Modeliranje softvera

Nikola Prica 1036/2014

Cilj prezentacije :

- Kako se grafički modeli koriste da prezentuju softver
- Razumevanje zašto su različiti tipovi modela potrebni i koji su njihovi pogledi na sistem
- Osnovni koncepti dijagrama konteksta, interakcije, strukture i ponašanja
- Upozanvanje sa idejama softverskog inženjerstva zasnovanog na modelima

Uvod

- Modelovanje sistema je pravljenje apstraktne slike sistema grafičkim elementima (UML dijagramima)
- Postoje i formalne metode modelovanja sistema
- Mogu se modelovati postojeći sistemi i sistemi koji tek treba da se naprave

Za šta služe modeli?

- Za analizu i raspravu o novom ili postojećem sistemu
- Kao vid dokumentacije
- Za detaljno opisivanje sistema iz kog se može generisati kod

Pogledi na sistem

- Spoljašnji
- Strukturni pogled sistema
- Pogled na interakcije sistema
- Pogled na ponašanje sistema

UML dijagrami

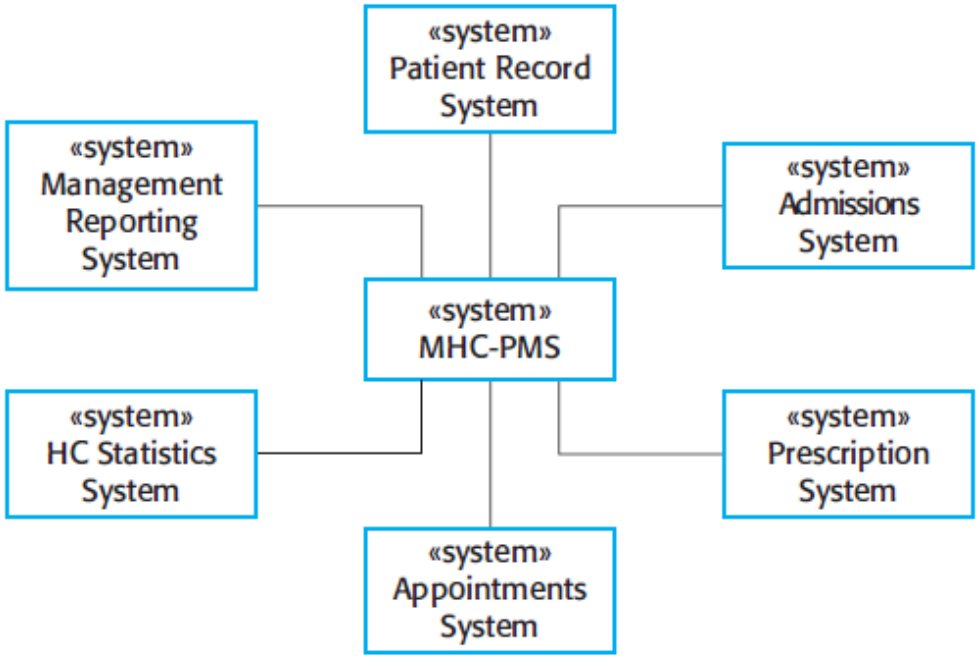
- Za osnovno predstavljanje sistema dovoljno je korišćenje 5 osnovnih UML dijagrama:
  - Slučajeva upotrebe (Use case diagrams)
  - Klasa (Class diagrams)
  - Sekvence (Sequence diagrams)
  - Aktivnosti (Activity diagrams)
  - Stanja (State diagrams)

Modelovanje konteksta

- Vrši se na samom početku radi postavljanja granica sistema
- Ulagač odlučuje koje funkcionalnosti će sistem imati
- Usklađivanje novog sistema sa već postojećim
- Smanjuje troškove i vreme izgradnje sistema

