UNIVERZITET METROPOLITAN BEOGRAD

SE101 - RAZVOJ SOFTVERA I INŽENJERA SOFTVERA

Modeliranje softvera primenom UML-a

Lekcija 06

PRIRUČNIK ZA STUDENTE

SE101 - RAZVOJ SOFTVERA I INŽENJERA SOFTVERA

Lekcija 06

MODELIRANJE SOFTVERA PRIMENOM UML-A

UVOD

UML je grafički jezik za opisivanje softverskog sistema

U ovoj lekciji se iylažu UML modeli (u vidu dijagrama) koji prikyacuju struktury ili ponašanje softverskog sistema.

Poglavlje 1

Uvod u UML

UNIFIED MODELING LANGUAGE (UML)

Objašnjenje šta je UML i čemu služi.

. UML ili UnifiedModeling Language je grafički jezik za vizualizaciju, specifikaciju, konstruisanje i dokumentovanje sistema programske podrške koji je postavljen kao standard.

Ako malo fleksibilnije želimo to da objasnimo, moramo naglasiti da UML nije programski već grafički jezik. Njime je moguće pratiti razvoj neke građevine, izraditi plan uzgajanja voća u voćnjaku itd. UML predstavlja kolekciju najuspešnijih inženjerskih metoda kojima je iskustveno dokazano, pojednostavljeno modelovanje velikih i složenih sistema.

UML definiše modele koji opisuju strukturu ili ponšanje nekog sistema.

UML model je apstrakti pogled na sistem koji ignoriše neke detalje sistema. Mogu se razviti komplementarni modeli sistema da bi se prikazao kontekst sistem, interakcije, struktura i ponašanje.

Definisano je sedam ciljeva kojima UML kao jezik teži:

1. Pružiti korisniku brz jezik za vizuelno modelovanje kojim će moći u relativno kratkom vremenu napraviti i razmenjivati modele sa određenim značenjem,
2. Pružiti korisniku mogućnost proširenja i stvaranja specijalizovanih delova,
3. Biti nezavisan od programskih jezika i razvojnih procesa,
4. Pružiti formalne osnove za razumevanje jezika za modelovanje,
5. Podsticanje rasta i razvoja objektno orijentisanih programskih jezika,
6. Podrška visoko pozicioniranih razvojnih pojmova kao što su saradnja, okvirni rad, uzorci i komponente,
7. Integrisanje i nadopunjavanje praktičnim iskustvom.

Od decembra 1997 do danas, UML je imao više verzija, kao rezultat njegovog razvoja i proširenja. Najnovija verzija UML-a je verzija 2.5.1. Njenu specifikaciju možete naći na https://www.omg.org/spec/UML

DVA TIPA UML DIJAGRAMA

UML 2.5 definiše 14 dijagrama, grupisani u dva tipa dijagrama: dijagrami strukture i dijagrami ponašanja.

Da bi smo razumeli ulogu ovih dijagrama, moramo da naglasimo da se UML 2.5. sastoji od dva tipa gradivnih blokova i samih dijagrama, elemenata i relacija, i to:

1. Dijagrami strukture, koji prikazuju strukturu softverskog sistema, te se smatraju i statičkim dijagramima
2. Dijagrami ponašanja, koji pokazuju ponašanje (funkcionalnost) softverskog sistema, te se smatraju dinamičkim dijagramima.

Svaki softverski sistem u suštini se predstavlja jedinstvenim UML modelom, sa svim elementima i njihovim vezama, a dijagrami samo prikazuju različite poglede na taj jedinstven, integrisan model softverskog sistema. U ovoj lekciji ćemo opisati sam najčepže korišćene UML dijagrame.

Slika 1.1 Dijagrami strukture i dijagrami ponašanja [1.1]

UVOD U UML (VIDEO)

Trajanje: 7:20

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 2

Dijagram klasa

UML OZNAKA KLASE

Klasa u UML-u se predstavlja pravougaonikom sa tri dela. U prvom delu se daje naziv klase, u dugom se navode atributi, a u trećem operacije, tj. metod

Klasa je klasifikator koji predstavlja objekte koji dele ista svojstva, ograničenja i semantiku. U UML-u klasa se predstavlja pravougonikom sa imenom klase (skraćeni oblik) ili sa pravougaonikom koji u sebi ima tri dela (ili pravouganika) i kome prvi deo prikazuje ime klase, drugi deo prikazuje njegove atribute, a treći deo njegove operacije, tj. metode (slika 1)

Slika 2.1 Predstavljanje klase u UML-u [1.1]

Kao što se može primetiti na slici, klasa Okvir ima tri celine. Prva celina predstavlja naziv same klase, drugi deo predstavlja mesto za deklarisanje svih promenljivih odnosno atributa klase i njihovih tipova podataka. Treća sekcija predstavlja definisanje metoda odnosno operacija (funkcija).

Pre naziva metode i dalje stoje simboli za enkapsulaciju koji određuju vidljivost metode, a nakon imena same metode posle dve tačke stoji tip podatka koji ta metoda vraća.

Da bi se ograničilo pravo pristupa nekom atributu ili metodu nekog objekta, koriste se tri stepena zaštićenosti, tj. kontrole pristupa:

- public (+) – javan, dozvoljen pristup sa drugih objekata;
- private (-) – privatan, dozvoljen pristup samo iz te klase;
- protected (#) – zaštićen, dozvoljen pristup i iz podklasa ove klase

Kontrola pristupa definiše vidljivost klase, promenljive, ili metoda – od strane drugih objekata. Slika 2 daje drugi primer klase u UML-u.

Slika 2.2 Klasa Sudent u UML-u [1.1]

ASOCIJACIJA

Asocijacija označava da dve klase mogu da razmenjuju poruke, odnosno, da razmenjuju servise, ako je asocijacija dvosmerna.

Kod veze tipa asocijacije jedan objekat upotrebljava servise drugog, ali ga ne poseduje. Asocijacije su relacije koje se manifestuju u vreme izvršavanja (realizacije) time što dozvoljavaju razmenu poruka među objektima. Asocijacija je veza tipa klijent-server. Upotrebljavaju se kada:
Jedan objekat upotrebljava servise drugog, ali on nije sadržan u njemu
Životni ciklus upotrebljene klase ne zavisi od klase koja je koristi.

- Upotrebljen objekat se deli podjednako sa puno drugih objekata.
- Objekti šalju informaciju drugim objektima slanjem parametara.
- Asocijacije označavaju veze objekata koji razmenjuju poruke.

Kada se šalje parametar iz objekta A, poziva se metod objekta B koji prima parametar. Definicija klase treba da odredi da takva veza postoji i koji tip informacije se može preneti
Na slici 3 prikazan je primer asocijacije dve klase pomoću UML jezika. Asocijacija ima svoj naziv koji se upisuje na sredini linije, a uz objekte se upisuju dva podatka: (1)uloga koju objekat igra u odnosu na objekat sa kojim je povezan (2) multiplikator (ili kardinalost) koji pokazuje broj mogućih objekata koji mogu biti u vezi. Na slici 4 prikazan je primer asocijacije izmedju klase Kupac i klase Narudžbina.

Slika 2.3 Asocijacija dve klase [1.1]

Slika 2.4 Primer asocijacije izmedju klasa Kupac i Narudžbenica [1.1]

DIJAGRAMI KLASA SA ASOCIJACIJAMA

Dijagrami klasa se koriste pri razvoju modela objektno-orijentisanog softverskog sistema da bi se pokazale klase sistema i asocijacije između ovih klasa

Dijagrami klasa se koriste pri razvoju modela objektno-orijentisanog softverskog sistema da bi se pokazale klase sistema i asocijacije između ovih klasa. Zbog toga, svaka klasa ima neko saznanje o klasi sa kojom je u asocijaciji.

U početnim fazama projektovanja softvera, objekti predstavljaju nešto što postoji u realnom svetu (npr. pacijent, recept, doktor i dr.). Sa daljim razvojem softvera, tj. njegovom implementacijom (kodiranjem), moraju se definisati najčešće dodatni objekti koji obezbeđuju dodatnu funkcionalnost.

Dijagrami klasa se mogu kreirati sa različitim nivoima detaljnosti. U početku se navode samo stvarni objekti, sa svojim klasama, bez mnogih detalja koji se kasnije dodavaju, sa napretkom projektovanja sistema. Slika 8 prikazuje jedan jednostavni dijagram klasa, sa samo dve klase: Pacijent i Dosije pacijenta. Dijagram pokazuje i koliko objekta može biti povezano asocijacijom. U prikazanom dijagramu, to je asocijacija tipa 1:1, tj. samo jedan objekt može da bude u asocijaciji sa drugim. To znači da pacijent može da ima samo jedan zdravstveni dosije. Ako se u dijagramu stavi, to označava neograničen broj objekata može biti u asocijaciji na toj strani asocijacije, što je i slučaj na primeru dijagrama klasa prikazanog na slici 9 .

Slika 2.5 UML klase i asocijacija [1.2]

Slika 2.6 Klase i asocijacije u MHC-PMS sistemu [1.2]

AGREGACIJA I KOMPOZICIJA

Primena veze agregacije i kompozicije na UML klasnom dijagramu

Agregacija
Agregacija se primenjuje kada jedan objekat fizički ili koncepcijski sadrži drugi.
Na slici 12a prikazan je UML model veze tipa agregacija između dva objekta. Objekat-vlasnik sadrži objekat-komponentu, ali objekat- komponentu mogu istovremeno da koriste i druge klase, tj. objekat- komponenta može da bude i u vlasništvu više različitih klasa- vlasnika. Na slici 12b prikazan je primer ovakve veze. Na fakultetetu radi više nastavnika. Kako veza tipa agregacije dozvoljava da objekat – komponenta ima odnose i sa drugim klasama, to se jasno vidi iz ovog dijagrama da nastavnik može da radi i na nekom drugom fakultetu, tj. da deli svoje radno vreme na dva ili više fakulteta, odnosno, da ima više poslodavaca.
Kompozicija
Kompozicija je jak oblik agregacije. Upotrebljava se za aktivne objekte – objekte koji su izvori upravljanja (slika 13). Ona kreira proces upravljanja, i njene komponente izvršavaju taj proces. Veza tipe agregacije se u UML obeležava izokrenutim rombom, koji je popunjen crnom bojom (slika 13). Opet, kao primer, navodimo fakultet. Fakultet može da ima više odseka, ali za razliku od nastavnika, ti odseci ne mogu da budu i delovi drugih fakulteta ili organizacija. Zato je ovde korišćen jači oblik agregacije, a to je kompozicija. Ona ne dozvoljava komponenti da bude deo nekog drugog objekta ili da ima veze tipa agregacije sa drugim objektima.

Slika 2.7 Primer agregacije [1.1]

Slika 2.8 Primer kompozicije [1.1]

GENERALIZACIJA

Generalizacija omogućava da se zajedničke informacije (atributi i metode) definišu i održavaju samo na jednom mestu (klasi roditelju).

Generalizacija je tehnika koja se koristi da bi se na jednostavan način rešavale složene situacije. Uopštene klase sadrže atribute i operacije (metode) koje imaju druge klase, koje su onda podklase uopštene klase. Podklase nasleđuju atribute i metode svojih klasa-roditelja (nad klasa) te se na taj način pojednostavljuje opis klasa sistema, jer su česti slučajevi kada više klasa imaju iste atribute i metode (operacije).

Na slici 14 prikazana je grupa klasa povezana znacima generalizacije (bela strelica) koja ukazuje na uopštenije (generalizovane) klase. Klase na vrhu imaju zajednička svojstva

(atribute i metode), a klase na dnu imaju specifična svojstva, kao i svojstva koja nasleđuju od generalizovanih klasa (roditelja). Znači, atribute i metode koje imaju generalizovane klase (pokazane sa belim strelicama) imaju i sve klase povezane ispod (sa belim strelicama) generalizovanih klasa

Slika 2.9 Hijerarhija generalizacije klasa [1.2]

INTERFEJS

Način kreiranja, svrha i upotreba interfejsa

interfejs nekog objekta prezentuje kako se može njime manipulisati ili kako se sa njim može komunicirati. To je sredstvo pomoću kojeg drugi objekti mogu da utiču na stanje objekta, ili da saznaju njegovo stanje. Interfejs nekog softverskog objekta se definiše programima (metodima) klase kojoj objekat pripada. To znači da interfejs sadrži popis metoda neke klase koji su javni, tj. dostupni drugim objektima. Na taj način, ove metode se mogu pozvati iz metoda koje su definisane drugim klasama. .

Na slici 25 prikazan je interfejs Student koji nudi service koje realizuju klase RedovniStudent i InternetStudent. To je primer kada dve klase koriste jedan zajednički interfejs. Na slici 26 prikazan je zajednički interfejs Person za za klase Professor i Student.

