada. Oni ne opisuju šta sistem radi, već kako to radi. Među najčešćim nefunkcionalnim zahtjevima su zahtjevi za performanse, sigurnost, dostupnost i skalabilnost sistema.

Ovi zahtjevi su često teže mjerljivi od funkcionalnih, ali imaju značajan uticaj na ukupni kvalitet softverskog sistema. Neadekvatno definisani nefunkcionalni zahtjevi mogu dovesti do sistema koji tehnički funkcioniše, ali ne zadovoljava korisnička očekivanja.

Primjer nefunkcionalnog zahtjeva:

„Sistem mora odgovoriti na korisnički zahtjev u roku kraćem od dvije sekunde.“

Iako ne opisuje funkcionalnost, ovaj zahtjev ima direktan uticaj na kvalitet korisničkog iskustva.

4.4. KORISNIČKI I SISTEMSKI ZAHTJEVI

Korisnički zahtjevi predstavljaju opis potreba sistema iz perspektive krajnjeg korisnika. Oni su obično formulisani jednostavnim jezikom kako bi bili razumljivi

osobama koje nemaju tehničko znanje. Njihova svrha je da jasno izraze šta korisnik očekuje od sistema.

Sistemski zahtjevi su detaljniji i tehnički precizniji opis korisničkih zahtjeva. Oni služe kao osnova za dizajn i implementaciju softverskog sistema i namijenjeni su prvenstveno razvojnim timovima (Robertson & Robertson, 2013).

Primjer korisničkog zahtjeva:

„Korisnik želi da može pregledati svoje prethodne narudžbe."

Sistemski zahtjevi su detaljnija i tehnički preciznija verzija korisničkih zahtjeva.

Primjer sistemskog zahtjeva:

„Sistem mora omogućiti prikaz liste prethodnih narudžbi korisnika, sortirane po datumu kreiranja."

4.5. DOKUMENT SA SOFTVERSKIM ZAHTIJEVIMA (SRS)

Dokument sa softverskim zahtjevima predstavlja formalni zapis svih identifikovanih zahtjeva sistema. Ovaj dokument definiše funkcionalne i nefunkcionalne zahtjeve, kao i ograničenja sistema, i koristi se kao referenca tokom cijelog životnog ciklusa razvoja softvera.

Kvalitetno izrađen dokument sa softverskim zahtjevima treba da bude potpun, konzistentan i razumljiv svim učesnicima u projektu. Njegova uloga je da smanji rizik od nesporazuma i olakša validaciju razvijenog sistema (Robertson & Robertson, 2013).

Primjer sadržaja SRS dokumenta:

- Opis sistema
- Funkcionalni zahtjevi
- Nefunkcionalni zahtjevi

- Ograničenja sistema

Ovakva struktura omogućava jasnu komunikaciju između korisnika i razvojnog tima.

4.6. ZAINTERESOVANE STRANE (STAKEHOLDERS)

Zainteresovane strane su svi pojedinci ili organizacije koje imaju interes u razvoju i korištenju softverskog sistema. To mogu biti krajnji korisnici, menadžeri, investitori, ali i članovi razvojnog tima. Svaka od ovih grupa može imati različite zahtjeve i očekivanja od sistema.

Identifikacija zainteresovanih strana predstavlja ključni korak u procesu prikupljanja zahtjeva, jer omogućava sagledavanje sistema iz različitih perspektiva i smanjuje rizik od izostavljanja važnih zahtjeva.

Primjer zainteresovanih strana:

- Krajnji korisnici softvera
- Projektni menadžer
- Razvojni tim
- Investitor projekta

Svaka od navedenih strana može imati različite zahtjeve i prioritete.

4.7. KVALITET SOFTVERSKIH ZAHTJEVA

Kvalitet softverskih zahtjeva ogleda se u njihovoj jasnoći, nedvosmislenosti i provjerljivosti. Dobro definisani zahtjevi trebaju biti razumljivi svim učesnicima projekta i omogućiti provjeru da li je sistem ispravno implementiran.

