Razvoj softvera

dr.sc. Emir Mešković

VII predavanje

Sadržaj predavanja

- Pregled procedura za dizajn softvera
- UML
  - Dijagrami slučajeva upotrebe (use case diagrams)

Inženjerske metodologije

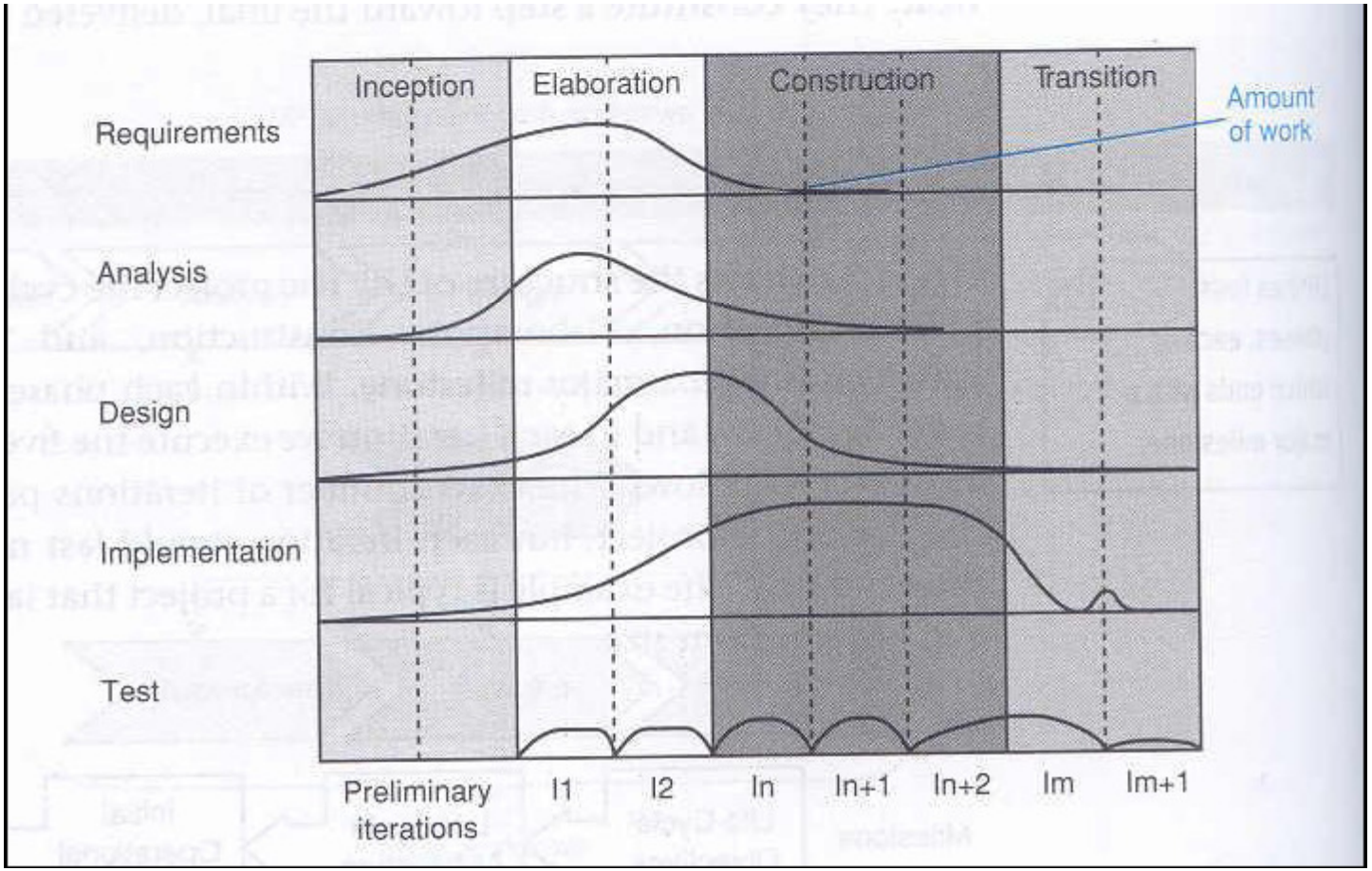
- Faze za klasični inženjerski projekat
  - Analiza zahtjeva
  - Dizajn
  - Konstrukcija (implementacija)
  - Testiranje
- Krajnji ciljevi za softverski proizvod
  - Korisnost
  - Pouzdanost
  - Fleksibilnost
  - Preglednost
  - Efikasnost

Proces razvoja softvera

- *Unified Process* (UP) – metodologija koja određuje kako se treba odvijati proces razvoja softvera
- Sastoji se od određenih faza, s time da se svaka faza realizira kroz simultano izvršavanje određenih temeljnih aktivnosti
- Propisuje rezultate koje pojedine faze odnosno aktivnosti trebaju proizvesti, no ne određuje način kako se ti rezultati trebaju dokumentirati
- Obično se koristi za razvoj objektno-orijentisanih sistema, makar je primjenjiv i na druge sisteme.
- Predstavlja labaviju "public domain" verziju komercijalne metodologije RUP (*Rational Unified Process).*

Struktura UP-a

- Proces razvoja softvera u skladu s UP prikazan je sljedećom slikom.



Struktura UP-a

- Struktura UP ili životni ciklus projekta je podijeljen u četiri faze – *Inception*, *Elaboration*, *Construction* i *Transition*
- Pojedina faza završava kad dostigne njen prag (*milestone* – skup uslova koje treba zadovoljiti), dakle kad se postignu njeni ciljevi
- U okviru svake faze može biti jedna ili više iteracija i u svakoj iteraciji se izvršava pet temeljnih aktivnosti (*core workflows*)
- Ipak, u svakoj fazi postoji jedna ili dvije aktivnosti na koje je stavljen posebni naglasak i koje se obavljaju s povećanim intenzitetom

Temeljne aktivnosti u UP

- Zahtjevi (*requirements*). Utvrđuje se šta sistem treba raditi.
- Analiza (*analysis*). Zahtjevi se analiziraju, profinjuju i strukturiraju.
- Oblikovanje (*design*). Predlaže se građa sistema koja će omogućiti da se zahtjevi realiziraju.
- Implementacija (*implementation*). Stvara se softver koji realizira predloženu građu.
- Testiranje (*test*). Provjerava se da li stvoreni softver zaista radi onako kako bi trebao.