Slika 2.10 Interfejs Student povezuje klase RedovniStudent i InternetStudent [1.1[

Slika 2.11 Interfejs Person koriste klase Professor i Student [1.2]

PRIMER DIJAGRAMA KLASA - ŠKOLA

Dijagram klasa škole sadrži 5 klasa (Škola, Katedra, Predavač, Kurs i Učenik) i relacije (veze) koje ih povezuju u sistem.

Na slici 27 prikazan je jedan skup klasa uzetih iz informacionog sistema jedne škole. Polazeći od leve donje strane tog dijagrama videćete klase nazvane Učenik, Kurs i Predavač. Postoji asocijacija između Učenik i Kurs, koja specifikuje da učenici pohađaju kurseve. Pored toga se vidi da svaki student pohađa koliko god želi kurseva i da svaki kurs može pohađati bilo koji broj učenika.

Svih pet klasa su označene kao postojane. To znači da je predviđeno da njihovi konkretni primerci egzistiraju u bazi podataka ili nekoj drugoj formi postojanog skladištenja. Na dijagramu su prikazani i atributi (primitivni tipovi) svake od tih klasa.

U dvema od ovih klasa (Škola i Katedra) precizirano je više operacija za manipulisanje njihovim delovima. One su uključene zbog toga što su važne za održavanje integriteta podataka (dodavanje ili uklanjanje klase Katedra, na primer, prouzrokovaće neke druge promene).

Slika 2.12 Dijagram klasa škole [1,1]

Mogli biste da u ovim i drugim klasama uzmete u obzir mnoge druge operacije, kao što je preispitivanje preduslova za neki kurs pre pridruživanja nekog učenika. To su ipak više poslovna pravila nego operacije vezane za integritet baze podataka, pa ih je najbolje staviti na viši nivo apstrakcije nego što je ova šema.

UML DIJAGRAM KLASA (VIDEO)

Trajanje: 16:46 min.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 3

Dijagram komponenata

UML KOMPONENTA

Komponente predstavlja deo softvera koji predstavlje najčešće dve ili više klase i može da sadrži portove sa ulaznim i izlaznim iterfejsima.

Komponenta obično predstavlje neki deo softvera. Ona predstavlja viši nivo apstrakcije od klase, i obično jedna komponenta predstavlja dva ili više klasa (ili objekata). Na slici 1 prikazana je UML 2.5 oznaka jedne komponente. Komonenta je predstavljena klasifikatorom u obliku pravougaonika sa ključnom rečju <<component>> . Drugi način za prikazivanje komponente je korišćenje jednog simbola u gornjem desnom uglu pravougaonika koji predstavlja komponentu, kao što je prikazano na slici 1.

Slika 3.1 Dva načina označavanja komponente u UML 2.5 [1.3]

Dve komponente mogu biti povezane konektorom spajanja (assembly connector). Konektor spajanja dve komponente prikazuje zahtevani interfejs od strane jedne kompoinente i obezbeđen interfejs od strane druge komponente. Na slici 2 prikazan je slučaj da komponenta Component1 zahteva interfejs koji obezbeđuje komponenta Component2.

Slika 3.2 Konektor spajanja spaja dve komponente [1.3]

UML element Port se može povezati sa komponentom da bi se označio servis (ponašanje) koji komponenta obezbeđuje svom okruženju, ili označava servis koji komponenta zahteva od svog okruženja. Portovi označavaju ulaze i izlaze jer mogu da rade obostrano, tj. u oba smera. Na slici 3 prikazana je komponenta sa portovima za pružanje onlajn servisa sa dva interfejsa. Jedan prihvata naloge (narudžbenice) a drugi obezbeđuje plaćanje naručenog proizvoda.

Slika 3.3 Komponenta sa portovima za naručivanje i za plaćanje proizvoda [1.3]

DIJAGRAMI KOMPONENATA

Dijagrami komponenata sadrže povezane komponente, često sa konektorima spajanja i portovima, i pokazuju strukturu jednog softverskog sistema

Dijagrami komponenata pokazuju delove softvera koji čine softverski sistem. Komponente predstavljaju gradivne blokove softverskog sistema i mogu da obuhvate značajne delove sistema.

Na slici 4 prikazan je dijagram komponenata sa povezanim komponentama. Konektori spajanja tipa "link" sa interfejsima koje obezbeđuju komponente **"Product"** i **"Customer"**, kao i sa zahtevanim interfejsima specifiranih od strane komponente **"Order".** Veza zavisnosti (dependency relationship) povezuje detalje računa kupca sa zahtevanim interfejsom **"Payment",** komponente **"Order".**

Za razliku od dijagrama paketa, dijagrami komponenta su semantički bogatiji, jer obuhvataju elemente koji imaju samo privatan pristup (pristup "private"), dok dijagram paketa sadrži elemente samo sa javnim pristupom.

.

Slika 3.4 Dijagram komponenata softvera za naručivanje proizvoda [1.4]

PRIKAZ ELEMENATA DIJAGRAMA KOMPONENATA

Dijagram komponenata prikazuje komponente, obezbeđene i zahtevane interfejse, portove i relacije između njih

Dijagram komponenata (component diagram)prikazuje komponente (slika 5), obezbeđene i zahtevane interfejse, portove i relacije između njih. Ovi dijagrami se koriste kod razvoja softvera primenom komponenata (Component-Based Development - CBD) radi opisa sistema sa servisnom-orijentisanom arhitekturom (Service-Oriented Architecture - SOA).

Komponente mogu da predstavljaju:

- **lokalne komponente** (komponente procesa, posla) i
- **fizičke komponente** (npr., CORBA, EJB, COM, WSDL komonente)

Zajedno sa artifaktima koji ih primenjuju i čvorovima po kojima su raspoređeni i u kojima se izvršavaju. Komponente se obično kreiraju uradi primene određene tehnologije i finkconalnosti, primene određenog hardvera i softverskog okruženja.

Slika 3.5 Prikaz dijagrama komponenata sa tipičnim elementima [1.4]

DIjagram komponenti najčešće koristi sledeće elemente: komponeta (component), interfejs (interace) obezbeđen i zahtevani interfejs (provided interface, required interface), klasa (class), port (port), konektor (connector), artifact (artifact), relizacija komponente (componentrealization), zavisnost (dependency), upotreba (usage).

DIJAGRAMI KOMPONENATA (VIDEO)

Georgia Tech predvanje - Komponente

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 4

Dijagram paketa

UML OZNAKA PAKETA

Paket sadrži semantički povezane elemente, grupisani po grupama da bi se obezbedila bolja struktura modela sistema.

Paket (package) klasifikator koji sadrži elemente koji su semantički povezani i koje mogu zajedno menjati. Služe za organizovanje elemenata po grupama da bi se obezbedila bolja struktura modela sistema.

Paket može da sadrži i druge pakete. U okviru istog paketa, svi elementi u njemo moraju da imaju jedinstvena imena. Međutim, ako su elementi različitog tipa, oni mogu da imaju ista imena.

Paket može da unese (importuje) pojedinačne članove drugih paketa ili sve članove drugih paketa. Paket se može spojiti sa drugim paketima.

Na slici 1.a prikazana je UML oznaka za jedan paket. Paket može da prikaže svoje članove. U tom slučaju ime paketa se prikazuje u levom tablu oznake paketa (slika 1.b)

Slika 4.1 UML oznaka paketa bez (a) i sa prikazom svojih članova (b) [1.4]

Članovi paketa se mogu prikazati i van samog paketa ako se povežu linijama od paketa do članova, kao što je prikazano na slici 2. Znak plus (+) unutar kruga označava na obodu paketa i sa njim se povezuju linijama članovi paketa, kada se prikazuju van okvira paketa.

Slika 4.2 Prikaz članova paketa van pravouganika koji označava paket [1.4]

Unutar paketa se mogu prikazati sledeći ULM elementi: **Tip** (Type), **Klasifikator** (Classifier), **Klasa** (Class), **SLučaj korišćenja** (Use Case), Komponenta (Component), **Paket** (Package), **Ograničenje** (Constrain), **Zavisnost** (Dependency) i **Događaj** (Event).

SPAJANJE PAKETA

Dva paketa se mogu spojiti u treći koji sadrži sadržaje spojenih paketa. Označene sa ključnom rečju <<merge>>

Spajanje paketa (package merge) u jedan je odnos dva paketa koji ukazuje da se sadržaj jednog paketa, proširuje i sadržajem drugog paketa.

Spajanje paketa se može smatrati i kao operacija koja uzima sadržaje dva paketa i proizvodi novi paket koji kombinuje sadržaje oba paketa koja su se spojila.

Ovaj mehanzam se može koristiti kada elementi defisani u različitim paketima imaju isto ime i predstavljaju isti koncept. Na primer, sadrže različite definicije istog koncepta, a za različite namene.

Dati osnovni koncept se širi inkrementalno sa spajanjem sa paketom koji sadrži jedan inkrement. Spajanjem različitih inkremenata na ovaj način, može se formati prilagođena definicija nekog koncepta za neki specifični slučaj.

Spajanje paketa je posebno korisno pri meta-modeliranju i mnogo se koristi kod definisanja UML meta modela.

Na slici 3 je prikazano spajanje paketa upotrebom isprekidane linije sa strelicama ka paketima koji se pripajaju. Ključna reč ***<<merge>>*** , tj. "spajanje" se stavlja blizu isprekidane linije.

Slika 4.3 Spajanje paketa Kernel i Profile, a dobijeni paket Constructs sa onda unosi u paket PrimitiveTypes [1.4]

UML MODEL

Model je specijalizovani UML paket koji opisuje sistem sa određenog ugla gledanja (pogleda).

Model je specijalizovani UML paket koji opisuje sistem sa određenog ugla gledanja (pogleda). Modeli se koriste radi crtanja dijagrama modela.

Model se obeležava u UML-u sa UML simbolom običnog paketa (ikonica foldera) sa malim trouglom u gornjem desnom uglu velikog pravougaonika (slika 4a).

Ako modela ima prikaz sadržaja u pravougaoniku, onda se trougao stavlja u tab pored imena (slika 4.b).

Model se može obeležiti i kao paket sa ključnom rečju <<model>> postavljene iznad imena modela (slika 5).

Slika 4.4 Dve varijante UNL oznake modela Business Layer [1.4]

Slika 4.5 UML oznaka modela sa stereotipom > [1.4]

PRIMER 1 DIJAGRAM PAKETA

Primer dijagrama paketa za slučaje kupovine preko veba.

Na slici 6 su prikazani neki glavni elementi dijagrama paketa.

- Web Shopping

- Mobile Shopping
- Phone Shopping
- Mail Shopping

Paketi Mail Shoping i Phone Shoping se spajaju u paket Shoping Cart.

Četiri ista paketa upotrebljavaju paket Payment i Shopping Cart unose (import) i druge pakete

Slika 4.6 Dijagram paketa [1.4]

PRIMER 2 DIJAGRAMA MODELA

Dijagram modela pokazuje neke apstrakcije ili neki specifičan pogled na sistem, da bi opisao neke arhitektonske, logičke ili ponašanje aspekta sistemA

Na slici 7 je prikazan primer jednog dijagrama modela.

Dijagram modela je pomoćni UML dijagram strukture koji pokazuje neke apstrakcije ili neki specifičan pogled na sistem, da bi opisao neke arhitektonske, logičke ili ponašajnje aspekte sisteme.

Na slici se vide neki glavni elementi dijagrama modela.Lazered Application. Lazered Application je kontejner model koji sadrži tri druga modela: Presentation Layer, Business Layer i Data Layer. Postoje zavisnosti koje su definisane između tih modela.

Modeli obično sadrže pakete. Paketi mogu da imaju zavisnosti ili druge relacije, kao na primer, import, koje su definisane između ovih paketa.

Slika 4.7 Dijagram modela [1.4]

PACKAGE DIAGRAM IN UML

Trajanje: 3:54 minuta

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 5

Dijagram raspoređivanja

FIJAGRAM RASPOREĐIVANJA, ČVOROVI I ARTIFAKTI

Artifakt je izvor raspoređivanja po čvorovima

Dijagram raspoređivanja (depoyment diagram) pokazuje izvršnu arhitekturu sistema sa rasporedom artifacta (najčešće softverskih komponenata) po čvorovima sistema

Čvorovi (nodes) predstavljaju ili hardverske uređaje ili izvršna okruženja softvera. Oni se povezuju komunikacionim putevima i čine mrežu sistema sa proizvoljnom složenošću.