Primer

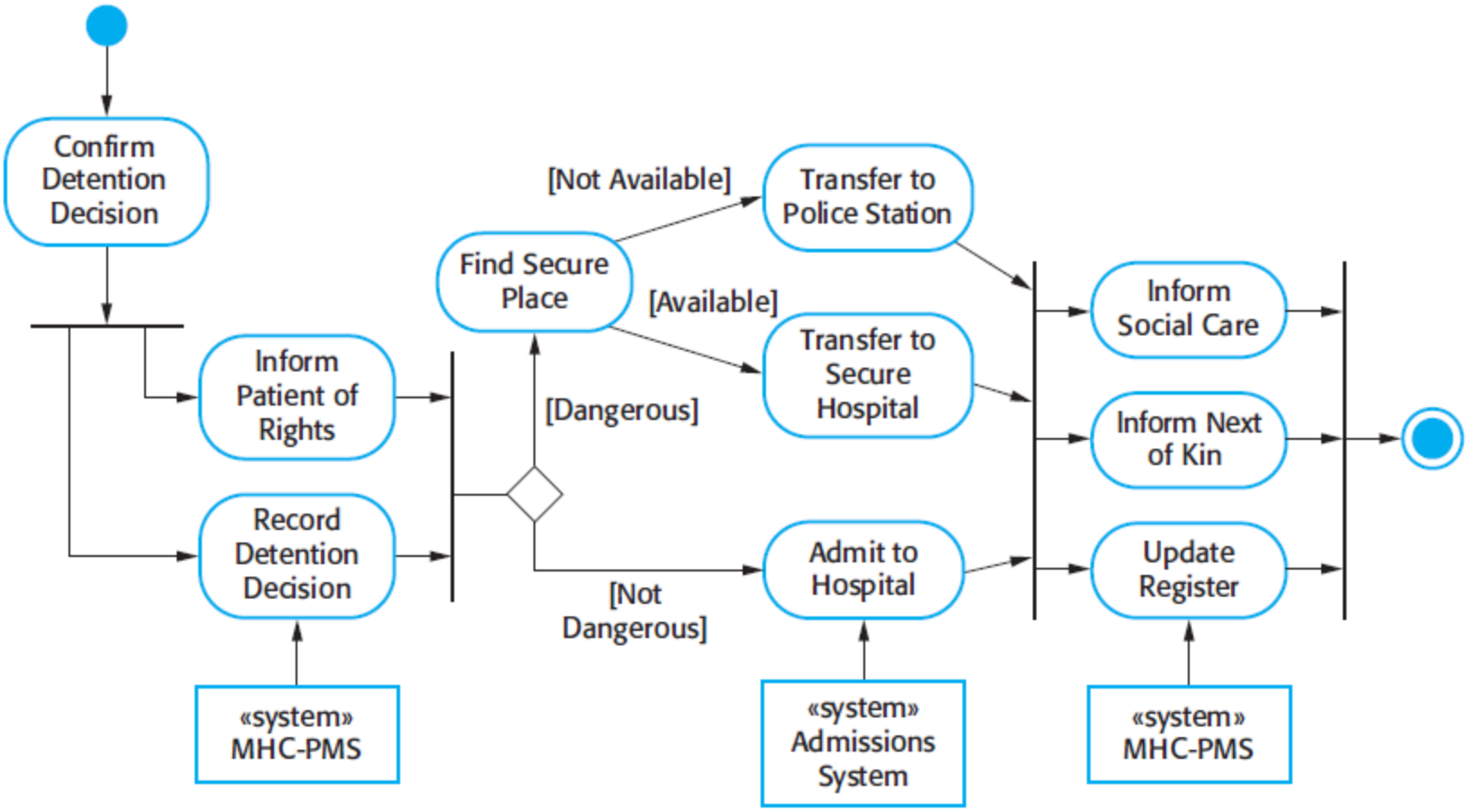
- Upravljanje informacijama o pacijentima klinike i njihvoim tretmanima



Slika 1: Dijagram konteksta Mentalne ustanove

- Da li treba da uključi informacije o doktorima?
- Da li treba da se prikupljaju personalne informacije o pacijentu?
- Da li se koriste neki već postojeći sistemi i kako se ti podaci obrađuju?
- Da li se sva obrada podataka izvršava centralizovano?

- Kada se ispitaju granice sistema i njegov kontekst pravimo dijagram konteksta (slika 1) koji ne prikazuju vrste veza između sistema.
- Za prikazivanje veza između sistema se mogu koristiti dijagrami Aktivnosti
- Dijagrami aktivnosti prikazuju tok aktivnosti nekog procesa i ko je zaduzen za određenu aktivnost



Slika 2: Dijagram aktivnosti prinudne kazne

Modeliranje interakcija

- Služi za predstavljanje međusobne interakcije:
  - Učesnika sa sistemom
  - Sistema koji se razvija sa drugim sistemima
  - Komponenata sistema
- Može da naglasi problem u komunikaciji koji se može pojaviti
- Pokrićemo dva povezana pristupa modeliranju interakcija:
  - Dijagrami slučaja upotrebe (Interakcija sistema sa spoljašnjim učesnicima)
  - Dijagram sekvence (Interakcija sistema sa svojim komponentama i nekim korisnikom)