Prema savremenim istraživanjima, veliki broj neuspješnih softverskih projekata može se povezati sa loše definisanim ili nepotpunim zahtjevima, što dodatno naglašava važnost njihove pravilne formulacije (Robertson & Robertson, 2013).

Primjer lošeg zahtjeva:

„Sistem treba da radi brzo.“

Primjer kvalitetnog zahtjeva:

„Sistem mora obraditi korisnički zahtjev u roku kraćem od dvije sekunde.“

Drugi primjer je jasan, mjerljiv i provjerljiv, što ga čini kvalitetnim zahtjevom.

5. PRIMJENA PISANJA SOFTVERSKIH ZAHTJEVA U PRAKSI

5.1. ULOGA SOFTVERSKIH ZAHTJEVA U RAZVOJU SOFTVERSKIH SISTEMA

Pisanje softverskih zahtjeva ima ključnu ulogu u praktičnom razvoju softverskih sistema jer predstavlja osnovu za sve naredne faze razvoja, uključujući dizajn, implementaciju, testiranje i održavanje sistema. Jasno definisani zahtjevi omogućavaju razvojnim timovima da razumiju potrebe korisnika i da ih pravilno implementiraju u softversko rješenje.

U praksi, softverski zahtjevi služe kao referentna tačka za donošenje odluka tokom razvoja projekta. Svaka promjena u sistemu se procjenjuje u odnosu na postojeće zahtjeve, čime se smanjuje rizik od nekontrolisanih izmjena i povećanja troškova razvoja (Bourque & Fairley, 2014).

5.2. PRIMJENA SOFTVERSKIH ZAHTJEVA U FAZAMA ŽIVOTNOG CIKLUSA SOFTVERA

Softverski zahtjevi se koriste u svim fazama životnog ciklusa softvera. Tokom faze analize, zahtjevi se prikupljaju i dokumentuju na osnovu potreba korisnika. U fazi dizajna, zahtjevi se prevode u arhitekturu sistema, dok se u fazi implementacije koriste kao smjernice za razvoj funkcionalnosti (Bourque & Fairley, 2014).

U fazi testiranja, softverski zahtjevi predstavljaju osnovu za definisanje test slučajeva. Svaki test ima za cilj provjeru da li je određeni zahtjev ispravno implementiran. Ovakav pristup omogućava objektivnu evaluaciju kvaliteta sistema (ISTQB, 2018).

Primjer:

Ako zahtjev glasi:

„Sistem mora omogućiti korisniku registraciju putem e-mail adrese“, testiranje će se fokusirati na provjeru funkcionalnosti registracije i validaciju unesenih podataka.

5.3. PRIMJENA ZAHTJEVA U TRADICIONALNIM I AGILNIM METODOLOGIJAMA

U tradicionalnim metodologijama razvoja softvera, kao što je vodopadni (waterfall) model, softverski zahtjevi se definišu detaljno na početku projekta i rijetko se mijenjaju tokom razvoja. Ovakav pristup omogućava precizno planiranje, ali je manje fleksibilan u slučaju promjena korisničkih potreba.

Nasuprot tome, agilne metodologije podrazumijevaju iterativno prikupljanje i unapređivanje zahtjeva. U agilnom okruženju zahtjevi se često zapisuju u obliku korisničkih priča, koje se postepeno razrađuju kroz saradnju sa korisnicima.

Primjer korisničke priče:

„Kao registrovani korisnik, želim da mogu promijeniti lozinku kako bih povećao sigurnost svog naloga.“

5.4. PRIMJENA SOFTVERSKIH ZAHTJEVA U RAZLIČITIM OBLASTIMA

Softverski zahtjevi se primjenjuju u različitim oblastima, uključujući obrazovanje, zdravstvo, bankarstvo i e-upravu. Svaka od ovih oblasti ima specifične zahtjeve i ograničenja koja moraju biti jasno definisana kako bi softverski sistem bio funkcionalan i pouzdan.

