Razvoj alata za potporu aktivnosti specificiranja softverskih zahtjeva

Belina, Zvonimir

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Zvonimir Belina

RAZVOJ ALATA ZA POTPORU AKTIVNOSTI SPECIFICIRANJA SOFTVERSKIH ZAHTJEVA

ZAVRŠNI RAD

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Zvonimir Belina

Matični broj: 0016149673

Studij: Informacijski sustavi

RAZVOJ ALATA ZA POTPORU AKTIVNOSTI SPECIFICIRANJA SOFTVERSKIH ZAHTJEVA

ZAVRŠNI RAD

Mentor/Mentorica:

Dr. sc. Marko Mijač

Varaždin, rujan 2023

Zvonimir Belina

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Ovaj rad istražuje pojam softverskih zahtjeva u kontekstu aktivnosti specificiranja softverskih zahtjeva te razvoj prikladnog alata. Rad započinje uvodom koji daje pregled tema i ciljeva rada iza čega slijedi drugo poglavlje, gdje su pojašnjene metode i tehnike rada pomoću kojih su se provela istraživanja i razradila tema. Treće poglavlje posvećeno je inženjerstvu zahtjeva s poglavljima koji opisuju softverske zahtjeve, proces definiranja zahtjeva, smjernice za pisanje zahtjeva te generički proces za inženjerstvo zahtjeva.

Iza teorijskog okvira, slijedi četvrto poglavlje koje analizira pet postojećih alata za specifikaciju softverskih zahtjeva, a to su QVScribe, ReqSuite® RM, RQA – QUALITY Studio®, ScopeMaster® i ReqView. Peto poglavlje predstavlja praktični dio rada, u kojem se primjenjuju koncepti inženjerstva zahtjeva na stvarnom projektu. Opisuje se opseg softverskog rješenja, perspektiva proizvoda, karakteristike korisnika, ograničenja te se specificiraju funkcionalni i nefunkcionalni zahtjevi izrađenog softverskog rješenja. Također, pojašnjava se dizajn softverskog sustava preko različitih UML dijagrama te isječaka samog programskog koda uz demonstraciju rješenja pomoću nekoliko snimaka zaslona.

Naposljetku, sedmo poglavlje donosi sažete zaključke razrade teme rada. Ovaj rad doprinosi dubljem razumijevanju softverskih zahtjeva te izrazito bitnom procesu pisanja, odnosno specifikacije softverskih zahtjeva.

Ključne riječi: softverski zahtjev; inženjerstvo zahtjeva: specifikacija zahtjeva; softver; sustav; smjernice; alat za specifikaciju;

Sadržaj

1. Uvod

U današnje vrijeme, softver igra vrlo veliku ulogu u gotovo svim aspektima života i poslovanja. Iz tog razloga, kako bi se isporučili funkcionalni, pouzdani i učinkoviti softverski proizvodi, potrebno je razumjeti i na pravilan način definirati zahtjeve koji su temelj razvoja tih proizvoda. Inženjerstvo zahtjeva predstavlja ključan korak u procesu razvoja softvera te je od presudnog značaja za uspješnost projekata.

Iz osobnog iskustva u području razvoja softvera u obliku timskih studentskih projekata na više kolegija fakulteta, prepoznao sam potrebu za olakšanjem i unapređenjem procesa definiranja zahtjeva studentima pomoću novog alata. Ova potreba proizašla je iz izazova s kojima se često susreću inženjeri softvera, uključujući teškoće u elicitiranju, dokumentiranju i upravljanju zahtjevima.

Ovaj rad istražuje problematiku inženjerstva zahtjeva i razvoja alata koji podržavaju aktivnosti specifikacije softverskih zahtjeva. Proučava se teorijska podloga inženjerstva zahtjeva, od procesa definiranja zahtjeva, uključujući razliku između funkcionalnih i nefunkcionalnih zahtjeva, preko smjernica za pisanje i upravljanje zahtjevima do generičkog procesa za inženjerstvo zahtjeva.

S ciljem boljeg razumijevanja funkcionalnosti praktičnog dijela analizira se pet postojećih alata, koji se između ostalog koriste za specificiranje softverskih zahtjeva. Nakon analize slijedi praktični dio koji se odnosi na razvoj novog alata za podršku specifikaciji softverskih zahtjeva.

U konačnici, cilj ovog rada je doprinijeti razumijevanju i unapređenju procesa definiranja softverskih zahtjeva te razvoju alata koji studentima olakšava ovu kompleksnu i ključnu fazu razvoja softverskih proizvoda.

2. Metode i tehnike rada

Pri razradi teme „Razvoj alata za potporu aktivnosti specifikacije softverskih zahtjeva“ korištene su sljedeće metode i tehnike:

1. ***Analiza literature*** provedena je pregledavanjem relevantne literature, u ovom slučaju knjiga, internetskih članaka te vodiča. Ovime se stekao uvid u teorijsku podlogu inženjerstva zahtjeva.
2. ***Ispitivanje postojećih alata*** provelo se u ranom periodu pisanja rada. Ovo ispitivanje je uključilo traženje relevantnih alata, pregled njihovih mogućnosti i funkcionalnosti, traženje zajedničkih kategorija usporedbe te naposljetku usporedba alata. Rezultat ovog ispitivanja pridonio je stvaranju ideje za implementaciju praktičnog dijela.

Za izradu praktičnog dijela korišteni su sljedeći programski alati i aplikacije:

1. ***Visual Paradgim Community Edition*** je besplatan UML alat za sve vrste nekomercijalnog modeliranja. Pomoću ovog alata izrađeni su razni UML dijagrami za vizualizaciju projekta prije realizacije, ali i kao dokumentacije poslije.
2. **Visual Studio** je integrirano razvojno okruženje (eng. *Integrated Development Environment - IDE*) tvrtke Microsoft. U ovom okruženju je izrađen praktični dio u obliku C# .NET Windows obrasci (eng. *forms*) aplikacije.
3. **Github** platforma i servis u oblaku (eng. *Cloud service*) za softverski razvoj te verzioniranje koristeći Git, pomoću koje se sav kod verzionirao.

3. Inženjerstvo zahtjeva

„Inženjerstvo zahtjeva: predmetni podskup inženjerstva sustava s otkrivanjem, razvijanjem, praćenjem, analiziranjem, kvalificiranjem, komuniciranjem i upravljanjem zahtjevima koji definiraju sustav na uzastopnim razinama apstrakcija.“ (Dick, Hull, Jackson, 2017, str. 25). Iz ove definicije možemo vidjeti da se inženjerstvo zahtjeva sastoji od nekoliko ključnih aktivnosti koje se smatraju ispravnima za inženjerstvo zahtjeva. Otkrivanje je povezano s elicitacijom zahtjeva, praćenje uključuje povezivanje zahtjeva s drugim artefaktima te validaciju zahtjeva s obzirom na potrebe iz stvarnog svijeta. Kvalificiranje je osiguravanje da rješenje ima odgovarajuću kvalitetu preko raznih testiranja zahtjeva. Komuniciranje označava da su zahtjevi zapravo način komunikacije između raznih osoba (korisnici, developeri, dobavljači i sl.) koje sudjeluju u ostvarivanju zahtjeva. Naposljetku, pod razine apstrakcije podrazumijevaju različite slojeve zahtjeva i odnosa među njima gdje su na najvišem sloju problemi koji se trebaju riješiti i potrebe koje se trebaju zadovoljiti prema dogovoru s dionicima. Svi naredni slojevi definiraju sistem ili dio sistema u smislu praktičnih rješenja prema zahtjevima sloja iznad. Inženjerstvo zahtjeva disciplina je koja se proteže kroz cijeli životni ciklus razvoja te se u kasnijim fazama isplati uložiti trud u ranijim fazama jer poboljšanjem zahtjeva poboljšavamo i kvalitetu proizvoda.

3.1. Softverski zahtjevi

Zahtjevi (eng. *requirements*) su ključni elementi svakog softvera. Oni opisuju funkcionalnosti koje softver treba imati kako bi zadovoljio korisničke potrebe. Dobro definirani zahtjevi su važni jer omogućuju razumijevanje problemske domene, identifikaciju korisničkih potreba i očekivanja te definiranje jasne vizije za razvoj softvera. Istraživanja su pokazala da su precizno definirani zahtjevi jedan od najvažnijih faktora uspjeha softverskih projekata. U početnoj fazi projekta zahtjevi su potrebni za provođenje procjene izvedivosti, planiranje projekta, upravljanje rizicima, dodjelu prioriteta zadataka te definiranje kriterija za prihvaćanje softvera. S druge strane, nepotpuni i nerealni zahtjevi česti su razlozi neuspjeha softverskih projekata. Ako softver ne zadovoljava korisničke potrebe, neće biti uspješan ni koristan. To se ne može ispraviti ni najnovijim alatima i tehnologijama, niti primjenom najboljih praksi u dizajnu, implementaciji ili testiranju. Unatoč ključnoj ulozi zahtjeva, njihovo upravljanje često ne dobiva dovoljno pažnje.

U tradicionalnim softverskim procesima definiranje zahtjeva prva je faza softverskog procesa. Međutim, zahtjevi nisu izolirani od ostalih faza i aktivnosti, već usmjeravaju

dizajnerske, implementacijske i testne odluke. Često se javljaju povratne informacije koje mogu rezultirati identifikacijom novih zahtjeva ili izmjenom postojećih na temelju saznanja dobivenih tijekom dizajna, implementacije i testiranja. Za neki sustav softverski zahtjevi predstavljaju opis onoga što bi sustav trebao raditi, uslugu ili usluge koje pruža te ograničenja njegovog rada.

Valja napomenuti da postoji razlika između korisničkih zahtjeva i softverskih zahtjeva. „Softverski zahtjev je prema IEEE standardu 729 jedno od sljedećeg:

- Stanje ili sposobnost potrebna korisniku za rješavanje problema ili postizanje cilja.
- Uvjet ili sposobnost koju mora ispuniti ili posjedovati sustav ili komponenta sustava kako bi zadovoljili ugovor, standard, specifikaciju ili druge formalno nametnute dokumente.
- Dokumentirani prikaz stanja ili sposobnosti kao u prethodne dvije natuknice“ (02DCE, 2023).

Wiegers i Beatty (2013) navode da je korisnički zahtjev mogućnost korisnika da izvrši neki cilj ili zadatak sa sustavom ili atributom proizvoda. Drugim riječima, korisnički zahtjevi opisuju što korisnik treba i želi od sustava, dok softverski zahtjevi opisuju tehničke specifikacije i ograničenja koja sustav mora zadovoljiti kako bi ispunio te korisničke zahtjeve.

Neki tipovi informacija zahtjeva koje navode Wiegers i Beatty (2013) su:

- Poslovni zahtjev (eng. *Business requirement*),
- Poslovno pravilo (eng. *Business rule*),
- Ograničenje (eng. *Constraint*),
- Zahtjev za vanjskim sučeljem (eng. *External interface requirement*),
- Značajka,
- Funkcionalni zahtjevi,
- Nefunkcionalni zahtjevi,
- Atribut kvalitete (eng. *Quality attribute*),
- Zahtjev sustava,
- Zahtjev korisnika.

Poslovni zahtjev cilj je visoke razine organizacije koja izrađuje proizvod ili kupca koji ga nabavlja. Poslovno pravilo je politika, smjernica, standard ili propis koji definira ili ograničava neki aspekt poslovanja te je izvor nekoliko vrsta softverskih zahtjeva. Ograničenje se odnosi na izbore dostupne programeru za stvaranje proizvoda. Zahtjev za vanjskim sučeljem opisuje vezu između samog sustava i korisnika, drugog sustava ili hardverskog uređaja. Značajka predstavlja mogućnost ili logički povezane mogućnosti sustava koje daju vrijednost korisniku.

Funkcionalni i nefunkcionalni zahtjevi će biti detaljnije objašnjeni u sljedećim poglavljima zato što su ti zahtjevi oni kojima se bavim u praktičnom dijelu te predstavljaju određenu osnovu za softvere kojima se bavimo na fakultetskim kolegijima. Atribut kvalitete je vrsta nefunkcionalnog zahtjeva koji opisuje uslugu ili karakteristiku performanse proizvoda. Zahtjev sustava je zahtjev najviše razine za proizvod s više podsustava, bio to samo softver ili softver i hardver. Zahtjev korisnika je cilj ili zadatak koji određene klase korisnika moraju moći izvršiti sa sustavom ili željenim atributom proizvoda.

3.1.1.Funkcionalni zahtjevi

Funkcionalni zahtjevi predstavljaju specifične funkcionalnosti koje krajnji korisnik zahtijeva od sustava kao mogućnosti, odnosno što softver treba raditi. Oni određuju osnovne operacije, manipulaciju podacima, poslovne procese, interakciju s korisnikom ili druge specifične funkcionalnosti koje definiraju glavne zadatke koje sustav treba obavljati. Svi ovi zahtjevi moraju biti nužno implementirani u sustavu kao dio ugovora. Funkcionalni zahtjevi najčešće su prikazani kao ulazni podaci koje korisnik daje sustavu, operacije koje se izvršavaju i očekivani izlaz. Oni su vidljivi izravno u konačnom proizvodu i razlikuju se od nefunkcionalnih zahtjeva. Na primjer, u sustavu za upravljanje bolnicom, funkcionalni bi zahtjev bio da liječnici mogu pristupiti podacima svojih pacijenata. Svaki funkcionalni zahtjev može uključivati više interakcija ili dijaloga između sustava i vanjskog svijeta te da bi se funkcionalni zahtjevi precizno opisali, potrebno je navesti sve moguće scenarije. Postoji više načina za izražavanje funkcionalnih zahtjeva, na primjer prirodni jezik, strukturirani ili formatirani jezik bez stroge sintakse i jezik formalne specifikacije s pravilnom sintaksom. U softverskom inženjerstvu funkcionalni se zahtjevi također nazivaju funkcionalnim specifikacijama, mogućnostima (eng. *capabilities*) ili značajkama (eng. *features*).

3.1.2.Nefunkcionalni zahtjevi

Nefunkcionalni zahtjevi predstavljaju ograničenja kvalitete koje sustav mora zadovoljiti prema projektnom ugovoru. Nisu usko povezani s funkcionalnostima sustava, već definiraju kako, a ne što sustav treba raditi. Još se nazivaju i nebihevioralnim zahtjevima jer se ne odnose na samu funkcionalnost sustava, već na njegove karakteristike izvan same operacije. Nefunkcionalni zahtjevi obuhvaćaju različite aspekte sustava kao što su prenosivost, sigurnost, mogućnost održavanja, pouzdanost, skalabilnost, performanse, ponovno korištenje, fleksibilnost i sl. Prioritet i opseg implementacije ovih čimbenika može varirati između različitih projekata. „Oni utječu na cjelokupnu arhitekturu sustava, a ne samo na individualne komponente. Zbog tog izrazitog utjecaja na kvalitetu softverskog rješenja, često se nazivaju i zahtjevima kvalitete, te mogu biti kritičniji od funkcionalnog zahtjeva“ (Mijač, 2023). Iako je

važno precizno izraziti i funkcionalne zahtjeve, još je važnije to učiniti za nefunkcionalne zahtjeve jer je teže objektivno utvrditi jesu li takvi zahtjevi ispunjeni ili ne. Zbog toga nefunkcionalni zahtjevi često sadrže i kvantitativne komponente koje služe kao pokazatelji ispunjenosti zahtjeva, poput broja transakcija u sekundi, prosječnog vremena do greške, vremena odziva, količine memorije, broja istovremenih korisnika i slično. Na taj se način smanjuje prostor za subjektivnu interpretaciju uspjeha ili neuspjeha u ispunjavanju zadanih zahtjeva. Ako propustimo identificirati i implementirati funkcionalni zahtjev postoji mogućnost da ga naknadno implementiramo uz određeni trud. S obzirom na to što takvi zahtjevi obuhvaćaju, naknadna implementacija nefunkcionalnog zahtjeva može biti izuzetno zahtjevna, čak i nemoguća. To može dovesti do izazova u vezi s vremenom i raspoloživim resursima.

3.2. Proces definiranja zahtjeva

3.2.1. Elicitacija zahtjeva

Elicitacija zahtjeva prvi je korak u procesu definiranja zahtjeva te je to aktivnost koja uključuje otkrivanje i prikupljanje zahtjeva od dionika (eng. *stakeholders*) softvera. Sa samim dionicima iznimno je važno uspostaviti i održavati dobru komunikaciju jer su njihove potrebe glavni izvor zahtjeva i ograničenja softvera. Cilj je ove aktivnosti razumjeti sve relevantne zahtjeve i potrebe dionika. Postoje različite tehnike kako bi se prikupile relevantne informacije kao što su intervjuiranje, radionice, analiza dokumentacije, upitnika i sl.

3.2.2. Analiza zahtjeva

Nakon elicitacije prikupljeni se zahtjevi detaljno analiziraju kako bi se utvrdila njihova valjanost, dosljednost i izvedivost. Ovdje se identificiraju mogući konflikti među zahtjevima te se razmatraju alternativna rješenja kao i ograničenja koja bi usporavala razvoj softvera. Također se vrši prioritizacija zahtjeva kako bi se odredili najvažniji i kritični zahtjevi. Cilj je ove aktivnosti dobiti bolje razumijevanje i uvid u svaki pojedini zahtjev.