Dijagrami slučaja upotrebe

- Prikazuju najjednostavnije kako korisnik vidi sistem.
- Svaki dijagram predstavlja nezavisnu celinu koja zahteva spoljasnjeg učesnika koji u njoj učestvuje

Slika 3: Dijagram slučaja upotrebe Transfera podataka

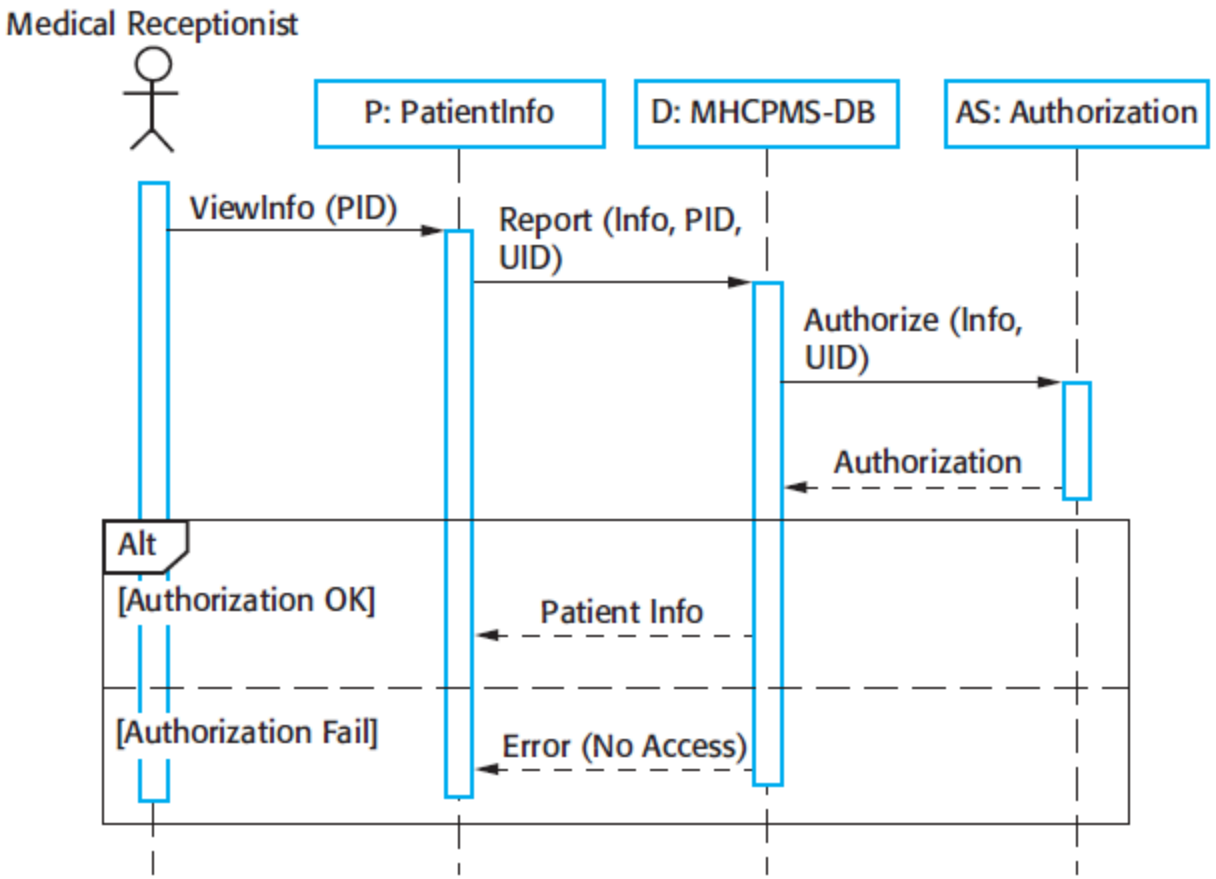
- Sami dijagrami ne daju mnogo informacija
- Potrebne dodatne informacije se dostavljaju u proizvoljnoj formi koje sadrže dodatne informacije

MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the MHC-PMS to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.
Data	Patient's personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