Primjer u obrazovnom sistemu:

„Sistem mora omogućiti nastavnicima unos i pregled ocjena za svakog učenika.“

Primjer u bankarskom sistemu:

„Sistem mora osigurati da su sve finansijske transakcije šifrovane.“

Ovi primjeri pokazuju kako se softverski zahtjevi prilagođavaju specifičnim potrebama različitih domena.

5.5. PROBLEMI U PRAKSI I ZNAČAJ PRAVILNO DEFINISANIH ZAHTJEVA

U praksi se često javljaju problemi zbog nejasno ili nepotpuno definisanih softverskih zahtjeva. Prema istraživanjima, veliki broj softverskih projekata ne uspijeva upravo zbog pogrešnog razumijevanja korisničkih potreba i loše komunikacije između učesnika projekta.

Pravilno definisani softverski zahtjevi doprinose smanjenju rizika, boljem planiranju resursa i većem zadovoljstvu korisnika. Zbog toga se pisanju zahtjeva u savremenim softverskim projektima posvećuje posebna pažnja.

5.6. PREDNOSTI PRIMJENE DOKUMENTOVANIH SOFTVERSKIH ZAHTJEVA

Primjena dokumentovanih softverskih zahtjeva omogućava bolju kontrolu nad razvojem softvera i olakšava komunikaciju između svih zainteresovanih strana. Dokumentovani zahtjevi služe kao pravni i tehnički dokaz dogovorenih funkcionalnosti sistema (Bourque & Fairley, 2014).

Pored toga, jasno definisani zahtjevi olakšavaju održavanje i nadogradnju sistema, jer omogućavaju lakše razumijevanje postojećih funkcionalnosti i ograničenja.

6. UPOTREBA RAČUNARA, METODA I ALATA U PISANJU SOFTVERSKIH ZAHTJEVA

6.1. ULOGA RAČUNARA U PROCESU PISANJA SOFTVERSKIH ZAHTJEVA

Računari imaju izuzetno važnu ulogu u savremenom procesu pisanja softverskih zahtjeva, jer omogućavaju efikasno prikupljanje, obradu i čuvanje velike količine podataka. Umjesto ručnog zapisivanja i papirne dokumentacije, danas se koriste digitalni alati koji omogućavaju lakšu organizaciju zahtjeva i njihovu brzu izmjenu tokom razvoja softverskog sistema.

Upotreba računara omogućava timovima da rade paralelno na istom dokumentu, što značajno unapređuje saradnju između analitičara, programera i korisnika. Također, digitalna dokumentacija omogućava praćenje promjena, verzionisanje dokumenata i povratak na prethodne verzije ukoliko dođe do grešaka ili nesporazuma.

Pored toga, računari omogućavaju jednostavnu integraciju softverskih zahtjeva sa drugim fazama razvoja, kao što su dizajn i testiranje, čime se osigurava kontinuitet i konzistentnost tokom cijelog životnog ciklusa softvera (Kotonya & Sommerville, 1998).

6.2. METODE PRIKUPLJANJA SOFTVERSKIH ZAHTJEVA

Prikupljanje softverskih zahtjeva predstavlja jedan od najvažnijih koraka u razvoju softverskog sistema. U praksi se koristi više metoda kako bi se obuhvatile različite perspektive korisnika i zainteresovanih strana. Izbor metode zavisi od vrste projekta, složenosti sistema i dostupnosti korisnika.

Jedna od najčešće korištenih metoda je intervju, tokom kojeg analitičar direktno razgovara sa korisnicima. Ova metoda omogućava detaljno razumijevanje potreba, ali

zahtijeva dobru pripremu pitanja i jasno definisan cilj razgovora. Intervjui mogu biti strukturisani ili nestrukturisani, u zavisnosti od nivoa slobode koji se ostavlja korisniku.