UML dijagram raspoređvanja prikazuje raspored artifakta po čvorovima, čvorove, uređaje, izvršna okruženja, kompozicije čvorova, komunikacione puteve, specifikaciju rasporeda softverskih komponenata, zavisnost specifikacije rasporeda i socijacije softverskog raporeda.

UML artifakat je klasifikator koji predstavlja neku fizičku jedinicu, neku informaciju koja se koristi ili koja je proizvod procesa raspoređivanja softvera, ili rezultat rapoređivanja i rada sistema. Artifakt je izvor raspoređivanja po čvorovima. Na primer, ovo mogu biti artifakti: tekstualni dokument, fajl sa izvornim kodom, skript, fajl sa izvršnim kodom, arhivski fajla (datoteka), tabela baze podataka....Ovi artifakti se u UML-u označavaju posebnim stereotipvima:

<<file>> - fizička datoteka (fajl) u kontekstu razvijenog sistema

Standardni stereotipi koji su podklase artifakta <<file>> su:

- ***<<document>>*** - neka opšta datoteka koja nije ***<<source>>*** ili ***<<executavle>>***
- ***<<source>>*** - datoteka sa izvornim kodom koje se može prevesti (kompilirati) u izvršni kod
- ***<<library>>*** - statička ili dinamička datoteka biblioteke
- ***<<executavle>>*** - datoteka sa izvršnim kodom koji se može izvršiti od strane računarskog sistema
- ***<<script>>*** - skript datoteka koja se može interpretirati od strane računarskog sistema

Slika 1 prikazuje primere UML oznaku za artifakte. U gornjem desnom uglu pravougaonika mora da se nalazi ikonica dokumenta. Poslednja oznaka u redu je alternativna oznaka artifakta.

Slika 5.1 Primeri oznaka artifakata [1.4]

ČVOR

Artifakti se raspoređuju po ciljevima raspoređivanja. Čvor (Node) je cilj raspoređivanja koji predstavlja kompjuterski resurs u kome se raspoređuju artifakti radi izvršenja sistema.

Artifakti se raspoređuju po ciljevima raspoređivanja. Cilj raspoređivanja (***deployment target***) je lokacija raspoređivanja artifakta. Na slici 2 prikazana je UML 2.4 definicija cilja raspoređivanja.

Slika 5.2 UML definicija cilja raspoređivanja [1.4]

Čvor (***Node***) je cilj raspoređivanja koji predstavlja kompjuterski resurs u kome se raspoređuju artifakti radi izvršenja sistema. Čvor se u UML-u prikazuje trodimenzionalnim paralelopipedom (slika 3).

Slika 5.3 Prikazivanje čvora u UML-u koji predstavlja aplikacioni server [1.4]

Čvor je povezan sa rasporedima artifakata i indirektno, sa upakovanim elementima koji se koriste u manifestacijama (impementacijama) artifakta koji je raspoređen u čvoru.

Čvorovi se mogu povezivati komunikacionim putevima. Na primer, sa komunikacionim putem se može predstaviti veza alikacionog servera i baze podataka. Komunikcioni put predstavlja vezi između dva čvora. Mogu se koristiti i specifične topologije mreža kao veza - linkova - između primeraka čvorova.

Čvor je definisan sa:

- **uređajem** (device) i
- **izvršnim okruženjem** (execution environment).

HIJERARHIJA ČVOROVA I IZVRŠNA OKRUŽENJA

Hijerarhija čvorova se može modelirati upotrebom kompozicije čvorova ili definicijom neke interne strukture čvorova.

Hijerarhija čvorova se može modelirati upotrebom kompozicije čvorova ili definicijom neke interne strukture čvorova (slika 4). Interna struktura čvorova se definiše sa delovima i konektorima. Delovi čvora mogu biti samo čvorovi.

Slika 5.4 Primer hijerarhije čvorova [1.4]

Izvršno okruženje (execution environment) je obično deo nekog opšteg čvora ili <<device>> elementa koji predstavlja okruženje fizičkog hardvera na kome se nalazi izvršno okruženje.

Izvršna okruženja se mogu povezivati (npr. izvršno okruženje baze podataka se povezuje se izvršnim okruženjem operativnog sistema).

Primerci izvršnog okruženja se dodaju primercima čvorova upotrebom kompozitnih asocijacija između čvorova i izvrših okruženja, pri čemu izvršna okruženja imaju ulogu dela.

Na slici 5 prikazan je serversjki uređaj kao skup nekoliko povezanih izvršnih okruženja.

Slika 5.5 Primer aplikacionog servera sa nekoliko povezanih izvršnih okruženja. [1.4]

KOMUNIKACIONI PUT

Komunikacioni put (communication path) je asocijacija između dva raspodeljena cilja preko koga oni mogu da razmenjuju signale i poruke

Komunikacioni put (communication path) je asocijacija između dva raspodeljena cilja preko koga oni mogu da razmenjuju signale i poruke

Komunikacioni put nije označen kao UML asocijacija, i nema neku dodatnu oznaku u odnosu na asocijaciju. Na slici 6 prikazan je komunikacioni put koji povezuje više aplikacionih servera i servera baza podataka.

Slika 5.6 Komunikacioni put koji povezuje do tri aplikaciona servera i do dva servera baza podataka [1.4]

Kada su ciljevi raspoređivanja fizički uređaji, komunikacioni put najčešće predstavlja fizičku vezu između čvorova (slika 7)

Slika 5.7 Komunikacioni put u vidu fizičke veze dva čvora (servera) [1.4]

Kada su ciljevi raspoređivanja izvršna okruženja, komunikacioni put predstavlja najčešće neki protokol (slika 8).

Slika 5.8 Komunikacioni put u vidu protokola [1.4]

RASPOREĐIVANJE

Raspoređivanje (deployment) je veza zavisnosti koja opisuje raspoređivanje nekog artifakta na neki cilj raspoređivanja (npr. čvor).

Raspoređivanje (deployment) je veza zavisnosti koja opisuje raspoređivanje nekog artifakta na neki cilj raspoređivanja (npr. čvor). Raspoređivanje se može definisati i na nivou instance (primerka) - kao alokaciju određene instance (primerka) artifakta na određenu instancu cilja raspoređivanja.

Raspoređivanje komponente je raspoređivanje jednog ili više artifakta, ili instanaca artifakta, opciona parametrizovanih primenom specifikacije raspoređivanja.

Raspoređivanje se prikazuje kao veza zavisnosti sa početkom na artifaktu (snabdevač) i krajem na cilju raspoređivanja (klijent) i sa oznakom ***<<deploy>>*** . Primer raspoređivanja atrifakta na čvoru je dat na slici 9.

Slika 5.9 Primer raspoređivanja artifakta na JSP serveru [1.4]

Na nivou instanci, instance artifakata se mogu rasporediti određenim instancama cilja raspoređivanja. Može se onda izostaviti primena podvučenih naziva artifakta (slika 10).

Slika 5.10 Primer raspoređivanja artifacta dvema instancama JSP servera [1.4]

Za modelovanje složenh ciljeva raspoređivanja , modeli koji sadrže čvoreve sa kompozitnom strukturom definisane primeno "parts", neko svojstvo (koje funkcioniše ka deo) takođe može da bude cilj raspoređivanja. Raspoređivanje takođe može da se prikaže sa raspoređenim artifaktima od strane cilja raspoređivanja (slika 17). Raspoređivanje se može obeležiti i upotrebom tekstualne liste upotrebljenih artifakta u nekom cilju raspoređivanja.

PRIMER: DIJAGRAM RASPOREĐIVANJA NA NIVOU SPECIFIKACIJE

Nivo specifikacije (zove se i nivo tipa) dijagrama raspoređivanja pokazuje određeni prikaz raspoređenih artifakata po ciljevima raspoređivanja.

Nivo specifikacije (zove se i nivo tipa) dijagrama raspoređivanja (slika 11) pokazuje određeni prikaz raspoređenih artifakata po ciljevima raspoređivanja, bez referenci na određene instance artifakata ili čvorova.

Slika 5.11 Primer prikaza dijagrama raspoređivanja na nivou specifikacije (nivo tipa) [1.4]

PRIMER 2: KLASTERSKO RASPOREĐIVANJE J2EE VEB APLIKACIJE

Primer raspoređivanja J2EE veb aplikacije sa balansiranjem opterećenja i klasterizacijom koja pokazuje korišćene instance servera.

Dolazeći HTTP zahtevi se prvo obrađuju od strane Apache veb servera (slika 12). Statički sadržaj, kao što je HTML stranice, slike, CSS i JavaScript se obarđuju od strane veb servera. Opterećenje izazvano zahtevima za JSP stranicama se uravnotežaba (balansira) i šalje na 2x2 Apache Tomacat servere upotrebom i horizontalne i vertikalne klasterizacije.

Svih 4 instanci Apache Tomacat servera primaju ili memorišu podatke koje dolaze ili odlaze na instancu Oracle 11g DBMS, koji je postao usko grlo za performnse, u slučaju intenzive obrade podataka.

All 4 instances of Apache Tomcat servers save/receive data to/from a single instance of Oracle 11g DBMS, which could become a performance bottleneck if web application is data-intensive (slika 12)

Slika 5.12 Dijagram raspoređivanja J2EE veb aplikacije sa uravnotežavanjem opterećenja i klasterizacijom arifakata [1.4]

PRIMER 3: VIŠESLOJNO URAVNOTEŽAVANJE OPTEREĆENJA J2EE SERVERA

Primer UML dijagrama raspoređivanja sa uravnotežavanjem opterećenja i klasterizacijom hardvera i softvera

Primer pokazuje 2 aktivna uravnoteživača opterećenja povezani sa 2x4 Sun Fire servera. Svaki server ima instalisane 3 instance IBM Web Sphere 7 J2EE aplikaciona servera

Uravnoteživač opterećenja mreže je uređaj koji deli oterećenje mreže na više servera. Primer pokazuje hardver jetNEXUS ALB-X uravnoteživača opterećenja. On kombinuje funkcije uravnotežavanja opterećenja OSI nivoa 7 (nivo aplikacija), HTTP kompresiju, SSL offload i keširanje sadržaja - sve u okviru jednog rešenja (uređaja).

Prikazana konfiguracija ima 2 aktivna hardverska uravnoteživača opterećenja povezanih sa 2x4 SunFire servera (slika 13)

Slika 5.13 Primer dijagrama raspoređivanja za visšeslojno uravnotežavanje opterećenja J2EE veb aplikacije [1.4]

DIJAGRAMI RASPOREĐIVANJA(VIDEO)

Georgia Tech predvanje - Dijagrami raspoređivanja (Demployment)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 6

Slučajevi korišćenja sistema

SLUČAJ KORIŠČENJA

Slučaj korišćenja specificira određenu funkcionalnost u saradnji sa jednim ili sa više učesnika, a u okviru određenog subjekta.

Slučajevi korišćenja obuhvataju zahteve sistema koji se projektuje ili analizira, opisuju funkcionalnost sistema i određuju zahteve koje sistem postavlja svom okruženju.

Slučaj korišćenja (use case) je vrsta klasifikatora ponašanja koji specificira neku funkcionalnost (korisničku funkciju) koju ostvaruje jedan ili više subjekata na koje se slučaj korišćenja odnosi u saradnji sa jednim ili više učesnika, i koji vodi kao vidljivom rezultatu koji predstavljaju određenu vrednost ovim učesnicima svakog subjekta.
Učesnik (Actor) predstavlja ulogu koju korisnici korisničkih funkcija izvode kada su u interakciji sa tim korisničkim funkcijama. Uobičajeno je da izvođač predstavlja ulogu koju čovek, hardverski uređaj ili čak neki drugi sistem igra sa posmatranim sistemom. Izvršavanje slučajeva korišćenja počinje na zahtev jednog učesnika. U zvršavanju slučajeva korišćenja može učestvovati više učesnika i oni sa sistemom takođe razmenjuju poruke i podatke. Svaki učesnik mora biti povezan sa bar jednim slučajem korišćenja vezom asocijacije.

Korisnička funkcija (use case) predstavlja funkcionalan zahtev posmatranog sistema. *Korisnička funkcija izvršava određenu, sagledivu količinu posla. Iz perspektive datog izvođača, korisnička funkcija radi nešto što je izvođaču korisno.* Korisnička funkcija opisuje šta sistem radi ali ne definiše kako to radi.

Dijagram slučajeva korišćenja (Use Case Diagram) prikazuje spoljne učesnike, odnosno korisnike sistema i njihove veze sa korisničkim funkcijama koje sistem omogućava. Model slučajeva korišćenja obično se sastoji od više dijagrama slučajeva korišćenja. Slika 1 učesnike, slučajeve korišćenje i veze (asocijacije)

Slika 6.1 Dijagram slučaaj akorišćenja

VEZE UKLJUČIVANJA I PROŠIRENJA IZMEĐU SLUČAJA KORIŠĆENJA

Veze uključivanja i proširivanja između slučajeva korišćenja pojednostavljuju opis ponašanja sistema jer se ponavljajuća ponašanja definišu samo u jednom slučaju korišćenja.