3.2.3. Specifikacija zahtjeva

Kako bi se moglo efektivnije upravljati promjenama i komunicirati, potrebno je prikupljene zahtjeve pohraniti na lako razumljiv, jasan, postojan i dobro organiziran način što je i sam cilj ove aktivnost. Za pohranu se najčešće koristi prirodan jezik, no kako bismo osigurali ispravnu strukturu i potpunost informacija, također su u upotrebi posebni formalni jezici kao što je Jedinstveni jezik za modeliranje (eng. *Unified Modeling Language - UML*). Također se često funkcionalni i nefunkcionalni zahtjevi pohranjuju u specifikaciju softverskih

zahtjeva (eng. *software requirements specification - SRS*) koji, kako objašnjavaju Wiegers i Beatty (2013), opisuje funkcionalnosti i mogućnosti koje softverski sustav treba pružiti, kao i njegove karakteristike i ograničenja koja mora zadovoljiti. Jedna od preporučenih praksi kako pristupiti specifikaciji softverskih zahtjeva je standard IEEE 830-1998.

3.2.4.Provjera zahtjeva

Ovaj korak uključuje provjeru jesu li zahtjevi potpuni, dosljedni i točni, jesu li provjerljivi i ispunjavaju li potrebe i očekivanja dionika te uspješnost ostvarenja specificiranih zahtjeva u kasnijim fazama projekta. Provjera zahtjeva sastoji se od verifikacije i validacije. Cilj je validacije zahtjeva utvrditi jesu li specificirani zahtjevi u skladu s potrebama i očekivanjima dionika te jesu li pravilno definirani. Validacija osigurava da imamo ispravan skup zahtjeva koji će rezultirati softverom koji zadovoljava korisničke potrebe. Ključno je da se validacija provodi od radnih faza, kako bismo bilo kakve pogrešne zahtjeve lakše i jeftinije korigirali. Konačnu validaciju specificiranih zahtjeva obično provode sami korisnici prije nego što se softver preda.

Sukladno validaciji, važna je i aktivnost verifikacije za provjeru zahtjeva. Cilj je verifikacije provjeriti jesu li kasnije faze razvoja softvera u potpunosti i pravilno implementirale specificirane zahtjeve, odnosno jesu li svi specificirani zahtjevi uzeti u obzir prilikom dizajna softvera, ispravno implementirani i testirani. Za tu svrhu često se koristi matrica praćenja (eng. *traceability matrix*) koja uspostavlja vezu između svakog zahtjeva iz specifikacije i drugih softverskih artefakata poput UML dijagrama, programskog koda i testnih scenarija.

3.2.5.Upravljanje zahtjevima

Kroz životni ciklus softvera zahtjevi se mijenjaju što stvara potrebu da se ti zahtjevi održavaju i kontroliraju kako bi ostali relevantni. Osim toga, uključuje i dokumentiranje promjena, praćenje verzija, upravljanje sljedivosti između zahtjeva i drugih artefakata te osiguravanje da zahtjevi ostanu valjani i usklađeni s rastućim potrebama dionika. Ovo je kritičan korak u životnom ciklusu razvoja softvera jer pomaže osigurati da se razvija softverski sustav koji zadovoljava potrebe i očekivanja dionika te da se razvoj odvija unutar predviđenog vremenskog okvira, proračuna i kvalitete. Upravljanje zahtjevima također pomaže u sprječavanju širenja opsega projekta te osigurava da su zahtjevi usklađeni s ciljevima projekta.

3.3. Smjernice za pisanje zahtjeva

Vodiči za pisanje zahtjeva služe kao skup najboljih praksi i smjernica kako bi se osiguralo da su zahtjevi učinkovito dokumentirani, priopćeni i shvaćeni. Sukladno tome, smjernice za pisanje zahtjeva koriste se za pružanje sustavnog pristupa te strukture za izražavanje potreba i specifikacija određenog softvera, projekta ili proizvoda. Jedan od često korištenih vodiča tj. smjernica u praksi je vodič Međunarodnog Vijeća za Inženjerstvo Sustava (eng. *International Council on Systems Engineering - INCOSE*) za pisanje zahtjeva. Ryan, Wheatcraft, Zinni, Dick i Baksa (2019) u INCOSE vodiču za pisanje zahtjeva navode da se ovaj vodič fokusira na izražavanje koncepata, potreba i zahtjeva u kontekstu inženjerstva sustava te da je cilj kombinirati savjete iz relevantnih standarda poput ISO/IEC/IEEE 29148 te doprinosa autora i recenzenata kako bi se stvorio sveobuhvatan skup karakteristika, pravila i atributa za dobro oblikovane izjave potreba i zahtjeva. Uz INCOSE smjernice postoji i predložak za specifikacije zahtjeva Volere (eng. *The Volere Requirements Specification Template*). Predložak pruža osnovu za izradu specifikacije zahtjeva i sadrži odjeljke za svaku vrstu zahtjeva koja je relevantna za suvremene softverske sustave.

3.3.1.Karakteristike zahtjeva

U ovom se poglavlju definiraju karakteristike zahtjeva te pravila i atributi pridruženi svakoj karakteristici kojima se autorima pomaže da navedene karakteristike postignu. „Pri definiranju potreba i zahtjeva potrebno je paziti da se osigura da je izjava o potrebi ili zahtjevu pravilno oblikovana.“ (Ryan, Wheatcraft, Zinni, Dick, Baksa, 2019, str. 29). Pravila će biti objašnjena u poglavlju 5.3., a atributi u poglavlju 5.4.

Tablica 1: Prikaz karakteristika zahtjeva

Karakteristika	Definicija	Pravila	Atributi ostvarenja
C1 - Potreba	Bitna sposobnost, karakteristika ili ograničenje potreban za zadovoljenje koncepta, čijim će izostavljanjem nedostajati karakteristika koja se neće moći ispuniti implementacijom drugih zahtjeva.	R30	A1 – obrazloženje A4 – praćenje do roditelja A5 – praćenje od izvora A31 – praćenje do

			zahtjeva kolega
C2 - Prikladnost	Specifičnost i razina detalja zahtjeva prikladna je razini apstrakcije objekta na koji se odnosi.	R R31	/
C3 - Nedvosmislenost	Zahtjev mora biti tako napisan da se može protumačiti na samo jedan način od strane svih čitatelja.	R1 – R19 R22 – R24 R28 R32 – R37	A1 – obrazloženje A2 –Primarna metoda verifikacije ili validacije Sistem interesa (eng. *System of interes - SOI*) A3 – SOI verifikacijski ili validacijski pristup
C4 – Potpunost	Sposobnost, karakteristika, ograničenje ili faktor kvalitete zahtjeva se može razumjeti bez dodatnih informacija.	R6 – R9 R18 R24-R25 R33 – R35 R39	A30 - praćenje do definicije sučelja
C5 - Singularnost	Zahtjev bi trebao navesti jednu sposobnost, karakteristiku ili faktor kvalitete.	R9 R18 – R23 R39	/
C6 - Izvedivost	Zahtjev se treba realizirati unutar ograničenja kao npr. trošak, vrijeme, sigurnost uz prihvatljiv rizik.	R26 R33	A2 – SOI primarna metoda verifikacije ili validacije A3 – SOI verifikacijski ili validacijski pristup A5 – praćenje od izvora A34 – rizik implementacije

			A35 – Ublažavanje rizika
C7 – Provjerljivost	Struktura i formulacija zahtjeva je takva da se može provjeriti njegova ispunjenost.	R1 – R10 R15 R17 – R18 R24, R26, R28 R32 – R35	A2 – SOI primarna metoda verifikacije ili validacije A3 – SOI verifikacijski ili validacijski pristup A30 - praćenje do definicije sučelja
C8 – Točnost	Zahtjev mora biti točna reprezentacija korisnikove potrebe.	R6 R32 – R33 R36	A2 – SOI primarna metoda verifikacije ili validacije A3 – SOI verifikacijski ili validacijski pristup A6 – uvjet korištenja
C9 - Sukladnost	Zahtjev treba biti napisan u skladu s odobrenim uzorkom, stilom ili standardom za pisanje i upravljanje zahtjevima.	R12, R18 R30 R36 – R40	/

(Izrađeno na temelju Ryan, Wheatcraft, Zinni, Dick, Baksa, 2019, str. 29)

3.3.2.Karakteristike skupa zahtjeva

Kao i u prethodnom poglavlju, ovdje se definiraju karakteristike, ali za skup zahtjeva uz pridružena pravila i atribute. Osim pisanja pojedinačnih zahtjeva koji imaju karakteristike definirane u prethodnom odjeljku, sljedeće karakteristike dobro napisanog skupa zahtjeva također se moraju uzeti u obzir.

Tablica 2: Prikaz karakteristika skupa zahtjeva

Karakteristika	Definicija	Pravila	Atributi ostvarenja
C10 - Potpunost	Skup zahtjeva treba dostatno opisivati potrebne mogućnosti, karakteristike, ograničenja, sučelja, standarde, propise i faktore kvalitete da bi se zadovoljile potrebe bez potrebe za drugim skupom potreba.	R3 R29 R41	A4 – praćenje do roditelja A7 – stanja i načini rada A38 – vrsta /kategorija
C11 – Konzistentnost	Skup zahtjeva treba sadržavati jedinstvene potrebe, koje se ne preklapaju s drugima, a jedinice i mjerni sustavi koji se koriste su homogeni. Jezik koji koristi unutar skupa je dosljedan.	R4 R29 – R30 R36 – R41	A30 – praćenje do definicije sučelja A31 – praćenje do zahtjeva kolega
C12 – Izvedivost	Skup zahtjeva se treba moći realizirati unutar ograničenja kao trošak, vrijeme, oprema uz prihvatljiv rizik.	R26 R29 – R30 R33 – R34	A24 – stabilnost A29 – status implementacije A34 – rizik implementacije A36 - ključna potreba ili zahtjev pogona
C13 – Razumljivost	Skup zahtjeva treba biti napisan na taj način da su jasna očekivanja subjekta i njegov odnos prema sustavu.	R4 R18 R36 – R41	A1 - obrazloženje
C14 – mogućnost validacije	Treba biti moguće dokazati da će skup zahtjeva dovesti do ostvarenja potreba i zahtjeva više razine unutar ograničenja kao trošak, vrijeme, oprema uz prihvatljiv rizik.	R3 – R4 R36 – R41	/

(Izrađeno na temelju Ryan, Wheatcraft, Zinni, Dick, Baksa, 2019, str. 43)

3.3.3. Pravila za definiranje zahtjeva i skupova zahtjeva

U ovom se poglavlju utvrđuju pravila za pojedinačne zahtjeve i skupove zahtjeva čiji je cilj osigurati da se formuliraju s karakteristikama definiranim u prethodnim poglavljima. Svako je pravilo popraćeno objašnjenjem kao i karakteristikama koje su utvrđene tim pravilom.

Tablica 3: Prikaz pravila definiranja zahtjeva

Pravilo	Objašnjenje	Utvrđene karakteristike
R1 – struktura rečenice	Zahtjev mora biti u obliku potpune rečenice, najjednostavnijeg oblika: <subjekt> <glagol> <objekt>, gdje je „subjekt“ entitet koji poduzima radnju, „glagol“ je radnja koja se izvodi, a na „objekt“ se djeluje.	C3 – Nedvosmislenost C7 - Provjerljivost
R2 – aktivni oblik	Koristite aktiv prilikom pisanja zahtjeva tako da je subjekt onaj koji provodi radnju rečenice.	C2 – Prikladnost C3 – Nedvosmislenost C7 - Provjerljivost
R3 – prikladna razina subjekta i glagola	Razina subjekta i glagola mora biti prikladna onoj kojoj se odnosi. Npr. Zahtjevi koji se odnose na razinu poslovnog upravljanja će imati oblik „<upravljanje poslovanjem> će moći…“	C2 – Prikladnost C3 – Nedvosmislenost C7 – Provjerljivost C10 – potpunost C14 – mogućnost validacije
R4 – definiranje pojmova	Pojmove definirajte u nekom obliku ontologije, rječniku podataka ili sl., koji omogućuje ispravno razumijevanje onoga što se napisalo.	C3 – Nedvosmislenost C7 – Provjerljivost C11 – konzistentnost C13 - razumljivost C14 – mogućnost validacije
R5 – korištenje određenog člana (**engleski jezik**)	Koristite određeni član „the“ umjesto neodređenog „a“.	C3 – Nedvosmislenost C7 – Provjerljivost
R6 – jedinice mjere	Svi brojevi moraju imati navedene mjerne jedinice mjernog sustava ili stvari na koju se broj odnosi.	C3 – Nedvosmislenost C4 - Potpunost C7 – Provjerljivost C8 - Točnost
R7 – nejasni pojmovi	Izbjegavajte korištenje pojmova kao što su: neki, bilo koji, skoro, otprilike, itd.	C3 – Nedvosmislenost C4 - Potpunost C7 – Provjerljivost
R8 – klauzule bijega	Izbjegavajte korištenje klauzula bijega kao što su: koliko je moguće, što je manje moguće, prema potrebi, ako je izvedivo, itd.	C3 – Nedvosmislenost C4 - Potpunost C7 – Provjerljivost

R9 – otvorene klauzule	Izbjegavajte otvorene klauzule kao što je „i tako dalje“, umjesto da kažete što je još potrebno.	C3 – Nedvosmislenost C4 – Potpunost C5 - Singularnost C7 – Provjerljivost
R10 – suvišni infinitivi	Izbjegavajte pisanje zahtjeva kao što su „Sustav će biti dizajniran da može…“ ili „Sustav će biti dizajniran da bude sposoban za…“, umjesto „Sustav će…“.	C3 – Nedvosmislenost C7 – Provjerljivost
R11 – zasebne klauzule	Svaki zahtjev treba imati glavni glagol koji opisuje funkcionalnost ili potrebu te bi svaka nadopuna uvjeta ili ograničenja trebala imati zasebnu klauzulu.	C3 – Nedvosmislenost
R12 – ispravna gramatika	Neispravna gramatika dovodi do dvosmislenosti i nedostatku razumijevanja.	C3 – Nedvosmislenost C9 - Sukladnost
R13 – ispravan pravopis	Neispravan pravopis može dovesti do zabune.	C3 – Nedvosmislenost
R14 – ispravna interpunkcija	Neispravna interpunkcija može dovesti do zabune između podrečenica u zahtjevu te veći broj interpunkcije povećava šansu za dvosmislenošću.	C3 – Nedvosmislenost
R15 – logički uvjeti	Koristite definirane konvencije da biste izrazili logičke izraze, npr. “[X I Y]”, “[X ILI Y]”, [X XOR Y]”, “NE[X ILI Y]”.	C3 – Nedvosmislenost
R16 – izbjegavajte negaciju	Stavljanje negacije u zahtjev uzrokuje nešto što je nemoguće provjeriti, npr. *<Sustav> neće otkazati.*	C3 – Nedvosmislenost C7 – Provjerljivost
R17 - simbol „/“	Simbole kose crte „/“ označava da su dva podatka odijeljena kosom crtom iste vrijednosti, no ima toliko drugih značenja i uporaba da bi se trebao izbjegavati.	C3 – Nedvosmislenost C7 – Provjerljivost
R18 – jedna rečenica	Zahtjev bi trebao sadržavati jednu misao u jednoj rečenici, koja se nadopuni s dodatnim podrečenicama.	C3 – Nedvosmislenost C4 – Potpunost C5 - Singularnost C7 – Provjerljivost C9 - Sukladnost C13 - razumljivost

R19 – izbjegavajte veznike	Prisutnost veznika kao što su *i, ili, pa, te, ni, niti, ali, kao,* itd. može značiti da taj zahtjev sadrži više zahtjeva u sebi.	C3 – Nedvosmislenost C5 - Singularnost
R20 – izbjegavajte svrhu	Zahtjev u sebi ne mora sadržavati i svrhu svojeg postojanja, već svrhu treba obrazložiti u zasebnom dijelu.	C5 - Singularnost
R21 – izbjegavajte zagrade	Prisutnost zagrada najčešće označava da se u zahtjevu nalaze nepotrebne informacije ili pridonose dvosmislenosti.	C5 - Singularnost
R22 - nabrajanja	Ako se u zahtjevu govori o nekakvom skupu, uvijek je bolje da se svaki element tog skupa ispiše kao zaseban zahtjev.	C2 – Prikladnost C3 – Nedvosmislenost C5 - Singularnost
R23 - kontekst	U nekim slučajevima je teško razumjeti složen zahtjev te je jednostavnije priložiti dijagram ili model.	C3 – Nedvosmislenost C4 – Potpunost C5 - Singularnost
R24 – izbjegavajte zamjenice	Kako biste izbjegli dvosmislenost, koristite i ponavljajte imenice u cijelosti. Također izbjegavajte neodređene zamjenice kao što su *svi, svaki, bilo koji, ostali* i sl.	C3 – Nedvosmislenost C4 – Potpunost C7 – Provjerljivost
R25 – korištenje naslova	Izbjegavajte korištenje naslova kao objašnjenja ili dodatnih objašnjenja za zahtjeve.	C4 – Potpunost
R26 – izbjegavajte apsolutne vrijednosti	Korištenje apsolutnih vrijednosti kao što je „stopostotna dostupnost“ nije realno, a ni ostvarivo.	C7 – Provjerljivost C6 – Izvedivost C12 – Izvedivost
R27 - eksplicitnost	Eksplicitno navedite uvjet pod kojim je zahtjev primjenjiv.	C4 – Potpunost C7 – Provjerljivost
R28 – eksplicitnost liste	Za svaku akciju navedite eksplicitno uvjet umjesto da navedete uvjet i pobrojite akcije za taj uvjet.	C3 – Nedvosmislenost C7 – Provjerljivost
R29 - klasifikacija	Klasificirajte zahtjeve po tipu i kategoriji problemske domene ili samog rješenja.	C10 – potpunost C11 – konzistentnost C12 – Izvedivost
R30 – izrazite jednom	Svaki zahtjev napišite i izrazite samo jednom.	C1 - Potreba C9 - Sukladnost C11 – Konzistentnost