Faze UP

- *Inception (početak)*
  - ***Ciljevi:*** dokazati izvedivost i isplativost projekta, utvrditi ključne zahtjeve da bi se vidio kontekst (doseg) sistema, prepoznati rizike.
  - ***Naglasak:*** zahtjevi i analiza.
  - ***Prag:*** Ciljevi životnog ciklusa (*Life Cycle Objectives)*
- *Elaboration (razrada)*.
  - ***Ciljevi:*** stvoriti izvršivu jezgru arhitekture sistema. Profiniti procjenu rizika, definirati atribute kvalitete, evidentirati use case-ove koji pokrivaju 80% funkcionalnih zahtjeva, stvoriti detaljni plan za fazu konstrukcije.
  - ***Naglasak:*** zahtjevi, analiza, oblikovanje.
  - ***Prag:*** Arhitektura životnog ciklusa (*Life Cycle Architecture*)

Faze UP

- *Construction (izgradnja)*.
  - ***Ciljevi:*** dovršiti zahtjeve, analizu i oblikovanje, te dograditi jezgru arhitekture do konačnog sistema. Pritom treba očuvati integritet arhitekture i oduprijeti se pritiscima da se sistem dovrši na brzinu. Obaviti beta test i pokrenuti sistem.
  - ***Naglasak:*** oblikovanje, implementacija.
  - ***Prag:*** Početna operativna sposobnost (*Initial Operational Capability*)
- *Transition (prijelaz)*.
  - ***Ciljevi:*** popraviti uočene greške, pripremiti sistem za rad u korisničkoj okolini, distribuirati ga na korisnička radna mjesta, stvoriti korisničku dokumentaciju, organizirati podršku korisnicima, napraviti "*post-project review*".
  - ***Naglasak:*** implementacija, test.
  - ***Prag:*** Izdavanje prozivoda (*Produkt Release*)

UML definicija

- UML (*Unified Modeling Language*) je jezik za grafičku predstavu, vizualiziranje, konstruisanje i dokumentiranje softverskih sistema.
- UML nudi standard za projektovanje sistema
- UML je općeniti jezik za modeliranje koji se koristi grafičkim prikazom za izradu abstraktnog modela posmatranog sistema (UML model).
- Obično se koristi za modeliranje objektno-orijentisanih softverskih sistema, mada je primjenjiv i na druge sisteme
- Ne implicira nikakvu metodologiju za modeliranje, već samo omogućuje da se rezultati modeliranja dokumentuju na razumljiv način

projekat kuće -> kuća

UML dijagram sistema -> kod sistema

Veza između UML i UP

- Makar su formalno nezavisni jedan od drugog, UML i UP su u bliskoj vezi i skladno se nadopunjuju.
- UP određuje kako će se odvijati proces razvoja softvera, a UML omogućuje da se rezultati tog razvoja dokumentiraju.
- UML i UP su zapravo nastali zajedno, u okviru istog projekta, s namjerom da budu komplement jedan drugom.
- Ono što danas zovemo UML zapravo je dio tog projekta koji se odnosi na vizualni jezik, a ono što zovemo UP je dio projekta koji se odnosi na proces.
- Ipak, postoji jedna razlika: UML je prihvaćen kao svjetski standard, a UP nije.
- Većina softverskih inženjera doživljava UML i UP kao cjelinu i primjenjuju ih na razvoj OO sistema.

Građa UML modela

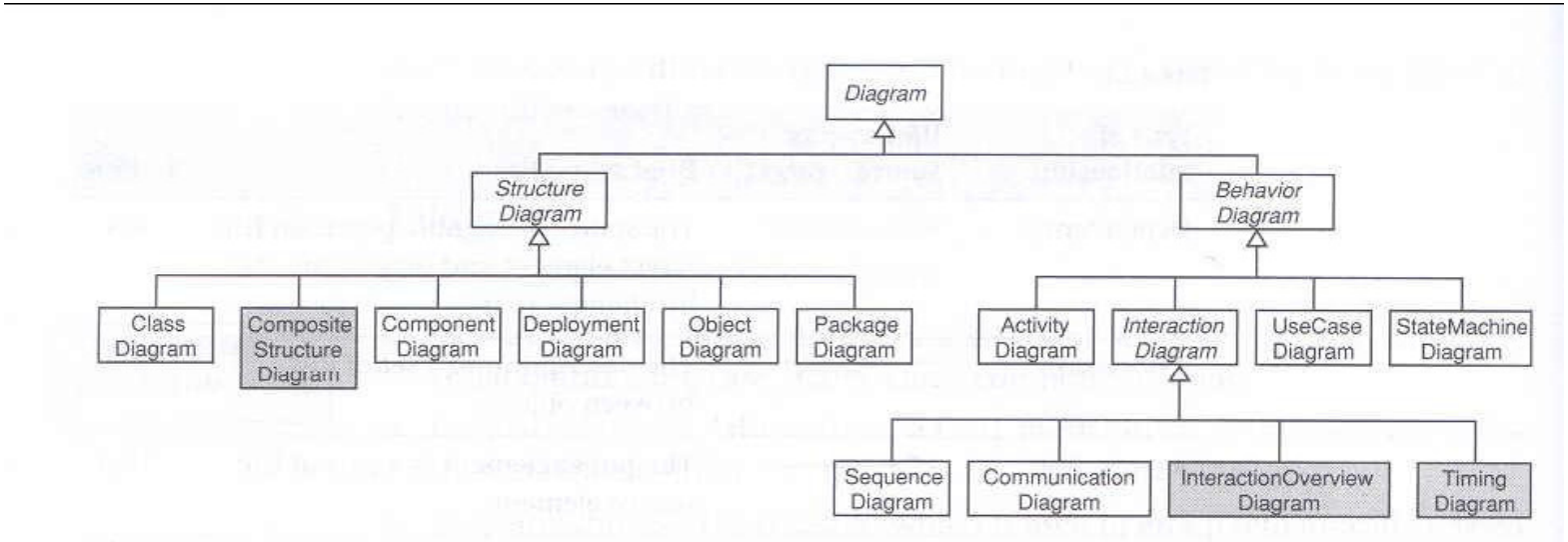
- Kao rezultat modeliranja nastaje UML *model sistema koji je sastavljen od sljedećih dijelova.*
  - Stvari (*things*) – elementi modela, na primjer klase, *interface-i*, komponente, računarski čvorovi, …
  - Veze (*relationships*) – povezuju stvari i određuju kako se stvari semantički odnose jedne prema drugima
  - Dijagrami – pogledi na model. Na njima su nacrtane neke stvari (kućice) i neke veze (spojnice). Služe za vizualizaciju strukture ili ponašanja sistema.
- Važno je naglasiti da dijagram ili skup dijagrama nije isto što i model, makar to ljudi često tako doživljavaju, te makar se model obično implicitno stvara baš crtanjem dijagrama.

