Primjena UML jezika i korisničkih priča u izradi specifikacije web portala

Klarić, Marijan

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij

PRIMJENA UML JEZIKA I KORISNIČKIH PRIČA U IZRADI SPECIFIKACIJE WEB PORTALA

Završni rad

Marijan Klarić

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Osijek, 14.06.2022.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Ime i prezime Pristupnika:	Marijan Klarić
Studij, smjer:	Preddiplomski stručni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	AR4668, 25.07.2018.
OIB Pristupnika:	82604163870
Mentor:	Izv. prof. dr. sc. Irena Galić
Sumentor:	Dr. sc. Hrvoje Leventić
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc. dr. sc. Bruno Zorić
Član Povjerenstva 1:	Dr. sc. Hrvoje Leventić
Član Povjerenstva 2:	Dr. sc. Krešimir Romić
Naslov završnog rada:	Primjena UML jezika i korisničkih priča u izradi specifikacije web portala
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada	Zadatak završnog rada je istražiti i opisati načine izrade razvojne specifikacije za razvoj web portala. Fokusirati se na agline principe kao što su korisničke priče i njihova primjena, te na primjenu UML jezika za razradu specifikacije dizajna i arhitekture portala. U praktičnom dijelu potrebno je izraditi specifikaciju web portala te implementirati takav portal prema specifikaciji. Tehnologija: web razvojni okvir po izboru Tema rezervirana za: Marijan Klarić Sumentor s FERIT-a: Hrvoje Leventić
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/bodaPostignuti rezultati u odnosu na složenost zadatka: 3 bod/bodaJasnoća pismenog izražavanja: 3 bod/bodaRazina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	14.06.2022.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.

	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 15.07.2022.

Ime i prezime studenta:	Marijan Klarić
Studij:	Preddiplomski stručni studij Računarstvo
Mat. br. studenta, godina upisa:	AR4668, 25.07.2018.
Turnitin podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Primjena UML jezika i korisničkih priča u izradi specifikacije web portala**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Irena Galić

i sumentora Dr. sc. Hrvoje Leventić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.

Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj:

1. UVOD

Tehnologija se svakodnevno vrlo brzo mijenja, postaje sve dostupnija ljudima, a tako raste i potražnja za programskim aplikacijama. Kako bi poduzeća mogla ponuditi što kvalitetnije proizvode potrebno je pronaći neka načela i koncepte kojih se treba pridržavati kako bi softver koji se razvija bio održiviji i kvalitetniji. Softver je doživio veliku ekspanziju sredinom 80-ih godina dvadesetog stoljeća. Velika ekspanzija dovodi do toga da se javljaju različiti načini modeliranja. Konstantno postoji potreba za pronalaskom jezika za modeliranje koji može zadovoljiti sve buduće potrebe korisnika.

U ovom radu opisuje se povijest nastanka danas najpopularnijeg jezika za modeliranje – UML (Unified Modeling Language). UML se koristi kako bi se uklonilo nerazumijevanje kompleksnih struktura i kako bi postojao jednostavan način prikaza različitih modela [1]. Kada se želi razviti ideja za određeni softver, vrlo je teško u datom trenutku odrediti što je važno i koliko će provedba takve ideje u stvarnost trajati. Korisničke priče su nastale da riješe taj problem. Korisničke priče pomažu u shvaćanju tko su trenutni i budući korisnici softvera te koje funkcionalnosti su potrebne upravo tim korisnicima. Tematika rada opisuje teoretska shvaćanja koja se kriju iza UML-a te korisničkih priča, prikazujući njihovu primjenu na web portalima.

1.1 Zadatak rada

Zadatak ovog rada je opisati UML-jezik i korisničke priče, zatim ih primijeniti na web portal. UML jezik pomaže jasnije prikazati i objasniti složene sustave. Kroz rad će se opisati različiti UML dijagrami. U radu će se obraditi i agilni principi kao što su korisničke priče. Nakon opisa UML jezika i korisničkih priči slijedi njihova primjena na web portalu.

2. OPĆENITO O POJMU UML

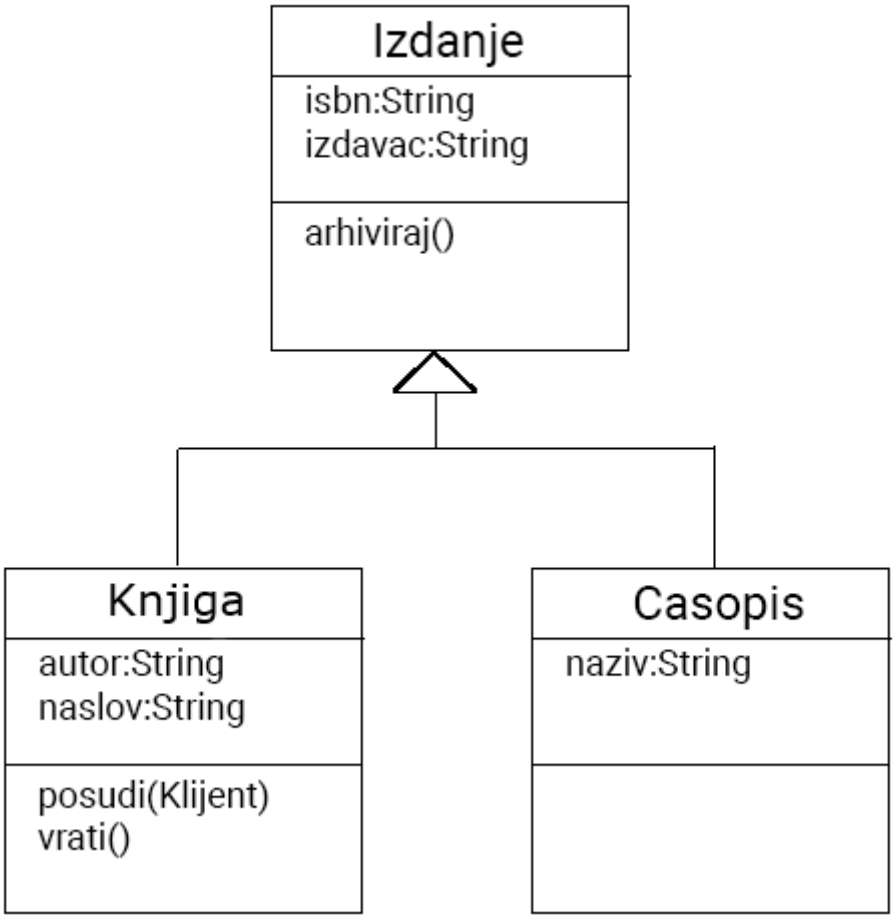
UML (engl. *Unified Modeling Language*) je skup grafičkih notacija, koje pomažu objasniti i dizajnirati programske sustave, posebice programske sustave koji koriste objektno orijentirano programiranje. U 1980-ima pojavljuju se objektno orijentirani programski jezici. U tom razdoblju se eksperimentiralo s različitim pristupima. Ključne knjige o objektno orijentiranim jezicima za grafičko modeliranje objavljene su u razdoblju između 1988. do 1992. Do 1994. broj metoda iznosio je više od 50. Autori su nudili uglavnom slična rješenja koja su se razlikovala u malim stvarima. Sve to dovelo je do velike konfuzije kod korisnika. Jedna od najpoznatijih metoda bila je OOSE (engl. *Object - Oriented Software Engineering*).

Tvorci OOSE metode bili su Booch i Jacobson. Još jedna poznata metoda bila je OMT (eng. *Object Modeling Technique*) čiji je tvorac Rumbaugh. Sredinom 1990. Ivar Jacobson, Grady Booch i James Rumbaugh počinju zajedno surađivati. Njihova ideja bila je da se pri izradi projekta koristi jedan programski jezik za modeliranje. Sredinom 1994. dolazi do razvijanja UML-a. Rumbaugh počinje surađivati s Boochem u Rational Software Corporation te nastaje prva verzija UML-a. Nedugo nakon toga pridružuje im se Jacobson pa je UML objedinio i OOSE. Godine1997. nekoliko organizacija (IBM, Oracle, Intellicorp, Texas Instruments i dr.) dale su prijedloge za metode koje bi olakšale međusobnu komunikaciju između modela. UML je tada postao jak, primjenjiv i utjecajan.

2.1. Dijagram klasa

Kako bi postojala mogućnost definiranja dijagrama klase, potrebno je prvo objasniti što je to klasa [1]. Klasa je apstrakcija skupova objekata koja imaju iste atribute i ponašanja. Kada bi se za primjer uzela klasa Automobil, ona bi sadržavala podatke o proizvođaču automobila, godini proizvodnje, prijeđenim kilometrima i broju vrata na vozilu. Također klasa Automobil sadržava i neke funkcije kao npr. funkcija voziti.

Ukoliko bi se nadodala još jedna klasa Vozač tada bi ona sadržavala podatke: imenu vozača, godini položenog vozačkog ispita i automobil koji taj vozač vozi, tada bi postojala mogućnost razmišljanja o vezi između klase Automobil i Vozač. Na primjer: svaki vozač posjeduje automobil. Dijagrami klasa pokazuju svojstva i operacije pojedine klase, ali i veze između različitih klasa. Moguće je modelirati različite veze, koje spadaju u dvije skupine: generalizacija i asocijacija.

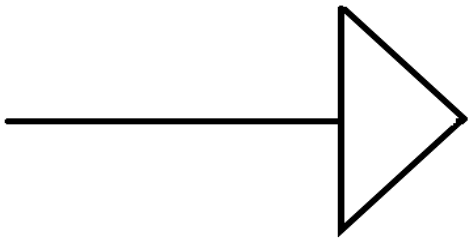


Slika 2.1. Prikaz dijagrama koji pokazuje hijerarhiju između klasa, takav prikaz se naziva generalizacija. Svaka klasa se sastoji od tri dijela: ime klase, lista atributa i operacije.

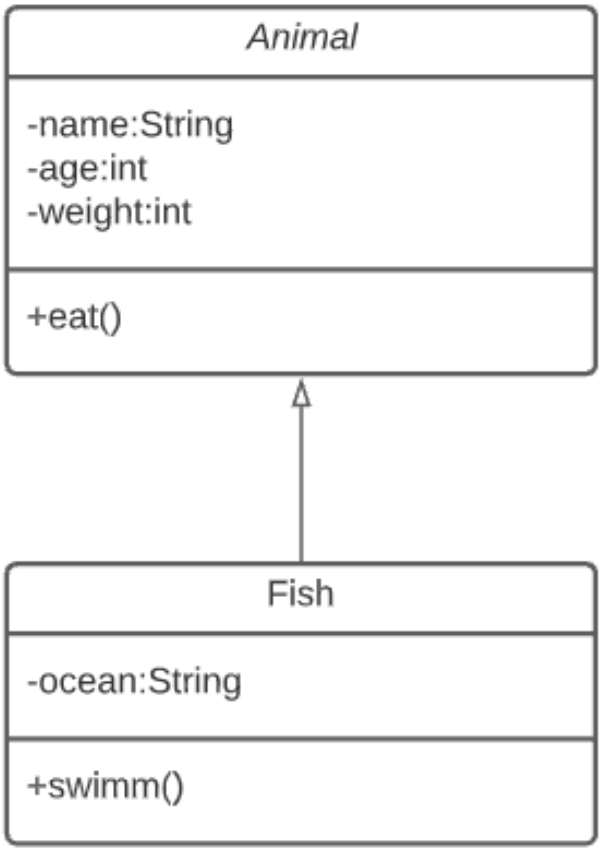
Različite veze između klasa prikazuju se pomoću:

- nasljeđivanja
- asocijacije
- agregacije
- kompozicije

2.1.1. Nasljeđivanje



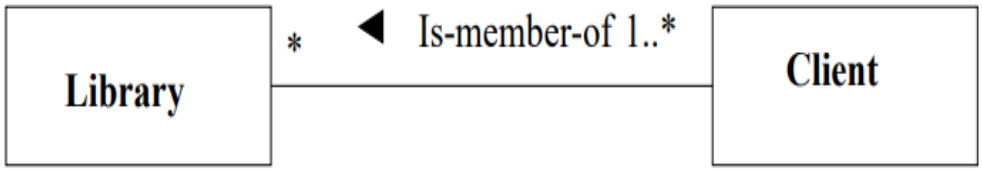
Slika 2.2 Prikaz nasljeđivanja se prikazuje pomoću strijelce na prikazanoj slici.



Slika 2.3. Prikaz primjera kako klasa Fish nasljeđuje sve atribute i funkcije klase Animal. Klasa Fish može i ne mora imati vlastite atribute i funkcije. Klasa Animal je apstraktna klasa. Apstraktne klase u klasnom dijagramu pišu se u *italic* stilu ili pomoću zagrada "<<Animal>>".