		C12 – Izvedivost
R31 – bez rješenja	Kada definirate zahtjeve izbjegavajte predlaganje rješenja. Aktivnosti dizajna se bave pitanjem „kako“ realizirati zahtjeve, dok se u definiranju zahtjeva bavite pitanjem „što“.	C2 – Prikladnost
R32 – univerzalna kvantifikacija	Kada želite obuhvatiti cjelokupnost nekog aspekta rješenja koristite *svaki* umjesto „svi“, „bilo koji“ ili „oba“.	C3 – Nedvosmislenost C7 – Provjerljivost C8 - Točnost
R33 – raspon vrijednosti	Koristite raspon vrijednosti kod pisanja zahtjeva te neka taj raspon bude unutar prihvatljive razine.	C3 – Nedvosmislenost C4 – Potpunost C6 – Izvedivost C7 – Provjerljivost C8 – Točnost C12 – Izvedivost
R34 – mjerljivost	Koristite konkretne, mjerljive vrijednosti kod pisanja zahtjeva. Izbjegavajte riječi „brzo“, „maksimalno“, „minimalno“, „optimalno“, „srednje veličine“ i sl.	C3 – Nedvosmislenost C4 – Potpunost C7 – Provjerljivost C12 – Izvedivost
R35 – vremenska neodređenost	Koristite određena vremenska ograničenja umjesto „na kraju“, „trenutačno“, „najkasnije“, „najranije“, „prije“, „nakon“ i sl.	C3 – Nedvosmislenost C4 – Potpunost C7 – Provjerljivost
R36 – konzistentnost pojmova	Pojmove i mjerne jedinice koje definirate u skladu s pravilima R4 i R6 moraju biti konzistentno korišteni u cijelom skupu zahtjeva.	C3 – Nedvosmislenost C8 – Točnost C9 - Sukladnost C11 – konzistentnost C13 – razumljivost C14 – mogućnost validacije
R37 – definirajte akronime	Ako koristite akronime u zahtjevima, primjena toga skupa mora biti konzistentna.	C3 – Nedvosmislenost C9 - Sukladnost C11 – konzistentnost C13 – razumljivost C14 – mogućnost validacije
R38 – izbjegavajte kratice	Korištenje kratica treba se izbjegavati zato što one samo dodaju dvosmislenost.	C9 - Sukladnost C11 – konzistentnost C13 – razumljivost C14 – mogućnost validacije
R39 – stilski vodič	Koristite stilski vodič za pisanje zahtjeva na razini projekta.	C5 - Singularnost C9 - Sukladnost

		C11 – konzistentnost C13 – razumljivost C14 – mogućnost validacije
R40 – srodni zahtjevi	Zahtjeve koji pripadaju zajedno grupirajte zajedno po funkciji, kvaliteti, tipu i sl.	C9 - Sukladnost C11 – konzistentnost C13 – razumljivost
R41 - struktura	Dobra struktura skupa zahtjeva omogućit će vam lakše razumijevanje.	C10 – potpunost C11 – konzistentnost C13 – razumljivost C14 – mogućnost validacije

(Izrađeno na temelju Ryan, Wheatcraft, Zinni, Dick, Baksa, 2019, str. 51)

3.3.4. Atributi zahtjeva

Posljednje poglavlje sadrži isječak popisa atributa koji se mogu povezati sa svakim zahtjevom. Svrha je popisa predstaviti atribute koji su se koristili i one koje koriste različite organizacije. Može se odabrati samo određeni skup atributa ili organizacija može imati svoje atribute koji se odnose na domenu projekata te organizacije. „Važno je da organizacije moraju definirati shemu atributa za svoj projekt. Ova shema atributa bit će specifična za domenu, projekt i tvrtku.“ (Ryan, Wheatcraft, Zinni, Dick, Baksa, 2019, str. 87). Za više informacija o atributima, ali i karakteristikama i pravilima, možete posjetiti web stranicu INCOSE-a (https://www.incose.org).

Tablica 4: Prikaz isječka atributa zahtjeva

A1 - obrazloženje	Obrazloženje ima svrhu pružiti razloge postojanja određene izjave o potrebi ili zahtjevu. Ono detaljno opisuje zašto je određena potreba ili zahtjev nužan, pružajući druge relevantne informacije koje pomažu boljem razumijevanju svrhe i namjere tog zahtjeva ili potrebe.
A2 – SOI primarna metoda verifikacije ili validacije	Metoda primarne verifikacije ili validacije za svaku potrebu ili zahtjev navodi planiranu primarnu metodu kojom će se provjeriti ili potvrditi da projektirani i izgrađeni sustav zadovoljava zahtjeve (verifikacija) ili potrebe dionika (validacija). Ova metoda može uključivati testiranje, demonstraciju, inspekciju ili analizu.

A3 – SOI verifikacijski ili validacijski pristup	Ovaj atribut predstavlja definiciju kriterija uspjeha za provjeru ispunjavanja zahtjeva sustava ili provjeru valjanosti zadovoljavanja potreba. U nekim alatima postoji mogućnost definiranja zasebnog skupa "zahtjeva za provjeru" ili "zahtjeva za provjeru valjanosti". Kroz taj pristup, ovaj atribut bi bio povezan s tim zahtjevima i služio kao veza koja opisuje kako će se provjeriti ili potvrditi da sustav zadovoljava zahtjeve ili valjanost potreba. Kriteriji uspjeha obično sadrže mjerljive ili opipljive elemente koji se koriste za ocjenu uspješnosti provjere ili validacije.
A4 – Praćenje do roditelja	Zahtjevi na razini sustava su rezultat transformacije jedne ili više potreba. Zahtjevi na nižoj razini arhitekture sustava implementiraju se kroz dodjelu zahtjeva s više razine. Dijete zahtjeva je onaj koji proizlazi ili se izvodi iz pripadajućeg roditeljskog zahtjeva. Postizanje svakog djetetovog zahtjeva rezultira ispunjenjem roditeljskog zahtjeva. Svaki od podređenih zahtjeva mora biti praćen sve do svog roditeljskog zahtjeva kako bi se osigurala cjelovitost i dosljednost sustava.
A5 – Praćenje do izvora	Svaki izvor potrebe i zahtjeva mora biti dostupan za pristup. Zahtjevi se razlikuju od praćenja nadređenih zahtjeva jer identificiraju njihovu specifičnu svrhu ili način na koji je sadržaj zahtjeva određen. Svaka potreba mora biti povezana s njenim izvorom kako bi se utvrdilo odakle je potreba nastala.
A6 – Uvjeti korištenja	Opis očekivanih radnih uvjeta uporabe u kojima se primjenjuje potreba ili zahtjev.
A7 – Stanja i načini rada	Stanje ili način rada sustava na koji se odnosi zahtjev. Neki sustavi imaju različita stanja ili načine rada, pri čemu svako stanje ili način ima svoj vlastiti skup zahtjeva koji se odnose na to specifično stanje ili način rada. U slučaju da je sustav strukturiran na taj način, ovaj atribut omogućuje dodjelu zahtjeva odgovarajućim stanjima ili načinima rada.
A24 - Stabilnost	Naznaka vjerojatnosti promjene zahtjeva. Neke potrebe ili zahtjevi imat će u tekstu oznake koje treba odrediti (eng. *to-be-determined - TBD*), koje treba izračunati (eng. *to-be-calculated - TBC*) ili koje treba riješiti (eng. *to-be-resolved - TBR*).
A29 – Status implementacije	Pokazatelj statusa implementacije ili realizacije zahtjeva u rezultatima dizajna. Moguće vrijednosti uključuju zahtjev implementiran u dizajnu; zahtjev nije

	proveden i nema plana da se to učini; i zahtjev nije proveden, ali postoji plan za to.
A30 – Praćenje do definicije sučelja	Interakcije dvaju sustava su opisane u definicijama sučelja koje su često sadržane u dokumentu. Zahtjevi za sučelje koji se nalaze u svakom skupu zahtjeva za sustave koji su u interakciji će uključivati referencu na mjesto gdje je ta interakcija definirana. Ovaj atribut pruža vezu koja omogućuje praćenje zahtjeva sučelja do mjesta u dokumentaciji gdje je ta interakcija detaljno opisana i definirana.
A31 – Praćenje do zahtjeva kolega	Ovaj atribut povezuje zahtjeve koji su srodni na istoj razini.
A34 – Rizik implementacije	Vrijednost koja se pridružuje svakoj potrebi ili zahtjevu te indicira rizik da se potreba ili zahtjev ne ispuni.
A35 – Ublažavanje rizika	Indicira je li potreba ili zahtjev dionika dio smanjenja rizika.
A36 - Ključna potreba ili zahtjev pogona	Za implementaciju ključna potreba ili zahtjev može imati velik utjecaj na troškove ili raspored.
A38 – Vrsta/kategorija	Svaka će organizacija definirati vrste ili kategorije kojima se potreba ili zahtjev uklapaju, na temelju načina na koji žele organizirati svoje zahtjeve.

(Izrađeno na temelju Ryan, Wheatcraft, Zinni, Dick, Baksa, 2019, str. 29)

3.4. Generički proces za inženjerstvo zahtjeva

Generički proces za inženjerstvo sustava bit će predstavljen u idealnom svijetu bez promjena, a zatim u realnijem obliku gdje ima promjena.

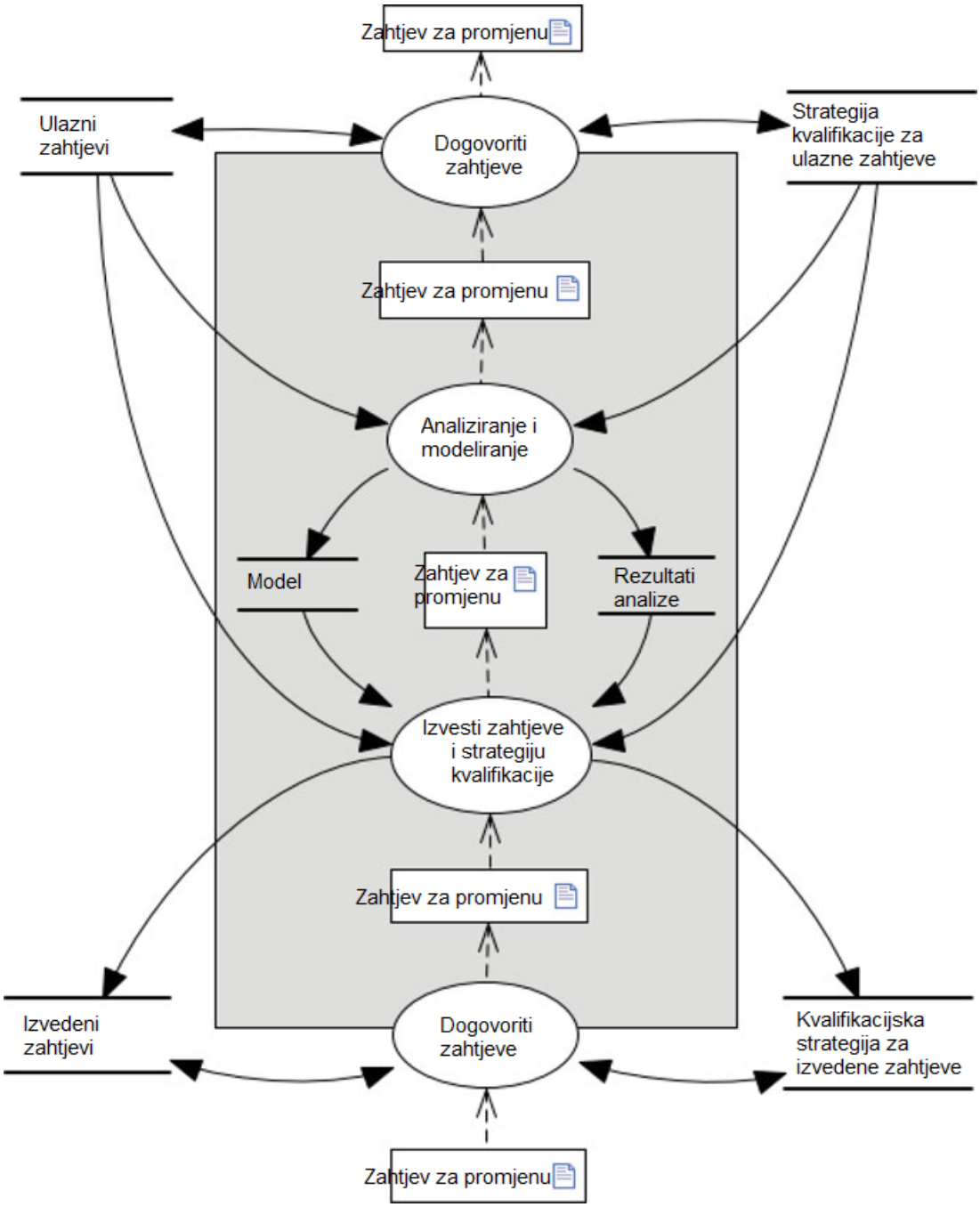
3.4.1. Idealan razvoj

Na slici 1 prikazan je proces inženjerstva zahtjeva u idealnom svijetu. Proces počinje dogovaranjem zahtjeva projekta s kupcem na razini iznad. Slijedi analiza ulaznih informacija i razmatranje kako razviti potrebne izlaze. Ova aktivnost najčešće uključuje kreiranje jednog ili više modela i izvješća analize koji zajedno pružaju temelj za provođenje zahtjeva.

3.4.2. Razvoj s promjenama

Poznato je da će u današnjem svijetu sve biti sklono promjenama i neće mirovati. Ovo se posebno odnosi na svijet razvoja sustava gdje svi konstantno mijenjaju mišljenja o tome što bi htjeli i ono što je dogovoreno više nije moguće izvesti. Zbog takvih se situacija prethodni generički model procesa morao modificirati kako bi se takvi slučajevi uzeli u obzir. Na slici 1 prikazan je inženjerstvo zahtjeva s promjenama. Iz slike je vidljivo da gotovo svaka aktivnost

može dovesti do stvaranje promjene. Te promjene ovdje teku prema gore što ne ukazuje na nedostatak želja za promjenom kod kupaca ili pojavu promjena samo na nižim razinama, već da se obrnuta putanja uzela u obzir. Obično se zahtjev za promjenom pojavljuje u koraku gdje se izvode zahtjevi i strategije kvalifikacije tako što se otkrije nepravilnost u izvješću ili modelu. Slično tome se u koraku prije, analizi i modeliranju, može pojaviti problem s ulaznim zahtjevom te će onda zahtjev za promjenom biti upućen procesu dogovora zahtjeva.



Slika 1: Proces inženjerstva zahtjeva u kontekstu promjena (Izvor: Dick, Hull i Jackson, 2017, str.42)

4. Analiza postojećih alata

Tema je ovoga rada razvoj alata za potporu aktivnosti specificiranja softverskih zahtjeva te je zbog toga veći naglasak na praktičnom dijelu rada. Iz tog je razloga jedan od koraka prije početka razvoja alata bio istražiti već postojeće alate slične namjene kako bih dobio ideju što bih implementirao. Saznao sam da se većina alata, između ostalog, fokusira na organiziranje i strukturiranje alata, a pri tome izostaju analiziranje i sugeriranje poboljšanja. Nakon malo dubljeg istraživanja, odlučio sam analizirati i usporediti pet alata čije su mi se funkcionalnosti činile najviše slične onome što bih trebao implementirati. Tih pet alata su QVscribe, ReqSuite® RM, RQA – QUALITY Studio®, ScopeMaster® i ReqView. U sljedećim će poglavljima navedeni alati biti uspoređeni po četiri kriterija: verzije, značajke, podrška i obuka te raspoređivanje i ovjera. Važna napomena je da su informacije koje su sadržane u kriterijima usporedbe većinom preuzete s web stranica navedenih alata te neki jednostavno nisu imali ponuđene informacije vezane uz te kriterije.

4.1. QVScribe

Alat QVScribe razvijen je od strane QRA Corp poduzeća te je to dodatak za Microsoft Word i Excel, Polarion, DOORS Next i Jama. Alat provodi analizu kvalitete najbolje prakse, analizu dosljednosti terminologije, analizu dosljednosti jedinica i analizu sličnosti zahtjeva za svaki pojedinačni zahtjev kako bi se poboljšala jasnoća, dosljednost te kvaliteta zahtjeva i tehničke dokumentacije. QVscribe pomaže timovima da napišu jasne zahtjeve u kraćem vremenu. QVscribe ocjenjuje pojedinačne zahtjeve za jasnoćom i poboljšava ukupnu dosljednost cijelih dokumenata („QRA Corp“, bez dat.). Ovaj alat nudi nekoliko verzija. Neke od njih se plaćaju, a postoje i dvije besplatne verzije koje služe kako bi se provjerilo odgovara li alat nekoj organizaciji. Verzije koje se plaćaju su „Enterprise“ za organizacije koje žele poboljšati procese zahtjeva preko više timova i konfigurirati rezultate analize, „Starter“ za individualne timove koji žele poboljšati proces zahtjeva i konfigurirati rezultate analize i „Solo“ za individualne korisnike koji žele poboljšati svoje zahtjeve te ne trebaju konfigurirati rezultate analize. Od besplatnih verzija postoji besplatna proba i demo, od kojih demo ima više mogućnosti. Što se tiče značajki, njihova dostupnost ovisi o verziji koja se kupi. Dakle, s „Enterprise“ verzijom dobivaju se sve značajke, dok se sa „Solo“ verzijom dobivaju samo prve tri. Ovaj alat pruža kvalitetnu podršku i obuku, baš kao i raspoređivanje i ovjeru. Dostupnost tih usluga ovisi o tome koja se verzija kupi.