Modeli i dijagrami

- Dijagram je djelimična reprezentacija sistema
- Model nadopunjuje dijagrame potrebnom dokumentacijom i opisima
- Postoje tri osnovna modela:
- Funkcionalni model
  - Prikazuje funkcionalnost sistema sa stajališta korisnika
  - Sadrži use case dijagrame
- Objektni model
  - Prikazuje strukturu i podstrukturu sistema koristeći se objektima, atributima, operacijama i relacijama
  - Sadrži class dijagrame
- Dinamički model
  - Prikazuje interno ponašanje sistema
  - Sadrži sequence i activity dijagrame

Modeli i dijagrami

- UML dijagrami se mogu podijeliti na:
  - Dijagrame ponašanja
    - Activity, state machine, interaction i use case dijagrami
  - Dijagrame strukture
    - Class, Component, Composite structure, Deployment, Object i Package dijagrami



Modeli i dijagrami

- *Use case* dijagram pokazuje funkcionalnost sistema u smislu glumaca (*actors*), njihovih ciljeva (*goals*) predstavljenih kao use-cases, i bilo koju zavisnost među tim slučajevima
- *Class* dijagram opisuje strukturu sistema prikazujući klase sistema, njihove atribute, kao i veze između klasa
- *Object* dijagram prikazuje potpun ili djelomičan pogled na strukturu modeliranog sistema u određenom trenutku vremena
- *Package* dijagram opisuje kako je sistem podijeljen u logičke grupacije na način da prikazuje zavisnosti među tim grupacijama
- *Component* dijagram opisuje kako je softverski sistem podjeljen u komponente i prikazuje vezu između tih komponenata
- *Deployment* dijagram služi da modelira hardver koji je korišten za implementaciju sistema izvršnog okruženja

Koncepti

- Struktura
  - Učesnik, Atribut, Klasa, Komponenta, Intefejs, Objekat, Paket
- Ponašanje
  - Aktivnost, Događaj, Poruka, Metoda, Operacija, Stanje
- Odnos
  - Asocijacija, Generalizacija, Ovisnost, Implementacija, Kompozicija

Zašto koristiti UML

- Zbog poboljšanja kvalitete softvera i smanjenja troškova i vremena potrebnog da se proizvod izbaci na tržište
- Zbog upravljanja složenosti sistema
- Za rješavanje arhitektonskih problema, kao što su fizičke distribucije, replikacije konkurentnosti, sigurnost, balansiranje opterećenja i tolerancija kvarova
- Da se fokusiramo na problem dizajniranja a ne na programiranje

Use case dijagram

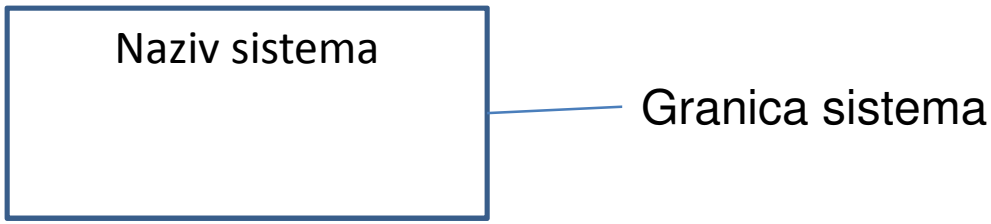
- Opisuje ponašanje sistema iz perspektive vanjskog posmatrača
- Odnosi se na pojedine scenarije koji opisuju šta se događa u sistemu kada se s njim stupi u interakciju
- Slučaj upotrebe (*Use Case*) je upravo skup scenarija koji povezuje određeni zadatak
- Učesnik (*Actor*) je neko ili nešto što inicira određeni događaj
- Najčešće se koristi za:
  - Uvid u funkcionalnost sistema
  - Identifikaciju korisnika sistema
  - Identifikaciju područja u kojem dolazi do interakcije između čovjeka i računara

Modeliranje slučajeva upotrebe

- Pronaći granice sistema
- Pronaći učesnike (aktere)
- Pronaći slučajeve upotrebe
  - Specificirati slučaj upotrebe
  - Kreirati scenarije
- Akteri – uloge koje obavljaju ljudi ili stvari koji koriste sistem
- Slučajevi upotrebe – stvari koje akteri mogu obaviti sa sistemom
- Veze – veze koje imaju značenje između aktera i slučajeva upotrebe
- Granica sistema – okvir iscrtan oko slučajeva upotrebe koji označava granicu sistema koji se modelira

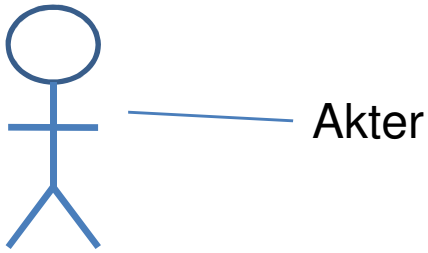
Granica sistema

- Odluka gdje se nalaze granice sistema
  - šta je sastavni dio sistema (unutar granica sistema), a šta vanjština sistema (izvan granica sistema)
- Definiše se od strane onoga ko koristi sistem (tj. aktera)
- Koje specifične koristi sistem nudi ovim akterima (tj. slučajeve upotrebe)
- Kako se definišu akteri i slučajevi upotrebe granica sistema postaje sve jasnije i jasnije definisana



Akteri

- Specificira ulogu koju neku vanjski entitet usvaja prilikom direktne interakcije sa sistemom
- Može predstavljati ulogu korisnika ili ulogu koju igra drugi sistem koji dotiče granicu sistema
- Uvijek su izvan sistema, ali sistem može čuvati neku internu reprezentaciju jednog ili više aktera
- Kako bi se identifikovali potrebno je razmotriti tko ili šta koristi sistem i koje uloge igraju u svojim interakcijama sa sistemom