Slučajevi korišćenja mogu da se poveyuju vezana uključivanja i proširivanja. Veza uključivanja, koja se u UML označava sa ključnom reči <<Include>> koij označava da se slučaj korišćenje koji je pokazan strelicom, uključuje u slučaj korišććenja iz koga polazi strelica, tako da se on izvršava kao program koji poziva potrpogram označen strelicom (slika 2)

Slika 6.2 Slučaj korišćenja Napravi porudžbinu ključuje slučaj Prihvatanje uslova plaćanja

Veza proširenja slučaja korišćenja se označava sa <<extend>> i omgućava da se ponašanje osnovnog slučaja korišćenja (iz koga polazi strelica) proširi ponašanje drugog slučaja korišćenja (pokazanan strelicom strelicom) ali samo ako su ispunjeni određeni uslovi kod osnovnog slučaja korišćenja (slika 3).

Slika 6.3 Slučaj korščenja Transakcija, u slučaju da je unet pogrešan PIN, se prošituje slučajem Pogrešan PIN

Primenom navedenih veza slučajeva korišćenja, koji označavaju slučajeve korščenja koji povemeno ili uključuju ili proširuju neko ponašanje , izbegava opisivanje tog ponašanja na više mesta.

DIJAGRAM SLUČAJEVA KORIŠĆENJA

Dijagram slučajeva korišćenja specificira spoljne zahteve sistema, funkcionalnosti sistema i zahteve koje sistem postavlja svom okruženju.

Dijagram slučajeva korišćenja sistema se koristi radi specificiranja::

- Spoljnih zahteva, zahtevane upotrebe sistema koji se projektuje ili analizira, s ciljem da se prikaže šta sve sistem treba da radi
- funkcionalnosti koju nudi sistem (subjekat) - šta sistem može da radi
- zahteve koje specificirani subjekat (sistem) postavlja svom okruženju - definisanjem kako okruženje bi trebalo da reaguje sa subjektom kako bi on obavio svoje servise.

Glavni elementi UML dijagrama slučajeva upotrebe su prikazani na slici 4.

Slika 6.4 Prikaz dijagrama slučajeva korišćenja sa opisom njehovih relacija (veza) [1.4]

DIJAGRAM SLUČAJA KORIŠĆENJA (VIDEO)

Trajanja: 12:46 min

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 7

Sekvencijalni dijagrami

LINIJA ŽIVOTA I PORUKE U SEKVENCIJALNOM DIJAGRAMU

Linja života je element koji predstavlja nekog individualnog učesnika u interakciji. Sekvencijalni dijagram opisuje razmenu poruka između linija života.

Sekvencijali dijagram (sequence diagram) je najčešća vrsta dijagrama interakcije koji se fokusira na razmenu poruka između određenog broja "linija života".

Sekvencijalni idjagram opisuje interakciju fokusirajući se na redosled poruka koje se razmenjuju, zajedno sa njihovim odgovarajućim specifikacijama pojavljivanja na linijama života.

Linja života (lifeline) je element koji predstavlja nekog individualnog učesnika u interakciji. Linije života mogu da predstavljaju samo jedan entitet interakcije.

Linija života se predstavlja simbolom u obliku pravougaonika koji predstavlja "glavu" vertikalne (najčešće isprekidane) linije koja predstavlja vreme živora učesnika (slika 1). Glava linije života ima oblik klasifikatora koga predstavlja linja života. Najčešće je glava linije života predstavljena belim pravougaonikom u kome je upisan naziv klase koju predstvalja.

Poruka (message) je imenovani element koji definiše specifičnu vrstu komunikacije između linija života jedne interakcije. Poruka specificira ne samo vrstu komunikacije, već i pošiljaoca i primaoce poruke. Pošiljalac i primaoci su dve tačke završetka linje koja predstavlja poruku (a u terminologiji UML specifikacije, one se nazivaju "specifične pojave").

Slika 7.1 Oznake linije života [1.4]

Poruka se označava linijom sa strelicom, koja povezuje dve linije života. (slika 2). Strelica pokazuje liniju života koja prima poruku, od pošiljaoca, a to može biti ili spoljni učesnik ili druga linija života. Krajevi linije sa strelicom određuju događaje slanja i primanja. Pravounik na liniji života označava vreme izvršenja aktivnosti kod pošiljaoca i primaoca proruke.

Slika 7.2 Poruka između dve linije života [1.1]

KREIRANJE I BRISANJE PORUKA, POVRATNA PORUKA

Poruka kreiranja kreira klasu - glavu linije života, a poruka uništenja - uklanja liniju života.

Poruka kreiraj se šalje liniji života da sam sebe kreira. Prikazuje se isprekidanom linijom sa srelicom (praznom) koja pokazuje me glavu kreirane linije života (slika 3.a).

Poruka obriši se šalje radi uklanjanja druge linije života. Linija života se izvršava krstićem u oblku slova X na kraju linije života. (slika 3.b)

Slika 7.3 Kreiranje i uklanjanje linje života klase Account [1.4]

Poruka odgovora na operaciju se prikazuje isprekidanom linijom sa otvorenom strelicom na kraju (slika 4).

Izgubljena poruka je poruka sa poznatim događajem slanja, ali bez događaja primanja. Takva poruke nikad ne dolazi do svog cilja (slika 5.a)

Slika 7.4 Povratna poruka [1.4]

Pronađena poruka ima događaj primanja ali nije poznat događaj slanja. Označava se kao na slici 5.b. Ovo je obično slučaj kada poruka dolazi van granica sistema (subjekta) ili kada nije bitno da se odredi njen izvor.

Slika 7.5 Izgubljena i pronađena poruka [1.4]

DIJAGRAM SEKVENCI

Dijagram sekvenci predstavlja objekte koji učestvuju u jednoj interakciji i koji su horizontalno poređani, kao i poruke koje oni razmenjuju tokom vremena.

Dijagram sekvenci pokazu vremenski redosled poruka između objekata. Kao što se vidi na slici 6, dijagram sekvence formira se postavljanjem objekata koji učestvuju u interakciji na vrh dijagrama, po X osi. Obično, objekat koji započinje interakciju stavljate na levu stranu, a one koji se kasnije uključuju, redom ka desnoj strani. Zatim unosite po Y osi poruke koje ti objekti šalju i primaju, od vrha ka dnu, kako vreme teče. To čitaocu daje jasnu vizuelnu predstavu o tome kako se odvija upravljanje tokom vremena.

Slika 7.6 Primer djagrama sekvenci [1.4]

ULOGA SEKVENCIJLNOG DIJAGRAMA U DEFINISANJU KLASA

Sekvencijalni dijagram pokazuje svojim porukama, koje operacije (metode) primajući objekti (ili klase) moraju da poseduju, jer treba da realizuju zahteve učesnika.

Sekvencijalni dijagram se može koristiti u prikazu interakcija:

- aktora i sistema u okviru opisa slučaja korišćenja
- interakcija pri konceptualnog projektovanju sistema, kada se utvrđuju osnovni entiteti (objekti, klase)
- pri detaljnom projektovanju interakcija između klasa, a radi utvrđivanja potrebnih metoda i atributa u klasama.

U stvari, ako posmatramo proces projektovanja softverskog sistema, kao procesa u više faza, onda u svakoj od njih sekvencijalni dijagram igra važnu ulogu, jer objašnjava razmenu poruka između aktera i objekata sistema, kao i između njih samih. Počev od vrlo apstraktnih interakcija, u svakoj sledećoj fazi, interakcije postaju konkretnije, te i objekti koji učestvuju u toj razmeni poruka, postaju detaljnije definisani.
Projektant sistema, uz korišćenje odgovarajućeg softverskih alata (kao na primer, Power Designer), kreiraju sve detaljnije sekvencijalne dijagrame, u saglasnosti sa njihovim napredovanjem u sagledavanju načina rada budućeg softverskog sistema, kao i njegovih elemenata. Konačan cilj je da se vrlo detaljno definišu svi potrebni objekti i sve poruke koje oni treba da razmene, kako bi se na taj način, definisali svi metodi (ili operacije u UML terminologiji) i svi atributi, koje treba da sadrže klase koje definišu ove objekte.

Na slici 7 prikazan je primer sekvencijalnog dijagrama koji opisuje interkaciju koja se odvija u slučaju izbora izbornog predmeta nekog programa od strane studenta (ref, Lethbridge/ Laganiee 2005).,

Slika 7.7 Poruke između klasa definišu njihove operacije [Lethbridge/Laganiee 2005]

Poruku koju objekat prima određuje operaciju (metod) koju objekat treba da poseduje, jer se očekuje da on realizuje zahtev koji poruka donosi. Nazivi poruka na slici 9 su istovremeno i nazivi operacije (metoda) koje primajući objekti moraju da poseduju. U prvim fazama projektovanja, to ne mora tako da bude. Međutim, u konačnoj fazi, fazi detaljnog projektovanja, to tako mora da bude.

SEKVENCIJALNI DIJAGRAM

Ovde se daje pregledno objašnjenje glavnih elemenata sekvencijalnih dijagrama

U cilju pojednostavljenja sekvencijalnog dijagrama, piojedini delovi, koji imaju posebnu funkcionalnost, mogu se smestiti u tzv, fragmente, koji se označavaju prevougaonicima koji u gornjem levom uglu imaju oynaku tipa interakceje. Fragmenti sadrže svoje sekvencijalne dijagrame, koji ne moraju da se prikayuju u fragment koji je deo nekog drugog fragmenta ili sekevancijalnog dijagrama.

Na slici 8 prikazane su dva fragmenta. Fragment **sf submit_comments** sadrži u sebi fragment **ref Handle Errors.**

Slika 7.8 Pregleni prikaz sekvencijalnog dijagram [1.4]

SEKVENCIJALNI DIJAGRAM (VIDEO)

Trajanje:12:33 min

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 8

Dijagrami mašine stanja

MAŠINE STANJA PONAŠANJA

Mašina stanja ponašanja je specijalizovano ponašanje koje se upotrebljava radi specifikacije diskretnog ponašanje nekog dela sistema koji se projektuje pomoću prelaza konačnih stanja.

Mašina stanja ponašanja (behavioral state machine) je specijalizovano ponašanje koje se upotrebljava radi specifikacije diskretnog ponašanje nekog dela sistema koji se projektuje pomoću prelaza konačnih stanja (finite state transitions). Ponašanje se modelira kao prelazak dijagrama stanja čvorova čvorova povezanih sa prelazima. *Prelazci iz jednog u drugo stanje je pokrenuto javljanjem serija događaja*. Za vreme prelaska nekog objekta, na primer, u novo stanje, mašina stanja može da izvršava neke aktivnosti.

Kontekst (context) definiše signal i poziv pokretanja za ovu mašinu stanja, a koji atributi i operacije su raspoložive u aktivnostima mašine stanja. Pokretač signala (trigger) je događaj koji menja stanje mašine stanja. Može imati uslov da bi se javio

Mašina stanja se može prikazati unutar okvira obeleženog kao mašina stanja (state machine) ili **stm** u skraćenom obliku. Sadržaj okvira čini mašina stanja, ali može da bude i bilo koji UML djagram.

Slika 8.1 Mašina stanja Bank ATM [1.4]

POJMOVI U DIJAGRAMIMA MAŠINE STANJA

Stanje je uslov ili situacija u toku života nekog objekta pri kojoj objekat zadovoljava neke uslove, izvršava neke aktivnosti ili čeka na neki događaj.

Stanje je uslov ili situacija u toku života nekog objekta pri kojoj objekat zadovoljava neke uslove, izvršava neke aktivnosti ili čeka na neki događaj. Objekat može da promeni svoje stanje ako se pojavi određeni događaj. U UML-u, stanje objekta se predstavlja pravougaonikom sa zaobljenim uglovima (slika 2 .a)

Događaj je specifikacija neke značajne pojave koja ima lokaciju u vremenu i prostoru. Događaj se opisuje sa: signalom, vremenom i promenom stanja. Signal predstavlja imenovani objekat koji jedan objekat asinhrono šalje, a drugi objekat ga prima **(**slika 2 .b). **Vremenski događaj** je događaj koji predstavlja određeni vremenski trenutak. **Događaj promene** je

događaj koji predstavlja promenu stanja ili zadovoljenje nekog uslova. Znači, kada pod određenim okolnosti objekat primi određeni signal, dolazi do promene njegovog stanja, tj. desio se određeni događaj. Promenom stanja, menjaju se vrednosti jednog ili više atributa objekta (slika2 1 .c) .