2.1.2. Asocijacija

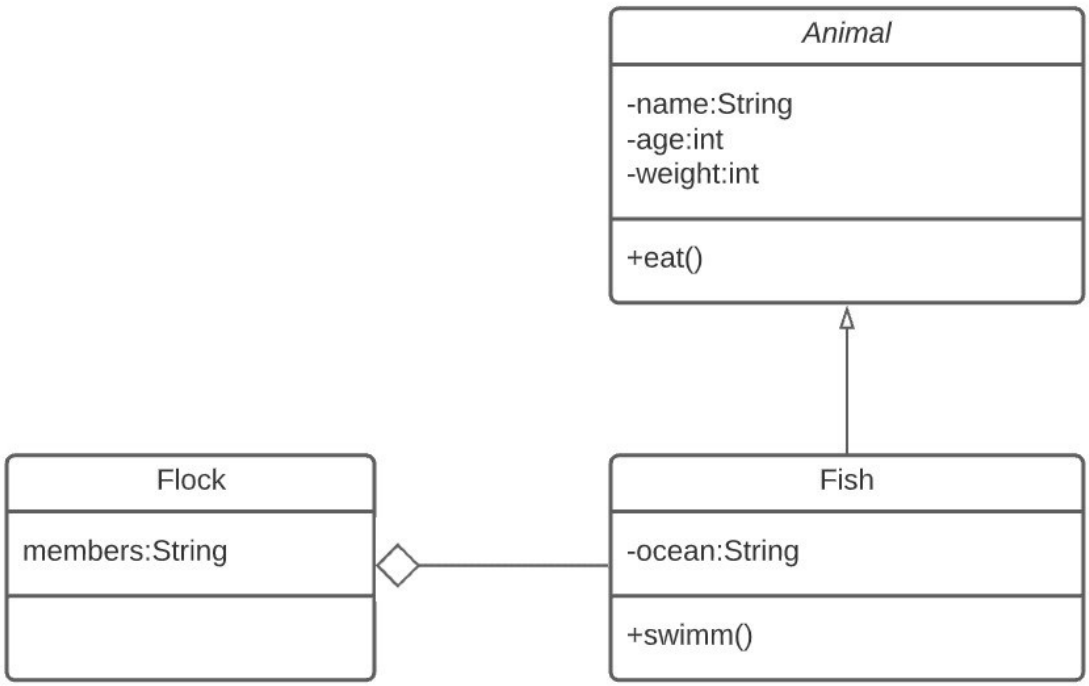
Asocijacija se prikazuje punom linijom koja povezuje dvije klase. Asocijacija može sadržavati i riječi kako bi bila još jasnija prilikom razmatranja osobi koja ju u datom trenutku promatra.



Slika 2.4. Prikaz asocijacije između klase Library i klase Client. Klijent može biti član više Library-a dok Library može imati više klijenata.

2.1.3. Agregacija

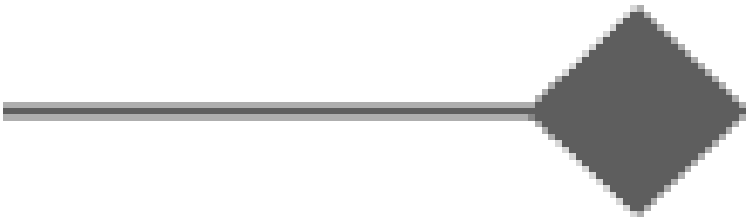
Slika 2.5. Prikaz oznake agregacije



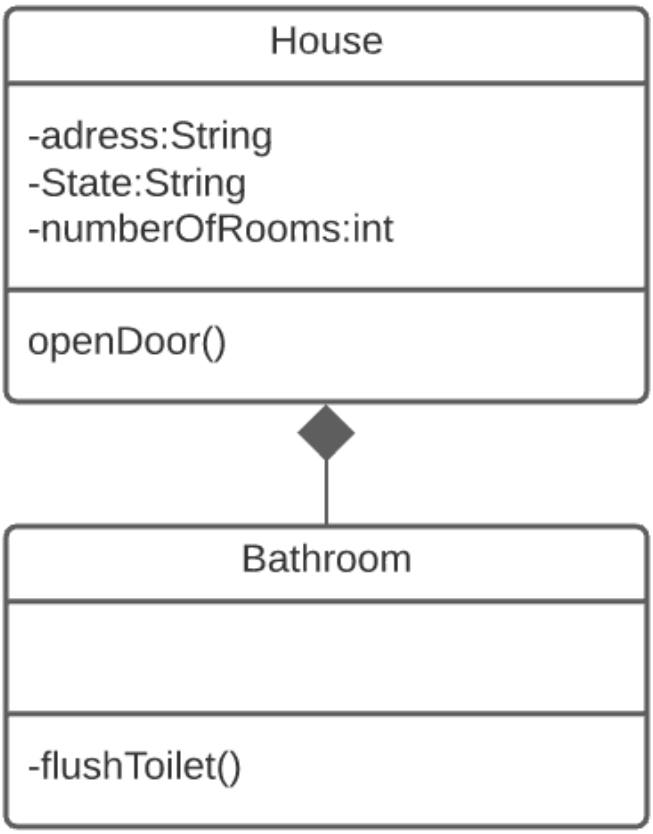
Slika 2.6. Prikaz agregacijske veze između klase Fish i klase Flock. Klasa Fish može, ali i ne mora biti član klase Flock. Agregacija pokazuje kako više pojedinaca iste klase mogu komunicirati.

2.1.4. Kompozicija

Kompozicija je vrsta agregacije gdje postoji odnos između cjeline i dijela. Razlikuje se od agregacije u tome što dio ne može postojati bez cjeline.



Slika 2.7. Prikazuje oznaku kompozicije



Slika 2.8. Prikaz primjera kompozicije. Klasa House može postojati bez klase Bathroom, ali klasa Bathroom ne može postojati bez klase House jer je dio nje.

2.1.5. Mnogostrukost

Mnogostrukost klase je indikator koliko objekata bi moglo popuniti klasu. Najčešći tipovi mnogostrukosti:

- 1 (automobil ima jednog vozača)

- 0..1 (automobil može i ne mora biti registriran)
- (automobil ne mora imati mehaničara koji ga održava, a može ih imati i nekoliko)

Mnogostrukost se definira sa donjom i gornjom granicom kao npr. 2..4 za broj igrača u video igri. Donja granica može biti bilo koji pozitivni broj. Gornja granica je pozitivni broj ili znak "*" koji označava neograničenost ili beskonačnost.

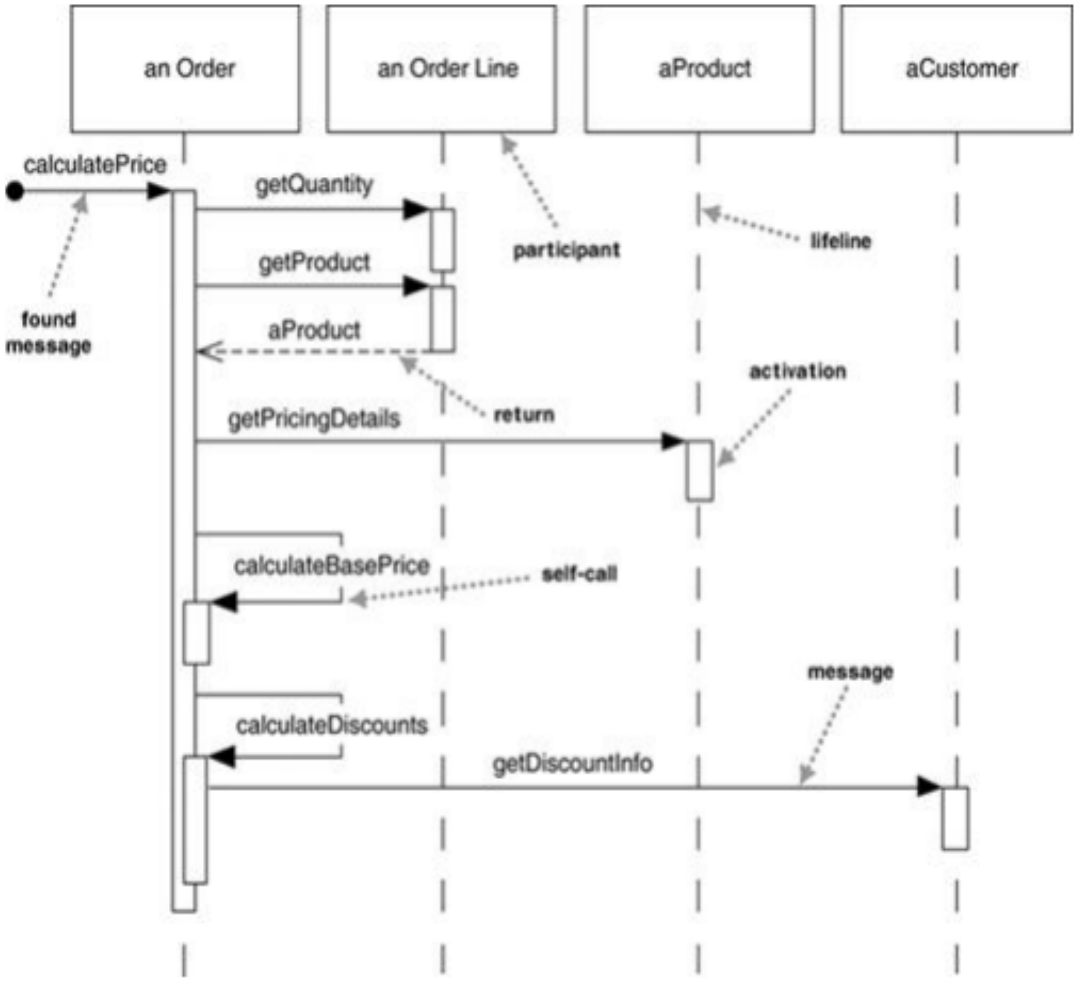
2.2.Dijagram slijeda

Dijagram slijeda obično opisuje ponašanje jednog događaja. Dijagram pokazuje objekte i poruke koje se razmjenjuju između objekata.

U sekvencijskom dijagramu se definiraju:

- Objekti
- Poruke
- Veze

Uzima li se u obzir idući scenarij: ukoliko postoji narudžba te se u nastavku želi pozvati metoda koja će izračunati srednju cijena proizvoda potrebno je pogledati sve artikle i cijene koje pripadaju tim artiklima. Na kraju je potrebno izračunati sumu s popustom koji je vezan za kupca. Sekvencijski dijagram pokazuje implementaciju takvog scenarija koji je detaljnije opisan na slici koja se nalazi u donjem dijelu teksta.



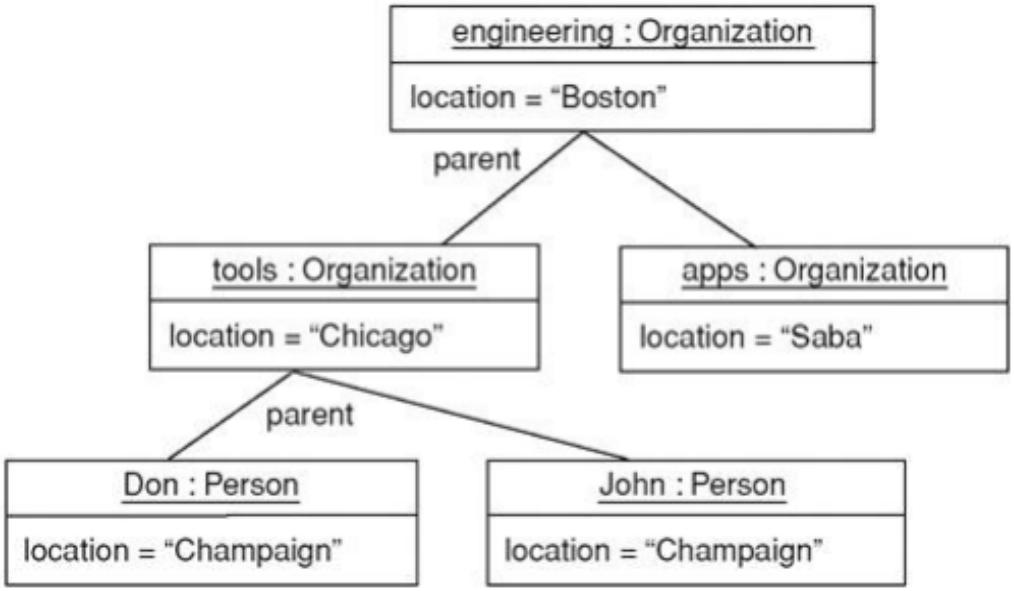
Slika 2.9. Prikaz primjera sekvencijskog dijagrama. Vertikalna isprekidana linija prikazuje vrijeme. Što se prikaz više bliži dnu to je prošlo više vremena. [2]

Uočljivo je kako getQuantity i getProduct traže informacije od "an Order Line" [2]. "An Order Line" vraća tražene podatke objektu "an Order". Objekt "an Order" traži od objekta "aProduct" informacije o cijeni. "An Order" tada pokreće funkcije "calculateBasePrice" i "calculateDiscounts." Metoda "getDiscountInfo" šalje poruku kupcu (aCustomer). Dijagram slijeda ne prikazuje u svim situacijama točan slijed izvršavanja pojedinih funkcija unutar dijagrama. Sekvence poruka getQuantity, getProduct, getPricingDetails i calculateBasePrice potrebno je izvesti za svaku liniju narudžbe dok je calculateDiscounts potrebno izvesti samo jednom. Ova situacija se može uočiti u gore prikazanom sekvencijskom dijagramu.