Tablica 5: Prikaz alata QVScribe po kriterijima usporedbe

Verzije	Enterprise Starter Solo Besplatna proba (eng. *free trial*) Demo
Značajke	Analiza kvalitete, konzistentnosti, sličnosti. Izvještaji o analizi u obliku prijenosnog formata dokument (eng. *Portable Document Format - PDF*). EARS (Easy Approach to Requirement Syntax) smjernice i provjere sukladnosti. Dijeljene prilagodljive konfiguracije analiza.
Podrška i obuka	30 minutna uvodna sesija 1 na 1. Pristup videozapisima za obuku i email podrška. 60 minutna sesija za grupnu obuku sa snimanjem. Poseban menadžer uspjeha.
Raspoređivanje i ovjera	Namjenski poslužitelj. Opcija lokalnog poslužitelja. Ovjera aktivnog direktorija.

4.2. ReqSuite® RM

ReqSuite® RM lokalno je rješenje za upravljanje zahtjevima temeljeno na oblaku koji pomaže tvrtkama u raspisivanju natječaja, nabavi, razvoju proizvoda. Osim toga, pomaže menadžmentu poslovnih aplikacija pojednostaviti procese povezane sa standardizacijom, suradnjom, konfiguracijom i drugim operacijama. ReqSuite® RM omogućuje članovima osoblja da automatski spreme promjene unesene u zahtjeve i generiraju usporedbe verzija za daljnju analizu. Nadzornici mogu definirati tijek rada statusa i pratiti status završetka projekta na temelju višestrukih kategorija, kao što je distribucija stanja stavki, dnevnik povijesti promjena i otvoreni zadaci. Alat dolazi s programskim sučeljem aplikacije (eng. *Application Programming Interface - API*) koji tvrtkama omogućuje integraciju platforme s nekoliko rješenja trećih strana, kao što su JIRA, GitLab, Enterprise Architect i MS Excel. („Software Advice, Inc.“, 2023). Kod ovog alata također postoje iste verzije koje se ne plaćaju, besplatna proba i demo, dok za plaćene verzije postoji jednokratna kupnja uz održavanje i pretplata. Obje plaćene verzije sadrže sve značajke koje su navedene u tablici. Nisam uspio pronaći informacije vezane uz podršku i obuku te raspoređivanje i ovjeru.

Tablica 6: Prikaz alata ReqSuite® RM po kriterijima usporedbe

Verzije	Jednokratna kupnja uz održavanje. Pretplata koja uključuje sva ažuriranja i podršku. Besplatna proba (eng. *free trial*) Demo
Značajke	Upravljanje zahtjevima i sličnim artefaktima pomoću slobodno definiranih kategorija. Automatsko spremanje svake modifikacije. Održavanje semantičkih veza između zahtjeva i povezanih artefakata. Standardiziranje biblioteka i ponovno iskorištenje istih za ostale projekte. Asistiranje i automatiziranje definiranju te pregledu zahtjeva pomoću UI. Kontrola napretka na projektima i kreiranje izvještaja te analiza. Dodjeljivanje zadataka pojedincima. Dvosmjerna sinkronizacija s ostalim alatima te pomoću REST API-a, uvoz i izvoz za Word, Excel i ReqIF. Prilagodba cjelokupnog alata prema potrebama korisnika.
Podrška i obuka	Nema informacija.
Raspoređivanje i ovjera	Nema informacija.

4.3. RQA – QUALITY Studio®

RQA – QUALITY Studio® pruža sredstva za povezivanje s velikim brojem inženjerskih alata (alati za upravljanje zahtjevima, alati za modeliranje UML/SysML, alati za simulaciju, MS Excel listovi, MS Word dokumenti...), dohvaćanje ključnih elemenata kojima se upravlja na tim alatima i pruža automatsku inspekciju prema kriterijima CCC-a. RQA - QUALITY Studio® pruža stotine metrika kvalitete za analizu različitih vrsta repozitorija zahtjeva: IBM DOORS (klasična i NG), PTC Integrity Lifecyle Manager, 3D Experience Requirements Manager, CATIA Reqtify, Visure Solutions, MS Excel, ReqIF i OSLC kompatibilan izvori informacija. Ove metrike pokrivaju CCC pristup i uključuju mehanizme proširivosti koji proširuju broj metrika korištenjem metrike koja se može parametrizirati (metrike koje krajnji korisnici mogu lako konfigurirati), metrike prilagođenog koda (metrike koje mogu kodirati napredni krajnji korisnici)

i metrike temeljene na popisu za provjeru (koje omogućuju ručno orijentiran pregled). („The REUSE Company“, bez dat.). Za ovaj alat sam nisam uspio pronaći niti verzije koje se plaćaju niti načine plaćanja, no saznao sam da se može rezervirati demo alata. Značajke alata popisane su u tablici, ali za podršku i obuku te raspoređivanje i ovjeru također nisam uspio pronaći informacije.

Tablica 7: Prikaz alata RQA – QUALITY Studio® po kriterijima usporedbe

Verzije	Demo
Značajke	Verifikacija i validacija zahtjeva. Višekorisnička suradnja. Pruža stotine metrika kvalitete za analiziranje različitih vrsta repozitorija zahtjeva. Metrike pokrivaju pristup točnost, konzistentnost i potpunost (eng. *Correctness, Consistency and Completeness - CCC*) i uključuju mehanizme proširenja koji proširuju broj metrike korištenjem parametrabilnih metrika (lako prilagodljive od krajnjih korisnika), metrike prilagođenog koda (mogu biti kodirane od strane naprednih krajnjih korisnika) i metrike temeljene na popisu za provjeru (omogućuju ručno orijentiran pregled).
Podrška i obuka	Nema informacija.
Raspoređivanje i ovjera	Nema informacija.

4.4. ScopeMaster®

ScopeMaster® je inteligentan alat za analizu softverskih zahtjeva. Čita tekst zahtjeva i umjesto vas obavlja dugotrajan rad na analizi. Raščlanjuje, tumači, testira, unakrsne reference i čak procjenjuje priče korisnika. Pronalazi potencijalne probleme kao što su: dvosmislenosti, duplikati, propusti, nedosljednosti i složenosti. ScopeMaster® transparentno koristi umjetnu inteligenciju (NLP) za ubrzavanje analize dok osigurava dosljedne rezultate. Ugrađena integracija za popularne alate kao što su Jira i Azure DevOps. REST API omogućuje fleksibilnu integraciju s bilo kojim alatom („ScopeMaster Ltd.“, 2023). Ovaj alat ima vrlo fleksibilne verzije i načine plaćanja. Dakle, uz demo i besplatnu probu sadrži i plaćanje po potrebi. U tablici možete vidjeti različite verzije te se one razlikuju po broju korisnika, korisničkih priča te cijeni.

Cijene se temelje na potrošnji (analizirana priča korisnika). Plaćajte samo ono što koristite. Kupite pakete kredita kada zatrebaju za analizu vaših korisničkih priča. Krediti se dijele među korisnicima u cijeloj organizaciji. Različiti paketi mogu se kombinirati i dijeliti u cijeloj organizaciji („ScopeMaster Ltd.“, 2023). Podrška i obuka su također kvalitetne. „Tier 1“ okruženje, kod sigurnog posluživanja softvera kao servis raspoređivanja i ovjere, označuje vrstu ili kategoriju okruženja pri planiranju okruženja.

Tablica 8: Prikaz alata Scopemaster® po kriterijima usporedbe

Verzije	Demo Besplatna proba (eng. *free trial*) Plaćanje po potrebi: • Jednokratna procjena – 1000 korisničkih priča, besplatno • „Startup“ paket – 500 korisničkih priča, do 3 korisnika, 5000£ • „Project“ paket – 2000 korisničkih priča, do 10 korisnika, 18 000£ • „Large“ paket – 7000 korisničkih priča, do 50 korisnika, 50 000£ • Enterprise – dogovor preko poziva za korisničke priče, pretplata, tipični ROI 10x unutar 2 mjeseca
Značajke	Automatizirani dijagrami slučajeva korištenja. Automatizirana kontrola kvalitete i bodovanje kvalitete korisničkih priča. Pronalaženje: dvosmislenosti, duplikata, zahtjeva koji nedostaju, nekonzistentnosti. Generiranje testova, predlaganje dijagrama sljedova testova. Sveobuhvatna izvješća o nizu korisničkih priča. Uvoz korisničkih priča preko vrijednosti odvojenih zarezom (eng. *Comma-separated values - CSV*), izvoz korisničkih priča, scenarija testova i rječnika podataka preko CSV-a ili REST API-a. Suradnja s kolegama, dodjeljivanje oznaka/labela za organiziranje korisničkih priča.
Podrška i obuka	Obuka korisnika. Savjeti za osiguranje projekta. Jednokratna procjena skupa zahtjeva.

Raspoređivanje i ovjera	Sigurno posluživanje softvera kao servis u „Tier 1“ okruženju. Opcionalno posluživanje na zahtjev.

4.5. ReqView

ReqView je softver za upravljanje zahtjevima temeljen na oblaku i lokalno dizajniran za pomoć tvrtkama svih veličina u upravljanju softverskim i sistemskim zahtjevima, praćenju, validacijskim testovima i više. Platforma omogućuje korisnicima da zabilježe zahtjeve pomoću prilagodljivih predložaka, PDF-ova, tekstova, tablica, slika i drugih datoteka. Administratori mogu koristiti ReqView za prilagodbu atributa projekta u skladu s tijekovima rada, povezati zadate provjere i validacije te članovima tima davati dozvole temeljene na ulogama. Nudi mnoštvo značajki kao što je pretraživanje cijelog teksta, usporedba projekata, predlošci dokumenata, privici datoteka, pregledavanje matrice sljedivosti i više. Dodatno, nadzornici mogu osigurati usklađenost sa sigurnosnim standardima i pratiti ciljeve, rizik i izmjene kako bi osigurali pravovremeni završetak projekata. („Software Advice, Inc.“, 2023). Ovaj alat za razliku od drugih ne nudi niti demo niti besplatnu probu, ali zato nudi besplatan plan koji je za manje projekte do 150 zahtjeva. Ostali planovi kao „Pro“ omogućuje spremanje zahtjeva bez ograničenja te praćenje sljedivosti između dokumenata i suradnju na projektu preko zajedničkog pohranjenog mrežnog diska. „Team“ verzija uz neke dodatne mogućnosti omogućuje lakšu suradnju u rastućim timovima te bolju integraciju alata za upravljanje zahtjevima. Naposljetku verzija „Enterprise“ sadržava sve značajke prijašnjih verzija uz fleksibilni model licenciranja za veće timove. Značajke koje alat sadrži su veoma slične ostalim alatima i nalaze se u tablici. Podrška za alat izvršava se preko email-a i portala podržana obukom za korisnike. Nisam uspio pronaći informacije o raspoređivanju i ovjeri.

Tablica 9: Prikaz alata ReqView po kriterijima usporedbe

Verzije	Besplatan plan: otvorite bilo koji projekt, ali samo za čitanje, uredite projekte s 1 dokumentom, do 150 objekata i jednim prilagođenim atributom. Pro: 390€ korisnik/godina, neograničeni dokumenti, objekti i atributi. Team: 510€ korisnik/godina sve značajke kao u Pro + integracija s Jirom, ReqIF (Requirements Interchange

	Format) uvoz/izvoz, povijest projekta, ostale značajke za bolju kolaboraciju tima. Enterprise: sve značajke kao u Team + neograničeni korisnici, „floating license server“, online i offline „floating license“.
Značajke	Ponovo iskoristive smjernice bazirane na standardima (ISO/IEC/IEEE 29148, Volere). Organiziranje projekta u strukturirane dokumente opisujući specifikacije zahtjeva. Uvoz iz Word-a preko HTML formata, Excel-a preko CSV formata. Izvoz u Word, Excel, HTML, CSV, ReqIF datoteku. Zabilježite izvor zahtjeva, vrstu, prioritet, kriterij prihvaćanja, metodu provjere ili druge informacije. Upravljanje zahtjevima u prilagođenim tijekovima rada. Prilagodljive matrice sljedivosti. Upravljanje rizicima preko analize načina kvara i učinaka (eng. *Failure Mode and Effects Analysis - FMEA*) metode. Offline suradnja, spremanje podataka lokalno. Spremanje projekata u čitljive datoteke s JSON formatom. Integracija s Jirom.
Podrška i obuka	Email podrška. Portal korisničke podrške. Prilagođena obuka, uvoz podataka, izvješća ili druge usluge.
Raspoređivanje i ovjera	Nema informacija.

5. Praktični dio

5.1. Opseg

Softversko rješenje EasyReqAssist bavi se problemskom domenom softverskih zahtjeva. Nedostatni, promjenjivi, krivo shvaćeni softverski zahtjevi jedan su od glavnih razloga zašto još danas softverski projekti nisu uspješni. Kako bi se ovaj problem što više ublažio, ovo rješenje osmišljeno je za studente STEM fakulteta kako bi im se pomoglo u izradi njihovih projekata, koristeći OpenAI tehnologiju za poboljšanje kvalitete i učinkovitosti zahtjeva. Cilj mu je olakšati pisanje softverskih zahtjeva mladim informatičarima, kako bi stvorili solidne temelje za pisanje softverskih zahtjeva u budućnosti. Rješenje nije novo, već pojednostavljeno i izolirano od alata spomenutih u prijašnjem poglavlju u smislu funkcionalnosti pisanja softverskih zahtjeva. EasyReqAssist omogućuje stvaranje novih projekata, operacije stvaranja, čitanja, ažuriranja, brisanja (eng. *Create, Read, Update, Delete* - *CRUD*) nad zahtjevima uz recenziranje pomoću OpenAI tehnologije unutar tih projekata te spremanje u različite datoteke. EasyReqAssist ne omogućuje složena praćenja zahtjeva, suradnje ili sinkronizacije s ostalim alatima.

5.2. Perspektiva proizvoda

EasyReqAssist je neovisno i samostalno softverskog rješenje osmišljeno kao alat koji bi pomogao nekome tko tek uči pisati softverske zahtjeve, primjerice studentima. Softver nije u odnosu s nekim većim sustavom, već s OpenAI API-em razvijenom od strane OpenAI, Inc. Samim time softver zahtjeva pristup Internetu kako bi se moglo komunicirati s unaprijed obučenim generativnim transformatorom (eng. *Generative Pre-training Transformer* - *GPT*) vrstom velikom jezičnog modela (eng. *Large Language Model* - *LLM*). Ne koristi nikakvu bazu podataka te je namijenjen za operacijski sustav Windows.

5.3. Karakteristike korisnika

Primarni korisnici softvera bi bili studenti STEM fakulteta te u manjem broju nastavnici/profesori. Od studenata i nastavnika posjeduju srednju do visoku razinu tehničke pismenosti te računalnih vještina. Studenti redovito upotrjebljuju softver za definiranje projektnih zahtjeva i potreba, dok nastavnici rijetko ili povremeno kako bi kreirali nastavne

materijale ili definirale zahtjeve za svoje projekte. Ne bi postojale različite dozvole za različite korisnike.

5.4. Ograničenja

Ovaj softver nema posebnih ograničenja u većem broju pogleda, što uključuje zakonske propise, hardverska ograničenja, druge sustave, sigurnosnu kritičnost. Očekuje se da će rješenje biti razvijeno u skladu s normama industrije.

