2. Analiza i specifikacija zahtjeva

2.1. Strukturiranje specifikacija zahtjeva

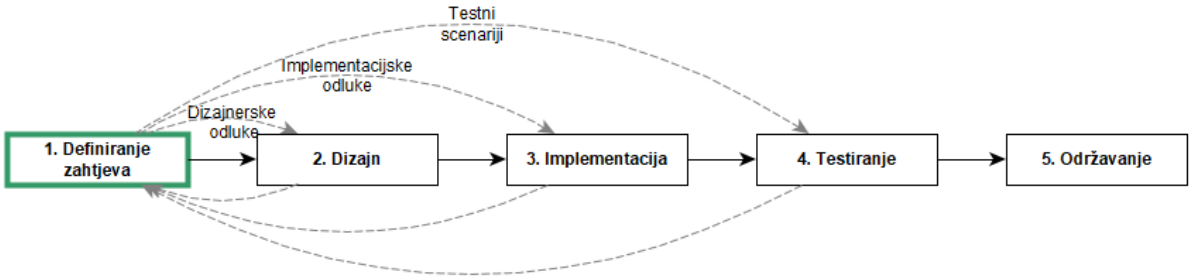
Sažetak

Jedna od prvih faza u razvoju softvera usmjerena je na analizu i specifikaciju zahtjeva koje korisnici imaju prema softveru, odnosno zahtjeva koje budući softver mora realizirati. Dobro definirani zahtjevi su temelj svakog softvera, jer ukazuju da dobro razumijemo sam problem, ali i da imamo jasna očekivanja od budućeg softverskog rješenja. Ukoliko su zahtjevi krivo ili loše definirani, niti najbolji alati nam neće pomoći da izradimo za korisnika prihvatljiv softver. Područje, ali i proces koji se bavi prikupljanjem, analizom, specifikacijom, te provjerom zahtjeva naziva se inženjerstvo zahtjeva. U ovoj lekciji obraditi ćemo osnovne koncepte povezane s inženjerstvom zahtjeva, s posebnim fokusom na izradu strukturirane specifikacije softverskih zahtjeva.

Uvod

Zahtjevi (engl. *requirements*) su temelj svakog softvera. Oni opisuju što softver treba moći napraviti kako bi ispunio potrebe korisnika softvera. Dobro definirani zahtjevi ukazuju da razumijemo problemsku domenu, da smo svjesni potreba i očekivanja korisnika, te da imamo jasnu viziju što softver treba raditi kako bi bio koristan. Istraživanja pokazuju da su jasno definirani zahtjevi jedan od najizrazitijih faktora uspjeha softverskih projekata. Već u samom začetku softverskog projekta potrebni su nam kako bismo mogli provesti procjenu izvodljivosti, planiranje projekta, upravljanje rizicima, dodjelu i prioritizaciju zadataka, definiranje kriterija prihvaćanja softvera, i sl. S druge strane, prema istim istraživanjima, nepotpuni i nerealni zahtjevi su neki od najčešćih razloga zašto softverski projekti propadaju. Naime, ukoliko softver ne ispunjava potrebe korisnika, neće niti imati korisnika, te je sukladno tome beskoristan. To nažalost nije moguće ispraviti niti korištenjem najmodernijih alata i tehnologija, niti primjenom najboljih praksi u dizajnu, implementaciji ili testiranju softvera. Unatoč kritičnoj važnosti zahtjeva, često se upravljanju zahtjeva ne posvećuje dovoljno pažnje.

U tradicionalnim procesima razvoja softvera (npr. vodopadni model), za definiranje zahtjeva je rezervirana prva faza softverskog procesa. Unatoč tome, zahtjevi nipošto nisu izolirani od ostalih tradicionalnih faza i aktivnosti softverskog procesa. Naprotiv, oni usmjeravaju aktivnosti donošenja dizajnerskih i implementacijskih odluka, kao i definiranje scenarija za testiranje softvera. Nerijetko dođe i do povratne veze, pa provedba aktivnosti dizajna, implementacije i testiranja rezultira novim spoznajama koje rezultiraju identificiranjem novih ili izmjenom postojećih zahtjeva.



Slika 1 Uloga zahtjeva u procesu razvoja softvera

Definiranje zahtjeva je sastavni dio procesa razvoja softvera čak i kada kao aktivnost nije provedena i dokumentirana eksplicitno, a možda čak niti svjesno. U većini slučajeva zahtjevi su ipak eksplicitno definirani s manjom ili većom razinom detalja, te na manje ili više formalan i sofisticiran način. Područje i proces koji se bavi sustavnim prikupljanjem, analizom, specifikacijom, te provjerom zahtjeva naziva se inženjerstvo zahtjeva (engl. *requirements engineering*). Iako ga gledamo kao područje unutar softverskog inženjerstva, inženjerstvo zahtjeva nije rezervirano isključivo za kontekst izgradnje softvera, nego se bavi zahtjevima sustava općenito.