2.3. Dijagram objekta

Dijagrami objekata opisuju objekt koji se nalazi unutar sustava u određenom vremenu. Prikazuje instancu klase, a ne klase kao u klasnom dijagramu. Objektni dijagrami su vrlo

korisni za prikazivanje povezanosti između objekata. Struktura se može definirati pomoću klasnog dijagrama, no i dalje su moguće nejasnoće u razumijevanju te strukture. U takvim situacijama postoji mogućnost prikaza nekoliko dijagrama objekta koji su u mogućnosti upotpuniti razumijevanje strukture.

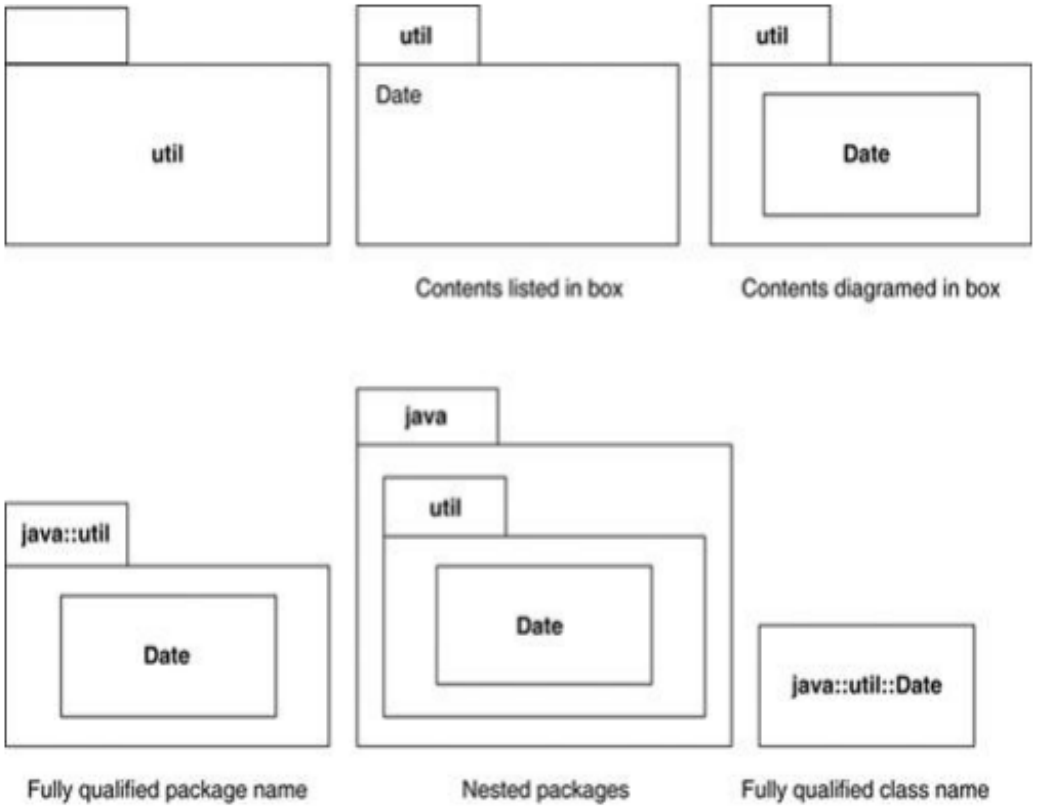


Slika 2.10. Prikaz instance klase [2]

Slika 2.10. prikazuje instance klase u kojima se može zamijetiti kako elementi objektnog dijagrama jesu instance specifikacija, a ne prave instance. Razlog je u tome što je dopušteno ostavljati obavezne atribute praznima ili prikazivati instance specifikacija apstraktnih klasa [2]. Instance specifikacija su djelomično definirane instance klasa.

2.4. Paketni dijagram

Paketni dijagram je konstrukcija za grupiranje koja omogućuje uzimanje bilo koje konstrukcije u UML-u i omogućuje grupiranje elemenata u skupine koje čine jedinice više razine [2]. Svaka klasa je dio jednog paketa. Paketi mogu biti dijelovi drugih paketa, što omogućuje hijerarhijsku strukturu. Paket može sadržavati druge pakete ili klase. Svaki paket ima naziv što znači da svaka klasa mora imati jedinstveno ime. Ukoliko postoji klasa Auto unutar paketa, jedini način definiranja nove klase Auto je da se nalazi unutar drugog paketa.



Slika 2.11. Prikaz načina paketa u dijagramima. [2]

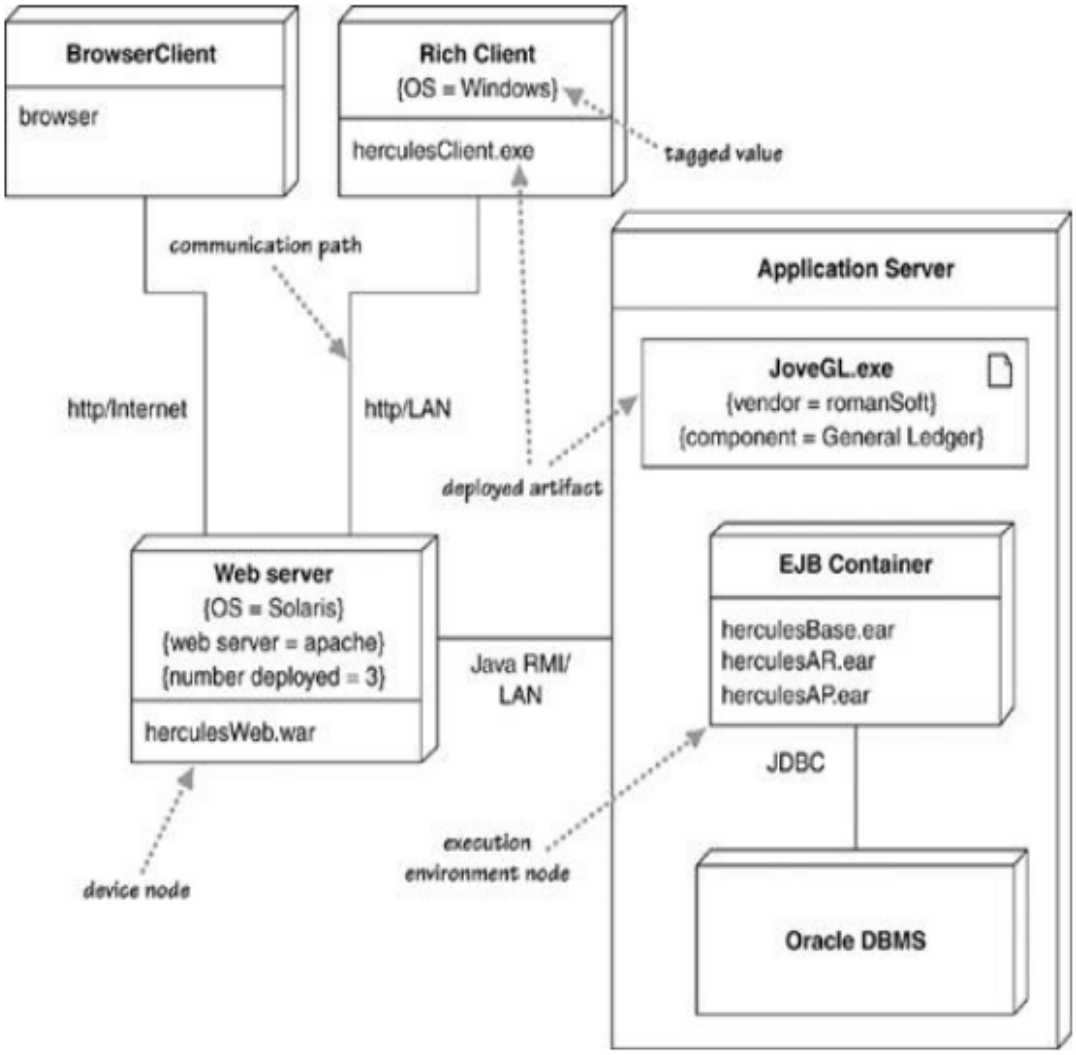
Paketi se prikazuju na dva načina:

- Jedan se paket crta unutar drugog paketa.
- Paketi se crtaju odvojeno te se povezuju putem veze ugnježđenja.

Postoje dvije vrste vidljivosti koje se označavaju sa "+" za javni i "-" za privatni način prikaza datoteke. Javni pristup znači da je element vidljiv iz svakog paketa. Privatni je vidljiv unutar paketa, ali je sakriven za druge pakete.

2.5. Dijagram rasporeda

Dijagram rasporeda prikazuje koji dijelovi programa se pokreću na pojedinim dijelovima hardvera [2]. Sastoji se čvorova koji su povezani komunikacijskim kanalima. Čvor može pokrenuti program. Čvorovi se javljaju u dva oblika. Uređaj je hardver, koji je povezan sa sustavom. Okruženje izvršenja (engl. *execution environment*) je program koji pokreće ili sadrži softver (npr. operacijski sustav).



Slika 2.12. Prikaz dijagrama rasporeda [2]

Čvorovi sadrže artefakte koji su ujedno i fizička manifestacija softvera. Moguće je prikazati artefakte pomoću klasnog kontejnera. Artefakti su najčešće implementacija komponente. Komunikacijski kanali govore kako se odvija komunikacija između čvorova.

2.6. Dijagram korištenja

Dijagrami korištenja pomažu razjasniti koje funkcionalnosti treba imati sustav [2]. Dijagrame korištenja najbolje je razvijati u što ranijim fazama planiranja. Detaljniji dijagrami korištenja treba razvijati prije početka razvoja tog dijela softvera. Važno je raspolagati informacijom koja ukazuje da dijagrami korištenja predstavlja vanjski pogleda na sustav. Posebno je potrebno ukazati na važnost teksta, a ne prvobitno dijagramima. Ukoliko je pažnja usredotočena na

dijagram, u detaljnijim postupcima analize se samo razumijevanje značenja može činiti suviše kompleksno.

Komponente dijagrama korištenja:

- **Pravokutnikom** se označava sustav, a sustav može predstavljati web stranicu, aplikaciju ili pak nešto drugo.
- **Prikazom koji ukazuje na čovjeka** se označavaju sudionici, oni su vanjski entiteti koji izravno ili neizravno povezani sa sustavom. Sudionici se dijele na primarne i sekundarne. Primarni sudionici iniciraju korištenje sustava, dok sekundarni sudionici odgovaraju na zahtjeve primarnog sudionika. Primarni sudionici se u pravilu nalaze lijevo od sustava, a sekundarni sudionici desno od sustava.
- **Eliptičnom kružnicom** je označena radnje koje se odvijaju u sustavu kada sudionik pokrene radnju sa sustavom.

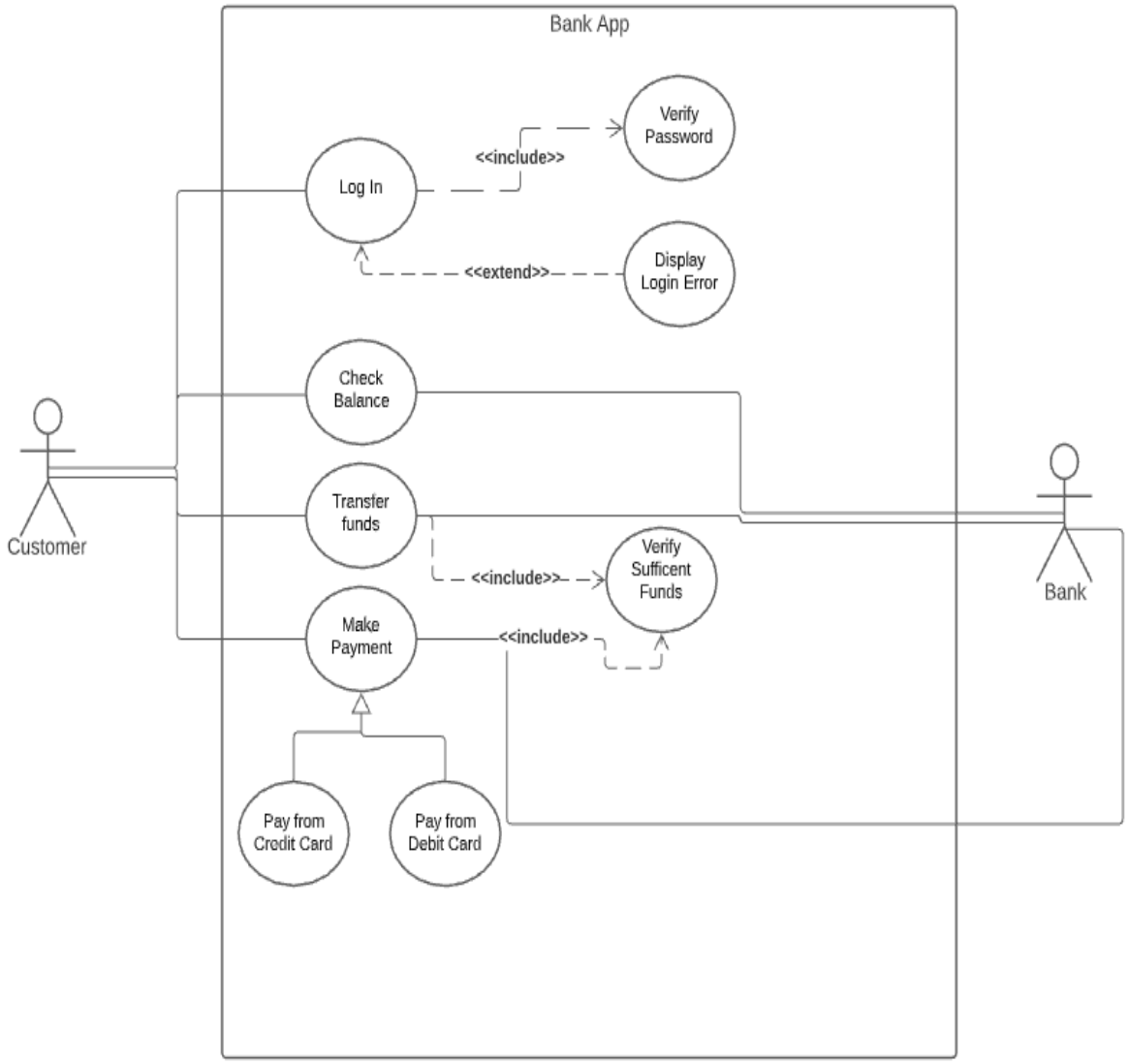
Različite vrste veza u dijagramu korištenja:

- asocijacija (engl. *association*)
- uključi (engl. *include*)
- proširi (engl. *extend*)
- generalizacija (engl. *generalization*)

Veza uključi ukazuje na to da svaki put kad se pokrene glavna radnja, automatski će se pokrenuti i uključene radnje. Uključena radnja je sastavni dio glavne radnje. Veza uključi se prikazuje uz pomoć strelice kojoj su linije isprekidane; također se označava tekstom strelica s <<include>> (hrv. uključi). Strelica pokazuje povezanost od glavne radnje prema uključenoj radnji. Na slici je prikazano kako radnja login ima vezu uključi s radnjom Verify Password.

Veza proširi označava radnju koja se može dogoditi, ali ne mora se izvoditi svaki put. Na primjeru se uočava kako postoji radnja Display Login Error. Nije potrebno svaki puta prikazati tu radnju, nego samo onda kada korisnik unese krivu lozinku. Veza proširi se crta tako da strelica pokazuje prema glavnoj radnji te se označava s <<extend>> (hrv. proširi). Generalizacija omogućava nasljeđivanje karakteristike glavne radnje i proširuje mogućnosti. Na primjeru se uočava kako na radnji Make Payment postoje dvije opcije: Pay from Credit

Card i Pay from Debit Card. Svaka radnja dijeli karakteristike s glavnom radnjom Make Payment. Generalizaciju se označava sa zatvorenom strelicom.



Slika 2.13. Prikaz dijagrama korištenja

2.7. Dijagram stanja

Dijagramom stanja prikazuje stanje u kojem će se pronaći objekt za vrijeme trajanja tog objekta [2]. Dijagram stanja koristi se prilikom definiranja složenih objekata.

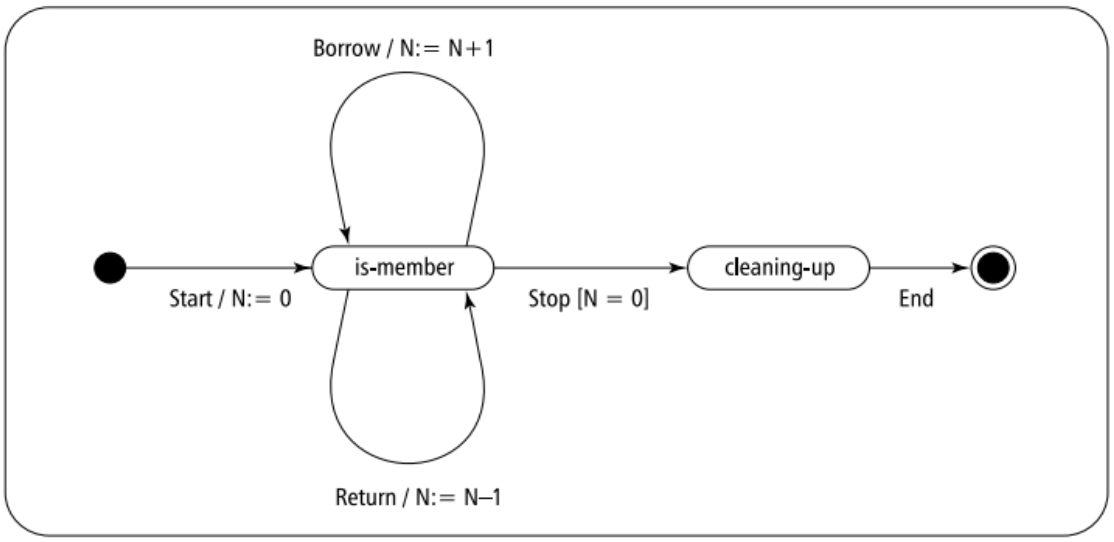
Dijagram stanja prikazuje:

- Događaje koji utječu na objekt
- Različita stanja objekta

Događaj mijenja stanje ili ponašanje objekta.

Vrste događaja:

- poziv
- signal
- promjena
- vremenski događaj



Slika 2.14. Prikaz dijagrama stanja [2]

Stanje se prikazuje pomoću pravokutnika zaobljenih rubova. Početno stanje prikazuje se s krugom. Završno stanje oslikava se s kružnicom unutar koje se nalazi krug. Tranzicija je prikazana strelicom. Ona se sastoji od oznake za pokretanje tranzicije, takva tranzicija može biti zaštićena s boolean oznakom. U tom slučaju tranzicija će se dogoditi samo ukoliko boolean bude istinit (engl. *true*).

2.8. Dijagram aktivnosti

Dijagramom aktivnosti se opisuje logika, procesi i tijek rada [2].

Elementi dijagrama aktivnosti:

- **Početno i završno stanje**

  Početno stanje označava se punim krugom, a završno stanje prikazuje se kružnicom u kojem se nalazi ispunjeni krug.

Slika 2.15. Primjer početnog i završnog stanja

- **Aktivnosti**

  Postoje dvije vrste aktivnosti: paralelne i slijedne. Kod paralelnih aktivnosti redoslijed izvršavanja aktivnosti nije bitan.

Slika 2.16. Prikaz aktivnosti unutar pravokutnika

- **Prijelaz**

Slika 2.17. Simbol kojim se prikazuje prijelaz

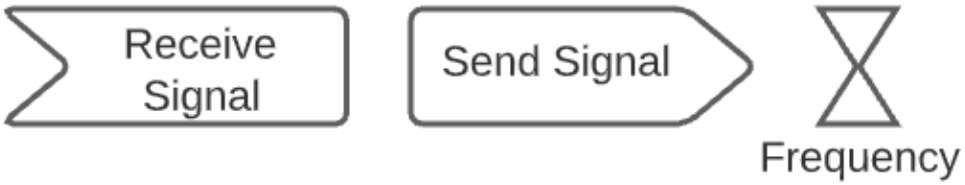
- **Odluka**

  Koristi se kada postoji uvjet.

Slika 2.18. Simbol kojim se prikazuje odluka

- **Signal**

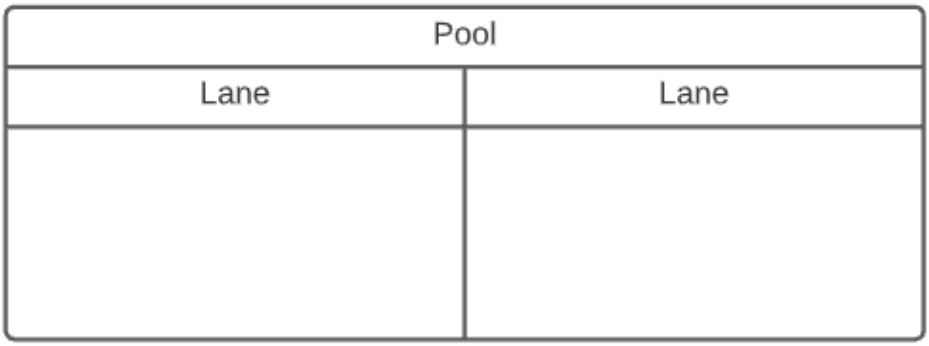
  Signal služi za prikazivanje neke aktivnosti, gdje djeluju događaji koji su nastali tijekom vanjskog procesa. Postoje primajući signali, šaljući signali i vremenski signali.



Slika 2.19. Simboli kojima se prikazuju primajući, šaljući i vremenski signali

- **Plivaće staze (horizontalne i vertikalne)**

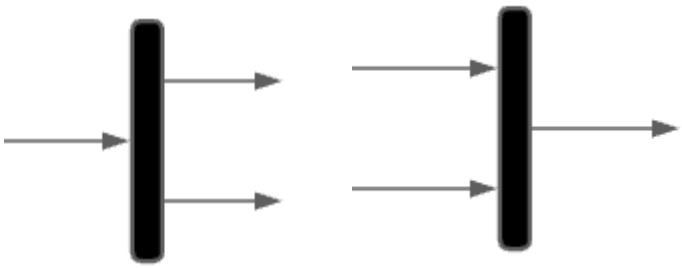
  U plivačkoj stazi navedeni su sudionici i stanja koja pripadaju stazama. Važne su kada postoji namjera prikazivanja koji od dijelova sustava je zadužen za određeni zadatak.



Slika 2.20. Primjer plivaće staze

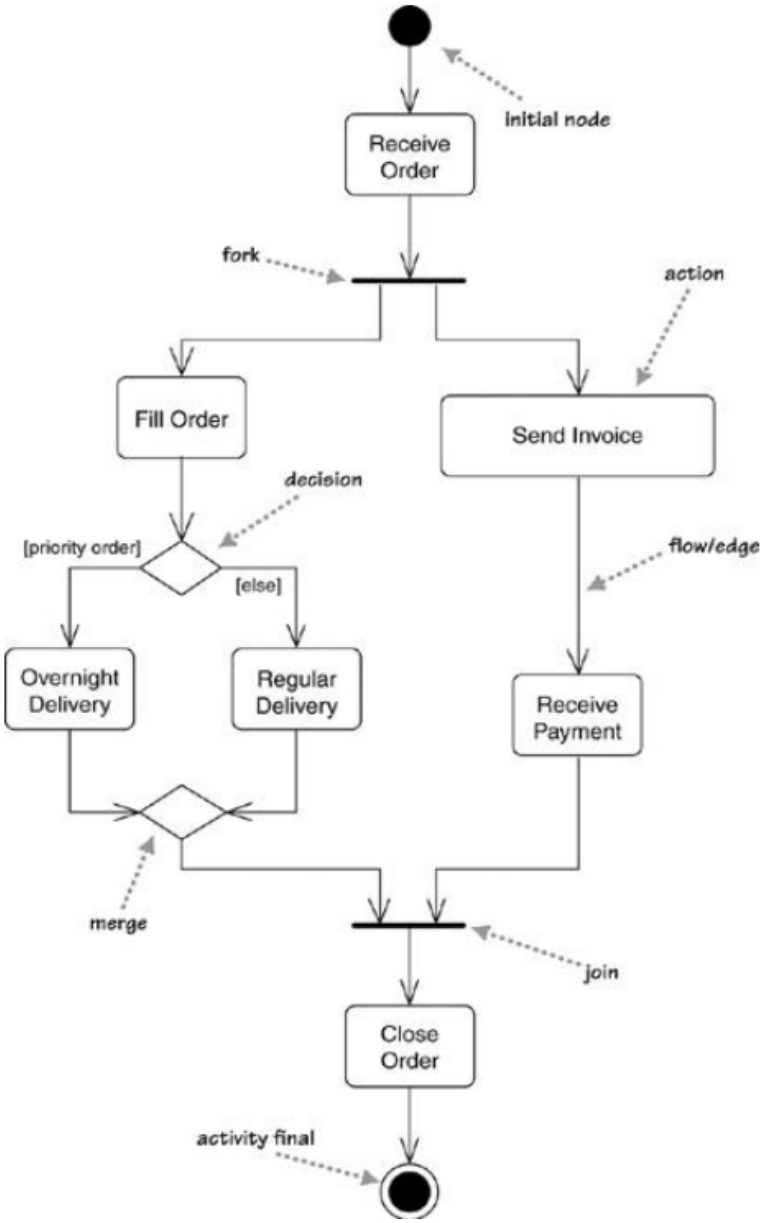
- **Račvanje i skupljanje**

Račvanje ima jedan ulazni tok i više izlaznih tokova. Skupljanje se koristi kako bi se sinkronizirala paralelna aktivnost. Odlazni tok se pokreće kada svi dolazni tokovi dođu do skupljanja.