5.5. Funkcionalni zahtjevi

Tablica 10: Funkcionalni zahtjev 1

Identifikator	FZ-1
Zahtjev	Sustav će omogućiti kreiranje projekta.
Obrazloženje	Kako bi korisnik mogao kreirati zahtjeve, mora prije toga definirati projekt koji će sadržavati zahtjeve.
Način provjere	Korisnik na ekranu operacija sa zahtjevima vidi svoj projekt odabran u padajućoj listi iznad samog popisa zahtjeva.
Prioritet [1-3]	1
Izvor	Programer – Zvonimir Belina

Tablica 11: Funkcionalni zahtjev 2

Identifikator	FZ-2
Zahtjev	Sustav će omogućiti kreiranje zahtjeva.
Obrazloženje	Kako bi se obnašala glavna funkcionalnost ovog softvera, korisnik mora biti u mogućnosti kreirati softverski zahtjev koji su dio nekog projekta.
Način provjere	Nakon što je korisnik kreirao zahtjev on se prikazuje na ekranu operacija sa zahtjevima u popisu zahtjeva za odabrani projekt.
Prioritet [1-3]	1
Izvor	Programer – Zvonimir Belina

Tablica 12: Funkcionalni zahtjev 3

Identifikator	FZ-3
Zahtjev	Sustav će omogućiti pregled zahtjeva za odabrani projekt.
Obrazloženje	Kako bi korisnik mogao vidjeti napravljene zahtjeve za svaki projekt, oni moraju biti prikazani u nekom obliku.
Način provjere	Kada korisnik kreira zahtjev on bi se trebao prikazati na formi popisa zahtjeva.
Prioritet [1-3]	1
Izvor	Programer – Zvonimir Belina

Tablica 13: Funkcionalni zahtjev 4

Identifikator	FZ-4
Zahtjev	Sustav će omogućiti provjeru/recenziju zahtjeva, pomoću LLM-a u toku kreiranja zahtjeva.
Obrazloženje	Kako bi korisnik mogao provjeriti ispravnost svojeg zahtjeva, može na klik gumba pitati OpenAI LLM za recenziju, koji mu onda daje prijedloge kako poboljšati zahtjev.
Način provjere	U toku kreiranja zahtjeva kada korisnik napiše zahtjev i klikne na gumb provjere, nakon nekoliko trenutaka u prozoru desno mu se prikazuje odgovor LLM-a.
Prioritet [1-3]	1
Izvor	Programer – Zvonimir Belina

Tablica 14: Funkcionalni zahtjev 5

Identifikator	FZ-5
Zahtjev	Sustav će omogućiti brisanje zahtjeva iz popisa kreiranih zahtjeva.
Obrazloženje	Kako bi korisnik mogao maknuti zahtjev iz svojeg projekta, mora ga moći obrisati iz popisa kreiranih zahtjeva.
Način provjere	Klikom na zahtjev u popisu zahtjeva te zatim klikom na gumb za brisanje, zahtjev se briše iz popisa te više nije prikazan.
Prioritet [1-3]	2
Izvor	Programer – Zvonimir Belina

Tablica 15: Funkcionalni zahtjev 6

Identifikator	FZ-6
Zahtjev	Sustav će omogućiti pregled detalja i izmjenu napravljenih zahtjeva.
Obrazloženje	Kako bi korisnik mogao vidjeti cjelokupnost informacija zahtjeva i u slučaju potrebe promijenio te informacije, mora ga moći vidjeti u detaljnijem obliku.
Način provjere	Odabirom kreiranog zahtjeva na popisu i klikom na gumb za detalje otvara se forma s popunjenim informacijama zahtjeva koje se mogu mijenjati te spremiti.
Prioritet [1-3]	2
Izvor	Programer – Zvonimir Belina

Tablica 16: Funkcionalni zahtjev 7

Identifikator	FZ-7
Zahtjev	Sustav će omogućiti spremanje kreiranih zahtjeva u obliku tekstualne, CSV te PDF datoteke.
Obrazloženje	Kako bi korisnik svoje zahtjeve imao pohranjene na nekom trajnom mjestu, mora ih moći spremiti u nekom obliku.
Način provjere	Odabirom vrste datoteke i klikom na gumb za spremanje otvara se dijaloški okvir za spremanje te nakon unošenja naziva datoteke i odabira mjesta spremanja datoteka se pohranjuje.
Prioritet [1-3]	1
Izvor	Programer – Zvonimir Belina

5.6. Nefunkcionalni zahtjevi

5.6.1. Izgled softvera

- **NFZ-1** - Sustav će koristiti grafičko sučelje ravnog dizajna prilagođeno korisniku (eng. u*ser friendly*).

5.6.2. Upotrebljivost softvera

- **NFZ-2** - Sustav će imati provjere i upozorenja na svim relevantnim mjestima kako bi se smanjio broj mogućih grešaka,
- **NFZ-3** - Vrijeme potrebno da se korisnik osposobi za korištenje aplikacije ne smije prelaziti 15 minuta.

5.6.3. Performanse softvera

- **NFZ-4** – Sustav mora osigurati odziv na korisničke interakcije unutar 1 sekunde, osim provjere zahtjeva,
- **NFZ-5** – Kapacitet pohrane ovisi o kapacitetu diska računala na kojem je softver instaliran.

5.6.4. Izvođenje softvera i okruženje

- **NFZ-6** – Aplikacija mora biti kompatibilna s Windows operacijskim sustavima.

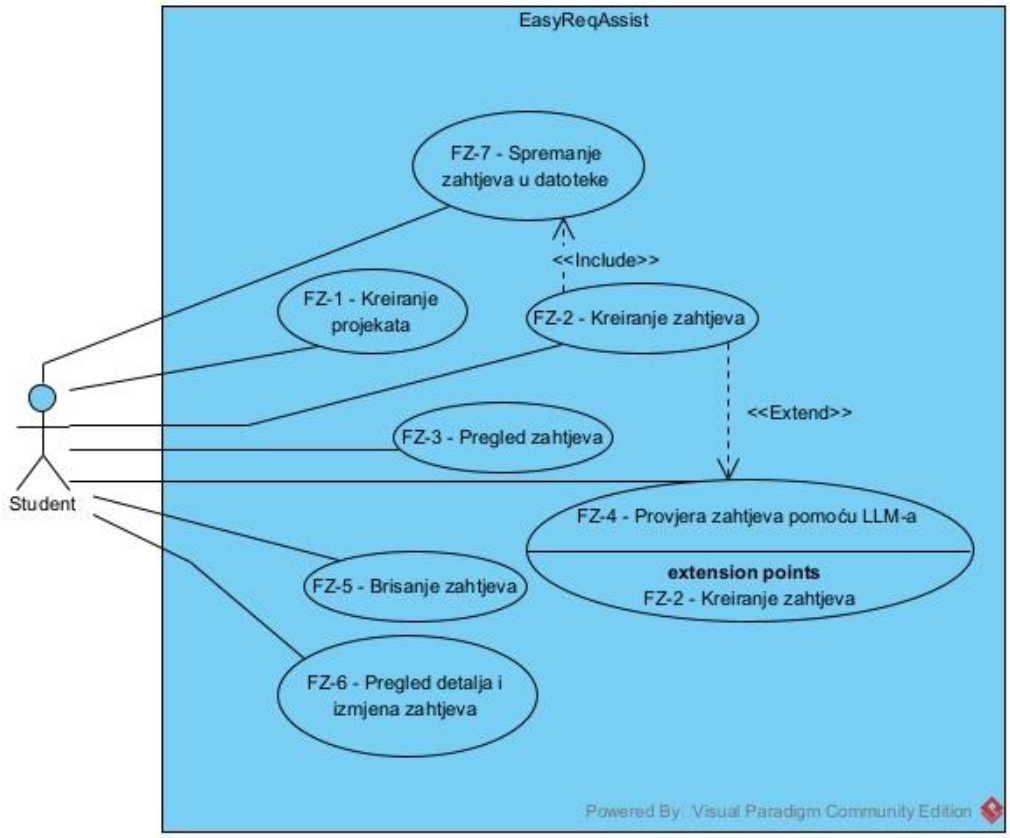
5.6.5. Sigurnost i privatnost

Nema nefunkcionalnih zahtjeva glede sigurnosti i privatnosti.

5.7. Dizajn softverskog sustava

5.7.1. Slučajevi korištenja

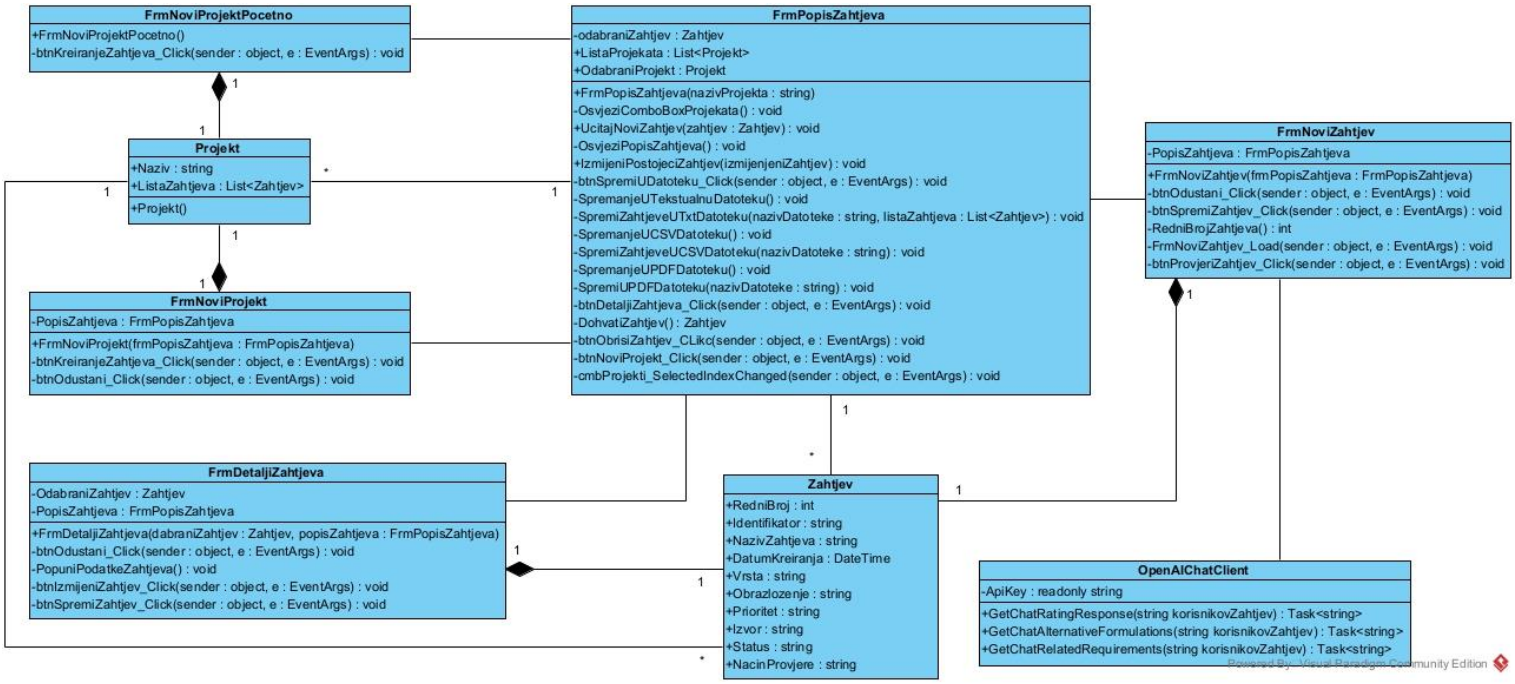
Za pregled ključnih funkcionalnosti sustava i ograničenja koriste se dijagrami slučajeva. Dijagrami slučajeva služe za pregled interakcija između sustava i njegovih korisnika (eng. *actors*). Iako slučajevi korištenja i korisnici u ovim dijagramima detaljno opisuju što sustav postiže i način korištenja od strane korisnika, oni ne ulaze u detalje funkcioniranje sustava. Na slici 2 prikazan je dijagram korištenja softverskog rješenja EasyReqAssist. Možemo vidjeti da slučaj korištenja FZ-2 – Kreiranje zahtjeva uključuje (eng. *include*) slučaj korištenja FZ-7 Spremanje zahtjeva u datoteke jer to podrazumijeva uobičajeno ponašanje korisnika aplikacije. Također, kreiranje zahtjeva može uključivati slučaj korištenja FZ-4 Provjera zahtjeva pomoću LLM-a zato što korisnik može birati želi li provjeriti zahtjev koji je napisao.



Slika 2: Dijagram korištenja cjelokupnog softvera EasyReqAssist (autorski rad)

5.7.2. Osnovna struktura softvera

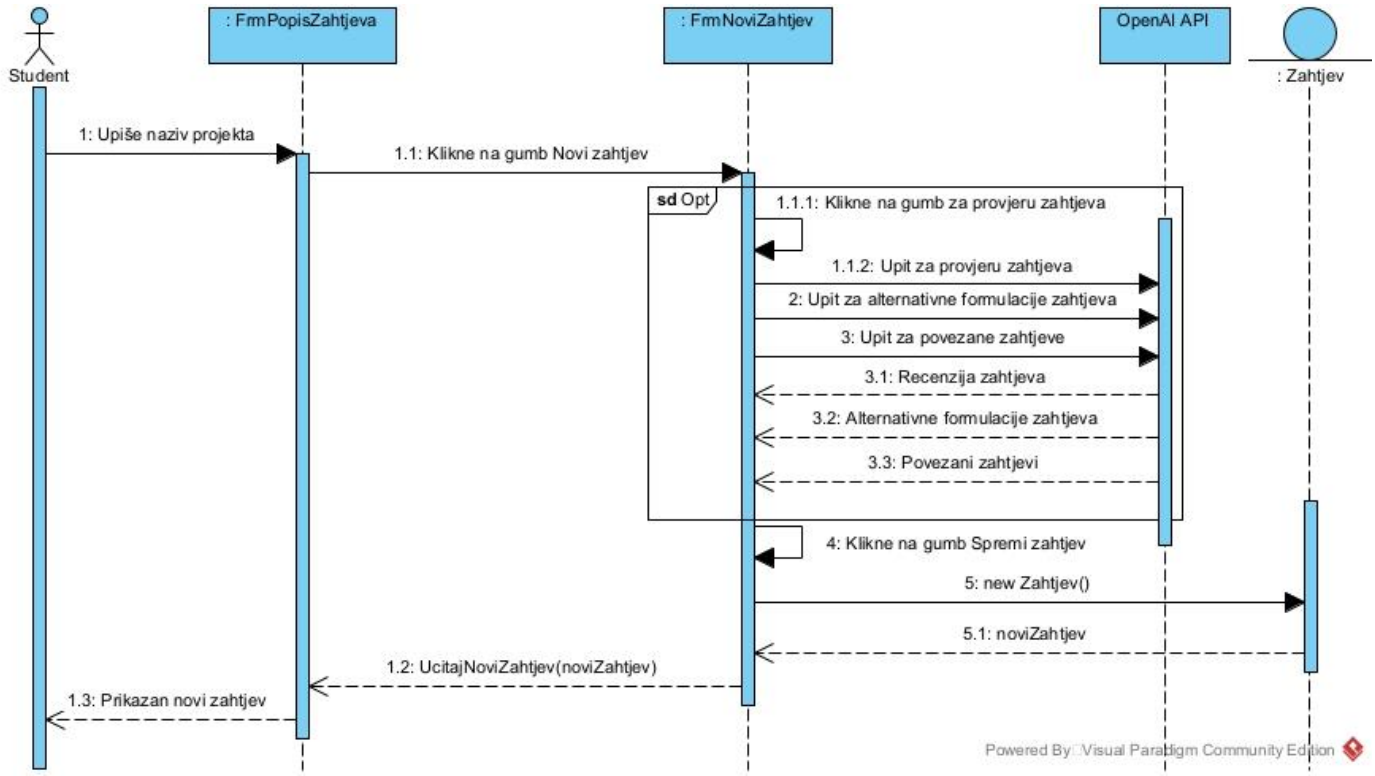
Dijagram klasa je UML dijagram koji se u praksi najviše koristi. Služi tome kako bi prikazao klase i objekte koji omogućavaju realizaciju funkcionalnosti softvera. Također, on prikazuje kako su dijelovi sustava povezani te vrstu tih veza. Na slici 3 prikazan je dijagram klasa softverskog rješenja EasyReqAssist.



Slika 3: Dijagram klasa cjelokupnog softvera EasyReqAssist (autorski rad)

5.7.3. Opis procesa stvaranja novog zahtjeva

Proces stvaranja novog zahtjeva bit će opisan koristeći dvije vrste dijagrama, dijagrama slijeda te dijagrama aktivnosti. Dijagram slijeda prikazuje ovisno o domeni modeliranja različite sustave, osobe, uređaje ili sl. kao paralelne vertikalne linije, koje su nekoj interakciji i preko horizontalnih strelica redom poruke koje razmjenjuju. Može se prikazivati npr. interakcija između osobe koja je došla u restoran i konobara, nekog sustava koji kontaktira neki servis ili računala koje komunicira s printerom, sve ovisi o domeni koja se modelira. Na slici 4 je prikazan dijagram slijeda za proces stvaranja novog zahtjeva u softverskom rješenju EasyReqAssist.

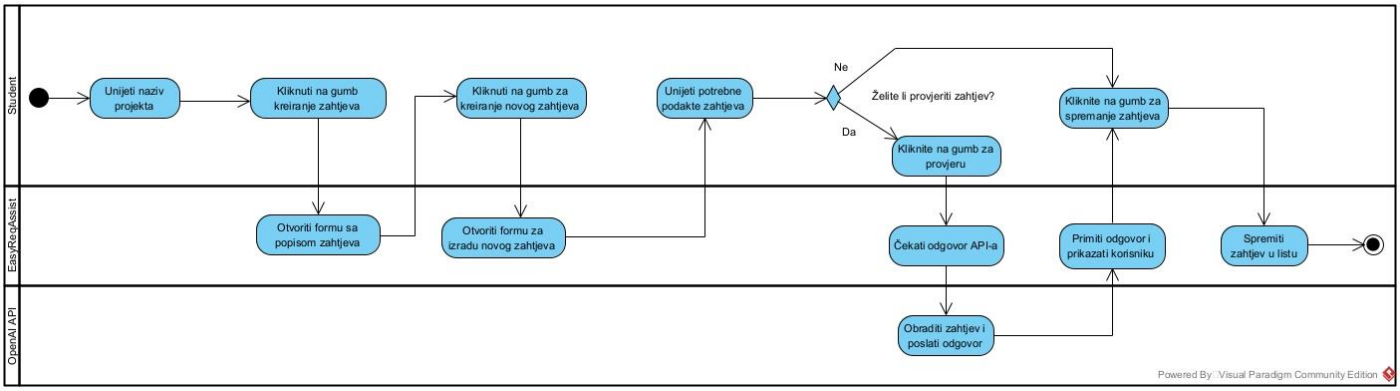


Slika 4: Dijagram slijeda za proces kreiranja novog zahtjeva (autorski rad)

Možemo vidjeti da se na ovom dijagramu slijeda nalazi pet sudionika (eng. *lifelines*), Student, entitet Zahtjev, dvije forme, FrmPopisZahtjeva, FrmNoviZahtjev i OpenAI API. Prvi korak je da student ili korisnik upiše naziv projekta za koji će kreirati zahtjeve što ga dovodi na formu FrmPopisZahtjeva, gdje se nalaze glavne funkcije softvera. Kako bi kreirao zahtjev klikne na gumb Novi zahtjev koji ga dovodi na formu FrmNoviZahtjev gdje će kreirati sve svoje zahtjeve. Prilikom pisanja relevantnih informacija za zahtjeve student može provjeriti sveobuhvatnost napisanog zahtjeva, pomoću umjetne inteligencije. Dakle, napisani zahtjev se šalje do bota za razgovor (eng. *chatbot*) ChatGPT-a koji onda na temelju INCOSE smjernica određuje može li se zahtjev poboljšati. Detalji implementacije bit će objašnjeni u narednim poglavljima. Ti prijedlozi se onda ispisuju studentu na ekranu te ih može iskoristiti kako bi poboljšao svoj zahtjev. Bilo da se student odluči provjeriti zahtjev ili ne, klikom na gumb za spremanje zahtjeva, zahtjev se pohranjuje u prije imenovani projekt te se prikazuje na listi zahtjeva.