Pored intervjua, često se koriste ankete koje omogućavaju prikupljanje podataka od većeg broja korisnika u kraćem vremenskom periodu. Radionice i grupne diskusije omogućavaju razmjenu ideja između korisnika i analitičara, dok posmatranje korisnika u stvarnom radnom okruženju pomaže u identifikaciji implicitnih zahtjeva koji se ne mogu lako verbalizovati (Young, 2004).

6.3. MODELOVANJE SOFTVERSKIH ZAHTJEVA

Modelovanje softverskih zahtjeva predstavlja tehniku kojom se zahtjevi prikazuju grafički, pomoću različitih dijagrama i modela. Ovakav pristup omogućava lakše razumijevanje sistema, naročito kada je riječ o složenim funkcionalnostima i velikom broju interakcija.

Jedan od najčešće korištenih alata za modelovanje je UML, koji obuhvata različite vrste dijagrama, kao što su dijagrami slučajeva korištenja, dijagrami aktivnosti i dijagrami sekvenci. Dijagrami slučajeva korištenja posebno su korisni jer prikazuju način na koji korisnici komuniciraju sa sistemom i koje funkcionalnosti sistem mora podržavati.

Modelovanje smanjuje rizik od pogrešnog tumačenja zahtjeva i omogućava lakšu validaciju sa korisnicima, jer vizuelni prikaz često olakšava razumijevanje u odnosu na tekstualni opis (Pressman, 2010).

6.4. UPOTREBA STATISTIKE U ANALIZI SOFTVERSKIH ZAHTJEVA

Statističke metode imaju važnu ulogu u analizi softverskih zahtjeva, naročito kada je u pitanju rad sa velikim brojem korisnika ili zahtjeva. Statistika se koristi za obradu

podataka prikupljenih putem anketa, upitnika i drugih metoda, kako bi se identifikovali najčešći i najvažniji zahtjevi.

Na osnovu statističke analize moguće je izvršiti prioritizaciju zahtjeva, odnosno odrediti koji zahtjevi imaju najveći značaj za korisnike ili najveći uticaj na kvalitet sistema. Ovakav pristup omogućava racionalnije planiranje resursa i efikasnije donošenje odluka tokom razvoja.

Statistika se također koristi za procjenu rizika i uspješnosti implementacije zahtjeva, čime se povećava pouzdanost cjelokupnog procesa razvoja (Wiegers & Beatty, 2013).

6.5. SOFTVERSKI ALATI ZA UPRAVLJANJE ZAHTJEVIMA

U savremenom softverskom inženjerstvu koriste se specijalizovani alati za upravljanje softverskim zahtjevima. Ovi alati omogućavaju evidenciju zahtjeva, njihovo povezivanje sa test slučajevima i praćenje statusa tokom razvoja sistema.

Primjena softverskih alata doprinosi boljoj organizaciji rada, smanjenju grešaka i povećanju transparentnosti projekta. Takođe, alati omogućavaju lakšu saradnju između članova tima i zainteresovanih strana, čime se unapređuje ukupni kvalitet softverskog sistema.

7. VLASTITI PRIMJER I EKSPERIMENT U OBLASTI PISANJA SOFTVERSKIH ZAHTJEVA

7.1. CILJ I SVRHA EKSPERIMENTA

Cilj ovog eksperimenta je ispitati uticaj načina pisanja softverskih zahtjeva na razumijevanje sistema, kvalitet implementacije i proces testiranja softverskog rješenja. Eksperiment ima za svrhu da pokaže razliku između neformalno definisanih zahtjeva i formalno, jasno dokumentovanih softverskih zahtjeva.

Eksperiment je osmišljen kao simulacija realnog softverskog projekta, čime se omogućava praktična primjena teorijskih znanja iz oblasti prikupljanja i pisanja softverskih zahtjeva.

7.2. OPIS SISTEMA KOJI JE PREDMET EKSPERIMENTA

Kao predmet eksperimenta odabrana je jednostavna web-aplikacija za prijavu ispita na visokoškolskoj ustanovi. Aplikacija je namijenjena studentima i nastavnom osoblju, a osnovna svrha sistema je digitalizacija procesa prijave ispita.