Proces definiranja zahtjeva (engl. *requirements process*)

Elicitacija zahtjeva

Elicitacija zahtjeva je aktivnost koja ima za cilj identificirati zahtjeve koje buduće softversko rješenje treba ispuniti. U ovoj aktivnosti je od izuzetne važnosti dobra komunikacija sa dionicima softvera, jer su upravo oni i njihove potrebe glavni izvor zahtjeva i ograničenja softverskog rješenja. Dionici (engl. *stakeholders*) su pojedinci, grupe ili organizacije koji imaju izravan ili neizravan interes prema softverskom rješenju. To mogu biti svi oni koji zbog uspjeha ili neuspjeha softverskog rješenja mogu ostvariti dobit ili gubitak (ne nužno financijski), te svi oni koji su odgovorni za nastajanje i korištenje softvera. To uključuje: menadžere koji će odobriti i upravljati projektom razvoja softvera, investitore koji će financirati razvoj softvera, korisnike koji će softver koristiti, tehničko osoblje koje će instalirati i servisirati softver, edukatore koji će osigurati ispravno korištenje softvera, i dr. Zbog njihove uključenosti u sam proces razvoja softvera ili kasnije korištenje softvera, dionici su jedan od glavnih izvora softverskih zahtjeva. Podatke od dionika možemo prikupiti provođenjem intervjua, upitnika, fokus grupa, radionica, te promatranjem korisnika u poslovnom okruženju. Neki od problema koji se pri tome mogu javiti su npr. da dionici nisu u potpunosti svjesni svojih potreba, da teško artikuliraju svoje potrebe, i imaju nerealne ili konfliktne zahtjeve.

Osim preko dionika, do zahtjeva možemo doći i analizom tržišta, poslovne dokumentacije, softverskih rješenja u uporabi ili sličnih softverskih rješenja, prijavljenih problema i sugestija vezanih uz postojeća softverska rješenja, prototipa i sl.

Specifikacija zahtjeva

Kada su identificirani zahtjevi koje buduće softversko rješenje treba ispuniti, potrebno ih je dokumentirati na jasan, lako razumljiv, konzistentan i nedvojben način. Upravo je to cilj specifikacije zahtjeva kao aktivnosti. Prilikom dokumentiranja zahtjeva, tj. izrade specifikacije najčešće se oslanjamo na korištenje prirodnog jezika kako bismo imali fleksibilnost i bogatstvo izričaja. Ipak, da bismo osigurali da specifikacija zahtjeva bude korektno strukturirana i da sadrži sve potrebne informacije, oslanjamo se na različite smjernice (npr. EARS ili MOSCOW) te gotove predloške (npr. Volere, ili IEEE 830-1998 SRS). Uz prirodni jezik, za specificiranje zahtjeva se mogu koristiti i pristupi s višom razinom strukture (npr. UML dijagramske tehnike), te formalni i matematički pristupi (npr. VDM ili Z jezik).

Provjera zahtjeva

Jedna od aktivnosti koju provodimo prilikom provjere zahtjeva je tzv. *validacija* zahtjeva. Ona ima za cilj utvrditi odražavaju li specificirani zahtjevi potrebe i očekivanja dionika, te jesu li zahtjevi specificirani na korektan način (npr. jesu li jasni, precizni, izvedivi i sl.). Validacija će osigurati da imamo ispravan skup zahtjeva koji će rezultirati softverom po mjeri korisnika. Ukoliko propustimo validirati zahtjeve u ranim fazama procesa razvoja softvera, cijena ispravka pogrešnog zahtjeva u kasnijim fazama će biti višestruko veća. Konačnu validaciju specificiranih zahtjeva će provesti sami korisnici prije primopredaje izrađenog softvera.

Osim validacije, za provjeru zahtjeva je bitna i aktivnost *verifikacije*. Ona ima za cilj provjeriti jesu li kasnije faze procesa razvoja softvera u potpunosti i na adekvatan način realizirale specificirane zahtjeve. Drugim riječima, verifikacije treba pokazati da su svi specificirani zahtjevi uzeti u obzir prilikom dizajna softvera, da su adekvatno implementirani i pokriveni testovima. U tu svrhu se često izrađuje tzv. matrica praćenja (engl. *traceability matrix*), koja mapira svaki zahtjev iz specifikacije sa ostalim softverskim artefaktima kao što su UML dijagrami, programski kôd i testni scenariji.