Dijagram aktivnosti koristi se kako bi se vizualizirao tok aktivnosti i procesa unutar nekog sustava od početne do završne točke. Na ovakvom UML dijagramu prikazuju se akcije koje se događaju unutar procesa te čvorovi odlučivanja koji ovisno o ishodu stavlja tok na

drukčiji put. Na slici 5 je prikazan dijagram aktivnosti za proces stvaranja novog zahtjeva u softverskom rješenju EasyReqAssist. Pošto dijagram prikazuje isti proces, a to je kreiranje novog zahtjeva nema potrebe za ponovnim objašnjavanjem cijelog procesa. Ono što možemo vidjeti je dodatan korak recenzije zahtjeva. Klikom na gumb za provjeru, sustav poziva API uz proslijeđeni zahtjev te čeka na njegov odgovor, što omogućava korisniku da nesmetano radi u sustavu dok se njegov zahtjev ne recenzira. Nakon što API recenzira zahtjev, šalje odgovor sustavu, koji ga zatim prikazuje korisniku. Nakon spremanja zahtjeva, proces kreiranja zahtjeva završava.



Slika 5: Dijagram aktivnosti za proces kreiranja novog zahtjeva (autorski rad)

5.8. Implementacija rješenja

5.8.1. Operacije stvaranja, čitanja, ažuriranja i brisanja zahtjeva

CRUD operacije predstavljaju četiri bitne funkcionalnosti koje omogućavaju stvaranje, čitanje, ažuriranje i brisanje podataka ili u ovom slučaju zahtjeva u nekom sustavu. U ovom softveru operacije stvaranja i ažuriranja imaju zasebne forme, ***FrmNoviZahtjev*** i ***FrmDetaljiZahtjeva***, dok se čitanje i brisanje nalazi na formi ***FrmPopisZahtjeva***. Slijede isječci koda koji prikazuju navedene operacije, počevši od kreiranja. Metoda ***btnSpremiZahtjev_Click()*** nalazi se u formi za novi zahtjev te je to metoda koja se poziva na događaj klika na gumb ***btnSpremiZahtjev***. Na početku se nalazi uvjet koji provjerava je li napisan identifikator zahtjeva te sam zahtjev. Ako nisu, polja se zacrvene te nije moguće spremiti zahtjev, u suprotnom, kreira se novi zahtjev koji se sprema pomoću metode ***UcitajNoviZahtjev()*** varijable ***PopisZahtjeva***.

```
private void btnSpremiZahtjev_Click(object sender, EventArgs e)
{
    if(txtIdentifikator.Text == "" || txtZahtjev.Text == "")
    {
        MessageBox.Show("Morate upisati identifikator i zahtjev!",
"Upozorenje", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        txtIdentifikator.BackColor = Color.IndianRed;
        txtZahtjev.BackColor = Color.IndianRed;
    }
    else
    {
        Zahtjev noviZahtjev = new Zahtjev {
            RedniBroj = RedniBrojZahtjeva(),
            Identifikator = txtIdentifikator.Text,
            NazivZahtjeva = txtZahtjev.Text,
            DatumKreiranja = dtpDatumZahjteva.Value,
            Vrsta = txtVrsta.Text,
            Obrazlozenje = txtObrazlozenje.Text,
            Prioritet = txtPrioritet.Text,
            Izvor = txtIzvor.Text,
            Status = txtStatus.Text,
            NacinProvjere = txtNacinProvjere.Text
        };

        PopisZahtjeva.UcitajNoviZahtjev(noviZahtjev);
        Close();
    }
}
```

Prije samog spremanja moguće je provjeriti sveobuhvatnost zahtjeva pomoću umjetne inteligencije klikom na gumb ***btnProvjeriZahtjev***, kojim se poziva metoda na klik događaj tog gumba ***btnProvjeriZahtjev_Click***. Metoda sadrži provjeru je li zahtjev upisan u potrebno polje te ako prođe prelazi se na provjeru zahtjeva. Metoda ***OcistiTextBoxeve()*** briše tekst koji je bio napisan prije u slučaju ako korisnik želi ponoviti provjeru. Zatim, stvara se objekt ***chatClient*** klase ***OpenAIChatClient()*** o kojoj će biti više riječi u narednom poglavlju. Slijede tri varijable tipa „string“ koje spremaju odgovore za recenziju, alternativne formulacije zahtjeva te srodne zahtjeve, dobivene od metoda ***chatClient***-a. Pomoću posljednje metode ***DodajOdgovoreUTextBoxeve()*** ti se odgovori prikazuju korisniku u svojim poljima na formi.

```
private async void btnProvjeriZahtjev_Click(object sender, EventArgs e)
{
    if(txtZahtjev.Text == "")
    {
        MessageBox.Show("Morate napisati zahtjev kako bi se mogao
provjeriti!", "Upozorenje", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    else
    {
        OcistiTextBoxeve();

        var chatClient = new OpenAIChatClient();
        string odgovorRecenzija = await
chatClient.GetChatRatingResponse(txtZahtjev.Text);
        string odgovorAlternativa = await
chatClient.GetChatAlternativeFormulations(txtZahtjev.Text);
        string odgovorPovezaniZahtjevi = await
chatClient.GetChatRelatedRequirements(txtZahtjev.Text);

        DodajOdgovoreUTextBoxeve(odgovorRecenzija, odgovorAlternativa,
odgovorPovezaniZahtjevi);
    }
}
```

PopisZahtjeva je objekt koji predstavlja formu ***FrmPopisZahtjeva*** i njezinu metodu ***UcitajNoviZahtjev()***. Možemo vidjeti da se novokreirani zahtjev sprema unutar liste ***ListaZahtjeva*** objekta ***OdabraniProjekt***, gdje je ***OdabraniProjekt*** projekt unutar kojeg korisnik trenutno kreira zahtjeve. Metoda ***OsvjeziPopisZahtjeva()*** će uskoro biti detaljnije obrazložena, a ovdje služi tome da prikaz zahtjeva uvijek bude ažuriran.

```
public void UcitajNoviZahtjev(Zahtjev zahtjev)
{
    OdabraniProjekt.ListaZahtjeva.Add(zahtjev);
    OsvjeziPopisZahtjeva();
}
```

Na formi popisa zahtjeva obavljaju se operacije čitanja i brisanja zahtjeva. Čitanje zahtjeva obavlja se već preko spomenute metode ***OsvjeziPopisZahtjeva()***, a zahtjevi su

prikazani unutar ***DataGridView***-a. Prije svega, kako bi se spriječile greške imamo provjeru sadržaja liste zahtjeva odabranog projekta. Zatim, postavlja se izvor podataka ***DataGridView***-a na „null“, kako bismo resetirali sve zadane postavke prikaza i omogućili generiranje drugih u skladu s novim podacima. Slijedi sortiranje zahtjeva unutar liste prema rednom broju zahtjeva, postavljanje izvora podataka ***DataGridView***-a na sortiranu listu te metoda ***Refresh()*** istog kako bi se potvrdilo ispravno generiranje prikaza.

```
private void OsvjeziPopisZahtjeva()
{
    if (OdabraniProjekt.ListaZahtjeva != null)
    {
        dgvZahtjevi.DataSource = null;
OdabraniProjekt.ListaZahtjeva = OdabraniProjekt.ListaZahtjeva.OrderBy(z =>
z.RedniBroj).ToList();
        dgvZahtjevi.DataSource = OdabraniProjekt.ListaZahtjeva;
        dgvZahtjevi.Refresh();
    }
}
```

Brisanje zahtjeva obavlja se klikom na gumb ***btnObrisiZahtjev()*** te sljedeća metoda prikazuje događaj klika na taj gumb. Uvjet prije brisanja je da korisnik odabere zahtjev koji želi obrisati, inicijalizacijom varijable **odabraniZahtjev** preko metode ***DohvatiZahtjev()***, koja vraća zahtjev koji je odabran u prikazu ili „null“ ako ništa nije odabrano. Ovdje imamo dvije provjere koje obuhvaćaju slučaj praznog prikaza zahtjeva i neodabranog zahtjeva. Ako zahtjev postoji i odabran je, briše se iz liste zahtjeva pomoću metode ***Remove()***, nakon čega slijedi ažuriranje popisa zahtjeva.

```
private void btnObrisiZahtjev_Click(object sender, EventArgs e)
{
    odabraniZahtjev = DohvatiZahtjev();

    if (dgvZahtjevi.Rows.Count == 0)
    {
        MessageBox.Show("Nemate zahtjeva za obrisati!", "Upozorenje",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    else if (odabraniZahtjev == null)
```

```
    {
        MessageBox.Show("Morate odabrati zahtjev da biste ga mogli obrisati!", "Upozorenje", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    else
    {
        OdabraniProjekt.ListaZahtjeva.Remove(odabraniZahtjev);
        OsvjeziPopisZahtjeva();
    }
}

private Zahtjev DohvatiZahtjev()
{
    if (dgvZahtjevi.CurrentRow != null)
    {
        return dgvZahtjevi.CurrentRow.DataBoundItem as Zahtjev;
    }
    return null;
}
```

Pregled detalja zahtjeva i ažuriranje obavlja se na formi ***FrmDetaljiZahtjeva***. Uspješno otvaranje ove forme zahtijeva postojanje barem jednog zahtjeva, odabir željenog zahtjeva i klik na gumb za detalje zahtjeva. Otvaranjem forme tekstualna polja nije moguće odmah mijenjati, već je potrebno kliknuti na gumb ***btnIzmijeni***, kako bi se polja otključala za promjenu. Kada korisnik izmijeni informacije koje je htio, klikne na gumb ***btnSpremiZahtjev***. Metoda je jednaka metodi za prvotno spremanje zahtjeva, s razlikom u metodi koja se poziva iz objekta ***PopisZahtjeva***, a to je metoda ***IzmijeniPostojeciZahtjev()***.

```
private void btnSpremiZahtjev_Click(object sender, EventArgs e)
{
    if (txtIdentifikator.Text == "" || txtZahtjev.Text == "")
    {
        MessageBox.Show("Morate upisati identifikator i zahtjev!", "Upozorenje", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        txtIdentifikator.BackColor = Color.IndianRed;
        txtZahtjev.BackColor = Color.IndianRed;
    }
    else
```

```
    {
        Zahtjev azuriraniZahtjev = new Zahtjev {
            RedniBroj = OdabraniZahtjev.RedniBroj,
            Identifikator = txtIdentifikator.Text,
            NazivZahtjeva = txtZahtjev.Text,
            DatumKreiranja = dtpDatumZahjteva.Value,
            Vrsta = txtVrsta.Text,
            Obrazlozenje = txtObrazlozenje.Text,
            Prioritet = txtPrioritet.Text,
            Izvor = txtIzvor.Text,
            Status = txtStatus.Text,
            NacinProvjere = txtNacinProvjere.Text
        };

        PopisZahtjeva.IzmijeniPostojeciZahtjev(azuriraniZahtjev);
        Close();
    }
}
```

U metodi ***IzmijeniPostojeciZahtjev()*** prvo spremamo indeks zahtjeva kojeg smo odabrali iz popisa te nam on služi kako bismo pronašli taj isti zahtjev u listi te ga zamijenili, iza čega slijedi ažuriranje popisa zahtjeva.

```
public void IzmijeniPostojeciZahtjev(Zahtjev izmijenjeniZahtjev)
{
    int index = OdabraniProjekt.ListaZahtjeva.FindIndex(z => z.RedniBroj ==
odabraniZahtjev.RedniBroj);
    OdabraniProjekt.ListaZahtjeva[index] = izmijenjeniZahtjev;
    OsvjeziPopisZahtjeva();
}
```

5.8.2. Operacije spremanja zahtjeva u datoteke

Zahtjeve koje korisnik stvori za svoje projekte bi bilo korisno u nekom obliku spremiti. Za ovu implementaciju se ne koristi baza podataka, već se zahtjevi spremaju u obliku tekstualnih, CSV ili PDF datoteka. Kreirani zahtjevi mogu se spremiti na formi za popis zahtjeva. Klikom na gumb ***btnSpremiUDatoteku*** poziva se srodna metoda ***btnSpremiUDatoteku_Click()*** zaslužna za operaciju spremanja. Ako nema zahtjeva za spremiti, korisniku se prikazuje prikladno upozorenje. Odabirom jednog do troje radio gumba

korisnik odabire u koju vrstu datoteke želi spremiti zahtjeve. Ovisno o odabiru poziva se jedna od triju metoda za spremanje, ***SpremanjeUTekstualnuDatoteku()***, ***SpremanjeUCSVDatoteku()*** te ***SpremanjeuPDFDatoteku()***. Ove tri metode služe kao dijalog za spremanje datoteke pa su stoga gotovo jednake.

```
private void SpremanjeUTekstualnuDatoteku()
{
    SaveFileDialog saveDialog = new SaveFileDialog();
    saveDialog.Filter = "Tekstualne Datoteke (*.txt)|*.txt";
      // CSV Datoteke (*.csv)|*.csv, PDF Datoteke (*.pdf)|*.pdf

    if (saveDialog.ShowDialog() == DialogResult.OK)
    {
      // SpremiZahtjeveUCSVDatoteku(), SpremiUPDFDatoteku()
        SpremiZahtjeveUTxtDatoteku(saveDialog.FileName,
OdabraniProjekt.ListaZahtjeva);

      // Sukladno vrsti datoteke prikazuje se poruka
        MessageBox.Show("Zahtjevi su uspješno spremljeni u tekstualnu
datoteku!", "Spremanje završeno", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }
}
```

Slijede kratka objašnjenja metoda ***SpremiZahtjeveUTxtDatotek()***, ***SpremiZahtjeveUCSVDatoteku()*** i ***SpremiUPDFDatoteku()***. ***SpremiZahtjeveUTxtDatotek()*** ***k***oristi ***StreamWriter*** klasu kako bi se potrebne informacije napisale u nekom toku teksta (eng. *stream*). Pomoću petlje se prolazi kroz listu zahtjeva te se za svaki zahtjev piše linija teksta sukladno informaciji, no prije petlje se zapiše projekt kojem pripadaju zahtjevi.

```
private void SpremiZahtjeveUTxtDatoteku(string nazivDatoteke, List<Zahtjev>
listaZahtjeva)
{
    using (StreamWriter writer = new StreamWriter(nazivDatoteke))
    {
        writer.WriteLine("Projekt: " + OdabraniProjekt.Naziv);
        writer.WriteLine();
```

```
        foreach (Zahtjev zahtjev in listaZahtjeva)
        {
            writer.WriteLine("Identifikator: " + zahtjev.Identifikator);
            writer.WriteLine("Zahtjev: " + zahtjev.NazivZahtjeva);
            writer.WriteLine("Datum kreiranja: " + zahtjev.DatumKreiranja);
            writer.WriteLine("Vrsta zahtjeva: " + zahtjev.Vrsta);
            writer.WriteLine("Obrazloženje: " + zahtjev.Obrazlozenje);
            writer.WriteLine("Prioritet: " + zahtjev.Prioritet);
            writer.WriteLine("Izvor: " + zahtjev.Izvor);
            writer.WriteLine("Status: " + zahtjev.Status);
            writer.WriteLine("Način provjere: " + zahtjev.NacinProvjere);
            writer.WriteLine(); // Prazan red između svakog zahtjeva
        }
    }
}
```

SpremiZahtjeveUCSVDatoteku() koristi ***StringBuilder*** klasu, koja predstavlja promjenjiv niz znakova. Na početku se kao i prije svega napiše naziv projekta na kojem se radilo. Koriste se dvije petlje, u prvoj se dodaje zaglavlje popisa zahtjeva, a u drugoj redak za svaki redak na popisu te se u obje između vrijednosti dodaju zarezi kako bi to zapravo bila CSV datoteka. Naposljetku se sav sadržaj zapiše u datoteku.

```
private void SpremiZahtjeveUCSVDatoteku(string nazivDatoteke)
{
    StringBuilder csvSadrzaj = new StringBuilder();

    csvSadrzaj.AppendLine("Projekt: " + OdabraniProjekt.Naziv);
    csvSadrzaj.AppendLine();

    for (int i = 0; i < dgvZahtjevi.Columns.Count; i++)
    {
        csvSadrzaj.Append(dgvZahtjevi.Columns[i].HeaderText);
        if (i < dgvZahtjevi.Columns.Count - 1)
        {
            csvSadrzaj.Append(",");
        }
    }
    csvSadrzaj.AppendLine();
```

```
    foreach (DataGridViewRow redak in dgvZahtjevi.Rows)
    {
        for (int i = 0; i < dgvZahtjevi.Columns.Count; i++)
        {
            csvSadrzaj.Append(redak.Cells[i].Value);
            if (i < dgvZahtjevi.Columns.Count - 1)
            {
                csvSadrzaj.Append(",");
            }
        }
        csvSadrzaj.AppendLine();
    }
    File.WriteAllText(nazivDatoteke, csvSadrzaj.ToString());
}
```