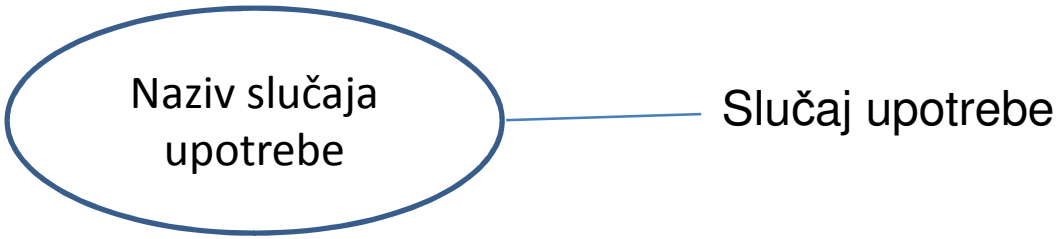


Akteri

- Prilikom modeliranja aktera potrebno je upamtiti sljedeće:
  - Akteri se uvijek nalaze izvan sistema
  - Akteri vrše interakciju direktno sa sistemom
  - Akteri predstavljaju uloge koje ljudi i stvari igraju u relaciji sa sistemom, a ne specifične ljude i specifične stvari
  - Jedna osoba ili stvar može igrati više uloga u relaciji sa sistemom, simultano ili tokom vremena (npr. trener i autor kursa)
  - Svaki akter zahtjeva kratak naziv koji ima smisla s poslovne perspektive.
  - Svaki akter mora imati kratak opis (jedna ili dvije linije) koji opisuje šta je taj akter sa poslovne perspektive
  - Kao i klase, akteri mogu imati odjeljak koji prikazuje atribute aktera i događaje koje može prihvatiti

Slučajevi upotrebe

- Definiše se kao specifikacija sekvenci akcija koje sistem, podsistem ili klasa može izvoditi pri interakciji sa vanjskim akterima
- Nešto što akter želi da sistem obavlja
- To je slučaj upotrebe sistema od strane specifičnog aktera:
  - uvijek su pokrenuti od strane aktera
  - uvijek se pišu sa tačke gledišta aktera
- Razmotriti kako će svaki akter koristiti sistem
- Kako se identifikuju slučajevi upotrebe također se mogu otkriti i novi akteri



Detaljna specifikacija slučaja upotrebe

- Nakon kreiranja use case dijagrama i identifikacije aktera i ključnih slučajeva upotrebe
- Ne odvija se egzaktnim redoslijedom
- Specificira se samo neki ili svi slučajevi upotrebe
- Izlaz ove aktivnosti je detaljniji use case
- Ne postoji UML standard za specifikaciju slučajeva upotrebe

Detaljna specifikacija slučaja upotrebe

- Naziv slučaja upotrebe – jednoznačno određuje slučaj upotrebe
- Identifikator slučaja upotrebe – koristan ukoliko se imena vremenom mijenjaju
- Kratak opis slučaja upotrebe – navodi cilj i bitna svojstva slučaja upotrebe
- Popis aktera
  - Primarni – pokreću slučaj upotrebe
  - Sekundarni – stupaju u interakciju sa slučajem upotrebe nakon što je već pokrenut

Detaljna specifikacija slučaja upotrebe

- Preduslovi (*preconditions*) – stvari koje moraju biti istinite prije nego se slučaj upotrebe može izvršiti
  - ograničenja na stanje sistema
  - jednostavni iskazi o stanju sistema koji će se evaluirati kao istina ili laž (true ili false)
- Tok događaja – koraci u slučaju upotrebe koji su deklarativni, numerisani i vremenski uređeni
- Naknadni uslovi (*postconditions*) – stvari koje moraju biti istinite na kraju slučaja upotrebe

Detaljna specifikacija slučaja upotrebe

Use case: Plaćanje PDV-a	Naziv slučaja upotrebe
ID: UC1	Jedinstveni identifikator
Akteri: Vrijeme Vlada	Akteri uključeni u slučaj upotrebe
Preduslovi: 1. Kraj poslovnog kvartala	Stanje sistema prije početka slučaja upotrebe
Tok događaja: 1. Slučaj upotrebe počinje na kraju poslovnog kvartala 2. Sistem određuje visinu poreza na dodanu vrijednost (PDV) koja se plaća vladi 3. Sistem šalje elektroničko plaćanje vladi	Stvarni koraci slučaja upotrebe
Postuslovi: 1. Vlada dobija tačan iznos PDV-a	Stanje sistema kada se okonča slučaj upotrebe

Detaljna specifikacija slučajeva upotrebe

- U toku događaja mogu se koristiti ključne riječi:
  - *if* za indiciranje grane u toku događaja
  - for i while za ponavljanje akcija u toku događaja
- Kada nije jednostavno izraziti grananje korištenjem *if*
  - Dodaje se nova sekcija na kraj slučaja upotrebe za alternativni tok koja mora sadržavati sljedeće:
    - Tok događaja za altrenativni tok - najviše nekoliko koraka i uvijek mora početi sa Boolean uslovom koji uzrokuje da tok bude izvršen
    - Postuslove za ovaj tok događaja

Scenariji

- Kada se uoči složenost koja se ne može reducirati moraju se formulisati složene slučajeve upotrebe
- Jednostavnije je i manje sklono greškama modelirati mreže grananja kroz glavne tokove kao odvojene scenarije
- Scenariji su drugi način viđenja slučajeva upotrebe.
- Scenario je jedan specifičan put kroz slučaj upotrebe
- Važna osobina scenarija je da se oni ne granaju
- Svaki slučaj upotrebe ima tačno jedan primarni scenario
- Svaki slučaj upotrebe također ima više sekundarnih scenarija – ovo su alternativni putevi primarnom scenariju kroz tok događaja

Scenariji

Use case: Odjavljivanje
ID: UC14
Akteri: Kupac
Preduslovi:
Primarni scenario: 1. Slučaj upotrebe počinje kada kupac odabere „idi na odjavu“ 2. Sistem prikazuje narudžbu kupca 3. Sistem zahtjeva identifikator kupca 4. Kupac unosi validan identifikator kupca 5. Sistem dohvata i prikazuje detalje o Kupcu 6. Sistem zahtjeva informacije o kreditnoj kartici – ime na kartici, broj kartice i datum isteka 7. Kupac unosi informacije o kreditnoj kartici 8. Sistem zahtjeva potvrdu narudžbe 9. Kupac potvrđuje narudžbu 10. Sistem debituje kreditnu karticu 11. Sistem prikazuje račun
Sekundarni scenariji: PogrešanIdentifikatorKupca PogrešniDetaljiKreditneKartice PremašenLimitKreditneKartice IsteklaKreditnaKartica
Postuslovi:

Scenariji

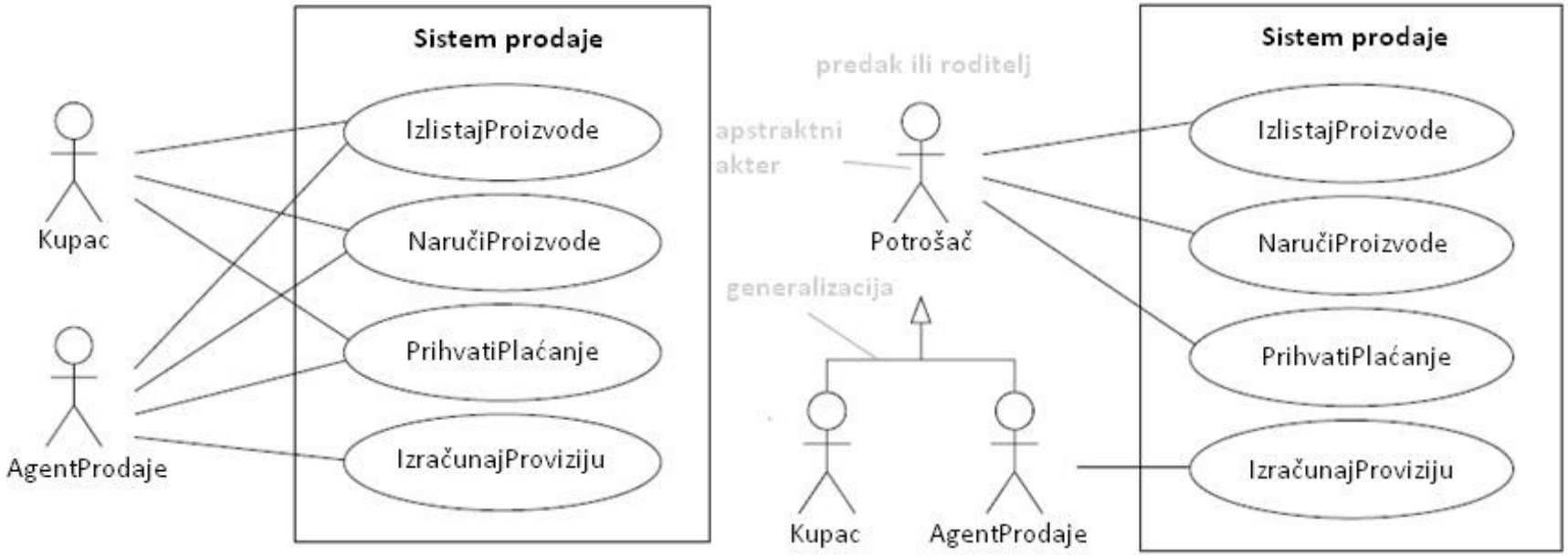
- Sekundarni scenario se specificira na isti način na koji se specificiraju slučajevi upotrebe
- Uvijek se mora jasno izraziti kako scenario počinje i osigurati da predstavlja samo jedan specifičan put bez grananja kroz tok događaja.
- Iz svakog sekundarnog scenarija mora se moći vratiti nazad u njegov slučaj upotrebe
- U svakom koraku primarnog scenarija treba potražiti:
  - Moguće alternativne tokove
  - Greške koje se mogu pojaviti
  - Prekidi koji se mogu pojaviti u toku – stvari koje se mogu dogoditi u bilo kom trenutku
- Pokušati ograničiti broj sekundarnih scenarija na neophodan minimum

Scenariji

Use case: Odjavljivanje **Sekundarni scenario: PogrešanIdentifikatorKupca**
ID: UC15
Akteri: Kupac
Preduslovi:
Sekundarni scenario: 1. Slučaj upotrebe počinje u koraku 3 slučaja upotrebe Zaključivanje kada kupac unese porešan identifikator kupca 2. For tri pogrešna unosa   1. Sistem traži od Kupca da ponovo unese identifikator kupca 3. Sistem informiše kupca da njegov identifikator kupca nije prepoznat
Postuslovi:

Generalizacija aktera

- Dosta istovjetnosti između dva aktera, u pogledu načina na koji vrše interakciju sa sistemom – postoji zajedničko ponašanje aktera koje bi moglo biti izvedeno u generalizovanog aktera

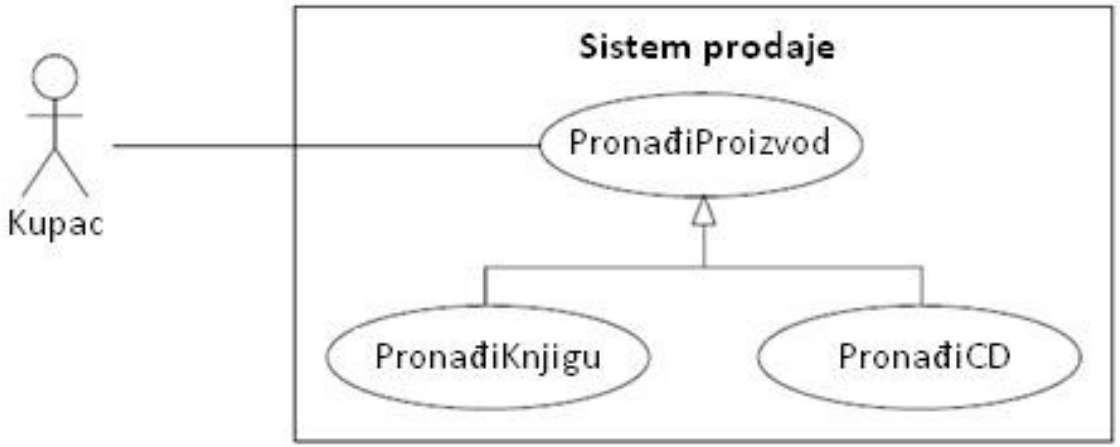


Generalizacija slučaja upotrebe

- Koristi se kada postoji jedan ili više slučajeva upotrebe koji su specijalizacije opštijeg slučaja
- Slučaj upotrebe dijete predstavlja specifičniju formu roditelja
- Djeca mogu:
  - Naslijediti osobine od slučaja upotrebe roditelja (normalan tekst u UML-u)
  - Dodati nove osobine (podebljani tekst u UML-u)
  - Predefinisati (promijeniti) naslijeđene osobine (ukošeni tekst u UML-u)