Tablica 1 Matrica praćenja

Zahtjev	Dizajn	Implementacija	Testiranje
FZ-1	UML: UC#1, DC#1, DSeq#1	Klasa X, metoda X1, komponenta 2	Jedinični testovi: #1-#12 Testovi prihvaćanja: AT#1,#2
FZ-2	UML: UC#1, DAct#1, DC#2	Klasa Y, metoda Y1, komponenta 2	Jedinični testovi: #14-#18 Testovi prihvaćanja: AT#3-#5
…	…	…	…

Definiranje strukture specifikacije zahtjeva

Ključni rezultati provedbe procesa definiranja zahtjeva su dokumenti specifikacija korisničkih zahtjeva (engl. user requirements specification - USR) i specifikacija softverskih zahtjeva (engl. software requirements specification - SRS). Specifikacija korisničkih zahtjeva sadrži zahtjeve više razine apstrakcije koji opisuju korisničke potrebe i očekivanja koje korisnik ima prema sustavu. Obično je pisana u obliku prirodnog jezika uz upotrebu terminologije iz problemske domene. Upotreba tehničkih izraza je svedena na minimum kako bi dokument bio razumljiv naručitelju, krajnjim korisnicima, menadžerima i drugim ne-tehničkim dionicima koji svi skupa predstavljaju ciljanu skupinu čitatelja dokumenta. Samu specifikaciju korisničkih zahtjeva može izraditi isporučitelj softvera na temelju analizirane dokumentacije, istraživanja tržišta i komunikacije s dionicima. Međutim, nerijetko (a pogotovo u visoko-reguliranim domenama kao što je javni sektor) i sam naručitelj izrađuje specifikaciju korisničkih zahtjeva ukoliko ima tehnička znanja. Specifikacija softverskih zahtjeva proširuje ono što je specificirano u korisničkim zahtjevima na način da detaljno opisuje što bi softver trebao raditi kako bi ispunio potrebe i očekivanja korisnika. Često se na temelju jednog korisničkog zahtjeva definira više softverskih zahtjeva. Iako je i specifikacija softverskih zahtjeva uobičajeno pisana prirodnim jezikom, s obzirom da cilja pretežito na tehnički potkovane dionike (programere, softverske arhitekte, testere i sl.) ne izbjegava se upotreba tehničke terminologije i dijagramskih tehnika. Pri tome je potrebno voditi računa da se specifikacija softverskih zahtjeva zadrži na definiranju što softver treba raditi, a ne kako. Drugim riječima informacije vezane uz dizajn i implementaciju softvera ne trebaju biti dio specifikacije softverskih zahtjeva.

+

Slika 2 Ciljane skupine specifikacija korisničkih i softverskih zahtjeva

Kako bi specifikacija zahtjeva bila izrađena kvalitetno, treba sadržavati informacije potrebne ciljanoj publici, te organizirati te informacije u konzistentan i dobro strukturiran dokument. Takva organizacija zahtjeva i njima povezanih informacija pomaže u boljem razumijevanju zahtjeva, minimiziranju broja zahtjeva, identificiranju duplikata, eliminiranju konflikata između zahtjeva te provjeri kompletnosti skupa zahtjeva. Za organizaciju i strukturiranje dokumenta specifikacije zahtjeva se vrlo često koriste predlošci (u originalu ili prilagođeni za vlastitu upotrebu) kao što je Volere predložak ili IEEE 830-1998 SRS predložak.

Predložak temeljen na IEEE 830-1998 dokumentu

Za potrebe ove lekcije definirali smo predložak za specifikaciju softverskih zahtjeva temeljen na dokumentu IEEE 830-1998. Sva poglavlja predloška te upute kako ih napisati se mogu pronaći u sljedećoj tablici.

Tablica 2 Predložak za izradu specifikacije zahtjeva

Rbr.	Naziv poglavlja	Upute
1.	**Uvod**	
1.1.	Svrha	Opišite svrhu SRS dokumenta (ne softverskog rješenja). Navedite kome je namijenjen dokument, tj. tko treba dokument čitati i razumjeti.
1.2.	Opseg	Ukratko objasnite problemsku domenu kojom se rješenje bavi. U kojem kontekstu će se softversko rješenje upotrebljavati? Dajte ime/naziv softverskom rješenju kojim će ga se u ostatku dokumenta moći naslovljavati, te po potrebi verziju. Objasnite radi li se o potpuno novom rješenju ili nadogradnji. Navedite što će softver raditi i što neće raditi. Koje dobrobiti i unaprjeđenja očekujemo da će softversko rješenje donijeti.
1.3.	Definicije, akronimi i skraćenice	Definirajte osnovne pojmove, akronime i skraćenice koji se koriste u specifikaciji, a koje su potrebne da bi se moglo ispravno razumjeti SRS dokument.
1.4.	Reference	Referencirajte sve dokumente, web stranice, standarde, druge specifikacije koji se spominju u SRS-u.
1.5.	Struktura dokumenta	Ukratko opišite kako je ostatak dokumenta organiziran i što sadrži.