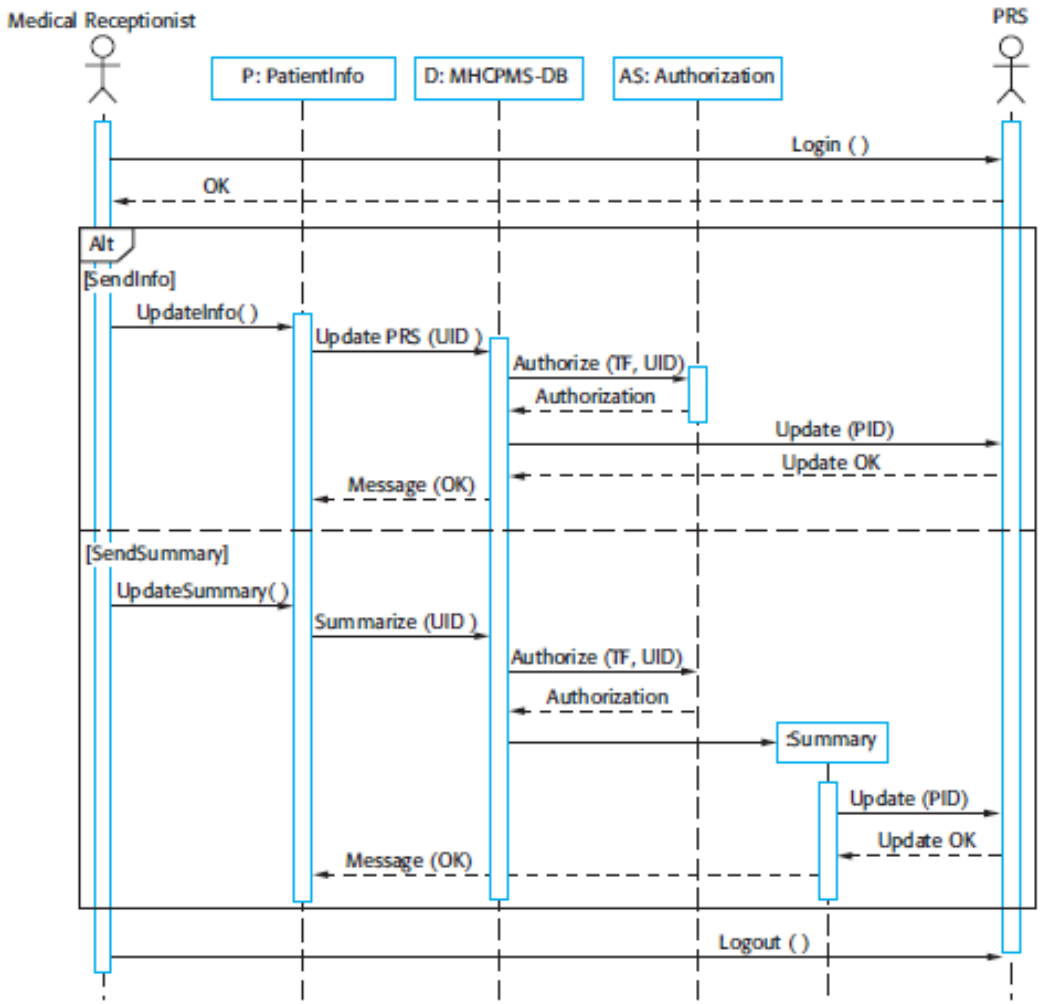
Slika 4: Tabela sa dodatnim podacima o slučaju upotrebe "Transfer podataka"

Dijagrami sekvence

- Primarno su napravljeni za modelovanje odnosa između objekata sistema i učesnika u sistemu, ali i za međusobnih odnosa između objekata.
- Prikazuju sekvencu akcija koje se izvršavaju tokom jednog slučaja upotrebe.



Slika 5: Dijagram sekvence za slučaj upotrebe Pregled informacija o pacijentu



Slika 6: Dijagram sekvence za transfer podataka

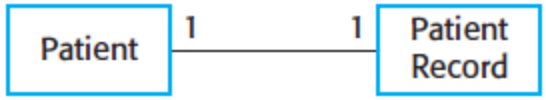
- Ako se dijagrami sekvence ne koriste za automatsko generisanje koda ili detaljnu dokumentaciju, onda se ne moraju navesti sve akcije .
- U ranim fazama razvoja projekta prikazuju se samo okvinre funkcionalnosti koje će tek kasnije zahtevati dodatnu implementaciju.