Slika 2.21. Prikaz skupljanja i račvanja

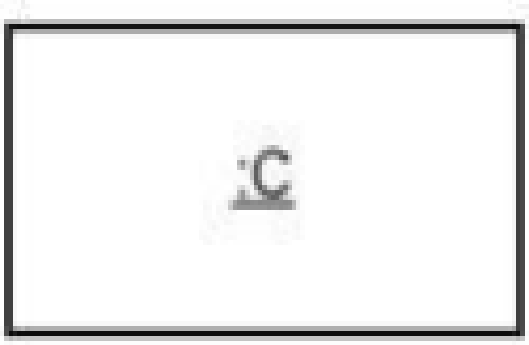
Izradi se radnja Receive Order. Kada se ta radnja izvrši dolazi se do račvanja. Radnje Fill Order i pošalji Send Order se javljaju istovremeno. Ukoliko se dvije radnje javljaju istovremeno onda redoslijed izvođenja tih radnji nije važan. Dijagram aktivnosti omogućuje da izvršitelj radnji bira sam redoslijed izvršavanja tih radnji. Kada se pojavi paralelizam onda je potrebno sinkronizirati procese. No, nije u cilju narudžbu poslati prije nego li je plaćena. To će se prikazati na dijagramu pomoću "pridruži" prije radnje Close Order.
Na taj način je omogućeno izvršavanje radnje Close Order samo kada je narudžba dostavljena i plaćena. Kada postoji uvjet onda se izvršava samo jedan dio grane. Na slici je prikazano kako postoji uvjet za prioritet narudžbe. Ukoliko je je prioritet narudžbe Overnight Delivery izvršava se radnja Overnight Delivery. U slučaju da se radi o Regular Delivery također se izvršava samo grana na kojoj se nalazi radnja za običnu dostavu. Sjedinjenje ima nekoliko ulaznih tokova, ali samo jedan izlaz. Sjedinjenje obilježava kraj uvjeta koju je pokrenuo neki uvjet.



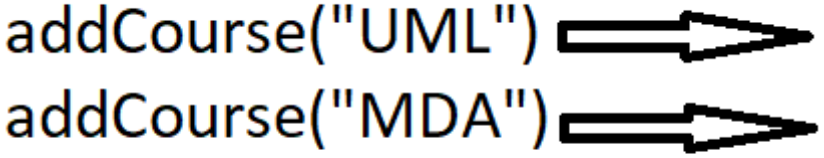
Slika 2.22. Prikaz dijagrama aktivnosti [2]

2.9. Komunikacijski dijagram

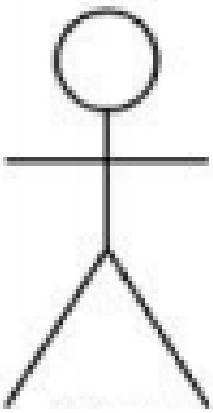
Komunikacijski dijagram prikazuje kako dijelovi sustava komuniciraju i koje se poruke razmjenjuju. Dijagram se prikazuje pomoću objekata, veza i aktera.



Slika 2.23. Prikaz objekta koji se označava pomoću slova :c



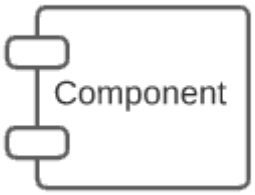
Slika 2.24. Prikaz veze koja povezuje objekte i aktere



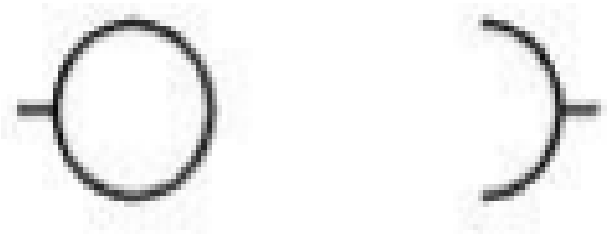
Slika 2.25. Prikaz aktera koji označava klijenta

2.10. Dijagram komponente

Dijagram komponenti prikazuje pogled na model u sustavu. Dijagram komponenti se koristi kada postoji potreba da se sustav podijeli na komponente te se treba prikazati njihova međusobna veza kroz sučelje [2].

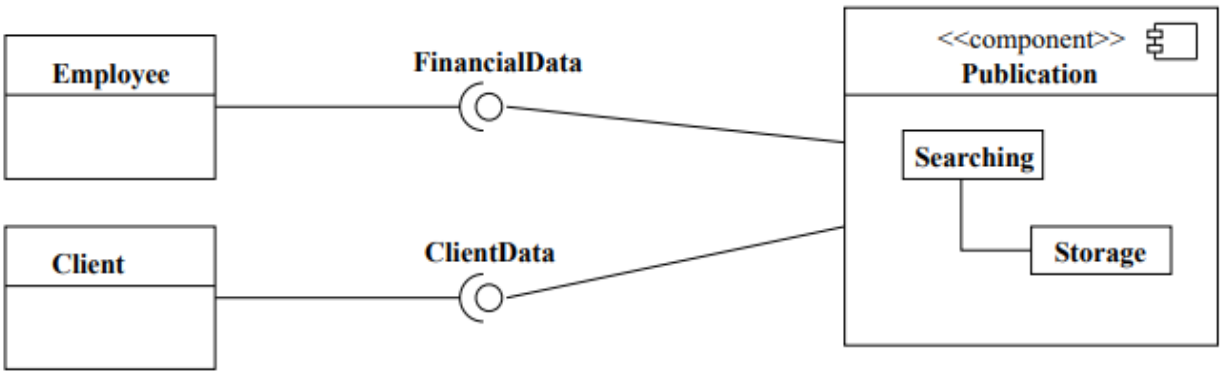


Slika 2.26. Prikaz komponente



Slika 2.27. Prikaz sučelja

Lijeva slika prikazuje sučelje koje je osigurano tj. komponenta može funkcionirati i bez tog sučelja. Slika desno prikazuje sučelje koje je potrebno, bez njega komponenta ne bi mogla funkcionirati samostalno.



Slika 2.28. Prikaz komponente Publication koja sadrži dvije klase: Searching i Storage. Komponente su povezane sučeljima [2]

2.11. Dijagram interakcije

Dijagram interakcije je spoj dijagrama aktivnosti i dijagrama slijeda [2]. Njegova svrha je prikazivanje odnosa između interakcija. Njegovi elementi su: čvor početka, čvor završetka,

čvor spajanja, čvor odlučivanja i čvor priključivanja. Ovaj dijagram predstavljen je u UML 2. Još uvijek nema veliku primjenu u svakodnevnim situacijama.

2.12. Dijagram vremena

Dijagram vremena je vrsta sekvencijskog dijagrama koji se fokusira na vremenska ograničenja za jedan objekt ili za više objekata [2]. Koriste se i za prikazivanje vremenskih ograničenja između životnih tokova.

Elementi vremenskog dijagrama:

- Horizontalna os
- Vertikalna os
- Promjena stanja objekata

Horizontalna os predstavlja vrijeme. Vertikalna os prikazivanje stanja i životne tokove. Promjenu stanja označavamo s izlomljenom crtom.

3. KORISNIČKE PRIČE

Korisnička priča opisuje funkcionalnosti koje su važne korisniku sustava ili softvera.

Sastoje se od tri dijela:

- opisa priče koja služi za planiranje i kao podsjetnik
- razgovora koji će omogućiti da se spomenu i detalji
- testova koji će utvrditi kada je završena korisnička priča

Korisničke priče su uglavnom napisane na papiru. Korisničke priče bi trebale predstavljati korisničke zahtjeve. Velika korisnička priča je dobila naziv epic, a on se može podijeliti u dvije ili više kraćih priča.

Zašto pisati korisničke priče:

- Naglašavaju verbalnu komunikaciju, a ne pisanu
- Razumljive su programerima ljudima koji ne dolaze iz tehničkog područja
- Odgovarajuće su veličine
- Rade s iterativnim razvojem softvera
- Odgađaju detalje proizvoda sve dok se ne shvati što je zapravo najvažnije za projekt

Jedna od prvih stvari koja se može primijetiti kod kreiranja softvera i korištenja korisničke priče, jest komunikacija između developera i korisnika u ranim fazama izrade proizvoda [3]. Korisnici imaju aktivnu ulogu tijekom cijelog ciklusa kreiranja softvera. Korisničke priče moguće je pisati u bilo kojem dijelu projekta, uglavnom se pišu na početku, prije nego li se započne s kreiranjem softvera. Tijekom pisanja priči važan je brainstorming, gdje bi trebalo napisati što više priči. Kada se priče formiraju, developeri procjenjuju koliko će vremena trebati da se svaka pojedina priča realizira.

Korisnici bi trebali sudjelovati s developerima dok se priče realiziraju. Tijekom iteracija korisnici sudjeluju u testiranju softvera i rade s developerima u automatizaciji testova. Prvi rok za korisničku priču najčešće je pogrešan, jer je teško odrediti na samom početku projekta točno vrijeme trajanja. Kako bi uspješno planirali datum izlaska nužno je sortirati priče u hrpe i svaka hrpa predstavljati će iteraciju. Priče najvišeg prioriteta idu u prvu hrpu. Kada se hrpa napuni priče nižeg prioriteta idu u drugu hrpu itd. Kako bi uspješno planirao datum izlaska, korisnici također moraju postaviti prioritete korisničke priče.

Vrlo je važno obratiti pažnju na:

- važnosti značajki široj bazi korisnika
- važnosti značajki vrlo važnim korisnicima
- kohezivnosti priča u relaciji jedna s drugom. Na primjer, korisnička priča "odzumiraj" ne mora imati veliki prioritet, no ako veliki prioritet ima korisnička priča "zumiraj" onda je vrlo važno implementirati i korisničku priču "odzumiraj".

Prilikom slaganja prioriteta, korisnička priča ima vrlo važnu ulogu u cjelini. Procjenu cijene daje tim developera. Svakoj priči dodijeljeni su bodovi, koji označavaju kompleksnost i veličinu priča u odnosu jedne s drugom.

3.1 Testovi prihvaćanja

Proces u kojem se utvrđuje da značajke proizvoda rade točno onako kako su definirane u korisničkim pričama se naziva test prihvaćanja [3]. Jednom kada iteracija započne, developeri započinju s programiranjem, a korisnici počinju specificati testove. Najbolje je pridružiti člana korisničkom timu koji ima tehničkog znanja i koji je iskusan u testiranju softvera.

3.2. Pisanje korisničke priče

Korisničke priče moraju biti:

- nezavisne
- u skladu sa svojstvom raspravljanja
- vrijedne krajnjem korisniku
- procjenjive
- male
- provjerljive

Vrlo je važno voditi računa da svaka priča bude nezavisna [3]. Ukoliko jedna priča ovisi o drugoj može doći do velikih poteškoća prilikom stvaranja prioriteta i planiranja. Ukoliko postoji informacija da su dvije korisničke priče ovisne jedna o drugoj, dobra opcija je spojiti ih u jednu veću korisničku priču. Ispostavi li se da su te dvije spojene korisničke priče prevelike, tada se moraju podijeliti na drugačiji način. O pričama se mora dati pregovarati i raspravljati. Važno je shvatiti kako korisničke priče nisu ugovor na koji se bilo tko obvezuje. Raspravu oko korisničkih priča moguće je prekinuti te nastaviti kasnije.

Projekti često uključuje priče koje nisu važne korisnicima. Važno je imati na umu razliku između osoba koje kupuju softver i osoba koje ga koriste. Na primjer: Osoba koja kupuje softver za veliku tvrtku može biti zabrinuta gdje će se podaci softverskog programa spremati, dok korisniku tog programa to ne mora biti uopće važno. Developeri moraju moći procijeniti koliko će im vremena trebati za implementaciju svake korisničke priče.

Postoje tri najčešća razloga kada developeri neće moći procijeniti vrijeme implementacije:

- Developer nema potrebno znanje o području
- Developer nema potrebno tehničko znanje
- Korisnička priča je prevelika

Ukoliko developer nema dovoljno znanja o području može se dogoditi da će imati problema pri implementaciji značajki. Ako developer ne može shvatiti korisničku priču potrebno je obratiti se korisniku koji ju je napisao. Može se tražiti od developera da implementiraju neku značajku koja mora koristiti tehnologiju o kojoj nemaju znanje. Problem se može riješiti, ako jednom ili dva developera bude omogućeno vrijeme gdje će proučavati potrebnu tehnologiju. Potrebno je proučiti tek toliko kako bi mogli dati procjenu o trajanju implementacije takve značajke. Kada je korisnička priča prevelika, potrebno ju je razložiti u nekoliko manjih.

Epic često spada u dvije kategorije:

- sjedinjena priča
- kompleksna priča

Sjedinjena priča sadrži više kratkih priča. Ponekad je moguće da kad developeri krenu u izradu značajke, da tek onda shvate što su sve korisnici mislili kad su pisali priču. Primjer korisničke priče: Korisnik može objaviti oglas.