2.	**Općeniti opis**	
2.1.	Perspektiva proizvoda	Stavite softversko rješenje u kontekst i odnos s drugim povezanim sustavima. Navedite je li softversko rješenje neovisno i u potpunosti samostalno, ili predstavlja dio većeg sustava (što je čest slučaj), ili je zamjena za neki postojeći sustav. Ako je dio većeg sustava potrebno je staviti u odnos zahtjeve cjelokupnog sustava sa zahtjevima našeg softverskog rješenja, te jasno definirati sučelja između njih. Osim odnosa s eventualnim nadređenim sustavom, navesti i vanjske sustavima i softver koji vaše softversko rješenje koristi. To mogu biti npr.: DMBS, operacijski sustav, web servisi i API-ji. U slučaju da softversko rješenje izravno koristi hardver i komunikacijske tehnologije opišite i značajke sučelja s njima (vrsta hardvera, port, komunikacijski protokoli, bluetooth, NFC, IC, TCP, format razmjene podataka i sl.).
2.2.	Funkcije proizvoda	Navedite glavne funkcije koje softversko rješenje treba sadržavati, bez ulaženja u detalje svake od tih funkcija (detaljni zahtjevi će biti sadržani u poglavlju 3). Funkcije trebaju biti organizirane i opisane na način da već pri prvom čitanju budu razumljive svim čitateljima dokumenta, uključujući i naručitelja/korisnika. Za formulaciju funkcija softverskog rješenja na ovoj razini može poslužiti specifikacija više razine, npr. specifikacija korisničkih zahtjeva (ako postoji).
2.3.	Karakteristike korisnika	Identificirajte grupe/uloge korisnika za koje očekujemo da će koristiti softversko rješenje (npr. nastavnik, profesor, administrator, blagajnica, bankovni službenik, računovođa, …). Navedite opće karakteristike identificiranih korisnika, navedite po čemu se oni razlikuju. To može uključivati razinu obrazovanja, iskustva, računalne i tehničke pismenosti, zatim predviđenu učestalost korištenja softverskog rješenja, razinu dodijeljenih dozvola, skupa funkcija softvera kojeg će koristiti.
2.4.	Ograničenja	Navesti aspekte problemske domene i samog rješenja koji mogu djelovati ograničavajuće na razvoj softverskog rješenja. To može uključivati: zakonske i korporativne propise i regulative (npr. GDPR odredbe, fiskalizacija, sigurnosne politike, i sl.); hardverska ograničenja (npr. u razvoju za male uređaje – baterija, memorija, procesor, povezivost, i sl.); potrebu za prilagodbom drugim sustavima (npr. interakcija s postojećim i potencijalno zastarjelim sustavima); sigurnosnu kritičnost aplikacije (npr. u slučaju da aplikacija upravlja opasnim strojevima, medicinskim uređajima, ili radi sa osjetljivim i povjerljivim podacima, i sl.); zahtjeve za pouzdanošću (npr. može biti zahtijevano korištenje određenog pristupa, alata praksi i standarda u programiranju, modeliranju i testiranju softverskog rješenja i sl.)
2.5.	Pretpostavke i ovisnost	Navedite pretpostavke i otvorena pitanja (za razliku od poznatih činjenica) čiji naknadni ishod može utjecati na zahtjeve navedene u ovom dokumentu. To su vrlo često okolnosti koje nisu pod našom odgovornosti (npr. ako postoji vjerojatna mogućnost ili najava promjene zakonske regulative koja će rezultirati modificiranjem definiranih zahtjeva; ili izlazak nove verzije API-ja o kojem naš softver ovisi).
2.6.	Ostalo	Navedite ostale aspekte problemske domene i budućeg softverskog rješenja koje smatrate bitnima a koji nisu predviđeni u ostalim dijelovima dokumenta.
3.	**Funkcionalni zahtjevi**	Definirajte funkcionalne zahtjeve za softversko rješenja i to na način da pružite dovoljno informacija dizajnerima i programerima da mogu započeti sa osmišljavanjem i implementacijom rješenja, a testerima da osmisle testne slučajeve. Zahtjevi navedeni ovdje se temelje na funkcijama proizvoda opisanim u poglavlju 2.2., ali su opisani s višom razinom detalja. Fokus je na funkciji i ograničenjima sustava. Svakom zahtjevu treba dodijeliti jedinstven identifikator. Zahtjeve je moguće i

<table>
<tr><td></td><td></td><td>grupirati po različitim kriterijima, kao npr. korisnicima (nastavnik, student, blagajnik, administrator…), slučaju korištenja, domenskim konceptima, i sl.
Za svaki zahtjev potrebno je ispuniti sljedeću tablicu:

<table>
<tr><td>Identifikator</td><td>Jedinstveni identifikator zahtjeva.</td></tr>
<tr><td>Zahtjev</td><td>Opis zahtjeva u obliku „Sustav će omogućiti <funkcionalnost> <objekt> uz <ograničenja>“. Stil pisanja treba biti ujednačen. Prilikom formulacije vodite se prihvaćenim smjernicama za definiranje zahtjeva, kao npr. onima navedenim u INCOSE Smjernicama za pisanje zahtjeva.</td></tr>
<tr><td>Obrazloženje</td><td>Obrazloženje zašto zahtjev postoji/zašto je potreban.</td></tr>
<tr><td>Način provjere</td><td>Kriterij provjere ili testni scenarij koji će omogućiti utvrđivanje je li zahtjev ispunjen ili nije.</td></tr>
<tr><td>Prioritet [1-5]</td><td>Prioritet zahtjeva (1 – najveći prioritet, 5 najmanji prioritet)</td></tr>
<tr><td>Izvor/Porijeklo</td><td>Naziv dokumenta kojim je zahtjev propisan ili dionika koji je podnio zahtjev.</td></tr>
</table>
</td></tr>
<tr><td>3.1.</td><td>Dinamika realizacije zahtjeva</td><td>Navedite hoće li svi zahtjevi biti realizirani u inicijalnoj verziji softvera, ili će neki biti ostavljeni za buduće verzija. Navedite ukoliko postoje još neki zahtjevi koje se planira realizirati u budućnosti.</td></tr>
<tr><td>4.</td><td>Nefunkcionalni zahtjevi</td><td></td></tr>
<tr><td>4.1.</td><td>Izgled softvera</td><td>Navedite zahtjeve (ukoliko postoje) koji su povezani s izgledom i vizualnim stilom softvera. To može uključiti zahtjeve da se npr. koristi formalan i korporativan stil (u slučaju poslovne aplikacije), ili zaigran stil (npr. računalna igra za djecu); zahtjev za korištenjem konkretne palete boja ili kontrola kako bi aplikacija bila u skladu s brendiranjem poduzeća i sl.</td></tr>
<tr><td>4.2.</td><td>Upotrebljivost softvera</td><td>Navedite zahtjeve (ukoliko postoje) koji su povezani s lakoćom učenja i korištenja softvera, prilagodbom za osobe s poteškoćama, lokalizacijom. To može uključiti zahtjeve vezane uz krivulju učenja, brzinu korištenja aplikacije (npr. brzina unosa podataka), lakoću pamćenja opcija softvera, frekvenciju grešaka koje korisnik napravi u radu sa softverom, mogućnost odabira jezika, mogućnost korištenja od strane slijepih osoba i sl.</td></tr>
<tr><td>4.3.</td><td>Performanse softvera</td><td>Navedite zahtjeve (ukoliko postoje) koji su povezani s performansama softvera. To može uključiti zahtjeve vezane uz brzinu procesiranja zadataka, odziv, preciznost rezultata, kapacitet pohrane, skalabilnost, dostupnost sustava i sl.</td></tr>
<tr><td>4.4.</td><td>Izvođenje softvera i okruženje</td><td>Navedite zahtjeve (ukoliko postoje) koji su povezani s izvođenjem softvera i okruženjem u kojem se softver izvodi. To može uključivati zahtjeve vezane uz fizičko okruženje u kojem se nalazi sustav (glasno okruženje, jako osvjetljenje, prašina i sl.), drugi postojeći sustavi unutar kojih se softver treba izvoditi ili s kojima treba imati interakciju (operacijski sustav, drugi softver od kojeg preuzimamo podatke ili ih šaljemo).</td></tr>
<tr><td>4.5.</td><td>Sigurnost i privatnost</td><td>Navedite zahtjeve (ukoliko postoje) koji su povezani sa pitanjima sigurnosti i privatnosti podataka, te standardima i propisima vezanima uz tu problematiku. To može uključivati zahtjeve za korištenjem propisanih sigurnosnih procedura, tehnologija, usklađenost sa pravnim okvirima i sl.</td></tr>
</table>

4.6.	Ostalo	Navedite ostale nefunkcionalne zahtjeve koji nisu prethodno navedeni.
5.	**Skice zaslona**	Vizualizirajte značajke interakcije između krajnjeg korisnika i softverskog rješenja kroz skice zaslona (engl. wireframe). Svrha skica je da na vizualan način predočimo i komuniciramo što aplikacija treba raditi, a ne da izradimo realan dizajn grafičkog sučelja. Pri tome skice možete na bilo koji način nacrtati (npr. ručno na papiru + slikanje s mobitelom, MS Paint, MS Word, Excel…)

Praktični primjer

Ovo potpoglavlje demonstrira izradu dokumenta specifikacije softverskih zahtjeva na praktičnom primjeru softvera za evidentiranje rezultata provedbe kontinuiranog praćenja na kolegiju Programsko inženjerstvo. Praktični primjer započinje kratkim opisom problemske domene danim od strane naručitelja (nastavnika na kolegiju), koji potencijalnim isporučiteljima softverskog rješenja (studentima na kolegiju) omogućava da se upoznaju sa problemskom domenom i potrebama korisnika. Na temelju opisane problemske domene pristupa se izradi dokumenta specifikacije softverskih zahtjeva. Sama struktura dokumenta je preuzeta iz predloška definiranog u prethodnom poglavlju. Potrebno je napomenuti da u ovoj lekciji izrađujemo samo prva dva poglavlja specifikacije, tj. Uvod i Općeniti opis. Preostali dio specifikacije softverskih zahtjeva koji uključuje funkcionalne i nefunkcionalne zahtjeve će biti obrađen u narednoj lekciji.

Opis problemske domene

Prilikom izvođenja nastave na kolegiju Programsko Inženjerstvo, nastavnici u okviru kontinuiranog praćenja na kolegiju Programsko inženjerstvo (2.g. IPS-a) ocjenjuju studente po više elemenata praćenja. S obzirom na broj studenata (očekivano oko 200) i broj elemenata praćenja (trenutno 5), broj individualnih ocjenjivanja doseže brojku 1000. Svako od tih ocjenjivanja rezultira brojem bodova koje nastavnici moraju evidentirati kako bi na kraju semestra mogli dati zaključnu ocjenu studentu.