Generalizacija slučaja upotrebe

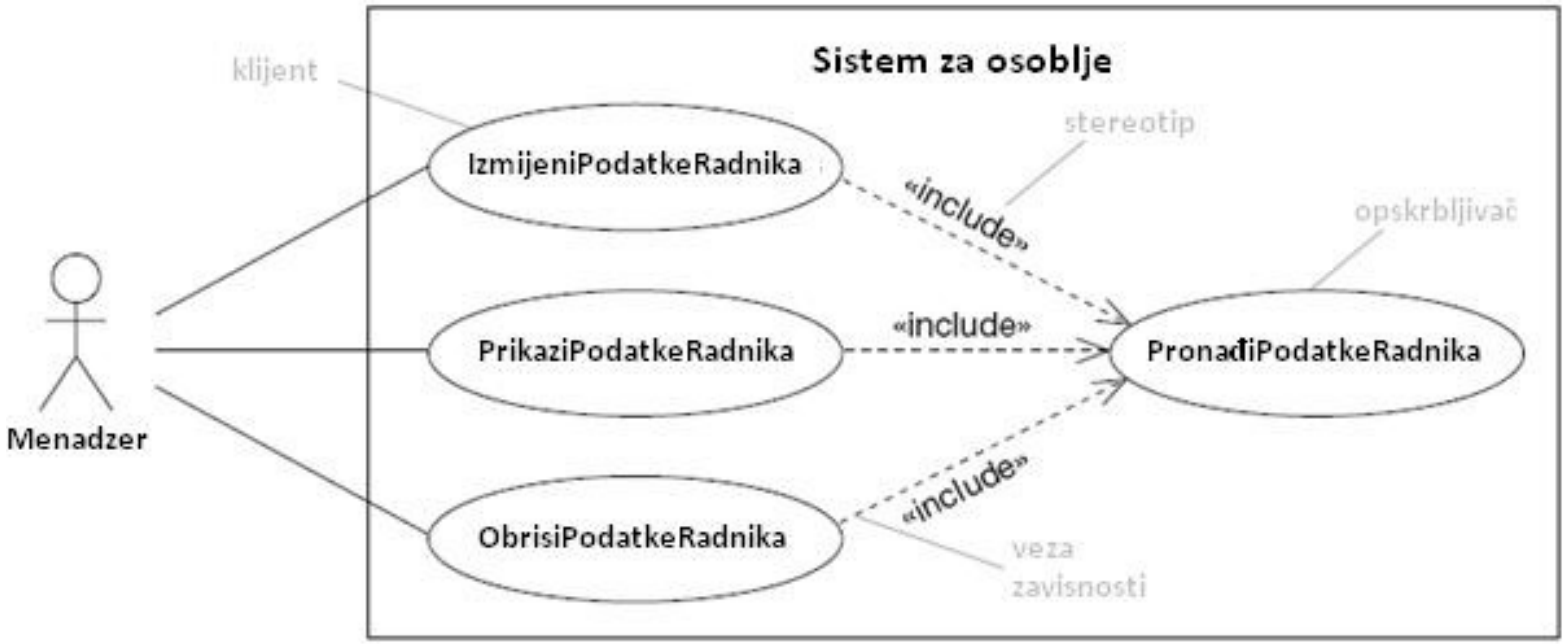
- Ukoliko slučaj upotrebe roditelj nema tok događaja, ili ima tok događaja koji je nekompletan, onda je to apstraktni slučaj upotrebe
- Oni nikada ne mogu biti izvršeni od strane sistema



Uključivanje manjeg slučaja upotrebe u veće - <<include>>

- Pisanje slučajeva upotrebe povremeno može biti sa dosta ponavljanja
- <<include>> veza između slučajeva upotrebe omogućava uključivanje ponašanja slučaja upotrebe *opskrbljivača* u tok *klijentskog* slučaja upotrebe
- Mora se specificirati egzaktna tačka u klijent slučaju upotrebe gdje se želi uključiti ponašanje slučaja upotrebe opskrbljivača
- Klijent slučaj upotrebe se izvršava do tačke uključivanja, onda se izvršavanje prenosi na slučaj upotrebe opskrbljivač. Kada opskrbljivač završi, kontrola se ponovo vraća klijentu

Uključivanje manjeg slučaja upotrebe u veće - <<include>>



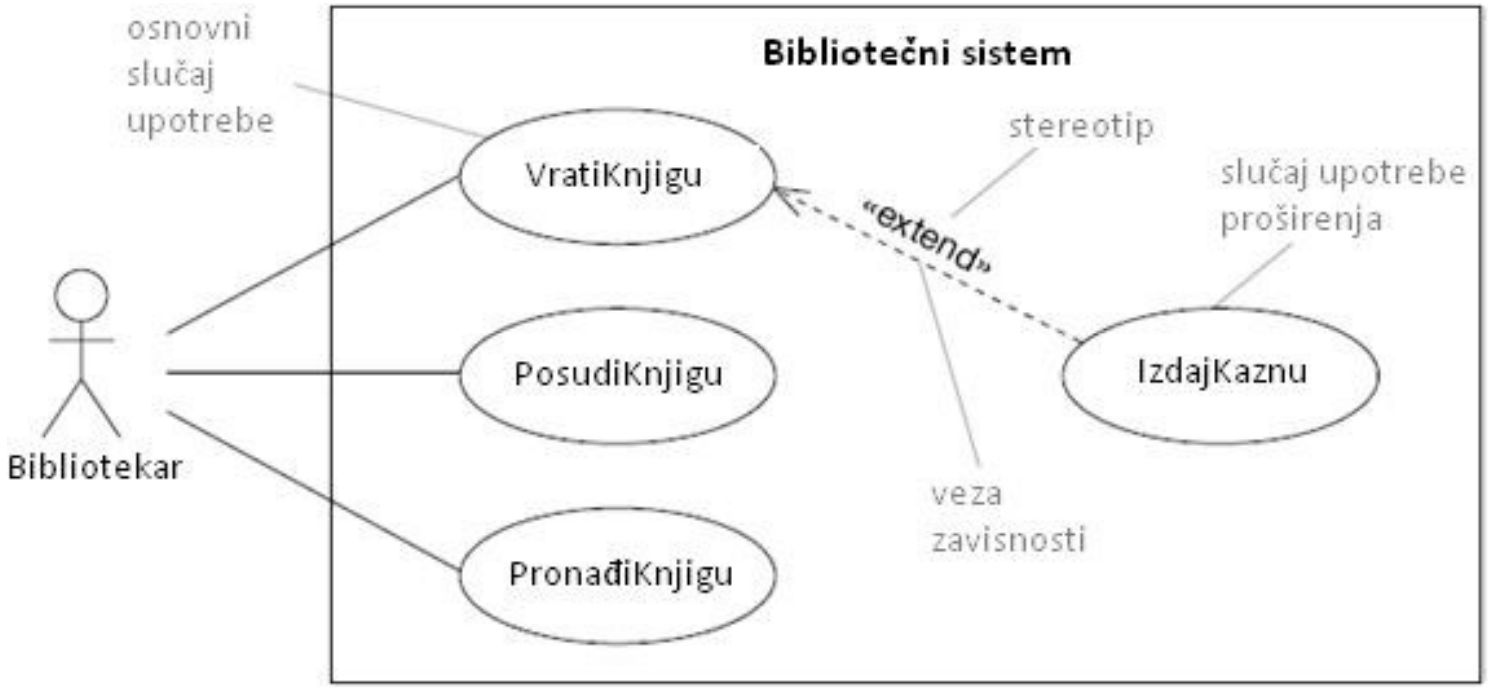
Uključivanje manjeg slučaja upotrebe u veće - <<include>>