Tranzicija **(ili prelazak)** je relacija između dva stanja koji pokazuje da objekat u prvom stanju izvršava neke akcije i ulazi u drugo stanje kada se navedeni događaj pojavi i kada se navedeni uslovi zadovolje (slika 2 .b),

Slika 8.2 Promena stanja objekta [1.1]

Događaj okidanja (trigger **)** – događaj čiji prijem kod objekta u početnom stanju izaziva tranziciju sposobnu da se realizuje, ako su njegovi povezani uslovi – zadovoljeni;

Uslovi čuvanja stanja (guard) – Logički izraz koji se izvršava kada je pokrenuta tranzicija prijemom događaja okidanja;

Akcija - Izvršavanje neke elementarne akcije koje direktno deluje na objekat koji poseduje mašinu stanja, i indirektno – na druge objekte koji su vidljivi objektu.

PROSTO I KOMPOZITNO STANJE OBJEKTA

Prosto stanje (simple state) je stanje koje nema podstanja - nema regiona i nema podmašinu stanja.

Prosto stanje (simple state) je stanje koje nema podstanja - nema regiona i nema podmašinu stanja. Prosto stanje je stanje prikazano pravougaonikom sa zaobljenim uglovima i sa imenom stanja unutar pravougaonika (slika 3).

Slika 8.3 Oznake stanja [1.4]

Opciono, stanje može da ima ime postavljeno unutar pridodatog taba. Tab sa imenom je pravougaoničkog oblika koji se obično nalazi van gornje strane stanja.

Kompozitno stanje (composite state) se definiše kao stanje koje ima podstanja (povezana stanja). Podstanja (substates) mogu biti sekvencijalna ili paralelna (istovremena).

Pseudostanje (pseudostate) je apstraktno stanje koje obuhvata različite tipove prelaznih stanja u grafu mašine stanja. Pseudostanja se koriste za povezivanje višestručkih prelazaka u složenije pitanje prelaska stanja ,kao što jsu početak i kraj, odlučivanje, račvanje, spajanje i dr. (slika 5)

Slika 8.4 Oznaka kompozitnog stanja sa podstanjima (a) i bez prikaza podstanja (b) [1.4]

Slika 8.5 UML oznake pseudostanja izbora (a i b), račvanja (c) i spajanja (d) [1.4]

DIJAGRAM PROTOKOLA MAŠINE STANJA

Dijagram protokola mašine stanja pokazuje koje operacije klasifikatora se mogu pozivati u svakom stanju posle prelaska klasifikatora u ciljno stanje.

Dijagrami protokola mašine stanja se koriste da pokažu upotrebu protokola ili životnog ciklusa nekog klasifikatora (npr. objekta) . DIjagram protokola mašine stanja (slika 6) pokazuje koje operacije klasifikatora se mogu pozivati u svakom stanju posle prelazaka klasifikatora u ciljno stanje.

Kako ovi dijagrami prikazuju životni ciklus stanja nekog klasifikatora (npr. objekta) oni su korisni za pokazivanje različitih stabilnih stanja klasa objekata koja mogu da postoje u nekom vremenu i da objasne kako objekti bi mogli da promene svoje stanje tokom vremena.

Glavni elementi dijagrama protokola mašine stanja predstavljaju *protokole stanja, prelaza i različite pseudostanja*, kao što je prikazano na slici.

Slika 8.6 Primer dijagram protokola mašine stanja, sa prikazom objašnjenja njegovih elemenata. [1.4]

MODELIRANJE VOĐENO DOGAĐAJIMA

Modeliranje vođeno događajima pokazuje kako sistem odgovara na eksterne i unutrašnje događaje. UML podržava modeliranje podržano događajima sa dijagramom stanja.

Modeiranje vođeno događajima pokazuje kako sistem odgovara na eksterne i unutrašnje događaje. Polazi se od pretpostavke da sistem ima konačan broj stanja i da događaji (kao podsticaji) prouzrokuju prelazak sistema iz jednog stanja u drugo. Ovo se naročito primenjuje kod sistema koji rade u realnom vremenu (tzv. sistemu u realnom vremenu).

UML podržava modeliranje podržano događajima upotrebom dijagrama stanja i događaja, vrše prelazak sistema iz jednog stanja u drugo. Oni ne pokazuju tok podataka unutar sistema, ali mogu da sadrže dodatne informacije vezane za proračune koji se rade u svakom stanju. Na slici 7 dat je primer dijagrama stanja u cilju modeliranjaa rada mikrotalasne peći, metodama modelovanja vođenim događajima. Svako stanje je označeno pravougaonikom sa zaobljenim uglovima.

.

Strelice pokazuju podsticaje (događaje) koji dovode do promene stanja sistema. Krgovi pokazuju početak i kraj promene stanja.

Slika 8.7 Dijagram stanja u mikrotalasnoj peći [1.2]

HIJERARHIJA STANJA

Super stanje sadrži više posebnih stanja.

Problem kod modelovanja stanja sistema je u tome što se broj mogućih stanja brzo povećava sa složenošću sistema. Kod velikih sistema, dijagram stanja, zbog toga, ne može da prikaže sve detalje, već samo pokazuje samo najvažnije informacije. Da bi se ovaj problem rešio, mogu se koristiti stanja na više nivoa, tj. koristiti tzv. super stanja. To znači da super stanje sadrži više posebnih stanja. U dijagramu stanja, super stanje izgleda kao i svako drugo stanje, ali se ono na „dodir“, može detaljnije prikazati sa svim detaljnim stanjima (kao pod stanjima). Stanje „Rad“ na slici 3 prikazano, kao super stanje, u detaljnom obliku, sa svojim posebnim (pod) stanjima: Provera, Kuvanje, Alarm i Urađeno.

Slika 8.8 Hijerarhija stanja [1.2]

PRIMER: DIJAGRAM MAŠINE STANJA PONAŠANJA ATM MAŠINE

Primer dijagrama mašine sanja ponašanja koji opisuje rad bankomata (ATM) na najvišem nivou stanja mašine

ATM (Automated Teller Machine) je u staru isključen (slika 9). Posle uključenje struje, ATM izvršava akcije uključenja i ulazi u stanje Self Test (samotestiranje). Ako test autotestiranja spremnosti za rad bude negativan, ATM dobija stanje Out od Service (sistem je u kvaru). Ako je test pozitivan, počinje prelaz bez trigera (u stanje Idle (čekanje). U ovom stanju ATM čeka interakciju korisnika.

Slika 8.9 Dijagram mašine stanja ponašanja ATM [2.1]

DIJAGRAM STANJA (VIDEO)

Trajanje: 12:56 min.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 9

Dijagram aktivnosti

AKTIVNOST

Aktivnost je parametrizovano ponašanje predstavljeno i koordinisano tokom akcija.

Aktivnost (Activity) je parametrizovano ponašanje predstavljeno i koordinisano tokom akcija. Tok izvršenja se modelira kao **čvor aktivnosti** koji je povezan **linijom sa strelicom** (activity edges).

Aktivnost sadrži sledeće čvorove aktivnosti: akcija (action), objekat (object) i kontrola (control). Akcija je jedan korak unutar aktivnosti Aktivnosti sadrže različite **vrste akcija.**

Slika 1 pokazuje UML grafički simbol aktivnosti (pravougaonik sa zaobljenim uglovima) sa imenom aktivnosti (boldiran) u gornjem levom uglu i sa čvorovima i strelama unutar granica pravougaonika.

Parametri aktivnosti se prikazuju na granici, ispod imena aktivnosti u formi: parameter-name: parameter-type (slika 1.b)

Slika 9.1 UML oznake aktivnosti (bez i sa parametrima) [1.4]

Aktivnost može da sadrži pred- i post-uslove ograničenja izvršavanja. koje sevode iza ključnih reči "precondition" i "postcondition". Ključna reč "singleExecution" se navodi u slučaju pojedinačne podeljenog izvršenja (single shared execution). Aktivnost se može prikazani i u obliku okvira sa oznakom **activity** ili **act** ispred imena u tagu imena (slika 2)

Slika 9.2 UML oznaka aktivnosti u formi okvira [1.4]

OBELEŽAVANJE AKTIVNOSTI I AKCIJA

Aktivnost i akcija, kao jedan korak unutar aktivnosti, obeležavaju se pravougaonikom sa zaobljenim uglovima.

Kao što je već napomenuto, aktivnost je parametizovan redosled ponašanja. Aktivnost se prikazuje pravougaonikom sa zaobljenim ugovima kojim se obuhvataju sve akcije koje se izvršavaju unutar jedne kativnosti (slika 3).

Slika 9.3 UML oznaka jedne aktivnosti [1.4]

Akcija je jedan korak unutra jedne aktivnosti. Obeležava se takođe pravougaonikom sa zaobljenim uglovima (slika 4)

Slika 9.4 UML oznaka akcije [1.4]

Ograničenja se mogu pridodati nekoj akciji. Na slici 5 prikazan je primer preduslova i postuslova definisanih za izvršenje jedna akcije.

Slika 9.5 Primer prikazivanja ograničenja za izvršenje akcije Dispense Drink [1.4]

KONTROLA PROTOKA

Kontrola protoka pokazuje tok kontrole od jedne do druge akcije i obeležava se linijom sa strelicom.

Kontrola protoka (control flow) pokazuje tok kontrole od jedne do druge akcije. Obeležava se linijom sa strelicom (slika 6a, slika 7).

Activity Edge obezbeđuje direktno povezivanje čvorova tokom objekata podataka. Strelice mogu da imaju ime, ali to ne moraju da budu unikatna imena unutar aktivnosti. Strelica ima ime koje se postavlja blizu strelice (slika 6.b).

Slika 9.6 Povezivanje aktivnosti [1.4]

Protok kontrole može da ima uslov (guard) - specifikaciju koja se ocenjuje u vreme izvršenja da bi se odredilo da li se se ivica može proći. Uslov mora da se oceni da bude true (istinito) da svaki token koji se nudi da prođe duže strele. Uslov granice aktivnosti (slika 6.c) se prikazuje u obliku zaokrenutog romba (slika 6.c) koji sadrži tokent.

Granica aktivnosti se može obeležiti upotrebom **konektora** (slika 6.d) koji je obeležen malim crnim krugom sa upisanim imenom.

Konektori se koriste da bi se izbegle duge kontrole protoka, jer se sa konektorima dele na dve ili više manjih tokova kontrole. Konektori su samo način obeležavanja i njihova upotreba ne menja model. Krugovi i linije u nekom dijagramu se mapiraju u jednu aktivnost u modelu. Svaki konektor sa datim imenom mora da se spoji sa drugim konektoroma koji koristi isto ime.

Slika 9.7 Tok kontrole između aktivnosti Send Payment i Accept Payment [1.4]

TOK OBJEKATA

Tok objekata je put prolaza objekata ili podataka između akcija

Tok objekata (object flaw) je put prolaza objekata ili podataka između akcija, Objekt se predstavlja prvougaonikom (slika 8).Tok objekata se obeležava linijom sa strelicom koja pokazuje smer kretanja objekta (slika 8a). Tok objekta mora da ima najmanje jedan objekat na svom kraju. Kraći način obeležavanja koristi ulazne i izlazne pinove (slika 9.b).

Slika 9.8 objekat

Slika 9.9 Tok objekta [1.4]

Početni čvor (initial node) obeležava se crnim krugom (slika 10)

Slika 9.10 Početni čvor [1.4]

Postoji dva tipa krajnjih čvorova: krajnji čvor aktivnosti i krajnji čvor toka. Krajnji čvor aktivnosti (activity final node) se prikazuje sa krugom sa upisanim manjim crnim krugom (slika 11.a). Krajnji čvor toka (flow final node) se obeležava krugom sa upisnim zaokrenutim krstom X (slika 11.b). Razlika između ovih tipova krajnjih čvorova je u tome što krajnji čvor toka označava kraj jednog toka kontrole, a krajnji čvor aktivnosti označava kraj svih tokova kontrole unutar jedne aktivnosti.

Slika 9.11 Krajnji čvor aktivnosti i krajnji čvor toka [1.4]

ČVOR ODLUČIVANJA

Čvor odlučivanja je čvor kontrole koji prima ulazni tok (kontrole ili objekata) i bira jedan izlazni tok od jednog ili više mogućih tokova, i to oni koji zadovoljava postavljen uslov.

Čvor odlučivanja (decision node) je čvor kontrole (upravljanja) jer prima ulazni tok (kontrole ili objekata) i bira jedan izlazni toka od jednog ili više mogućih tokova, i to oni koji zadovoljava postavljen uslov. Ustvari, čvor odlučiavnja je **čvor granjanja** toka.