Nastavnici trenutno evidenciju provode korištenjem tabličnog kalkulatora, međutim zbog niza nedostataka i problema koji se javljaju, željeli bi preći na za to posebno namijenjen softver. Neki od problema su: zahtjevna procedura postavljanja tabličnog kalkulatora za svaku akademsku godinu, pojedine radnje se teško automatiziraju, nedostatak automatiziranih izvještaja, složena prilagodba u slučaju izmjena u modelu praćenja i sl.

S obzirom da su podaci vezani uz rezultate vrednovanja osjetljivi (privatni), svakako je potrebno ograničiti pristup evidenciji tako da samo nastavnici mogu pristupiti. Same elemente praćenja i uvjete koje studenti moraju ostvariti nastavnici bi željeli moći definirati u slučaju potrebe za promjenom modela praćenja. U istom slučaju bi trebalo moći definirati i bodovnu skalu.

Na početku akademske godine svakako želimo evidentirati studente koji su upisani u kolegij. Pri tome je poželjno imati i mogućnost skupnog bržeg unosa studenata.

Nastavnici žele biti u mogućnosti evidentirati bodove studenata, i to za svaki definirani element praćenja pri čemu se mora znati koji je nastavnik unio bodove. A na kraju, na temelju ostvarenih

bodova i bodovne skale, želimo dobiti prijedlog ocjene za studenta. Naravno, ta ocjena je samo prijedlog, a nastavnik sam unosi ocjenu koju smatra ispravnom.

S obzirom da je prije otvaranja roka za kontinuirano praćenje nužno evidentirati zabranu potpisa u ISVU sustav, nastavnicima je potreban popis studenata koji nisu ostvarili uvjete za potpis. A u konačnici, za potrebe upisa ocjene/provedbe usmenog ispita na roku za kontinuirano praćenje nastavnicima je potreban prikaz detaljnih rezultata kontinuiranog praćenja za sve studente koji su ostvarili pravo na potpis.

Trenutačni model praćenja kolegija Programsko inženjerstvo propisuje ukupno 5 elemenata praćenja (vidi Tablica 3.). Za svaki element praćenja je definiran maksimalan broj bodova koji je moguće ostvariti, broj bodova koje je potrebno ostvariti kao uvjet za potpis, te broj bodova koje je potrebno ostvariti za ocjenu. Kao što je vidljivo, 1. i 2. teorijski kolokvij nisu uvjeti za potpis, dok sve tri zadaće jesu. S druge strane, svih pet elemenata praćenja su uvjet za ocjenu.

Tablica 3 Elementi praćenja kolegija Programsko inženjerstvo

Elementi praćenja	Max. bodovi	Uvjet za potpis	Uvjet za ocjenu
1. teorijski kolokvij	25	0	11
2. teorijski kolokvij	25	0	11
Zadaća 1	15	6	6
Zadaća 2	15	6	6
Zadaća 3	20	8	8
Ukupno	**100**		

U slučaju da student ostvari barem minimalan potreban broj bodova iz svakog elementa praćenja, ocjena se predlaže na temelju ukupnog broja ostvarenih bodova primjenom sljedeće bodovne skale:

Tablica 4 Bodovna skala za kolegij Programsko inženjerstvo

Bodovi od	Bodovi do	Ocjena
0	49	1
50	60	2
61	75	3
76	90	4
91	100	5

Specifikacija softverskih zahtjeva za Evaluation Manager

1. UVOD

1.1. Svrha

Ovaj dokument predstavlja specifikaciju softverskih zahtjeva za softver namijenjen evidentiranju rezultata provedbe kontinuiranog praćenja na kolegiju Programsko inženjerstvo. Specifikacija zahtjeva je izrađena na temelju inicijalnih korisničkih zahtjeva dostavljenih od strane nastavnika na kolegiju. Ciljana skupina specifikacije zahtjeva su projekt menadžeri koji će upravljati dinamikom izrade rješenja, dizajneri i programeri koji trebaju osmisliti i implementirati softversko rješenje, te testeri koji će provjeriti da rješenje uistinu ispunjava postavljene zahtjeve. Osim kao podloga za daljnji razvoj softverskog rješenja, ovaj dokument ima i svrhu ugovora između naručitelja (nastavnika na kolegiju Programsko inženjerstvo) i izvođača (odabranog poduzeća), pa ciljana skupina čitatelja uključuje i naručitelje.

Sama struktura dokumenta se temelji na predlošku definiranom u dokumentu *IEEE 830-1998 Recommended Practice for Software Requirements Specifications*.[1]

1.2. Opseg

Prilikom izvođenja nastave na kolegiju Programsko Inženjerstvo, nastavnici u okviru kontinuiranog praćenja na kolegiju Programsko inženjerstvo (2.g. IPS-a) vrednuju studente po definiranim elementima praćenja. S obzirom na broj upisanih studenata i broj definiranih elemenata praćenja, pojedinačnih vrednovanja je često više od 1000. Prilikom svakog vrednovanja nastavnik evidentira broj bodova koje je student ostvario. Po završetku kontinuiranog praćenja ostvareni bodovi iz pojedinačnih elemenata praćenja se zbrajaju, te se provjerava je li student ostvario pravo na potpis i ocjenu.