PromijeniPodatkeRadnika	PrikaziPodatkeRadnika
ID: UC1	**ID: UC2**
Akteri: Menadžer	**Akteri:** Menadžer
Preduslovi: 1. Menadžer je prijavljen na sistem	**Preduslovi:** 1. Menadžer je prijavljen na sistem
Tok događaja: 1. Menadžer unosi ID broj radnika 2. Include(PronađiPodatkeRadnika) 3. Menadžer bira dio podataka o radniku za izmjenu 4. ...	**Tok događaja:** 1. Menadžer unosi ID broj radnika 2. Include(PronađiPodatkeRadnika) 3. Sistem prikazuje podatke o radniku 4. ...
Postuslovi:	**Postuslovi:**

<<extend>>

- Obezbjeđuje način da se dodaju nova ponašanja postojećem slučaju upotrebe
- Osnovni slučaj upotrebe obezbjeđuje skup tačaka proširenja (*zakačke*) gdje se može dodati novo ponašanje
- Slučaj upotrebe proširenja obezbjeđuje skup segmenata koji mogu biti ubačeni u osnovni slučaj upotrebe
- Veza sama po sebi se može koristiti da tačno specificira koje tačke proširenja u osnovnom slučaju upotrebe će biti proširene

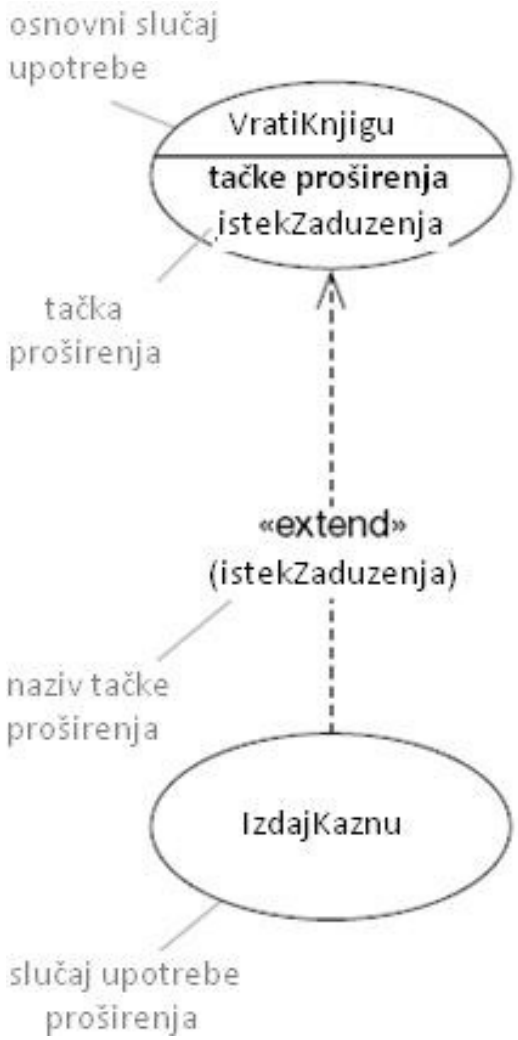
<<extend>>


osnovni
slučaj
upotrebe
Bibliotečni sistem
stereotip
VratiKnjigu
«extend»
slučaj upotrebe
proširenja
PosudiKnjigu
IzdajKaznu
Bibliotekar
veza
zavisnosti
PronađiKnjigu

<<extend>>

- Osnovni slučaj upotrebe ne zna ništa o slučajevima upotrebe proširenja
  - samo obezbjeđuje zakačke za njih
  - savršeno je kompletan bez svojih proširenja
- Tačke proširenja
  - ne ubacuju se u tok događaja osnovnog slučaja upotrebe, nego se umjesto toga dodaju na vrh toka događaja
  - mogu se prikazati izlistavanjem u novom odjeljku u ikoni osnovnog slučaja upotrebe
  - u glavnom toku nisu numerisane, nego se pojavljuju između numerisanih koraka u toku
- Tok osnovnog slučaja upotrebe ne zna gdje se vrši njegovo proširenje - proizvoljna i ad hoc proširenja toka

<<extend>> - primjer



VratiKnjigu
ID: UC9
Akteri: Bibliotekar
Preduslovi: 1. Bibliotekar je ispravno prijavljen na sistem
Tok događaja: 1. Bibliotekar unosi ID broj korisnika 2. Sistem prikazuje detalje o korisniku uključujujući i listu posuđenih knjiga 3. Bibliotekar u listi knjiga pronalazi onu koja će biti vraćena <istekZaduzenja> 4. Bibliotekar vraća knjigu 5. …
Postuslovi: Knjiga je vraćena