Nema mešanja tokova kontrole ili objekata. Svi tokovi u čvoru odlučivanja moraju biti istog tipa, tj. ili su svi tokovi kontrole, ili tokovi objekata. Oznaka čvora odlučivanja je zaokrenuti romb (slika 12.a)

Da bi se obezbedilo da bar jedan izlazni tok će uvek biti "pušten" na izlazu, pri definisanju uslova izlaznih tokova, treba na kraju koristiti uslov "else". To garantuje da će tako obeležena grana toka biti aktiviranja ako uslovi ostalih izlaznih tokova nisu zadovoljeni (slika 12.b)

Slika 9.12 Čvor odlučivanja - jedan od izlaznih tokova se mora izabrati [1.4]

Čvor odlučivanja može imati i definisano ulazno ponašanje (decision input behavior), tj. uslov ponašanja koji ulazni tok mora da zadovolji, da bi se onda odlučivalo koji od izlaznih tokova će biti odobren na osnovu njihovih uslova. Kriterijum za ulazne tokove se određuje i obeležava posle ključne reči ***<<decisionInput>>*** posle čega se u čvor upisuje jedan ili više uslova. Kako nema prostor da se to upiše u samom čvoru, to se uslov upisuje u vidu obaveštenja koje sa vezuje za čvor odlučivanja (slika 13.a).

Može se koristiti i ulazni tok kontole (decision input flow) koji uslovi izlaznih tokova moraju da uzmu u obzir pri odlučivanju (slika 13.b). Ulazni tok kontrole se upisuje posle ključne reči***<<decisionInputFlow>>***

Slika 9.13 Uslov ulaznog ponašanja (a) ili ulazni tok kontrola (b) se moraju zadovoljiti pre provere uslova izlaznih tokova [1.4]

ČVOR SPAJANJA

Čvor spajanja (merge node) ima na ulazu više alternativnih tokova (kontrole ili objekata) i bira jedan od njih koji onda izlazi iz čvora.

Čvor spajanja (merge node) ima na ulazu više alternativnih tokova (kontrole ili objekata) i bira jedan od njih koji onda izlazi iz čvora. Čvor spajanja se ne koristi za sinhronizaciju ulaznih tokova, jer on spaja ulazne tokove, već samo bira jedan od njih (slika 14.a).

Funkcionalnost čvora odlučivanja i spajanja se može kombinovati korišćenjem samo jednog čvora (slika 14.b). U ovom slučaju, čvor prima više ulaznih tokova i bira jedan od njih (funkcija spajanja), ali na izlazu ima više izlaznih tokova, koji imaju svoje definisane uslove aktiviranja. (svojstvo odlučivanja)..

Slika 9.14 Čvor odlučivanja bira jedan od ulaznih tokova (a), a ako je integrisan sa čvorom odlučivanja, može imati i više izkaznih tokova sa svojim uslovima (b) [1.4]

Čvorovi odlučivanja (decision nodes) i **čvorovi spajanja** (merge nodes) se obeležavaju na isti način (zaokrenurt romb) (slika 15). Oba moraju da imaju naziv tipa. Tokovi kontrole koji izlaze iz čvora odlučivanja imaju uslove (guard) kojim se vrši kontrola toka, jer je tok mogućan samo ako je uslov ispunjen.

Slika 9.15 Primena čvorova odlučivanja i spajanja [1.4]

ČVOR RAČVANJA I ČVOR SJEDINJAVANJA

Čvor račvanja (fork node) je čvor kontrole koji ima jedan ulazni tok, a više izlaznih tokova. Čvor sjedinjavanja ima na ulazu više istovremenih ulaznih tokova, a na izlazu samo jedan tok.

Čvor račvanja (fork node) je čvor kontrole koji ima jedan ulazni tok (kontrole ili objekata), a više izlaznih tokova. On se upotrebljava za podelu ulaznog toka na više istovremenih tokova. Čvor račvanja služi za podršku paralelizma u dijagramima aktivnosti. Ulazni tok se kopira u više izlaznih tokova, ali je najmanje jedan prihvaćen od svog ciljnog čvora koji treba da ga primi. (slika 16.a)

Čvor sjedinjavanja (join node) ima na ulazu više istovremenih ulaznih tokova, a na izlazu samo jedan. Izlazni tok je jedan od ulaznih, zavisno od toga koji je prvi došao do čvora prikupljanja. On se koristi za sinhronizaciju ulaznih istovremenih tokova (slika 16.b). Čvorovi

sjedinjavanja imaju specifikaciju sa logičknom vrednošću (istinito ili neistinito) koja koristi nazive ulaznih tokova kod određivanja uslova "prolaza", tj. prihvatanja tokena (objekta ili kontrolnog signala). Svaki ulazni tok mora da nudi bar jedan token. Specifikacija čvora prikupljanja se unosi između velikih zagrada kao {joinSpec=...}, (slika 16.c)

Slika 9.16 Čvor račvanja i čvor prikupljanja (b) koji može imati specifikaciju (c) [1.4]

Izlazni tok iz čvora sjedinjavanja ne može se izvršiti sve dok se svi ulazni tokovi kontrole se ne izvrše. Sjedinjavanje omogućava tokove kontrole kroz čvor. Ako se dva ili više dolazećih tokova primaju u čvoru sjedinjavanja, akcija koja je prikazana na izlaznom toku se izvršava dva ili više puta. Znači, račvanja (forks) i čvorovi sjedinjavanja (joins) bez nekih uslova propuštaju obeležene tokove kontrole. Uvek kad jedan tok kontrole dođe do čvora račvanja ili čvora sjedinjavanja, tok kontrole se javlja na izlazu. *Jedini uslov je da se izvrše svi dolazeći tokovi pri sjedinjavanju tokova kontrole*. (slika 17)

Slika 9.17 Čvor račvanje i cvor sjedinjavanja paralelnih tokova u dijagramu aktivnosti [1.4]

PRIKAZ ČVOROVA KONTROLE U DIJAGRAMIMA AKTIVNOSTI

Čvorovi kontrole u dijagramima aktivnosti određuju tok aktivnosti od početka do kraja dijagrama.

Radi sumiranja izloženog o čvorovima kontrole u dijagramima aktivnosti, na slici 18 prikazani su svi prethodno navedeni čvorovi kontrole u dijagramima aktivnosti:

- početni čvor (initial node)
- krajnji čvor toka aktivnosti (activity final node)
- čvor odlučivanja (decision node)
- čvor spajanja (merge node)
- čvor račvanja (čvor račvanja)
- čvor sjedinjavanja (join node)

Slika 9.18 Prikaz različiti vrsta čvorova kontrole u dijagramu aktivnosti [1.4]

VERTIKALNI PRIKAZ DIJAGRAMA AKTIVNOSTI

Dijagrami aktivnosti sa vertikalnim prikazom se čitaju odozgo na dole i imaju grananje i razdvajanje za opisivanje uslova i paralelnih aktivnosti.

Dijagram aktivnosti prikazuje tok aktivnosti kroz sistem. **Dijagrami aktivnosti sa vertikalnim prikazom** se čitaju odozgo na dole i imaju grananje i razdvajanje za opisivanje uslova i paralelnih aktivnosti. (Slika 19)

Na slici 20 pokazan je primer u kome je simbol odluke iskorišćen za prikaz iteracija.

Slika 9.19 Objašnjenje formiranja jednog dijagrama aktivnosti [1.4]

Slika 9.20 Primer dijagrama aktivnosti sa petljama i iteracijama [1.4]

PARTICIJA AKTIVNOSTI

Particija aktivnosti je grupa aktivnosti koje imaju neku zajedničku karakteristiku.

Particija aktivnosti (activity partition) je grupa aktivnosti koje imaju neku zajedničku karakteristiku. To može da bude po mestu izvršavanja (po organizacionim jedinicama), po nosiocu izvršavanju (uloga zaposlenog) i dr.

Partcija aktivnosti se prikazuje trakom sa dve paralelne (vertikalne ili horizontalne linije) sa njihovim nazivom u odeljku na njegovom kraju za upis imena (slika 21). Čvorovi aktivnosti, npr. akcije i strele se postavljaju između ovih linija koje se onda smatraju da su deo u particiji.

Slika 9.21 Načini grafičkog predstavaljanja traka u digramima aktivnosti. [1.4]

Particija može da se obeleži kao dimenzija sa svojim podparticijama. Na primer, neka aktivnost može jednu dimenzuju particija za lokaciju gde se ponašanja odvijaju,a drugu dimenziju za tropkove izvršenja te aktivnosti. Particije dimenzija ne mogu da budu sadržene o okviru bilo koje druge particije.

Primer na slici 21 opisuje kupovine nekog proizvoda

Slika 9.22 Dijagram aktivnosti procesa kupovine jednog proizvoda [1.4]

DIJAGRAM AKTIVNOSTI (VIDEO)

Trajanje:12:22 min

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 10

Pokazna vežba

PRIMER 1 - DIJAGRAM KLASA: DIJAGRAMA KLASA - ONLAJN PRODAVNICA

Primer pokazuje domenski model onlajn prodavnice. Customer, Account, Shoping Cart, Product, Order, Payment

Slika pokazuje UML dijagram klasa onlajn prodavnice. Svaki kupac (Customer) ima identitet veb korisnika. Veb korisnika mogao bi da bude u jednom od nekoliko stanja i trebalo bi da bude povezan sa kolicom za kupovinu.

Slika 10.1 Domenski dijagram klasa onlajn prodavnice [2.1]

PRIMER 2 - DIJAGRAM PAKETA: DIJAGRAM PAKETA ZA VIŠESLOJNU ARHITEKTURU

Primer prikazuje dijagram paketa koji predstavlja model jedne višeslojne aplikacije. Svaki sloj je predstavljen jednim UML modelom

Primer prikazuje dijagram paketa koji predstavlja model jedne višeslojne aplikacije, koja se oslanja na Microsoft Application Achitecture Guide, 2nd Ed.

Model aplikacije pokazuje nekoliko slojeva:

- prezentacioni sloj,
- servisni sloj,
- poslovni sloj
- sloj podataka
- drugi slojevi

Svaki sloj predstavlja jedan UML model

Slika 10.2 Dijagram paketa višeslojne aplikacijie [1.4]

PRIMER 3 - DIJAGRAM RASPOREĐIVANJA: DIJAGRAM MREŽNE ARHITEKTURE

Dijagrami mrežne arhitekture obično pokazuju umrežene čvorove i komunikacione puteve koji ih povezuju

UML nema posebne dijagrame sa opisivanje arhitekture mreže i ne obezbeđuje posebne elemente povezane sa mrežama. Dijagrami raspoređivanja se mogu koristiti za ove svrhe sa dodavanjem nekoliko mrežnih stereotipa. Dijagrami mrežne arhitekture obično pokazuju umrežene čvorove i komunikacione puteve koji ih povezuju.

Na slici 3 prikazan je dijagram mrežne arhitekture sa konfiguracijom koja se naziva "demilitarizovana zona sa dva zaštitna zida" . Demilitarizovana zona (DMZ) je segment servera ili mreže koji je lociran u neutralnoj zoni između Interneta i privatne mreže organizacije. Ona sprečava spoljnje korisnike da dobiju direktan pristup interne mreže organizacije bez direktnog prikaza na Internetu svog veb servera, e-mail servera ili DNS servera.

Prikazane ikone nisu deo UML standarda.

Slika 10.3 Primer dijagrama mrežne arhitekture sa raspoređenim uređajima [1.4]

PRIMER 4 - SEKVENCIJALNI DIJAGRAM: PREGLED INFORMACIJA O PACIJENTU

Primer pokazuje relizaciju osnovnog (primarnog) i alternativnog (sekundarnog) scenarija

Sekvencijalni dijagram na slici 4 na grafički način opisuje interakciju slučaja korišćenja “Pregled informacija pacijenta”. U gornjem delu je dat osnovni (primarni) scenario, a u donjem je dat alternativni, tj. pomoćni ili sekundarni scenario. On opisuje vanrednu situaciju do koje dolazi zbog prekida pristupa objektu A.S. U kome se vrši autorizacija.
Pored sekvencijalnog dijagrama, isti scenario se može opisati i tekstualno, navođenjem redosleda akcije aktera i sistema.