Tablica 5 Elementi praćenja kolegija Programsko inženjerstvo

Elementi praćenja	Max. bodovi	Uvjet za potpis	Uvjet za ocjenu
1. teorijski kolokvij	25	0	11
2. teorijski kolokvij	25	0	11
Zadaća 1	15	6	6
Zadaća 2	15	6	6
Zadaća 3	20	8	8
Ukupno	**100**		

Trenutačni model praćenja kolegija Programsko inženjerstvo propisuje ukupno 5 elemenata praćenja (vidi Tablica 5). Za svaki element praćenja je definiran maksimalan broj bodova koji je moguće ostvariti,

[1] http://ieeexplore.ieee.org/servlet/opac?punumber=5841

broj bodova koje je potrebno ostvariti kao uvjet za potpis, te broj bodova koje je potrebno ostvariti za ocjenu. Smatra se da student nije ostvario uvjet za potpis ili ocjenu ukoliko iz bilo kojeg elementa praćenja nije ostvario broj bodova dovoljan za potpis ili ocjenu. Kao što je vidljivo, 1. i 2. teorijski kolokvij nisu uvjeti za potpis, dok sve tri zadaće jesu. S druge strane, svih pet elemenata praćenja su uvjet za pozitivnu ocjenu.

U slučaju da student ostvari pravo na potpis, ocjena se predlaže na temelju ukupnog broja ostvarenih bodova primjenom sljedeće bodovne skale:

Tablica 6 Bodovna skala za kolegij Programsko inženjerstvo

Bodovi od	Bodovi do	Ocjena
0	49	1
50	60	2
61	75	3
76	90	4
91	100	5

Nastavnici trenutno evidenciju provode korištenjem tabličnog kalkulatora, međutim zbog niza nedostataka i problema koji se javljaju, željeli bi preći na za to posebno namijenjen softver. Neki od problema su: zahtjevna procedura postavljanja tabličnog kalkulatora za svaku akademsku godinu, pojedine radnje se teško automatiziraju, nedostatak automatiziranih izvještaja, složena prilagodba u slučaju izmjena u modelu praćenja i sl.

S obzirom na opisani problem i očekivanja korisnika, predlaže se izrada novog softverskog rješenja nazvanog Evaluation Manager, koje bi zamijenilo postojeću evidenciju rezultata kontinuiranog praćenja izrađenu u tabličnom kalkulatoru. Potrebno je naglasiti da Evaluation Manager obuhvaća isključivo kontinuirano praćenje, tj. evidentira rezultate samo onih studenata koji su odabrali model praćenja redovitih studenata (tzv. Model A). Praćenje studenata koji nisu dio kontinuiranog praćenja, tj. onih koji su odabrali model praćenja izvanrednih studenata (tzv. Model B) nije predviđeno ovim softverskim rješenjem. Također, softversko rješenje ne obuhvaća redovite niti izvanredne ispitne rokove nakon roka za kontinuirano praćenje.

1.3. Definicije, akronimi i skraćenice

- *Model praćenja* – model po kojem student pohađa nastavu tokom semestra. Definira zaduženja i aktivnosti koje student treba odraditi, te način na koji će se odrađene aktivnosti vrednovati.
- *Element praćenja* – pojedinačna aktivnost koju student treba odraditi samostalno ili u timu, a koja se boduje od strane nastavnika (npr. kolokvij, zadaća).
- *Uvjet za potpis* – kriterij nad pojedinim elementom praćenja koji mora biti zadovoljen kako student ostvario pravo na potpis iz kolegija.
- *Uvjet za ocjenu* – kriterij nad pojedinim elementom praćenja koji mora biti zadovoljen kako bi student ostvario pravo na ocjenu iz kolegija.
- *ISVU* – Informacijski sustav visokih učilišta koji fakultet koristi kao potporu prilikom provedbe poslovnih procesa vezanih uz nastavu.

1.4. Reference

1. "830-1998 - IEEE Recommended Practice for Software Requirements Specifications." IEEE, 1998. [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=5841
2. Dokument opisa problemske domene
3. Model praćenja kolegija Programsko inženjerstvo, 2022, [Online]. Available: https://nastava.foi.hr/course/214467

1.5. Struktura dokumenta

U *poglavlju 2* Evaluation Manager stavljamo u kontekst i opisujemo interakciju s korisnicima ali i drugim sustavima, softverskim rješenjima, hardverom, i komunikacijskim tehnologijama. Zatim, na sažet način opisujemo osnovne funkcije koje će Evaluation Manager izvršavati, karakteristike korisnika koji će koristiti softver, te ograničenja koja mogu utjecati na sam razvoj softverskog rješenja.