Sistem ima dvije osnovne grupe korisnika:

- studenti, koji prijavljuju ispite
- profesori, koji imaju uvid u prijavljene studente

Ovaj primjer je odabran jer predstavlja čest i lako razumljiv informacioni sistem, pogodan za analizu softverskih zahtjeva.

7.3. EKSPERIMENTALNI PRISTUP I METODOLOGIJA

Eksperiment je sproveden u dvije faze:

Faza 1 – Neformalno definisani zahtjevi

U prvoj fazi softverski zahtjevi su zapisani neformalno, bez preciznih kriterija i tehničkih detalja.

Faza 2 – Formalno definisani zahtjevi

U drugoj fazi isti sistem je opisan pomoću jasno strukturisanih, mjerljivih i provjerljivih softverskih zahtjeva.

U obje faze analizirani su:

- jasnoća zahtjeva
- mogućnost njihove implementacije
- mogućnost testiranja sistema

Ovakav metod omogućava direktno poređenje rezultata.

7.4. NEFORMALNO DEFINISANI ZAHTJEVI - ANALIZA PROBLEMA

U prvoj fazi eksperimenta identifikovani su sljedeći neformalni zahtjevi:

- „Student treba moći prijaviti ispit.“
- „Sistem treba biti jednostavan za korištenje.“
- „Profesor treba vidjeti prijave.“

Analizom ovih zahtjeva uočeni su brojni problemi:

- nije definisan način prijave ispita
- ne postoje kriteriji za jednostavnost sistema

- nije jasno koje podatke profesor vidi

Zbog nejasno definisanih zahtjeva, implementacija sistema bi zavisila od subjektivnog tumačenja programera, što povećava rizik od grešaka i nezadovoljstva korisnika.

7.5. FORMALNO DEFINISANI SOFTVERSKI ZAHTJEVI

U drugoj fazi eksperimenta zahtjevi su precizno definisani:

- „Sistem mora omogućiti studentu prijavu ispita putem korisničkog naloga, uz automatsku potvrdu uspješne prijave.“
- „Sistem mora omogućiti prijavu ispita najkasnije 48 sati prije termina ispita.“
- „Sistem mora prikazati profesoru listu prijavljenih studenata, uključujući ime, prezime i datum prijave.“
- „Sistem mora obraditi zahtjev za prijavu ispita u roku kraćem od dvije sekunde.“

Ovi zahtjevi su jasni, mjerljivi i provjerljivi, što omogućava njihovu direktnu implementaciju i testiranje.

7.6. POREĐENJE REZULTATA EKPERIMENTA

Poređenjem obje faze dobijeni su sljedeći rezultati:

- formalni zahtjevi su značajno smanjili nejasnoće u razumijevanju sistema
- implementacija sistema je bila lakša i preciznija
- testiranje sistema je moglo biti direktno povezano sa svakim zahtjevom

U drugoj fazi bilo je moguće jasno definisati test slučajeve za svaki zahtjev, što u prvoj fazi nije bilo moguće zbog neodređenosti zahtjeva.

7.7. ANALIZA I DISKUSIJA REZULTATA

Rezultati eksperimenta potvrđuju da način pisanja softverskih zahtjeva ima direktan uticaj na kvalitet softverskog sistema. Neformalni zahtjevi dovode do različitih interpretacija i povećavaju rizik od grešaka u razvoju.

Formalno definisani zahtjevi omogućavaju bolju komunikaciju između korisnika i razvojnog tima, kao i efikasnije planiranje i kontrolu razvoja softvera. Ovi rezultati su u skladu sa teorijskim nalazima iz oblasti softverskog inženjerstva.

7.8. OGRANIČENJA EKSPERIMENTA

Eksperiment je sproveden na hipotetičkom primjeru i nije uključivao stvarni razvoj softverskog sistema. Ipak, zbog realističnog scenarija i jasno definisanih faza, rezultati se mogu smatrati relevantnim za razumijevanje značaja pravilnog pisanja softverskih zahtjeva.