Slika 10.4 Prijem pacijenta (MHCPM slučaj) [1.2]

Iz dijagrama na slici 9 čita se sledeći scenario interakcije:

1. Medicinski recepcioner pokreće pogledajInfo operaciju u objektu P klase PatientInfo, ubacujući identifikator pacijenta PID, P je objekt za interakciju sa korisnikom, prikazan u vidu forme sa podacima o pacijentu.
2. Objekt P poziva bazu podataka D da vrati traženu informaciju, unoseći identifikacioni broj recepcionera radi sigurnosne provere.
3. Baza podataka proverava sa sistemom za proveru ovlašćenja A.S da li je korisnik (recepcioner) autorizovan za ovu akciju, tj. da li ima dozvolu za ovu akciju.

4. Lična informacija se šalje direktno sa objekta korisničkog interfejsa u PRS sistem. Alternativno, može se formirati slog iz baze podataka i onda se taj slog šalje (prikayom **alt** fragmenta.

Ako se dijagram interakcije ne koristi za generisanje izvornog koda ili detaljnu dokumentaciju, on ne mora da sadrži svaku interakciju. Često se detalji o interakcijama ostavljaju za fazu implementacije (kodiranja).

PRIMER 5 - SEKVENCIJALNI DIJAGRAM: DOBIJANJE OVLAŠĆENJA ZA PRISTUP PODACIMA PACIJENTA

Primer za bolje razumevanje razvijanja scenarija

Ako se dijagram interakcije ne koristi za generisanje izvornog koda ili detaljnu dokumentaciju, on ne mora da sadrži svaku interakciju. Često se detalji o interakcijama ostavljaju za fazu implementacije (kodiranja) te dijagrami interakcije u fazi definisanja zahteva ili koncepcijskog projektovanja ne sadrže mnoge poruke i detalje.To je ostavljeno za fazu implementacije. Na primer, odluka o tome kako dobiti identifikator za dobijanje ovlašćenja (autorizaciju) korisnika, slučaja datog na slici 5, je odložena. U fazi implementacije ovo je uključeno u interakciju sa objektom Korisnik, ali to nije važno u ovoj fazi i ne mora da bude uključeno u sekvencijalni dijagram

Slika 10.5 Dobijanje ovlašćenja za pristup podacima pacijenta (MHCPMS slučaj) [1.2]

PRIMER 6- DIJAGRAM AKTIVNOSTI: MAŠINA ZA PRODAJU AVIONSKIH KARATA

Primer UML dijagrama aktivnosti koji opisuje ponašanje slučaja korišćenja kupovine karte (Purchse Ticket) na mašini za prodaju karata (Ticket vending machine).

Aktivnost počinje kada učesnik Commuter želi da kupi kartu. Mašina za prodaju karata od njega zahteva da unese zahtev za putovanje. Zavisno od informacija koje sadrži ovaj zahtev (broj i tip karata, broj leta, destinacija putovanja i dr.) , kao i od izabranih opcija (opcije načina plaćanja; keš ili karticom), mašina izračunava cenu karte. Banka, kao drugi učesnik u ovom slučaju korišćenja, vrši autorizacijau transakcije, tj. odobrava izdavanje karte,

Pri plaćanju kešom, može doći do situacije da mašina mora da vrati kusur. Kada je plaćenja završeno, karta se isporučuje kupcu i na ekranu se pojavljuje tekst "Thank You".

Slika 10.6 Dijagram aktivnosti kupovine avio karte na mašini za prodaju [1.4]

PRIMER 7- DIJAGRAM AKTIVNOSTI: UML DIJAGRAM AKTIVNOSTI ONLAJN PRODAVNICE

Primer dijagrama aktivnosti onlajn prodavnice

Kupac onlajn prodavnice može da pretražuje i traži željeni artikl, da ga pregledam da ga doda u svoju korpu kupovine, da pregleda sadržaj korpe kupovine i da je ažurira, i da se na kraju odjavi. Kupac celo vreme treba da ima pogled na svoju korpu kupovine.

Slika 10.7 Dijagram aktivnosti onlajn prodavnice [1.4]

PRIMER 8-DIJAGRAM AKTIVNOSTI: SERVIS ZA DOBIJANJE ELEKTRONSKIH RECEPATA

Primer dijagrama aktivnosti EPS servisa za izdavanje eletronskih recepata koji je razvijen od strane Connecting for Health (NHS CFH) iz Engleske

EPS omogućava korisnicima - lekarima opšte prakse i sestrama - da pošalju recepte elektronskim putem apotekama za određenog pacijenta. Pristup EPSobezbeđuju NHS smart kartice, sa imenom, fotografijom i JMBG, i sa čipom. Kartica daje i nivo prava pristupa servisima, zavisno od uloge učesnika, tj. korisnika. On mora da unese i šifru za izdavanje lekova određenom pacijentu, daje uputsva korišćenja i koristi svoj elektronski potpis, koji se prenosi EPS-u. Recept se može odštampati i dati pacijentu u apoteci.

Isporučilac leka je unapred određena apoteka, ili ordinacija lekara opšte prakse i dr. kojoj se šalje elektronski recept. Apoteka dobija uvid u recept automatski štampanjem pristiglih recepata noću, ili ručnim unošešem ID recepta, ili skeniranjem barkoda isprintanog recepta. Lekovi se izdaju pacijentu ili njegovom predstavniku. Apotekar mora da upiše staus obrade recepta. Na kraju, svi lekovi moraju da imaju status "isporučen" ili "ne isporučen". Apoteka šalje poruku EPS-u o realizaciji recepta. Apoteka dobije raspore za slanje zahteva za plaćanje izdatih lekova. EPS elektronski daje podršku apoteci na pošalje zahtev za plaćanje osiguranju, tj. agenciji za plaćanje izdatih lekova, koja plaća lekova, a u skladu sa njenim raporedom plaćanja.

Slika 10.8 Dijagram aktivnosti servisa za izdavanje elektronskih recepata [1.4]

PRIMER 9- DIJAGRAM AKTIVNOSTI : DIJAGRAM AKTIVNOSTI ZA RAZREŠAVANJE PRIMEDBI NA PROJEKTNO REŠENJE SOFTVERA

Primer UML dijagrama aktivnosti postupka za rešavanje primedbi na projektno rešenje softvera.

Kada se kreira karta (ticket) od nekog koji ima ovlađćenje pristupa sistemu, i kada se problem se opisuje, određuje mu se identifikator i nalazi se način rrešavanja uočenig problema. Izdaje se potvrda da je problem rešen i proveren, i vrši se zatvaranje karte, ako je problem rešen.

U ovom primeru se ne koriste particije, te nije jasno ko je odgovoran za realizaciju određenih akcija.

This example does not use partitions, so it is not very clear who is responsible for fulfilling each specific action.

Slika 10.9 Dijagram aktivnostu za razrešavanje primedbi na projektno rešenje softvera [1.4]

PRIMER 10- DIJAGRAM AKTIVNOSTI: JEDINSTVEN PRISTUP GOOGLE APLIKACIJAMA

Primer UML dijagrama aktivnostima koji opisuje jedinstven pristip (Single-Sign-On) Google aplikacijama

Radi interakcije sa partnerskim kompanijama, Google upotrebljava jedinstven pristup aplikacijama (single-sign-on) u skladu sa OASIS SAMLL 2.0 protokolom. Google ima ulogu provajdera servisa, kao što su Gmail ili Start Pages. Partnerske kompanije umaju ulogu provajderi identiteta i kontrole imena korisnika, laozing i drugih informacija koje se upotrebljavaju za identifikaciju, proveru autentičnosti i odobravanje priistupa za određenog korisnika određene veb aplikacije koje postavljena na Google serveru ("hostovanje"). Svaki partner obezbeđuje URL svog SSO servisa kao i javni ključ koji će Google koristiti da bi verifikovao SAML odgovore.

Google šalje korisniku sistema zahteva za SAM proveru autentičnosti i šalje i zahtev nazad na veb pretraživač korisnika, kao i određenm provajderu identiteta. Zahtev za SAML proveru autentičnosti korisnika sadrži URL Google aplikacije koji korisnik želi da koristi.

Provajder identiteta vrši proveru autentičnosti korisnika tražiči od njega login podatke, ili proverava svoje kolačiće (cookies) za proveru autentičnosti. Partner generiše SAMPL odgovor i digitalno ga potpisuje i šanje Assertion Consumer Service (ACS) Googl-a. On verifikuje zahtev upotrebom javnog ključa partnera. Ukoliko je odgovor pozitivan, ACS usmerava korisnika na ciljni URL. U suprotnom, dobija poruku o grešci.

Slika 10.10 Jedinstven pristup (Single-Sign-On) Google aplikacijama [2.1]

PRIMER 11 - DIJAGRAM AKTIVNOSRU: PUŠTANJE PROBNOG PROIZVODA POD ZAŠTITOM SENTINEL HASP SL

Primer UML dijagrama aktivnosti procesa aktiviranja probnog proizvoda pod zaštitom Singtel HASP SL softvera.

Sentinel HASP vrši zaštitu od proizvoda od sofftverske piraterije i zaštitu intelektualne svojine od krađe. Softver koristi J2EE platformu.

Sentinel HASP softvare i uključuje Business Studio - moćan softver za izdavanje licenci i alat za upravljanje. Bussiness Studio koriste projektante proizvoda, marketing i inženjeri razvoja kod pripreme softverskog proizvoda za tržište i uključuje sve alate neophodne za licenciranje i zaključivanje aplikacije kod Senitel HASP SL hardvera ili HASP SL ključa softvera za uprvljanje licencama i praćenje promena, za kreiranje ključeva koji se kasnije koriste za aktiviranje proizvoda itd.

Dijagram ima 3 particije za 3 učesnika Order Management, Customer Service, and Customer. Customer ima instalisanu verziju za određeni period probem i može imati neke ograničenja funkcija softvera ili opcije. PO upotreba softvera u nekom periodu, korisnik odlučuje da li da aktivire zahtev za dobijanje i pune licence proizvoda. U tom slučaju, Order management kreira novi ključ za aktiviranje proizboda dok kupac treba da kreira i pošalje C2V file ("computer fingerprint").Onda odnos sa kupcima aktivira trajnu licencu instaliranom softveru, generiše V2C fajl, koga šalje kupcu.

Slika 10.11 Dijagram aktivnosti provere novog proizvoda sa zaštitom Singtel HASP SL [2.1]

PRIMER 12 - SLUČAJEVI KORIŠĆENJA: KONTROLA LANSIRANJA RAKETA

Na osnovu opisa sistema definisan je dijagram slučajeva korisćenja.

Prikazati dijagram slučajeva korišćenja (eng. use case diagram) na sici 1 za softverski sistem za kontrolu lansiranja raketa. Pilot može izvršiti izbor cilja, lansiranje rakete, proveru broja raketa. Svako lansiranje rakete potrebno je da potvrdi, pri čemu se potvrda može obaviti odgovarajućom glasnom komandom ili pritiskanjem odgovarajućeg tastera na upravljačkoj konzoli. Ukoliko se tokom lansiranja desi kvar na sistemu za lansiranje pilot će biti obavešten o ovome događaju i potrebno je da potvrdi prijem poruke.

Slika 10.12 Slučajevi korišćenja sistema za lansiranja raketa [Izvor: Autor]

PRIMER 13 - SEKVENCIJALNI DIJAGRAM: SISTEM ZA RAD SA FIGURAMA U 2D RAVNI

Na osnovu opisa sistema definisan je sekvencijalni dijagram.

UML-om modelovati sistem za rad sa figurama u 2D ravni. Od figura za sada postoje Kvadrat i Krug. Kvadrat je opisan dužinom stranice, dok je Krug opisan poluprečnikom. Ove veličine korisnik zadaje prilikom konstrukcije odgovarajućeg objekta. Svakoj figuri se može izračunati površina. Dodatno korisniku je na raspolaganju kolekcija figura koja može sadržati proizvoljan broj 2D figura. Kolekciji se takođe može izračunati površina kao suma površina svih figura koje sadrži. Na osnovu opisa sistema i dijagrama klasa sa sike 13 kreiran je i sekvencijalni dijagram za kreiranje i popunjavanje jedne KolekcijeFigura (slika 14)

Slika 10.13 Klasni dijagram [Izvor: Autor]

Slika 10.14 Sekvencijalni dijagram [Izvor: Autor]

PRIMER 4-DIJAGRAM AKTIVNOSTI: SISTEM ISPORUKE PORUDŽBINE

Na osnovu opisa sistema definisan je dijagram aktivnosti.

Na slici 15 je prikazan sistem za naručivanje, isporuku porudžbina. Dijagram aktivnosti prati aktivnosti procesa u zavisnosti od izbora odgovarajuće akcije. Na slici su prikazani i detalji koji opisuju komponente dijagrama aktivnosti.