U *poglavlju 3* definiramo funkcionalne zahtjeve za Evaluation Manager na onoj razini detalja koja je dovoljna dizajnerima i programerima da započnu sa osmišljavanjem i implementacijom rješenja, te testerima da osmisle testne slučajeve.

U *poglavlju 4* definiramo nefunkcionalne zahtjeve za Evaluation Manager koje dizajneri i programeri trebaju uzeti u obzir prilikom osmišljavanja arhitekture, odabira implementacijskih tehnologija i pristupa.

U *poglavlju 5* vizualiziramo način interakcije korisnika s Evaluation Manager-om na način da skiciramo grafičko korisničko sučelje.

2. OPĆENITI OPIS

2.1. Perspektiva proizvoda

Evaluation Manager je zamišljen kao samostalno softversko rješenje koje je zamjena za postojeći sustav (temeljen na tabličnom kalkulatoru) evidentiranja rezultata kontinuiranog praćenja. Softversko rješenje bi trebalo sadržavati klijentsku aplikaciju koja će se izvoditi na računalnu krajnjeg korisnika, dok bi baza podataka bila centralizirana zbog potrebe dijeljenja podataka između nastavnika. Izravna interakcija sa drugim sustavima koji nisu sastavni dio Evaluation Manager-a nije predviđena. S obzirom da je predviđen uvoz podataka o upisanim studentima iz ISVU sustava, postoji neizravna ovisnost prema ISVU sustavu s obzirom na format datoteke i podataka koje ISVU može isporučiti.

Evaluation Manager nema potrebu izravnog korištenja hardverskih ili komunikacijskih tehnologija. Predviđeno je da bilo kakva uporaba takvih resursa bude odrađena od strane operacijskog sustava ili "runtime" okruženja.

2.2. Funkcije proizvoda

Budući korisnici softverskog rješenja Evaluation Manager očekuju od softvera sljedeće mogućnosti:

- Ograničavanje pristupa evidenciji.
- Definiranje modela praćenja kolegija.
- Unos studenata koji su upisani u kolegij.
- Evidentiranje ostvarenih bodova studenta.
- Predlaganje ocjene za studenta.
- Ispis rezultata kontinuiranog praćenja.

2.3. Karakteristike korisnika

Korisnici koji će koristiti softversko rješenje Evaluation Manager su nastavnici na kolegiju Programsko inženjerstvo. S obzirom da će svi nastavnici koristiti softver na isti način i imati istu razinu prava, možemo reći da postoji jedna korisnička uloga – *nastavnik*. Ipak, od svakog nastavnika će biti zatraženo da se prijavi u aplikaciju prije korištenja sa vlastitim korisničkim podacima, kako bismo mogli razlikovati koje rezultate je evidentirao svaki od nastavnika. Svi nastavnici posjeduju naprednu razinu računalne i tehničke pismenosti.

2.4. Ograničenja

S obzirom da softversko rješenje Evaluation Manager podrazumijeva rad sa privatnim rezultatima studenata (ostvareni rezultati na kontinuiranom praćenju) koji su podložni GDPR odredbama,

potrebno je osigurati da samo ovlaštene osobe (nastavnici kolegija) imaju pristup navedenim podacima. S obzirom na problemsku domenu i karakteristike rješenja, nije uočeno postojanje dodatnih ograničenja: ne radi se o sigurnosno kritičnoj domeni; interakcija s drugim sustavima nije izravna i ne utječe u značajnoj mjeri na razvoj samog softverskog rješenja; hardverske karakteristike današnjih računala su više nego dovoljne za rad sa softverom. Od izvođača se očekuje da razvoj softverskog rješenja Evaluation Manager bude u skladu sa dobrim praksama struke, međutim, naručitelj ne postavlja specifična ograničenja s obzirom na metodološki pristup, alate i tehnologiju izrade.

2.5. Pretpostavke i ovisnosti

Model praćenja rada studenata u okviru kontinuiranog praćenja je unaprijed definiran i nije podložan promjenama u tekućoj akademskoj godini, stoga se ne očekuju izvanredne izmjene zahtjeva. Također, u vremenskom periodu izrade softverskog rješenja Evaluation Manager se ne očekuju takve izmjene u odabranoj tehnologiji kakve bi izazvale promjene na razini softverskih zahtjeva.

2.6. Ostalo

Nema potrebe za elaboracijom dodatnih aspekata.

Pitanja

1. Što je to zahtjev?
2. Čime se bavi inženjerstvo zahtjeva?
3. Što je to dionik (engl. stakeholder) u kontekstu softverskih zahtjeva? Tko sve može biti dionik?
4. Koje su osnovne aktivnosti procesa definiranja zahtjeva?
5. Koja je razlika između korisničkih zahtjeva i softverskih zahtjeva?
6. Zašto je bitno specificirati zahtjeve?
7. Na koji način su zahtjevi povezani sa ostalim aktivnosti procesa razvoja softvera?
8. Zašto je važno imati strukturiran dokument specifikacije zahtjeva?