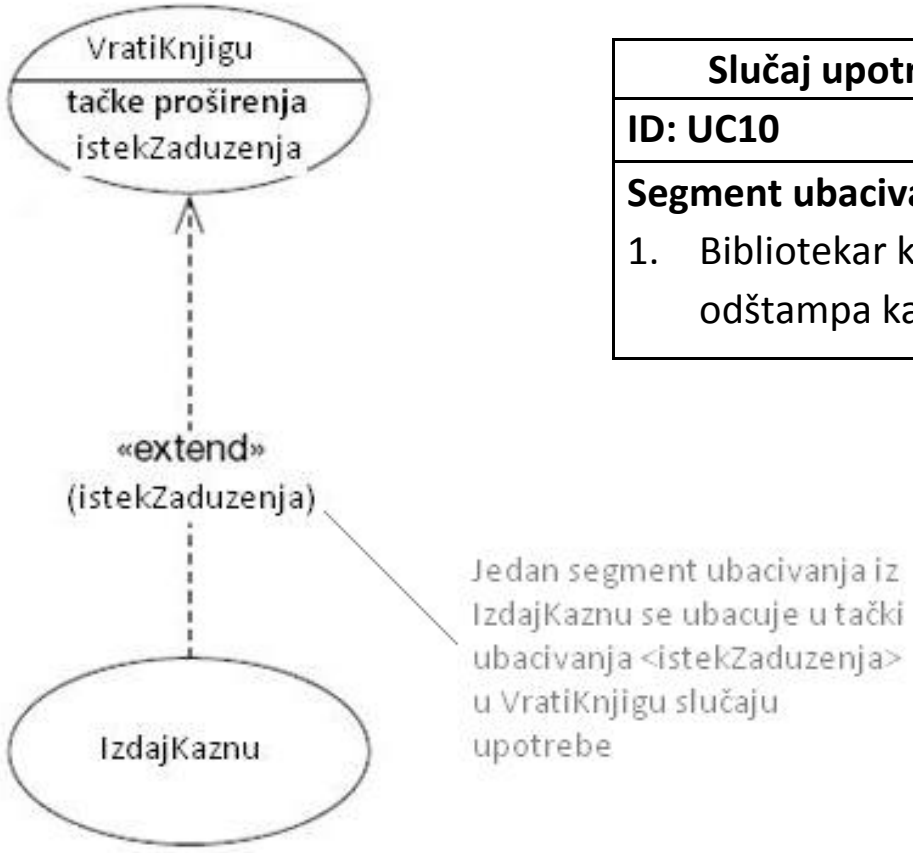
<<extend>>

- Kada se koristi <<extend>> osnovni slučaj upotrebe se ponaša kao modularni programski okvir u koji se mogu uključiti proširenja u predefinisanim tačkama proširenja
- Može se vidjeti da <<extend>> obezbjeđuje dobar način za bavljenje slučajevima iznimki ili slučajevima u kojima je potreban fleksibilan programski okvir jer se ne može predvidjeti (ili jednostavno ne može znati) unaprijed sva moguća proširenja

<<extend>>

- Slučajevi upotrebe proširenja nisu kompletni slučajevi upotrebe – ne mogu biti instancirani
- Sastoje se samo od jednog ili više fragmenata ponašanja (*segmenti ubacivanja*)
- <<extend>> veza specificira tačku proširenja u osnovnom slučaju upotrebe gdje će segment ubacivanja biti ubačen
- Primjenjuju se sljedeća pravila za <<extend>> vezu:
  - mora specificirati jednu ili više tačaka proširenja u osnovnom slučaju upotrebe ili se pretpostavlja da referiše na sve tačke proširenja
  - Slučaj upotrebe proširenja mora imati isti broj segmenata ubacivanja koliko je tačaka proširenja specificirano
  - Legalno je da dva slučaja upotrebe proširenja proširuju isti osnovni slučaj upotrebe u istoj tački proširenja – redoslijed kojim se proširenja izvršavaju je neodređen.

<<extend>>

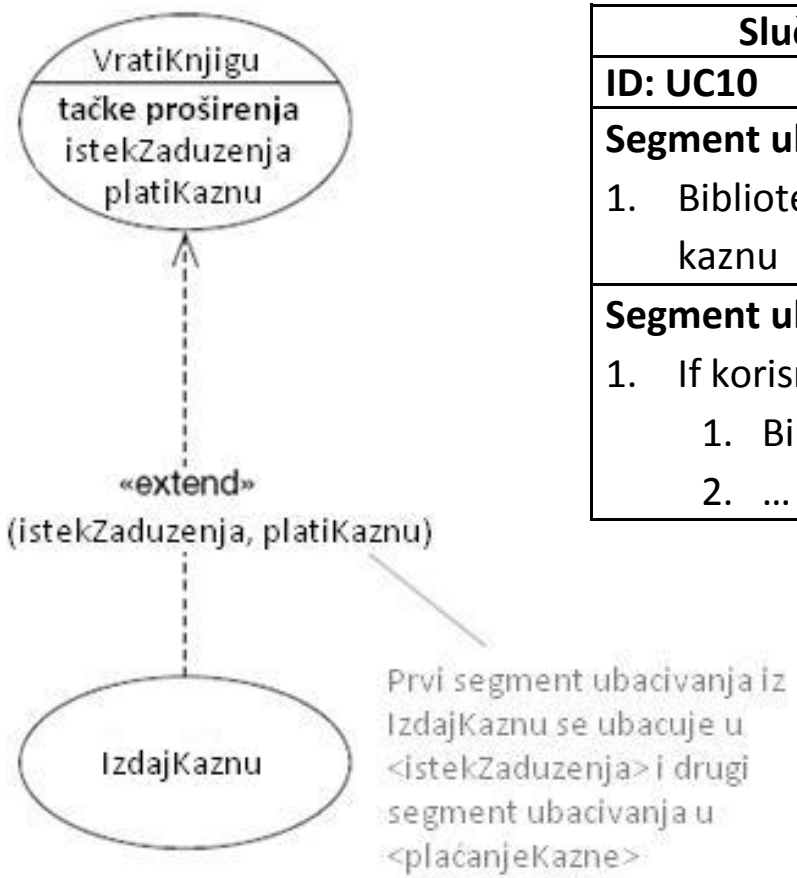


Slučaj upotrebe proširenja: IzdajKaznu
ID: UC10
Segment ubacivanja: 1. Bibliotekar koristi sistem da snimi i odštampa kaznu

<<extend>>

- U slučaju upotrebe proširenja može biti više segmenata ubacivanja.
- korisno kada se proširenje ne može obuhvatiti jasno u jednom segmentu ubacivanja zbog potrebe da se vrati u glavni tok osnovnog slučaja upotrebe da se nešto odradi.
- U primjeru, može se pretpostaviti da nakon snimanja i štampanja kazne se može vratiti u glavni tok da se obradi još knjiga sa istekom zaduženja, a potom konačno u tački proširenja <plaćanjeKazne> daje se opcija korisniku da plati ukupnu kaznu.
- Ovo je mnogo efikasnije nego da se mora štampati i prihvatiti plaćanje za svaku kaznu pojedinačno, što bi bio slučaj da su kombinovana dva segmenta u IzdajKaznu

<<extend>>



Slučaj upotrebe proširenja: IzdajKaznu
ID: UC10
Segment ubacivanja 1: 1. Biblioteka koristi sistem da snimi i odštampa kaznu

Segment ubacivanja 2:
1. If korisnik odabere da plati ukupnu kaznu
  1. Bibliotekar prihvaća plaćanje kazne
  2. …

<<extend>>

- Primjer - korisnicima se daje upozorenje prvi put kada se vraća knjiga sa istekom zaduženja, a kažnjavaju se samo naredni.
- Modelira se dodavanjem novog slučaja upotrebe proširenja i smještanjem uslova na <<extend>> veze.
- Uslovi su Boolean izrazi, a ubacivanje se obavlja ako i samo ako se izraz evaluira kao istinit.