Značajke na koje su korisnici mislili:

- oglas može biti sadržan u nekoliko kategorija
- oglas može biti označen kao neaktivan
- korisnik može imati nekoliko oglasa
- korisnik može uređivati oglas
- korisnik može izbrisati oglas

Ovisno o tome koliko će implementacija ovih značajki trajati, developeri mogu podijeliti korisničku priču na nekoliko njih. Može se dogoditi da postoji nekoliko vrlo malih korisničkih priča. Nekoliko vrlo malih priča se može spojiti u jednu. Korisničke priče se moraju testirati, inače ne bi postojalo znanje kada je završena njihova implementacija. Neprovjerene korisničke priče se prepoznaju po tome što ne sadrže nikakve funkcionalne značajke. Npr. korisnik se mora lako snalaziti po softveru. Testovi bi trebali biti automatizirani, to znači da je potrebno težiti 99% automatizaciji. Kada se softver razvija inkrementalno, vrlo lako se može dogoditi da softver koji je radio jučer ne radi danas u skladu sa svojim funkcijama.

3.3. Modeliranje korisnika

Na većini projekata, korisničke priče se pišu kao da postoji samo jedna vrsta korisnika. Takav pristup može dovesti do velikih poteškoća, jer ne obuhvaća one korisnike koji nisu definirani.

Nekoliko koraka kako identificirati vrste korisnika:

- brainstorming
- organizirati korisnike u kategorije
- utvrditi vrste
- "ispolirati" vrste

Kako bi se uspješno identificirale vrste korisnika, potrebno je napraviti sastanak gdje će se sastati korisnici i developeri. Svatko od sudionika treba na komad papira zapisati vrstu korisnika. Zatim taj papir staviti na pano. Kada se to napravi, svatko treba reći naglas što je napisao. Svaka osoba može napisati i više korisnika. Kada su zapisani korisnici, potrebno je organizirati korisnike u kategorije. Papire s korisnicima nužno je sortirati, tako da se poliježe jedan papir preko drugog ukoliko se korisnici preklapaju. Papiri se poliježu tako da prvi papir pokriva drugi toliko koliko se korisnici poklapaju. Potrebno je sažeti i grupirati korisnike. Najlakše je započeti s korisnicima čiji se papiri u potpunosti preklapaju. Moguće je to učiniti na dva načini: Prvi način je maknuti jednog od njih, a drugi način je staviti ih na jedan papir. Nakon što su utvrđene vrste korisnika, potrebno je dodati informacije o svakom od njih.

Informacije bi trebale sadržavati sljedeće:

- Učestalost posjećivanja softvera

- Znanje o domeni
- Razinu korisnikove računalne pismenosti
- Razina snalaženja s našim softverom
- Cilj koji korisnik želi ostvariti koristeći naš softver [3]

3.4. Tehnike za uspješno pisanje korisničkih priča

Tijekom izrade projekta korisničke priče će nastajati, ali isto tako neke od njih će biti nužno reducirati. Kako bi se korisničke priče efektivno skupile, potrebne su određene tehnike. Tehnike moraju biti jednostavne i nenametljive kako bi se mogle koristiti tijekom cijele izrade projekta.

Neke od najvrjednijih tehnika:

- **Intervjui s korisnicima**
  Korisnici bi trebali biti intervjuirani kad god je to moguće [3]. Najbolje je to napraviti s više različitih korisničkih uloga. Pitanja kao što je "Što želite?" treba izbjegavati. Korisnik rijetko može reći što on zaista treba. Najbolje je promatrati korisnika i vidjeti kako se snalazi sa softverom u određenim situacijama.
- **Upitnici**
  Kod postavljanja pitanja vrlo je važno voditi računa na koji način su pitanja postavljena. Pitanja koja daju korisniku opciju da odgovori s "Da" ili "Ne" treba izbjegavati. Pitanja trebaju biti otvorena i dopustiti korisniku da što više govori.
- **Promatranja**
  Promatrati korisnike dok koriste softver je odličan način shvaćanja koliko su potrebe samih korisnika reprezentativno prepoznate.
- **Radionice pisanja priči**
  Radionice pisanja priči mogu biti jedan od najbržih načina koji mogu osigurati veliki broj korisničkih priča. Korisničke priče trebaju biti kratke i bez detalja. Na radionicama bi trebalo sudjelovati što više ljudi koji će raditi na softveru i što više klijenata kako bi bio omogućen dolazak do što detaljnijih informacija.

3.5 Korisnički surogat

U projekt se moraju uključiti i stvarni korisnici. Nažalost, može biti kompleksno pronaći korisnika koji će pisati korisničke priče. Kada ne postoji mogućnost uključivanja stvarnih korisnika u projekt potrebno je napraviti surogat korisnika, koji neće biti korisnik, nego reprezentacija stvarnog korisnika. Surogat korisnika treba koristiti samo onda kada ne postoji mogućnost pronalaska stvarnih korisnika u timu [3]. U timu je najbolje imati nekoliko surogat korisnika.

3.5.1 Korisnikov menadžer

Ponekad tvrtka za koju se obavlja određeni posao, neće omogućiti pristup pravim korisnicima. Tada je najbolje okrenuti se menadžerima softvera. Iako menadžeri softvera ne koriste softver na jednak način kao korisnik i ima veća tehnička znanje nego običan korisnik i dalje može opisati poneki detalj važan za softver koji je u procesu razvijanja.

3.5.2 Voditelj razvoja

Voditelj razvoja može biti jedna od najgorih opcija za surogat korisnika, osim prilikom razvijanja softvera koji cilja na razvojne voditelje. Voditelj razvoja može stvarati prioritete priče drugačije samo kako bi se nova tehnologija implementirala što prije. Još jedna stvar koju treba imati na umu je to što voditelji razvoja nemaju jednako znanje o tom području za koji se softver razvija.

3.5.3 Prodavači

Prodavači teško mogu dati pravu reprezentaciju što pravi korisnik želi, jer je njima najvažnija stvar kako taj proizvod prodati. Prodavači imaju dobre veze s korisnicima i trebali biste ih iskoristiti. Pitajte ih da vas upoznaju s kupcima.

3.5.4. Stručnjaci

Stručnjaci su odličan izvor informacija jer razumiju važne stavke koji softver treba imati. Ponekad je teško razumjeti korisnikove želje, jer ne postoji dovoljno znanja o tom području. Stručnjaci mogu pomoći prilikom rješavanja nerazumijevanja i nedoumica. Sa stručnjacima postoji mogućnost rješavanja određenih nejasnoća oko softvera. Stručnjaci ne smiju postati jedini izvor informacija jer ne predstavljaju prosječnog korisnika.

3.5.5. Korisnici

Kupci softvera ne moraju biti i korisnici softvera koji su kupili. Treba uzeti u obzir i ljude koji kupuju softver, jer oni donose odluku koji će softver kupiti.

3.5.6. Tehnička podrška

Uzeti ljude za surogat korisnika može se činiti kao logičan odabir. Tehnička podrška provodi svoje vrijeme pričajući s korisnicima, tako da oni sigurno znaju koje su njihove potrebe. Međutim, to nije činjenica koja je utvrđena. Ljudi iz tehničke podrške će omogućiti ideje za aplikaciju kojim će lako podučavati korisnike.

3.5.7. Analitičari poslovanja ili sustava

Analitičari čine dobre surogat korisnike, jer provode puno vremena pričajući s korisnica i imaju dovoljno tehničkog znanja jer znaju na koji način sam sustav funkcionira.

Ukoliko se u ispitivanju koriste analitičari nužno je obratiti pozornost k tome da ne idu za vlastitom inicijativom nego da pomno i bez prestanka nastave slušati korisnike. Analitičari se ponekad znaju oslanjati na svoju intuiciju jer vjeruju kako imaju dovoljno iskustva da znaju što korisnici žele, a upravo to može predstaviti pogrešnu informaciju.

3.6. Testiranje prihvatljivosti korisničkih priča

Testovi bi trebali obuhvatiti što više detalja. S druge strane papira na kojem je napisana korisnička priča napisat će se testovi. Npr. testiraj s Visa, MasterCard i American Express kreditnim karticama. Testovi služe kako bi mogli odrediti jesu li uspješno implementirane korisničke priče. Definiranje testova prije pisanja programa može pomoći programerima da shvate poteškoće odmah i greške koje se mogu pojaviti u budućnosti [3].

Testovi se pišu:

- kad god developer i korisnik pričaju o korisničkoj priči i počnu ulaziti u detalje korisničke priče
- na početku iteracije, no prije što developeri počnu programirati
- kad se “otkriju” novi testovi tijekom ili poslije implementacije korisničke priče

Budući da se softver radi s ciljem zadovoljenja korisnika, korisnik je taj koji bi trebao specificirati testove prihvatljivosti. Korisnik treba raditi s developerom koji zna napisati test. Testeri trebaju opis softvera dobiti od korisnika, a ne od developera. Ukoliko tester dobije opis

od developera može se dogoditi da softver zaista funkcionira kako su developeri opisali, no to ne mora biti jednako onom opisu kojeg imaju korisnici. Developeri bi trebali raditi na tome da automatiziraju sve testove prihvatljivosti.

Treba uzeti u obzir i sljedeće testove:

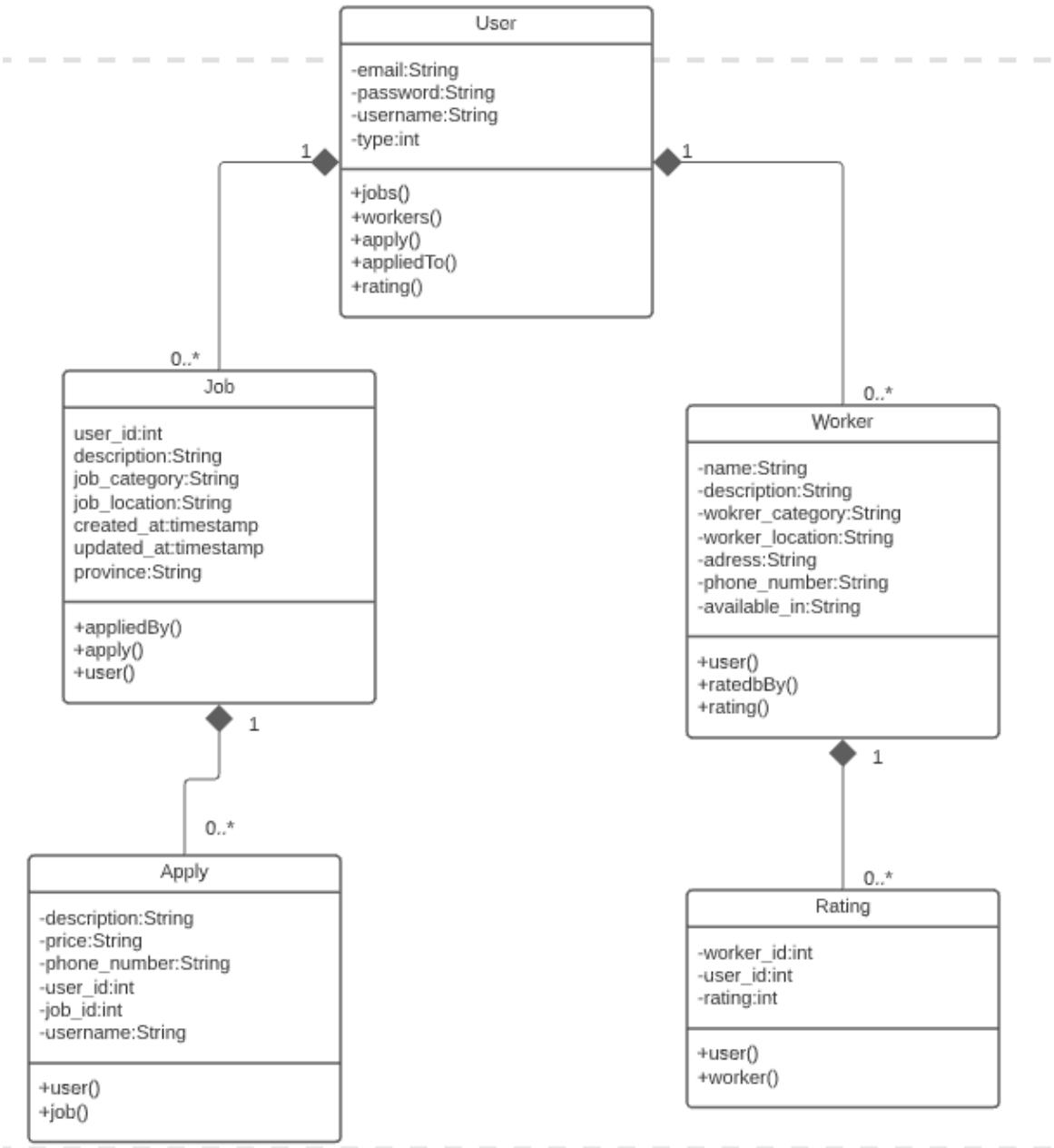
- testiranje UI (eng. *User Interface*) kako bi se uvjerili da se sve komponente ponašaju kako se to očekuje
- Ispitivanje upotrebljivosti, koliko se lako korisnik snalazi sa softverom
- Ispitivanje performansi, koliko se aplikacija ponaša dobro pod različitim opterećenjima unutar sustava

4. PRIMJENA UML DIJAGRAMA I KORISNIČKIH PRIČI NA WEB PORTALU

Workie je web portal koji omogućuje osobi da pronađe majstora ili majstoru da pronađe posao. Korisnici imaju dvije mogućnosti:

1. Osoba može odabrati županiju i vrstu majstora koja traži i portal će prikazati majstore.
2. Osoba može objaviti oglas u kojem opisuje kakvu uslugu treba te nakon toga njoj stižu ponude od majstora.