Strukturni modeli

- Prikazuju organizaciju sistema u vidu njegovih komponenti i odnosa među njima
- Mogu prikazivati statičku ili dinamičku sliku sistema
- Koriste se za dizajniranje arhitekture sistema

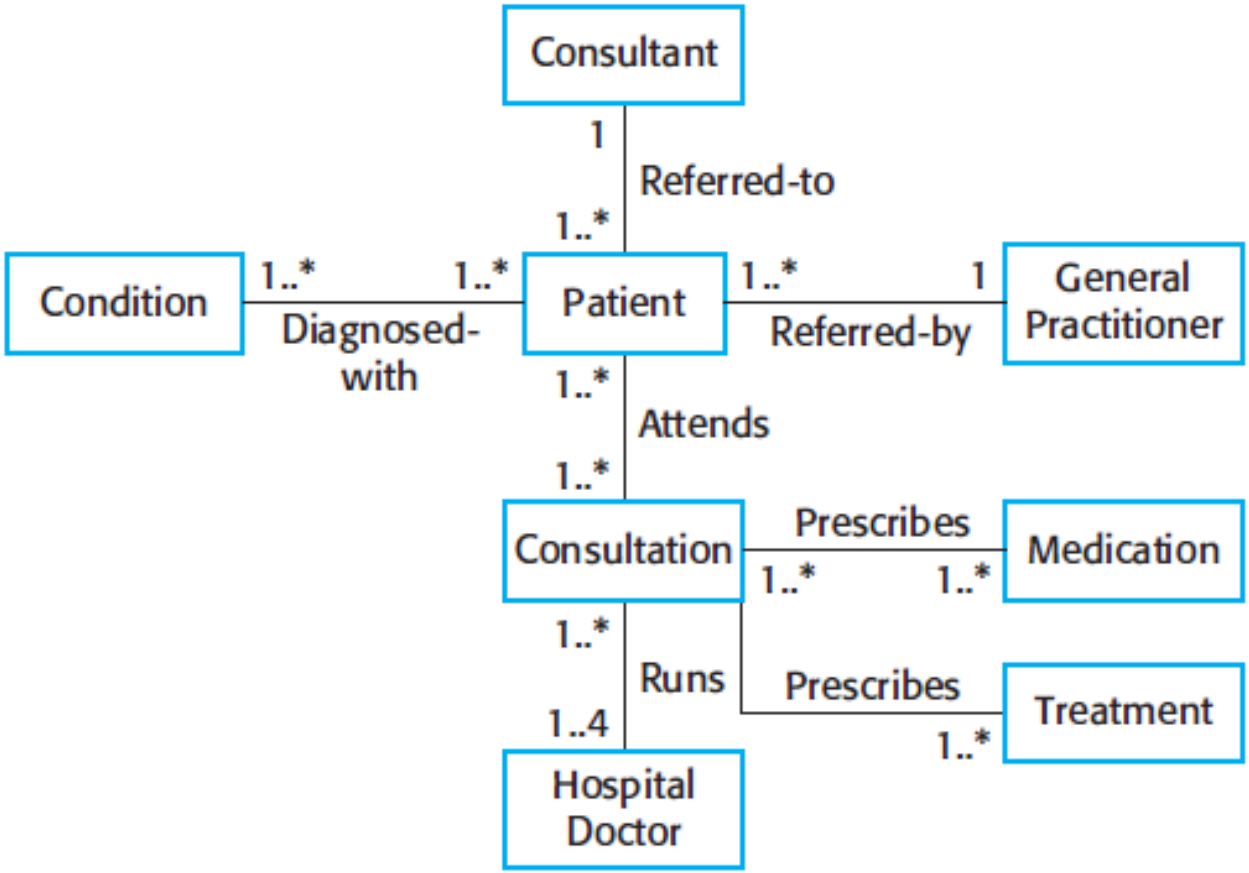
Dijagrami klasa

- Prikazuju statički deo sistema
- Sadrže klase koje se pojavljuju u sistemu i njihove odnose.
- Mogu biti prikazane sa određenom detaljnošću
- Pravljenje klasa počinje od jednostavnih objekata koje sistem modelira. Kasnije se dodaju osobine i operacije tih objekata i eventualno nove klase.