SpremiUPDFDatoteku() koristi biblioteku **iTextSharp** kako bi se pomoću **PdfWriter** klase informacije zapisale u PDF datoteku. Započinje se s kreiranjem instance PDF dokumenta, inicijalizacijom **PdfWriter** klase za pisanje te otvaranje PDF dokumenta. Opet se dodaje naziv projekta prije svega, a zatim imamo ugniježđenu petlju. Vanjska petlja služi stvaranju nove tablice za svaki redak i postavljanje širine retka. Unutarnja petlja služi za popunjavanje podataka, gdje u lijevi stupac idu zaglavlja iz popisa, a u desni podaci, uz vertikalna i horizontalna poravnanja.

```
private void SpremiUPDFDatoteku(string nazivDatoteke)
{
    Document pdfDokument = new Document();
    PdfWriter.GetInstance(pdfDokument, new FileStream(nazivDatoteke,
FileMode.Create));


    pdfDokument.Open();
    pdfDokument.Add(new Paragraph("Projekt: " + OdabraniProjekt.Naziv));
    pdfDokument.Add(new Paragraph(" "));


    for (int i = 0; i < dgvZahtjevi.Rows.Count; i++)
    {
```

```
            PdfPTable tablica = new PdfPTable(2);
            tablica.WidthPercentage = 100;

            float[] sirineStupaca = new float[] { 20f, 80f };
            tablica.SetWidths(sirineStupaca);



            for (int j = 0; j < dgvZahtjevi.Columns.Count; j++)
            {
                PdfPCell celijaKategorija = new PdfPCell(new
Phrase(dgvZahtjevi.Columns[j].HeaderText));
                PdfPCell celijaPodatak = new PdfPCell(new Phrase(dgvZahtjevi[j,
i].Value.ToString()));

                celijaKategorija.HorizontalAlignment = Element.ALIGN_LEFT;
                celijaKategorija.VerticalAlignment = Element.ALIGN_MIDDLE;
                celijaKategorija.BackgroundColor = BaseColor.LIGHT_GRAY;

                celijaPodatak.HorizontalAlignment = Element.ALIGN_LEFT;
                celijaPodatak.VerticalAlignment = Element.ALIGN_MIDDLE;

                tablica.AddCell(celijaKategorija);
                tablica.AddCell(celijaPodatak);
            }

            pdfDokument.Add(tablica);
            pdfDokument.Add(new Paragraph(" "));
        }
        pdfDokument.Close();
    }
```

5.8.3. Detalji komunikacije s programskim sučeljem aplikacije OpenAI

Kako bismo mogli bolje razumjeti kako softversko rješenje komunicira s OpenAI API-em potrebno je detaljnije ući u funkcionalnosti sustava, odnosno kod. Sav kod koji slijedi nalazi se unutar klase ***OpenAIChatClient.cs***. Ova klasa sadrži tri metode i samo jednu iznimno važnu samo za čitanje (eng. *readonly*) varijablu, tipa „string“ s privatnim modifikatorom pristupa, naziva ***ApiKey***. Ova varijabla predstavlja ključ API-a s kojim se identificira i ovjerava aplikacija ili u ovom slučaju korisnik, kako bi se moglo komunicirati sa samim API-em. Zbog

osjetljivosti informacije koju sadrži, sadržaj varijable čita se iz varijabli okoline (eng. *Environment variable*) pod nazivom „OPENAI_API_KEY“.

```
private readonly string ApiKey =
Environment.GetEnvironmentVariable("OPENAI_API_KEY",
EnvironmentVariableTarget.User);
```

Sve naredne metode rađene su pomoću neslužbene C# .NET biblioteke „***OpenAI-API-dotnet***“ (https://github.com/OkGoDoIt/OpenAI-API-dotnet). Prije svega, bilo je potrebno instalirati ***NuGet*** paket ***OpenAI***. Slijedi metoda pomoću koje se izvršava komunikacija s API-em. Metoda je javna, asinkrona, čime je i tip varijable „Task<string>“ te naziva GetChatRatingResponse. Prima jedan argument tipa „string“, koji predstavlja zahtjev koji je korisnik upisao.

```
public async Task<string> GetChatRatingResponse(string korisnikovZahtjev)
{
    OpenAIAPI api = new OpenAIAPI(ApiKey);

    var chat = api.Chat.CreateConversation();

    chat.AppendSystemMessage("Vi ste iskusni mentor sistemskog inženjeringa
specijaliziran za softverske zahtjeve. Vaš cilj je pružiti točne smjernice
i pomoći studentu da napiše sveobuhvatne i usklađene softverske zahtjeve
slijedeći ove tri karakteristike INCOSE smjernica: " +
"C3 - Need statements must be written such that the stakeholder intent is
clear. Requirement statements must be stated such that the requirement can
be interpreted in only one way by all the intended readers. " +
"C4 - The requirement statement sufficiently describes the necessary
capability, characteristic, constraint, or quality factor to meet the need
without needing other information to understand the requirement. " +
"C5 - The stakeholder need or requirement statement should state a single
capability, characteristic, constraint, or quality factor.");

    chat.AppendUserInput("Molim provjeri mi ovaj zahtjev: " +
korisnikovZahtjev);

    string response = await chat.GetResponseFromChatbotAsync();

    return response;
}
```

Postoji više načina kako se autentificirati, no ja sam odabrao način gdje se API ključ proslijedi u konstruktor klase ***OpenAIAPI***, što se može vidjeti u prvoj liniji koda metode. OpenAIAPI ulazna je točka OpenAI API-a. Iza toga se stvara varijabla „chat“ koja predstavlja ChatGPT API s kojim se generira tekst u obliku razgovora te se kreira razgovor pomoću metode ***CreateConversation()***. Pomoću varijable chat sada možemo započeti razgovor. ***AppendSystemMessage()*** kreira i dodaje poruku razgovora s ulogom sustava. Proslijeđeni argument pomaže pri postavljanju ponašanja asistenta, odnosno sugovornika u razgovoru. S druge strane ***AppendUserInput()*** isto kreira i dodaje poruku razgovora s ulogom korisnika. Proslijeđena poruka je zahtjev koji je korisnik upisao, a pomaže pri upućivanju asistenta. Zatim imamo varijablu „response“ u koju se sprema odgovor dobiven preko metode ***GetResponseFromChatbotAsync()***, a metoda poziva API da vrati odgovor trenutnog razgovora. Naposljetku se ta varijabla vraća kako bi se odgovor mogao prikazati korisniku. Sljedeće dvije metode rađene su po istom principu, ali su jednostavno promijenjeni proslijeđeni argumenti metodama ***AppendSystemMessage()*** i ***AppendUserInput()***, sukladno tome što očekujemo kao odgovor.

Metoda za dobivanje alternativnih formulacija zahtjeva, **GetChatAlternativeFormulations()**.

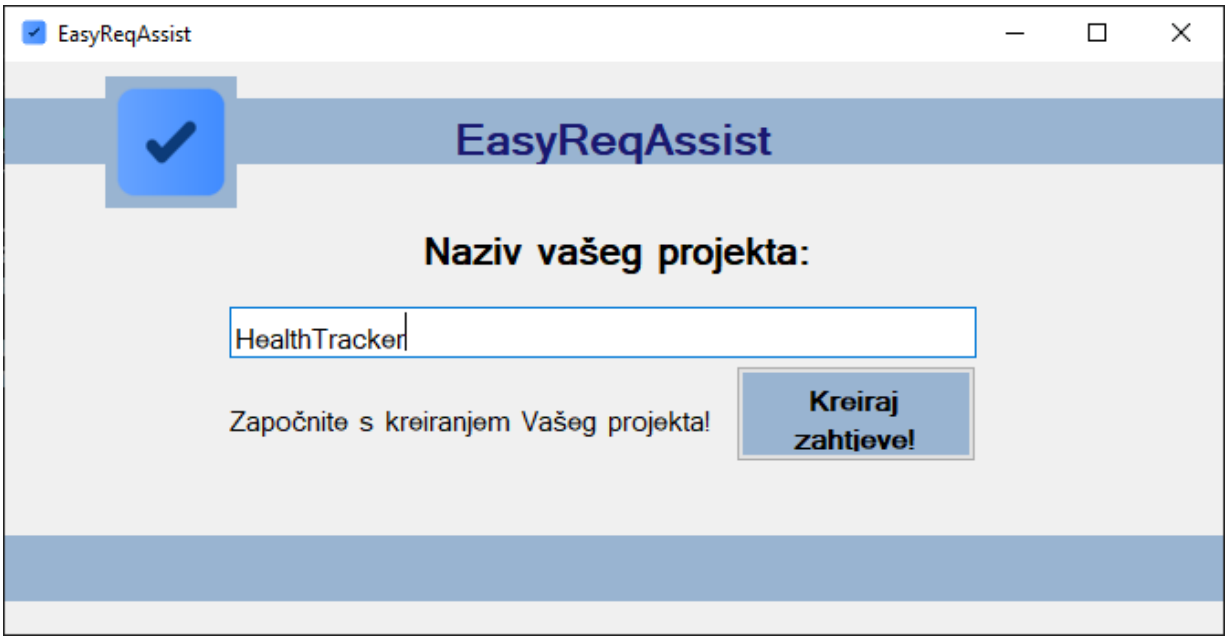
```
public async Task<string> GetChatAlternativeFormulations(string
korisnikovZahtjev)
{
    OpenAIAPI api = new OpenAIAPI(ApiKey);

    var chat = api.Chat.CreateConversation();

    chat.AppendSystemMessage("Vi ste iskusni mentor sistemskog inženjeringa
specijaliziran za softverske zahtjeve. Vaš cilj je pružiti točne smjernice
i pomoći studentu da napiše sveobuhvatne i usklađene softverske
zahtjeve.");

    chat.AppendUserInput("Napiši mi nekoliko alternativne formulacije za
sljedeći zahtjev: " + korisnikovZahtjev);

    string response = await chat.GetResponseFromChatbotAsync();

    return response;
}
```

Metoda za dobivanje zahtjeva koji su povezani s onim napisanim, ***GetChatRelatedRequirements()***.

```
public async Task<string> GetChatRelatedRequirements(string
korisnikovZahtjev)
{
    OpenAIAPI api = new OpenAIAPI(ApiKey);

    var chat = api.Chat.CreateConversation();

    chat.AppendSystemMessage("Vi ste iskusni mentor sistemskog inženjeringa
specijaliziran za softverske zahtjeve. Vaš cilj je pružiti točne smjernice
i pomoći studentu da napiše sveobuhvatne i usklađene softverske
zahtjeve.");

    chat.AppendUserInput("Napiši mi nekoliko drugih zahtjeva koji bi bili
povezani s ovim: " + korisnikovZahtjev);

    string response = await chat.GetResponseFromChatbotAsync();

    return response;
}
```

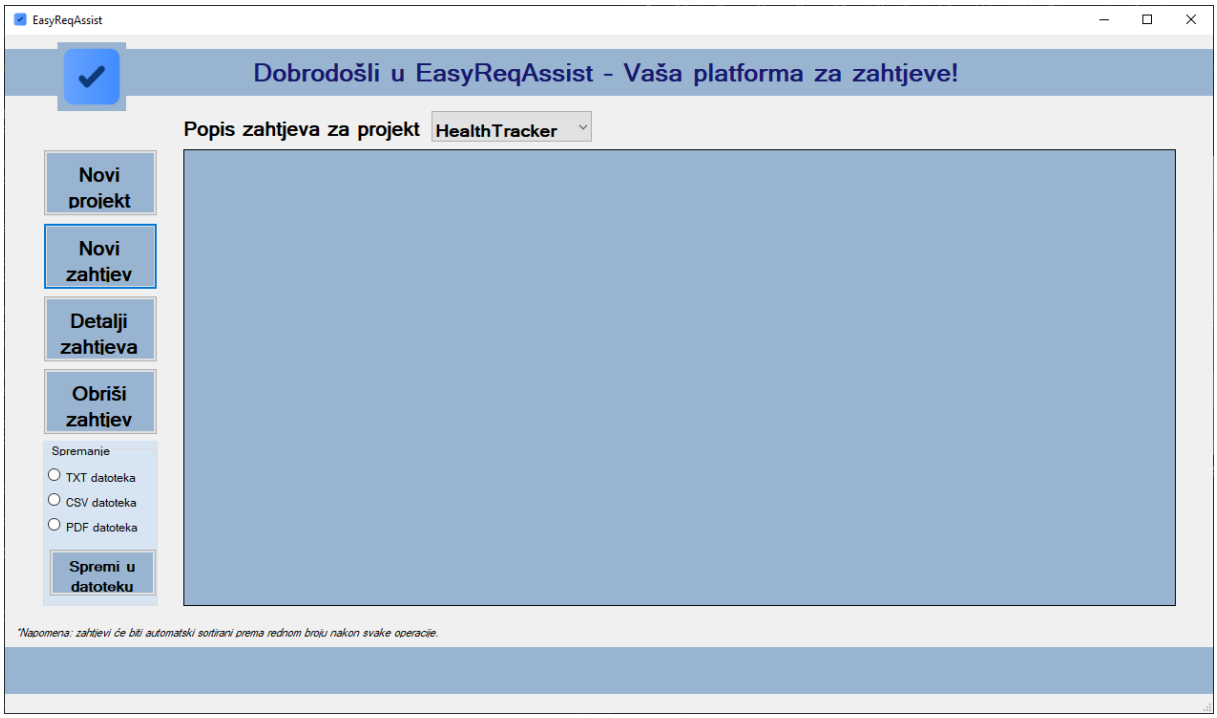
5.9. Demonstracija rješenja

U ovom poglavlju bit će prikazano nekoliko snimaka zaslona aplikacije kako bismo prikazali što aplikacija može i što radi. Počnimo sa zaslonom za upis naziva projekta koji se korisniku prikaže kada pokrene aplikaciju. Nazovimo projekt „HealthTracker“ i kliknimo na gumb „Kreiraj zahtjeve!“ kako bismo nastavili na sljedeći zaslon.



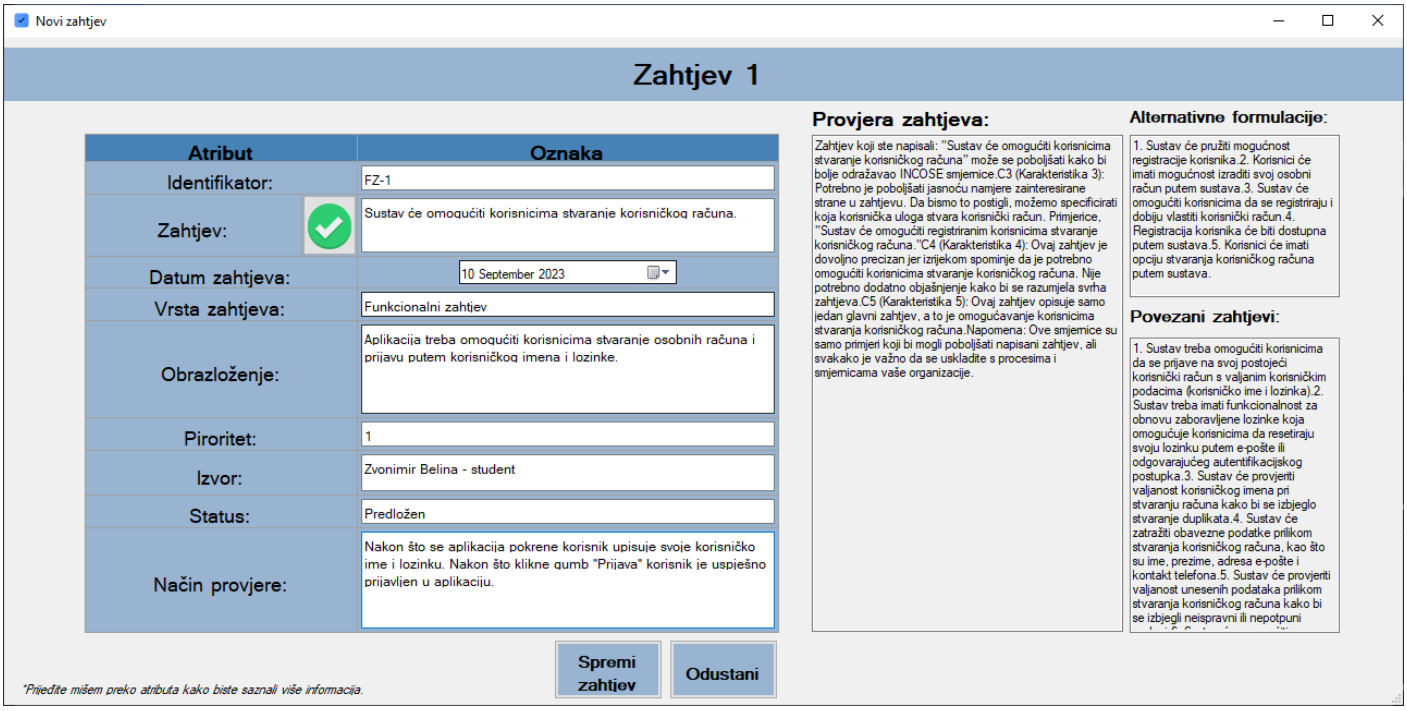
Slika 6: Početni zaslon softverskog rješenje EasyReqAssist (autorski rad)

Dolazimo na glavni zaslon gdje se odvija većina operacije aplikacije. Možemo vidjeti sve operacije koje se mogu izvoditi unutar aplikacije. Ako kliknemo na gumb „Novi zahtjev“ otvorit će nam se forma za kreiranje novog zahtjeva ovog projekta.