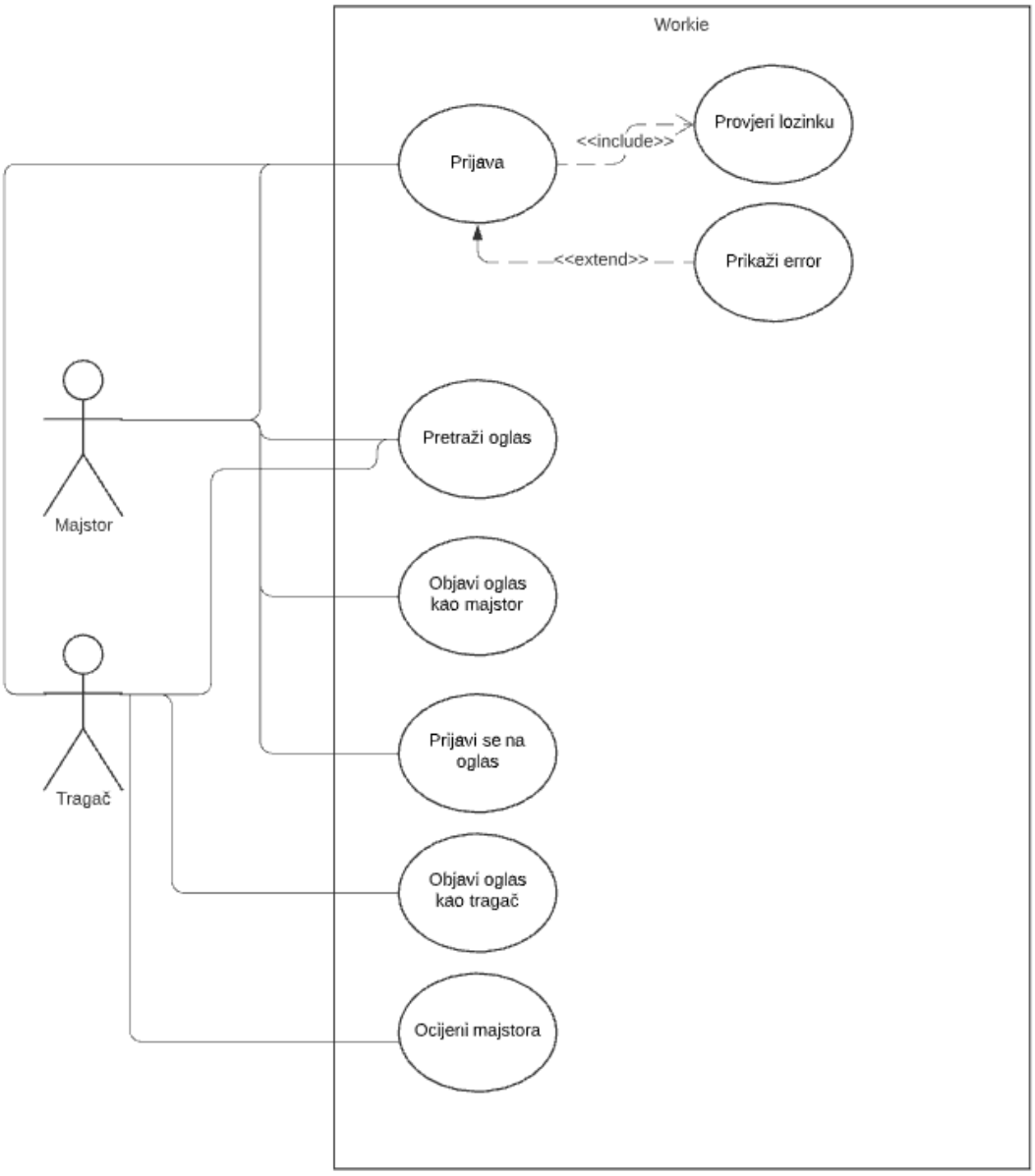
4.1. Dijagram klase za web portal



Slika 4.1. Prikaz dijagrama klase na primjeru web portala Workie

Klasa User može imati 0 ili više instanci klasa Job ili Worker. U slučaju da se instanca klase User obriše obrisale bi se i sve njoj podređene klase. Klasa Job može imati 0 ili više instanci klase Apply. U slučaju da se klasa Job izbriše sve instance klase Apply bi se također izbrisale. Klasa Worker može imati 0 ili više instanci klase Rating. Kad bi se obrisala instanca klase Worker obrisale bi se sve instance klase Rating.

4.2. Dijagram korištenja web portala



Slika 4.2. Dijagram korištenja web portala Workie

Postoje dva tipa korisnika: Majstor i Tragač. Kad se Majstor ili Tragač pokušaju prijaviti na portal dogodit će se provjera lozinke. Ukoliko lozinka bude netočna pokazat će se poruka s pogrešno unesenom lozinkom. Majstor može kreirati oglase gdje nudi svoje usluge, prijaviti se na oglas za posao, pretražiti oglase i prijaviti se na stranicu. Tragač može ocjenjivati majstore, postaviti oglas gdje traži uslugu, pretraživati oglase i prijaviti se na stranicu.

4.3.Primjena korisničkih priči na web portal

Kako bi uspješno istražile sve korisničke priče potrebno je napraviti surogat korisnika. Potrebna je osoba koja može reprezentirati korisnika.

4.3.1. Identificiranje korisničkih uloga

Potrebno je definirati tko će biti posjetitelj web portala kako bi zadovoljenje njihovih potreba bilo u potpunosti omogućeno i pravovremeno otkriveno.

Identificirane korisničke uloge:

- Iskusni majstor
- Internetski neobrazovan majstor
- Početni majstor
- Internetski obrazovan majstor
- Klasični tragač
- Napredni tragač

4.3.2 Sužavanje korisničkih uloga

Kad se identificiraju sve korisničke uloge, potrebno ih je razvrstati, te ukoliko postoje duplikati izbrisati. Najlakše je izbrisati karticu koja stoji jedna na drugoj. To znači da se te dvije korisničke uloge preklapaju 100%. Ne postoje korisničke uloge koje se 100% podudaraju.

Slika 4.3. Prikaz složenih kartica

Sa slike možemo vidjeti kako se Početni majstor i Internetski obrazovan majstor uvelike podudaraju. Početni majstor traga za poslovima pa će se on prijavljivati na oglase gdje osobe traže usluge koje on zadovoljava. Internetski obrazovan majstor će to također činiti. Njihove uloge su vrlo slične, tako da će se kartica Internetski obrazovan majstor ukloniti. Iskusni majstor je dugo u poslu, takvom majstoru Workie samo omogućuje da utvrdi potencijalne klijente. Takav majstor neće provoditi vrijeme na web portalu.

On će postaviti svoje usluge na Workie, a nakon toga neće više biti često aktivan na stranici. Internetski neobrazovan majstor također neće puno vremena provoditi na samoj stranicu, jer se teško sam snalazi na internetu. Takvom majstoru je internet novo područje u koje se može snaći samo uz pomoć druge osobe. Internetski neobrazovan majstor nije prioritet posluživanja na web portalu, pa je bilo nužno ukloniti i tu karticu. Nove korisničke uloge su prikazane na slici u daljnjem dijelu teksta.

Slika 4.4. Prikaz korisničkih uloga

4.3.3 Modeliranje korisnika

Ovisno o sljedećim karakteristikama korisnici će biti modelirani prema:

- Koliko će često koristiti softver
- Znanje o području iz kojeg dolazi
- Njegovo vještine s korištenjem interneta
- Njegova sposobnost da koristi naš softver
- Cilj koji želi postići koristeći naš softver

Definirane su četiri korisničke uloge:

1. **Iskusni majstor:** Iskusni majstor se dugo već bavi svojim poslom. Tijekom tih godina skupio je dovoljno kontakata tako da je uglavnom zauzet poslom mjesecima unaprijed. Kako uvijek traži izazove i bolje poslove objaviti će svoje usluge na našoj stranici. Iskusni majstor se neće prijavljivati na oglase gdje ljudi traže majstora.
2. **Početni majstor**: Vrlo vjerojatno je tek otvorio svoj obrt ili ga planira otvoriti. Shodno generaciji kojoj pripada na internetu se snalazi fluidno. Neće imati problema koristit web portal. On će postaviti oglas sa svojim uslugama, ali će se prijaviti i na oglase ljudi koji traže majstora, jer nema još veliku klijentelu.
3. **Klasični tragač:** Snalazi se na internetu, iako mu je povremeno potrebna pomoć u korištenju novih alata. On se neće najbolje snalaziti na stranici u području objavljivanja oglasa u kojem će opisati za što točno treba majstora. On će na tražilici portala pronaći majstora za kojeg smatra da će mu zadovoljiti trenutnu potrebu.

4. **Napredni tragač:** Odlično se snalazi na internetu te vrlo lako pronalazi majstora. Ukoliko nije siguran u majstore na web portalu, kreirati će oglas gdje će opisati koje su njegove potrebe.

Kada su definirane korisničke uloge, potrebno je odrediti koja je uloga najvažnija za web aplikaciju. Iskusni majstor neće koristiti stranicu često te on nije zainteresiran da puno vremena provodi na samoj stranici, on se dobro snalazi i bez nje. Iskusni majstor sigurno nije najvažniji korisnik za Workie. Početni majstor treba posao i on će učiniti sve da dođe do njega. Vješt je s računalima i posjećivati će Workie često. Koristit će sve mogućnosti web stranice. Početni majstor je vrlo važan za Workie. Klasični tragač neće posjećivati Workie često. Workie mu je opcija koju posjećuje tek kada iscrpi preporuke svojih prijatelja. Klasični tragač nema veliku važnost za web portal. Napredni tragač je po dobi između 25 i 40 godina, snalazi se na internetu te je svakodnevno na njemu. On se uzda u recenzije i preferira imati više opcija. Napredni tragač je uz Početnog majstora vrlo važan za Workie.

Na osnovu zaključka kreirana su dva korisnika Workia:

1. **Ivan -** otvorio je obrt prije 3 mjeseca. Iako u većini vremena ima posao koji obavlja, može se dogoditi da tjedno ne uspije pronaći dovoljan broj klijenata. U razdoblju kada nema posla Ivan će pomoću interneta pokušati pronaći posao, također će biti aktivan na Workie-u u tom razdoblju.
2. **Rafael** - preselio se iz Osijeka u Zagreb. Kupio je stan i želi ga renovirati. U potrazi je za majstorima i pronaći će ih preko interneta. Ne poznaje okolinu u kojoj se nalazi te nije u mogućnosti potražiti preporuku svojih poznanika. On će na web portalu Workie, uz pomoć recenzija drugih korisnika, uspjeti pronaći majstora za posao koji mu je potreban.

4.3.4 Korisničke priče

Budući da su za portal najvažniji majstori, započinje se s korisničkim pričama koje će zadovoljiti **Ivanove** potrebe.

Korisničke priče za korisnika Ivana:

- Majstor može pretraživati poslove po lokaciji i vrsti posla
- Majstor može vidjeti detalje o poslu
- Majstor se može prijaviti na oglas

- Majstor može kreirati oglas s vlastitim uslugama koje nudi
- Majstor može vidjeti svoje oglase na svom profilu
- Majstor može vidjeti oglase na koje se prijavio

Korisničke priče koje će zadovoljiti potrebe **Rafaela**:

- Tragač može tražiti majstore po lokaciji i vrsti posla
- Tragač može vidjeti detaljne informacije o majstoru
- Tragač može ocijeniti majstora
- Tragač može objaviti oglas gdje će navesti što mu točno treba
- Tragač može vidjeti majstore koji su mu se javili na oglas
- Tragač može vidjeti svoje oglase po profilu

4.3.5 Procjena trajanja implementacije korisničkih priči

U postupku su definirane i postavljene korisničke priče kojih ima ukupno dvanaest. One su prikazane u dolje navedenoj tablici. Nakon tog postupka potrebno je napraviti okvirni plan kada će aplikacija biti realizirana. U samom početku je potrebno poredati priče prema prioritetima.

Tablica 1. Korisničke priče poredane po prioritetima.

Sadržaj korisničkih priča
Majstor može pretraživati poslove po lokaciji i vrsti posla.
Tragač može tražiti majstore po lokaciji i vrsti posla.
Majstor može kreirati oglas s vlastitim uslugama koje nudi.
Tragač može objaviti oglas gdje će navesti što mu točno treba.
Majstor se može prijaviti na oglas.
Tragač može vidjeti majstore koji su mu se javili na oglas.
Majstor može vidjeti oglase na koje se prijavio.
Tragač može vidjeti detaljne informacije o majstoru.
Majstor može vidjeti detalje o poslu.
Tragač može vidjeti svoje oglase na profilu.
Tragač može ocijeniti majstora.
Majstor može vidjeti svoje oglase na svom profilu.