Slika 10.15 Dijagram aktivnosti sistema za naručivanje i isporuku. [Izvor: Autor]

Poglavlje 11

Individualna vežba

ZADATAK ZA INDIVIDUALNU VEŽBU - UML DIJAGRAMI SISTEMA TAKSI UDRUŽENJA (45 MINUTA)

Za opisani sistem kreirati tražene UML dijagrame - planirano za dva časa inividualnih vežbi

Sistem taksi udruženja omogućava poručivanje vožnji preko mobilne aplikacije. Naručilac taksija preko aplikacije poručuje vožnju, bira vrstu vozila (limuzina, karavan, mini van), unosi svoju adresu, odredište i dodaje napomenu ukoliko je ima. Adresa se može uneti na dva načina: ručno ili automatski preko GPS sistema koji postoji u mobilnom telefonu i treba biti uključen. Takođe, neophodna je stabilna internet konekcija.
Centralni informacioni sistem u bazi je složen sistem koji se sastoji od više programskih modula. Povezan je preko GPSa sa svim vozilima, prati njihovo kretanje, status (slobodan ili zauzet), lokaciju. Povezan je sa bazom koja čuva podatke o vozačima, vozilima, vožnjama, rasporedu rada i dr.
Nakon prijema poziva, operator ga analizira i na osnovu sistema za praćenje bira najbližeg slobodnog vozača i dodeljuje mu vožnju. Vozač na svom tabletu prima podatke o vožnji a poručilac dobija odgovor preko aplikacije o poručenoj vožnji: informacije o vozilu (broj, marka model), njegovoj lokaciji, koliko je udaljeno vozilo i za koliko vremena će stići, kao i informacije o dužini vožnje, očekivanom vremenu trajanja i ceni. Naručilac će moći da prati svoje vozilo u realnom vremenu dok se bude kretalo ka njemu, kao i tokom vožnje. Na odredištu korisnik bira način plaćanja: kartica, preko aplikacije ili keš. Plaćanje usluge je funkcionalnost koja koristi funkcionalnosti provere mogućnosti plaćanja i funkcionalnost realizacije plaćanja koje se izvršavaju preko bankarskog informacionog sistema. Registrovani korisnici imaju mogućnost plaćanja preko aplikacije koristeći kredit koji su prethodno uplatili. Takođe, registrovani korisnici imaju mogućnost davanja ocene nakon svake vožnje (o vozilu, vozaču, komentar, primedba, itd..). U sistemu će se beležiti svaka vožnja, vozačeve zarade, zarade udruženja, detalje o plaćanjima. Administrator ima mogućnost da preko sistema dodaje nova vozila, vozače, prati zarade i sve ostalo što se čuva u bazi.

Na osnovu navedenog opisa nacrtati:

1. Dijagram slučajeva korišćenja celokupnog sistema. (15 minuta)
2. Dijagram aktivnosti. (15 minuta)
3. Sistem sekvencijalni dijagram (15 minuta)

ZADACI 1 I 2 ZA SAMOSTALNI RAD - SEKVENCIJALNI DIJAGRAMI

Dati su zadaci koji se odnose na dijagrame interakcije.

Zadatak 1 (10 minuta): Opis aplikacije: Ova aplikacija namenjena je za zaposlene u privatnoj klinici (medicinska sestra, doktor, direktor klinike). Svaki korisnik se prijavljuje i na osnovu prijave ima različite opcije korišćenja sistema. Opcije su sledeće:

1. Prijavljivanje na sistem klinike
2. Otvaranje kartona pacijenta
3. Pristup kartonu pacijenta
4. Slanje pacijentovih podataka iz kartona lekaru
5. Prepisivanje terapije
6. Izdavanje recepta za leka
7. Davanje uputa za dalje lečenje
8. Pretraživanje zaposlenih
9. Potpisivanje naloga, zapisnika i pravilnika klinike
10. Prikaz pravilnika klinike

Modelovati sekvencijalni dijagram i dijagram za opisanu aplikaciju.

Zadatak 2 (10 minuta): Opis aplikacije: Aplikacija treba da omogući međusobnu interakciju korisnika putem društvene mreže na kojoj korisnici mogu da dele video sadržaje, četuju međusobno i prate sadržaje koji drugi korisnici dele na društvenoj mreži. Korisnik da bi postao član društvene mreže prvo mora da se registruje, proces registracije jeste jako jednostavan i ne zahteva previše podataka od korisnika već samo one osnovne. Kada se registruje korisnik će moći da deli svoj video sadržaj sa drugim korisnicima, prati sadržaje koji su drugi korisnici podelili, lajkuje i komentariše te iste sadržaje kao i mogućnost četovanja sa drugim registrovanim korisnicima. Takođe, registrovani korisnik će imati opciju da prati druge korisnike koji izbacuju video sadržaj na društvenoj mreži.

Modelovati sekvencijalni dijagram i kolaboracioni dijagram za opisanu aplikaciju.

ZADACI 3 I 4 ZA SAMOSTALNI RAD - SEKVENCIJALNI DIJAGRAMI

Dati su zadaci koji se odnose na sistemske dijagrame sekvenci.

Zadatak 3 (10 minuta):

Sistem za apoteku. Apoteka poseduje dva magacina koji u sebi sadrže lekove. Svaki lek u sebi sadrži informaciju o imenu i datumu isteka. Apoteka vodi evidenciju svoji klijenata. Više klijenata može kupovati više lekova. Od podataka klijenta čuvaju se informacije (ime, prezime, adresa). Napraviti odgovarajući klasni dijagram i transformisati taj dijagram u kod. Nakon transformacije dijagrama kreirati main metodu koja će demonstrirati prvo kreiranje magacina, zatim 3 leka u magacinu, kreiranje jednog klijenta i njegovou kupovunu 2 leka.

Modelovati sistemski dijagram sekvenci za opisani sistem.

Zadatak 4 (10 minuta):

Aplikacija za prodaju mobilnih uređaja. Aplikacija se bavi online prodajom (kupovinom) mobilnih uređaja, zavisno da li korisnik pristupa sistemu kao prodavac (ima mogućnost dodavanja uređaja za prodaju i odgovaranju poruka koje je dobio od kupaca) ili korisnik pristupa sistemu kao kupac (ima mogućnost kupovine i kontaktiranja prodavca). Korisnik ne može koristiti ovaj sistem bez naloga.

Modelovati sistemski dijagram sekvenci za opisanu aplikaciju.

ZADATAK 5 ZA SAMOSTALNI RAD - SEKVENCIJALNI DIJAGRAMI

Ovaj zadatak ima za cilj da studenti obnove znanje UML i njegove primene

Zadatak 5 (20 minuta):

Za potrebe video kluba, potrebno je definisati zahteve koje treba opisati pomoći dijagrama zahteva. Nakon toga potrebno je dizajnirati slučajeve kroišćenja. Kada se završi sa osnovnim modelovanjem zahteva, potrebno je kreirati dijagram klasa sa odgovarajućim atributima i metodama. Za kreirane dijagrame slučajeva korišćenja treba kreirati odgovarajuće SSD dijagrame za glavne scenarije. Nakon toga potrebno je definisati detaljne sekvencijalne dijagrame koji će biti izrađeni na osnovu scenarija koje student bude zadao. Iz sekvencijalnog dijagrama potrebno je izgenerisati dijagram kolaboracije.

Ovaj zadatak treba da se odradi korišćenjem Power Designer alata. Studentu se ostavlja na raspolaganje da razvije ideju i zahteve prema svojoj mašti. Potrebno je da zahtevi budu realni i obuhvataju osnovne scenarije rada video kluba kao što je poznato

ZADACI 6 I 7 ZA SAMOSTALNI RAD - DIJAGRAMI SLUČAJEVA KORIŠĆENJA

Dati su zadaci koji se odnose na modelovanje dijagrama slučajeva korišćenja.

Zadatak 6 (10 minuta):
Opis sistema: Sistem za prodaju PC komponenti. Sistem je namenjen firmama koje žele da naveliko kupuju PC komponente za svoje radnje. Napravljen je tako da olakša proces naručivanja i isporuke svih porudžbina. Podaci i informacije o svim poručivanjima čuvaće se u bazi podataka i time smanjiti broj potrebne dokumentacije za sve narudžbine.
Glavne komponente od kojih će sistem da se sastoji su komponenta za naručivanje i isporuku PC delova, komponenta za upravljanje proizvodima, komponenta za informisanje klijenata i komponenta za upravljanje nalogom.

Modelovati dijagram slučajeva korišćenja za korisnika. Izvršiti specifikaciju svakog identifikovanog slučaja korišćenja.

Zadatak 7 (10 minuta):
Opis sistema: Chatspace sistem simulira osnovne funkcionalnosti jedne chat aplikacije. Sistem omogućava korisniku da pristupi sistemu sa svojim nalogom na osnovu koga dobija sve privilegije korisnika sistema. Korisnik dobija mogućnost da razmenjuje poruke sa drugim korisnicima sistema. Poruke mogu biti u okviru teksta, priče, pesme, i ostalih tekstualnih objava. Korisnik ima mogućnost da poruke šalje drugim korisnicima. Takođe, korisnik ima mogućnost da čita poruke drugih korisnika kao i uvid u listu ostalih korisnika koji koriste sistem.
Modelovati dijagram slučajeva korišćenja za korisnika. Izvršiti specifikaciju svakog identifikovanog slučaja korišćenja.

ZADACI 8 I 9 ZA SAMOSTALNI RAD - DIJAGRAM MAŠINE STANJA

Dati su zadaci koji se odnose na dijagrame stanja.

Zadatak 8 (10 minuta):

Modelovati dijagram stanja za aplikaciju za naručivanje autodelova. Aplikacija omogućava svakom korisniku laku kupovinu bilo kog autodela koji mu je potreban i brzu isporuku na kućnu adresu.

Zadatak 9 (10 minuta):

Modelovati dijagram stanja za proizvoljnu aplikaciju. Prikazati hijerarhiju stanja i definisati događaje.

Poglavlje 12

Zaključak

ZAKLJUČAK

Rezime opisanih modela strukture i ponašanja softverskog sistema primenom UML-a.

1. **Mode**l je apstrakti pogled sistem koji ignoriše neke detalje sistem. Mogu se razviti komplementarni modeli sistema da bi se prikazao kontekst sistem, interakcije, struktura i ponašanje.
2. **Strukturni modeli** pokazuju organizaciju i arhitekturu sistema
3. **Dijagrami klasa** se koriste za definisanje statičke strukture klasa u sistemu i njihove interakcije.
4. **Dijagram paketa** - pokazuje kako je sistem podeljen na logičke grupe i prikazuje zavisnosti među ovim grupama.
5. **Dijagram komponenata** - prikazuje kako se neki softverski sistem podeljen na komponente i pokazuje zavisnosti između ovih komponenti.
6. **Dijagram raspoređivanja** - prikazuje izvršnu arhitekturu sistema koja predstavlja raspored elemenata softverskog sistema po hardverskim elementima sistema.
7. **Modeli ponašanja** se koriste radi opisa dinamičkog ponašanja izvršnog sistema. To se može modelovati iz perspektive obrade podataka od strane sistema, ili od strane događaja koji podstiču reakcije sistema.
8. **Dijagram slučajeva korišćenja** (use case diagram) prikazuje funkcionalnosti sistema koje sistem obezbeđuje u obliku aktera i njihovih ciljeva predstavljenih slučajevima korišćenja i njihovih relacija.Slučajevi korišćenja opisuju interakcije između sistema i spoljnih aktera
9. **Dijagrami interakcija** (interactivity diagrams) opisuju tok kontrole i podataka između objekta i koji čine sistem koji se modelira, a to su:.
10. **Dijagram sekvenci**, je dijagram interakcija koji prikazuje redosleda poruka među objektima-klasama (sequence diagram) u kome se daje i vremenski redosled poruka
11. **Dijagram mašine stanja** (state machine diagram) prikazuje stanja i pomene stanja sistema
12. **Dijagram aktivnosti** (activity diagram) opisuje poslovne tokove i tokove rada komponenata u sistemu.

LITERATURA

Daje se literatura koja je korišćena u pri pripremi ove klekcije.

1. Obavezna literatura:

1.1. Onlajn nastavni materijal za predmet SE201 Uvod u softversko inženjerstvo, školaska 2018/19., Univerzitet Metropolitan
1.2. Ian Sommervile, Software Engineering, Tenth Edition, Pearson Education Inc.,2016
1.3. Using UML - Software engineering with objects and components, 2nd edition, Pearson Education Limited, 2006
1.4. OMG, Unified Modeling Language (OMG UML;), Version 2.5.1, OMG Document Number: formal/2017-12-05,2017, http://www.omg.org/spec/UML/2.5.1

2. Veb:
2.1. https://www.uml-diagrams.org
2.2 https://sparxsystems.com.au/resources/uml2_tutorial/uml2_componentdiagram.html