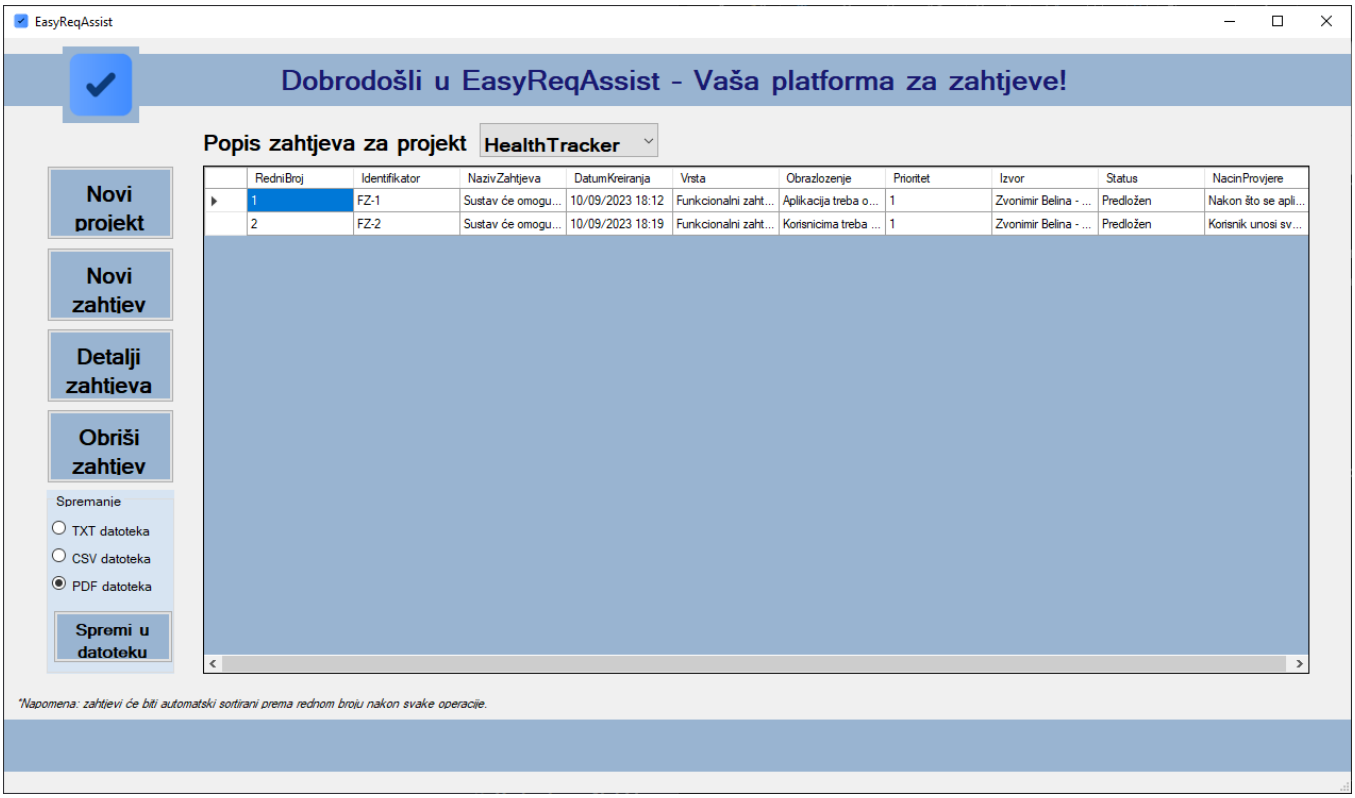
Slika 7: Glavni zaslon EasyReqAssist-a (autorski rad)

Otvaranjem forme za novi zahtjev možemo početi s kreiranjem novog zahtjeva. Ispunit ćemo zahtjev s potrebnim informacijama te ćemo provjeriti je li naš zahtjev sveobuhvatan pomoću umjetne inteligencije klikom na gumb sa zelenom kvačicom. Nakon toga ćemo spremiti zahtjev i napraviti još jedan.



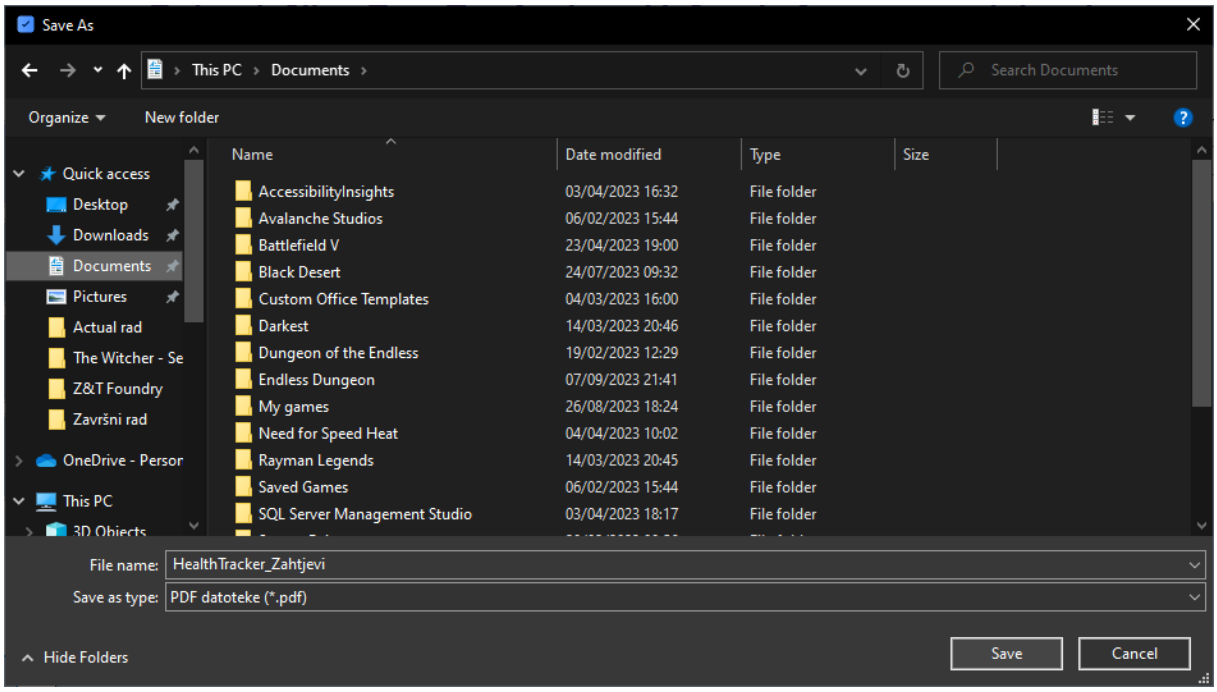
Slika 8: Zaslon kreiranja novog zahtjeva uz provjeru zahtjeva (autorski rad)

Nakon kreiranja oba zahtjeva možemo na popisu vidjeti da su nam oni pridruženi za projekt koji smo kreirali na početku. Sada ćemo spremiti kreirane zahtjeve u PDF datoteku odabirom na treći radio gumb te klikom na gumb „Spremi u datoteku“.



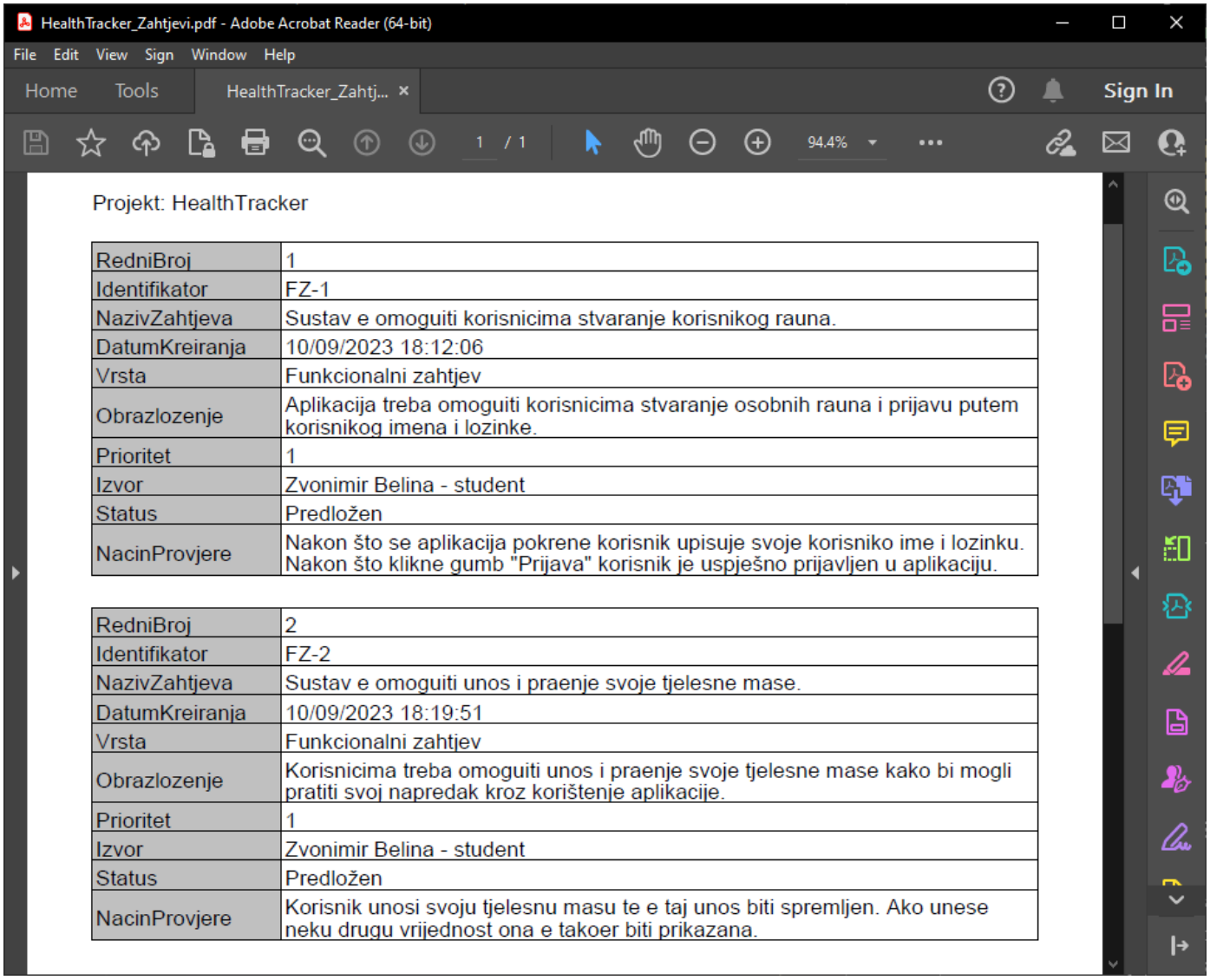
Slika 9: Glavni zaslon nakon kreiranja dvaju zahtjeva (autorski rad)

Otvara nam se dijalog za spremanje gdje odabiremo mjesto spremanja te naziv datoteke.



Slika 10: Dijalog zaslon za spremanje zahtjeva (autorski rad)

Ako otvorimo spremljenu PDF datoteku zahtjeva možemo vidjeti da su spremljeni u čitljivom formatu sa svim potrebnim informacijama koje smo ispunili u aplikaciji.



Projekt: HealthTracker

RedniBroj	1
Identifikator	FZ-1
NazivZahtjeva	Sustav e omoguiti korisnicima stvaranje korisnikog rauna.
DatumKreiranja	10/09/2023 18:12:06
Vrsta	Funkcionalni zahtjev
Obrazlozenje	Aplikacija treba omoguiti korisnicima stvaranje osobnih rauna i prijavu putem korisnikog imena i lozinke.
Prioritet	1
Izvor	Zvonimir Belina - student
Status	Predložen
NacinProvjere	Nakon što se aplikacija pokrene korisnik upisuje svoje korisniko ime i lozinku. Nakon što klikne gumb "Prijava" korisnik je uspješno prijavljen u aplikaciju.

RedniBroj	2
Identifikator	FZ-2
NazivZahtjeva	Sustav e omoguiti unos i praenje svoje tjelesne mase.
DatumKreiranja	10/09/2023 18:19:51
Vrsta	Funkcionalni zahtjev
Obrazlozenje	Korisnicima treba omoguiti unos i praenje svoje tjelesne mase kako bi mogli pratiti svoj napredak kroz korištenje aplikacije.
Prioritet	1
Izvor	Zvonimir Belina - student
Status	Predložen
NacinProvjere	Korisnik unosi svoju tjelesnu masu te e taj unos biti spremljen. Ako unese neku drugu vrijednost ona e takoer biti prikazana.

Slika 11: PDF dokument spremljenih zahtjeva (autorski rad)

6. Zaključak

Predmet istraživanja ovog rada bilo je inženjerstvo zahtjeva i razvoj alata za podršku aktivnostima specificiranja softverskih zahtjeva. Fokusirali smo se na definiciju, proces definiranja, što uključuje elicitaciju, analizu, specifikaciju, provjeru i upravljanje zahtjevima. Posebno smo istražili smjernice za pisanje zahtjeva uključujući karakteristike zahtjeva, karakteristike skupa zahtjeva, pravila za definiranje zahtjeva i skupova zahtjeva te atribute zahtjeva. Također smo proučili generički proces za inženjering zahtjeva, uključujući idealan razvoj i razvoj s promjenama.. Osim toga, istražili smo postojeće alate za upravljanje zahtjevima kao što su QVScribe, ReqSuite® RM, RQA – QUALITY Studio®, ScopeMaster®, i ReqView.

U praktičnom dijelu rada, razmotrili smo opseg, perspektivu proizvoda, karakteristike korisnika, ograničenja, te definirali funkcionalne i nefunkcionalne zahtjeve za razvoj softverskog sustava. Prikazali smo dizajn softverskog sustava koji uključuje dijagrame slučaja korištenja, klasa, slijeda za proces stvaranja novog zahtjeva, aktivnosti za isti proces te detalje implementacije CRUD operacija, spremanja zahtjeva i komunikacije s programskim sučeljem aplikacije OpenAI. Softversko rješenje je napravljeno u jeziku C# .NET okviru u programu Visual Studio.

Jasno definirani zahtjevi te praćenje i verifikacija zahtjeva tijekom životnog ciklusa projekta ključni su za uspješan razvoj softverskih sustava i postizanje zadovoljstva korisnika. Postoji mnogo alata na tržištu koji mogu olakšati upravljanje zahtjevima, ali odabir odgovarajućeg alata ovisi o specifičnim potrebama projekta. Izrada dobro definiranih zahtjeva ključna je za uspješan razvoj softverskih sustava i postizanje zadovoljstva korisnika.

Popis literature

Dick, J., Hull, E., Jackson, K. (2017). *Engineer Requirements process in context of changes, Requirements Engineering Fourth edition*. Cham, Švicarska: Springer Nature.

Dick, J., Hull, E., Jackson, K. (2017). *Requirements Engineering Fourth edition*. Cham, Švicarska: Springer Nature.

Institue of Electrical and Electronical Engineers [IEEE]. (1998). *IEEE Std. 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.*

02DCE (2023). *Software Engineering | Requirements Engineering Process – GeeksForGeeks*. Preuzeto 28.6.2023. s https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/

02DCE (2023). *Software Engineering | Classification of Software requirements – GeeksForGeeks*. Preuzeto 20.6.2023. s https://www.geeksforgeeks.org/software-engineering-classification-of-software-requirements/

Mijač M. (2023). *Analiza i specifikacija zahtjeva (2.dio).*

Mijač M. (2023). *Analiza i specifikacija zahtjeva – Strukturiranje specifikacije.*

QRA Corp. (bez dat.). *FAQs – QRA*. Preuzeto 18.07.2023. s https://qracorp.com/support/faqs/#gen1

Robertson, J., Robertson, S. (2012). *Volere Requirements Specification Template Edition 16-2012.*

Ryan, M., Wheatcraft, L., Zinni, R., Dick, J., Baksa, K. (2019). *Guide for Writing Requirements*. San Diego, CA, USA: International Council on Systems Engineering [INCOSE].

ScopeMaster Ltd. (2023.) *Pricing - Scopemaster.* Preuzeto 20.7.2023. s https://www.scopemaster.com/pricing/

ScopeMaster Ltd. (2023.) *Software Requirements Analysis, QA and Sizing – automated*. Preuzeto 20.7.2023. s https://www.scopemaster.com

Software Advice, Inc. (2023). *ReqSuite® RM Software Reviews, Demo & Pricing - 2023*. Preuzeto 18.07.2023. s https://www.softwareadvice.com/manufacturing/reqsuite-rm-profile/

Software Advice, Inc. (2023). *ReqView Software Reviews, Demo & Pricing - 2023*. Preuzeto 20.07.2023. s https://www.softwareadvice.com/requirements-management/reqview-profile/

The REUSE Company (bez dat.). *RQA – QUALITY Studio*. Preuzeto 18.07.2023. s https://www.reusecompany.com/rqa-quality-studio

Wiegers, K., Beatty, J. (2013). *Software Requirements, Third Edition*. Redmond, Washington: Microsoft Press.

Popis slika

Popis tablica