Kada je napravljen prioritet korisničkih priči potrebno je svaku priču bodovati tj. procijeniti koliko će implementacija priče trajati [3]. Što je priča kompleksnija to će bodovi biti veći i obrnuto.

Tablica 2. Korisničke priče koje su bodovane prema procjeni koliko će sama implementacija priče vremenski trajati

Sadržaj korisničkih priča	**Bodovi**
Majstor može pretraživati poslove po lokaciji i vrsti posla.	2
Tragač može tražiti majstore po lokaciji i vrsti posla.	2
Majstor može kreirati oglas s vlastitim uslugama koje nudi.	1
Tragač može objaviti oglas gdje će navesti što mu točno treba.	1
Majstor se može prijaviti na oglas.	1
Tragač može vidjeti majstore koji su mu se javili na oglas.	1
Majstor može vidjeti oglase na koje se prijavio.	1
Tragač može vidjeti detaljne informacije o majstoru.	0.5
Majstor može vidjeti detalje o poslu.	0.5
Tragač može vidjeti svoje oglase na profilu.	1
Majstor može vidjeti svoje oglase na svom profilu.	1
Tragač može ocijeniti majstora.	2

Bodovi su definirani uz pomoć referentne korisničke priče. Referentna korisnička priča mora biti određena korisnička priča za koju se većina developera slaže da nosi toliko broj bodova. U ovom slučaju je uzeta priča “Majstor može kreirati oglas s vlastitim uslugama koje nudi”, kao referentna priča. Mogla je to biti bilo koja korisnička priča s jednim bodom. Kada je određena referentna korisnička priča, težina drugih priči se može usporediti s onom referentnom.

4.3.6 Testiranje prihvatljivosti korisničkih priča

Kako bi bilo određeno jesu li korisničke priče uspješno implementirane, potrebno je provesti testove prihvatljivosti.

Test za tražilicu:

Korisnička priča:

- Majstor može pretraživati poslove po lokaciji i vrsti posla.

Test za korisničku priču **1**:

- Odaberi županiju npr. Vukovarsko-srijemsku te provjeri ispisuju li se oglasi samo za tu županiju.
- Odaberi kategoriju posla npr. elektroinstalater te provjeri ispisuju li se samo oglasi za tu kategoriju.
- Odaberi županiju npr. Zagrebačka i kategoriju npr. keramičar i provjeri ispisuju li se oglasi samo za navedene specifikacije.

Korisnička priča:

- Tragač može tražiti majstore po lokaciji i vrsti posla.

Test za korisničku priču **2**:

- Odaberi županiju npr. Vukovarsko-srijemsku te provjeri ispisuju li se oglasi samo za tu županiju.
- Odaberi kategoriju majstora npr. vodoinstalater te provjeri ispisuju li se samo oglasi za tu kategoriju.
- Odaberi županiju npr. Zagrebačka i kategoriju npr. bravar i provjeri ispisuju li se oglasi samo za navedene specifikacije.

Test za kreiranje oglasa:

Korisničke priče:

- Majstor može kreirati oglas s vlastitim uslugama koje nudi
- Tragač može objaviti oglas gdje će navesti što mu točno treba

Test za korisničku priču:

- Oglas je spremljen. Potrebno je potvrditi da se oglas spremio s unesenim podacima.

Korisnička priča:

- Majstor se može prijaviti na oglas

Test za korisničku priču:

- Prijavimo se na oglas te provjerimo postoji li prijava.
- Provjerimo podatke unesene na prijavi.

Korisnička priča:

- Majstor može vidjeti oglase na koje se prijavio

Test za korisničku priču:

- Provjerimo jesu li prikazani oglasi na kojima smo se mi zaista prijavili.

Korisnička priča:

- Tragač može vidjeti majstore koji su mu se javili na oglas

Test za korisničku priču:

- Provjerimo prikazuju li se svi majstori koji su se prijavili na oglas
- Provjerimo jesu li prikazane sve informacije prijave

Detalji o oglasima:

Korisnička priča:

- Majstor može vidjeti detalje o poslu

Test za korisničku priču:

- Provjerimo ispisuju li se sve informacije o oglasu

Korisnička priča:

- Tragač može vidjeti detaljne informacije o majstoru

Test za korisničku priču:

- Provjerimo ispisuju li se sve informacije o oglasu

Profil:

Korisnička priča:

- Tragač može vidjeti svoje oglase na profilu

Test za korisničku priču:

- Provjerimo jesu li prikazani svi oglasi koje je kreirao tragač.

Korisnička priča:

- Majstor može vidjeti svoje oglase na svom profilu

Test za korisničku priču:

- Provjerimo jesu li prikazani svi oglasi koje je kreirao majstor.

Ocjena majstora:

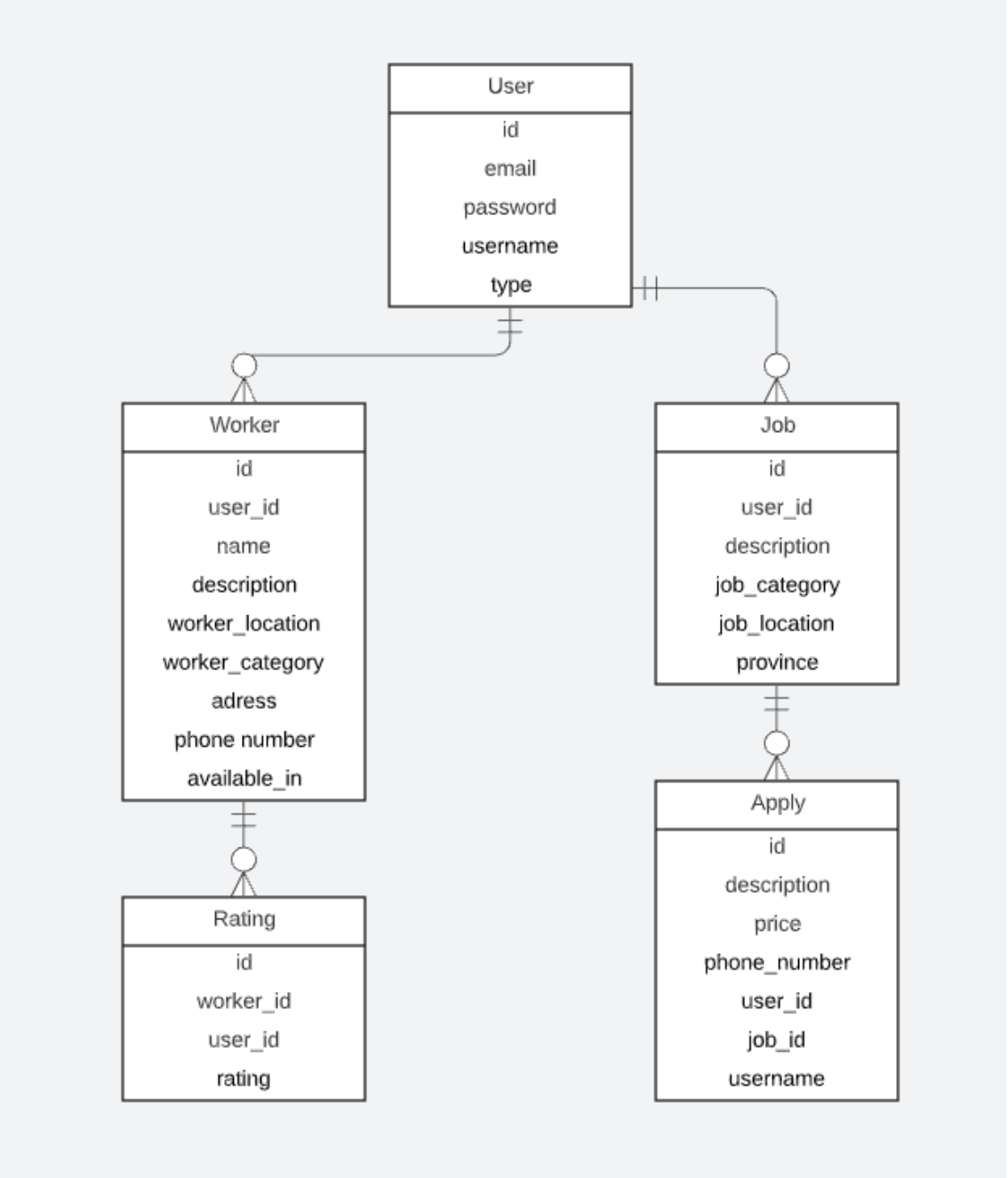
Korisnička priča:

- Tragač može ocijeniti majstora

Test za korisničku priču:

- Provjeri da li ocijenjen zaista majstor kojeg smo imali namjeru ocijeniti.
- Provjeri da li se prosjek ocjene ažurirao.
- Provjeri prikazuje li se ponovno traka za ocjenjivanje.

4.4 E-R dijagram web aplikacije



Slika 4.5. Prikaz E-R dijagrama za web aplikaciju

Korisnik može imati više oglasa za radnika ili oglasa za posao. Oglasi za radnika ili za posao pripadaju samo jednom korisniku. Ocjena pripada samo jednom radniku, ali radnik može imati više ili niti jednu ocjenu. Oglas za posao može imati više ili niti jednu prijavu za posao. Svaka prijava pripada samo jednom oglasu za posao.

5. ZAKLJUČAK

U ovom završnom radu obrađena je tematika UML dijagrama, korisničke priče te primjena istih na web portalu Workie. UML dijagrami pomažu da se kompleksnije stvari shvate na jednostavniji način. Često se događa da među developerima vladaju nedoumice. Pomoću UML dijagrama postoji mogućnost da se na grafički način predočite složeni sustavi i tako pojednostave shvaćanje samog sustava. Postoje različite vrste UML dijagrama. Ovisno o potrebi koristit će se različiti dijagrami. Najčešće za predočavanje kompleksnog sustava koristi se više različitih UML dijagrama.

Korisničke priče pomažu prilikom vrlo efikasnog pronalaska korisnika, pronalaženju značajki koje su najvažnije za implementaciju te posljednje, ali ne i manje bitno, pomažu prilikom izračuna koliko je vremena potrebno prilikom izrade jednog takvog sustava. Korisničke priče potiču komunikaciju i suradnju među timom. Ukoliko se slijedi koncept korisničkih priči neće biti potrebe trošiti vrijeme i resurse za značajke koje korisnicima nisu nužno potrebne. Pri završetku pisanja završnog rada primijenjeni su UML dijagrami i korisničke priče na web portal.

Primjenom ova dva koncepta potvrdila je njihovu upotrebljivost pri implementaciji sustava. Primjena UML korisničkih priči omogućava dobivanje kvalitetnijih softvera koji će biti razvijeni u što manje vremena te će biti efikasniji. Sama svrha pisanja i implementacije završnog rada aplicira se u korisnicima kojima mora biti olakšan način korištenja web portala ili bilo koje druge stranice ili aplikacije na internetu. Stvoriti kvalitetnu uslugu postaje sve kompleksniji posao s obzirom na konkurenciju, a onaj tko uspije najprije zadovoljiti korisnika uživat će u blagodati svoga rada.

LITERATURA

[1] Hans van Vliet „Software Engineering: Pinciples and Practice“

[2] Martin Fowler „UML distilled 3rd Edition: A brief guide to the standard object modeling language“

[3] Mike Cohn „User Stories Applied“

https://www.visual-paradigm.com/scrum/what-is-story-point-in-agile/ (Pristupljeno dana: 12.07.2021.)

SAŽETAK

Pojavom objektno orijentiranih programskih jezika pojavljuju se različiti pristupi dizajniranja programskih sustava. UML nastaje 1994. godine te nudi cjelovito rješenje u dizajniranju programskih sustava. UML se može primijeniti na velikom broju programskih domena(bankarski sustavi, financijski sustavi, zdravstveni sustavi i mnogim drugim sustavima). Danas se najčešće koriste: dijagram korištenja, klasni dijagrami i objektni dijagrami.

Korisničke priče spadaju u agilne metode razvoja. Stavljaju u fokus korisnika i njegove potrebe. Korisničke priče pomažu u definiranju zahtjeva koje sustav mora ispuniti kako bi krajnji korisnici bili zadovoljni sustavom.

Zadnju cjelina koju završni rad obrađuje jest primjena UML dijagrama i korisničkih priča na web portal.

KLJUČNE RIJEČI: UML, UML jezik, UML dijagram, korisničke priče, web portal

ABSTRACT

APPLICATION OF UML LANGUAGE AND USER STORIES IN CREATING WEB PORTAL SPECIFICATIONS

With the emergence of object-oriented programming languages, different approaches to designing programming systems appear. UML was created in 1994 and offers a complete solution in designing software systems. UML can be applied to many programming domains (banking systems, financial systems, health systems and many other systems). Today, the most commonly used are: usage diagram, class diagrams and object diagrams.

User stories belong to agile development methods. They focus on the user and his needs. User stories help define the requirements that the system must fulfill in order for end users to be satisfied with the system.

The last part that the final thesis deals with is the application of UML diagrams and user stories to the web portal.

KEYWORDS: UML, UML language, UML diagram, user stories, web portal