8. ZAKLJUČAK

U ovom seminarskom radu obrađena je tema pisanja i prikupljanja softverskih zahtjeva kao jednog od najvažnijih procesa u razvoju softverskih sistema. Kroz teorijski i praktični dio rada ukazano je na značaj pravilnog definisanja softverskih zahtjeva i njihov uticaj na uspješnost realizacije softverskih projekata. Posebna pažnja posvećena je razumijevanju osnovnih pojmova, istorijskom razvoju discipline, kao i savremenim pristupima u pisanju softverskih zahtjeva.

U teorijskom dijelu rada istaknuto je da su softverski zahtjevi temelj svake faze životnog ciklusa softvera. Kvalitetno definisani zahtjevi omogućavaju jasnu komunikaciju između korisnika i razvojnog tima, smanjuju rizik od pogrešnog tumačenja i doprinose efikasnijem planiranju i realizaciji projekta. Analizom osnovnih pojmova i tipova zahtjeva pokazano je da razlikovanje funkcionalnih i nefunkcionalnih zahtjeva, kao i korisničkih i sistemskih zahtjeva, ima ključnu ulogu u razumijevanju potreba sistema.

Poseban doprinos radu daje praktični dio, u kojem je kroz vlastiti primjer i eksperiment prikazan uticaj načina pisanja softverskih zahtjeva na razvoj softverskog sistema. Poređenjem neformalno definisanih zahtjeva sa formalno dokumentovanim zahtjevima jasno je uočeno da nejasni i nedovoljno precizni zahtjevi dovode do nesporazuma, otežane implementacije i problema u testiranju sistema. Nasuprot tome, formalno definisani, mjerljivi i provjerljivi zahtjevi omogućavaju efikasniji razvoj, lakšu validaciju i veći kvalitet krajnjeg softverskog rješenja.

Također, u radu je naglašena važnost primjene savremenih metoda, alata i računara u procesu pisanja softverskih zahtjeva. Upotreba specijalizovanih softverskih alata, modelovanja i statističkih metoda omogućava bolju organizaciju zahtjeva, njihovu analizu i praćenje tokom razvoja softvera. Na taj način se povećava transparentnost procesa i olakšava saradnja između svih zainteresovanih strana.

Na osnovu svega navedenog može se zaključiti da pisanje softverskih zahtjeva nije samo tehnički zadatak, već složen proces koji zahtijeva pažljivo planiranje, analizu i komunikaciju. Pravilno definisani softverski zahtjevi predstavljaju ključni faktor

uspješnosti softverskih projekata i imaju direktan uticaj na kvalitet, pouzdanost i održivost softverskih sistema. Zbog toga se ovoj fazi razvoja softvera mora posvetiti posebna pažnja, kako u akademskom okruženju, tako i u praksi savremenog softverskog inženjerstva.

9. LITERATURA

Bourque, P., & Fairley, R. (2014). *Guide to the Software Engineering Body of Knowledge Version 3.0.* IEEE.

IEEE Std 830. (1998). *IEEE Recommended Practice for Software Requirements Specifications.* USA: IEEE.

ISO/IEC/IEEE 29148. (2011). *Systems and software engineering - Life cycle processes - Requirements engineering* . Switzerland : ISO/IEC/IEEE .

ISTQB. (2018). *Certified Tester Foundation Level (CTFL) Syllabus.* ISTQB.

Kotonya, G., & Sommerville, I. (1998). *Requirements engineering : processes and techniques.* New York: John Wiley and Sons.

Pressman, R. (2010). *Software Engineering - A Practitioner's approach 7th ed.* The McGraw-Hill Companies.

Robertson, S., & Robertson, J. (2013). *Mastering The Requirements Process.* Pearson Education.

Sommerville, I. (2011). *Software engineering - 9th ed.* Pearson Education.

Wiegers, K., & Beatty, J. (2013). *Software Requirements 3rd ed.* Washington: Microsoft.

Young, R. (2004). *The Requirements Engineering Handbook.* Norwood: Artech house.