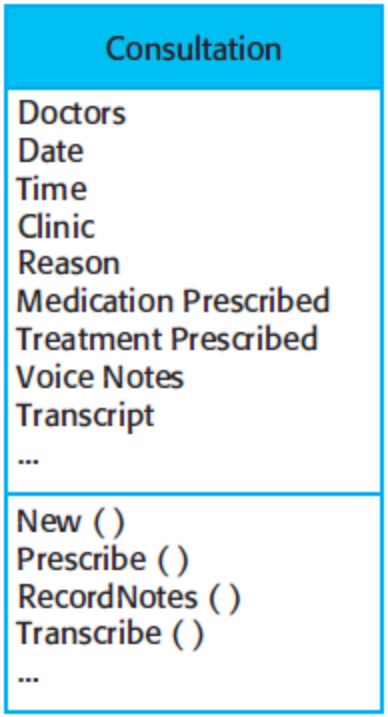
Slika 7 : Dijagram klasa i asocijacija

- Asocijacija, kao veza između dva objekta, naglšava postojanje veze između objekata.
- Na ovom nivou dijagrami klasa podsećaju na semantičke modele kojie reprezentuju bazu podataka.



Slika 8: Dijagram klasa u Mentalnoj ustanovi

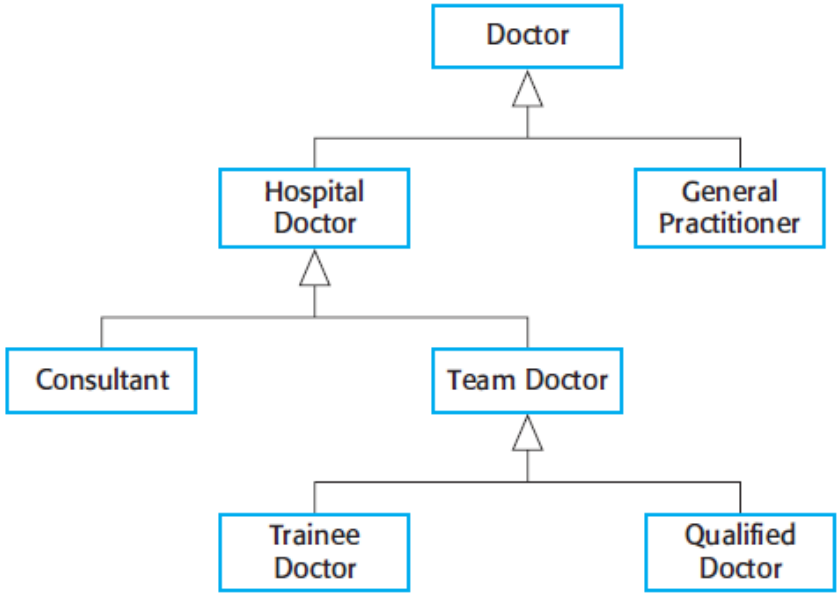
- Za prikazivanje veza između klasa najpogodnije je predstavljati klase sto jednostavnije
- Za detaljnije prikazivanje klasa dodaju im se atributi i njihove operacije.



Slika 9: Klasa koja opisuje konsultanta

Generalizacija

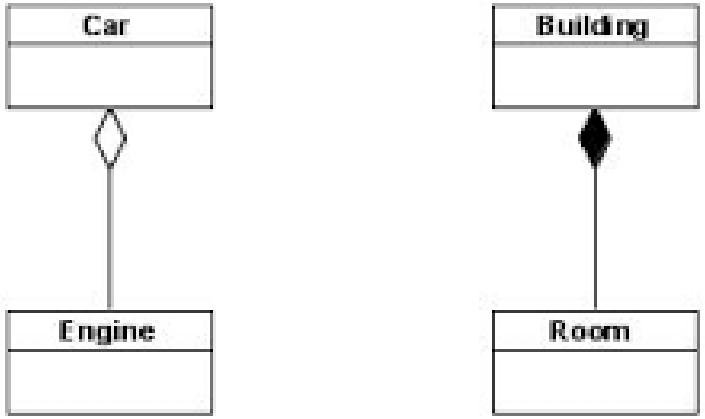
- Smanjuje kompleksnost klasa
- Zajedničke osobine jedne klase stavlja u natklasu
- Olakšava proces izmena koda



Slika 10: Generalizacija

Agregacija

- Služi za predstavljanje odnosa između dva objekta tako da jedan objekat sadrži drugi jedan ili više puta.
- Označava se popunjenim ili praznim rombom.



Slika 11: Primer agregacije

Modeliranje ponašanja

- Prikazuju dinamičko ponašanje sistema kada se izvršava
- On oslikava kako sistem reaguje na određene događaje iz okoline
- Sistem može da reguje na:
  - Podatke (Neki podatak je dostavljen sistemu)
  - Događaje(Mogu da šalju neke podatke)

Modeli ponašanja zasnovani na podacima

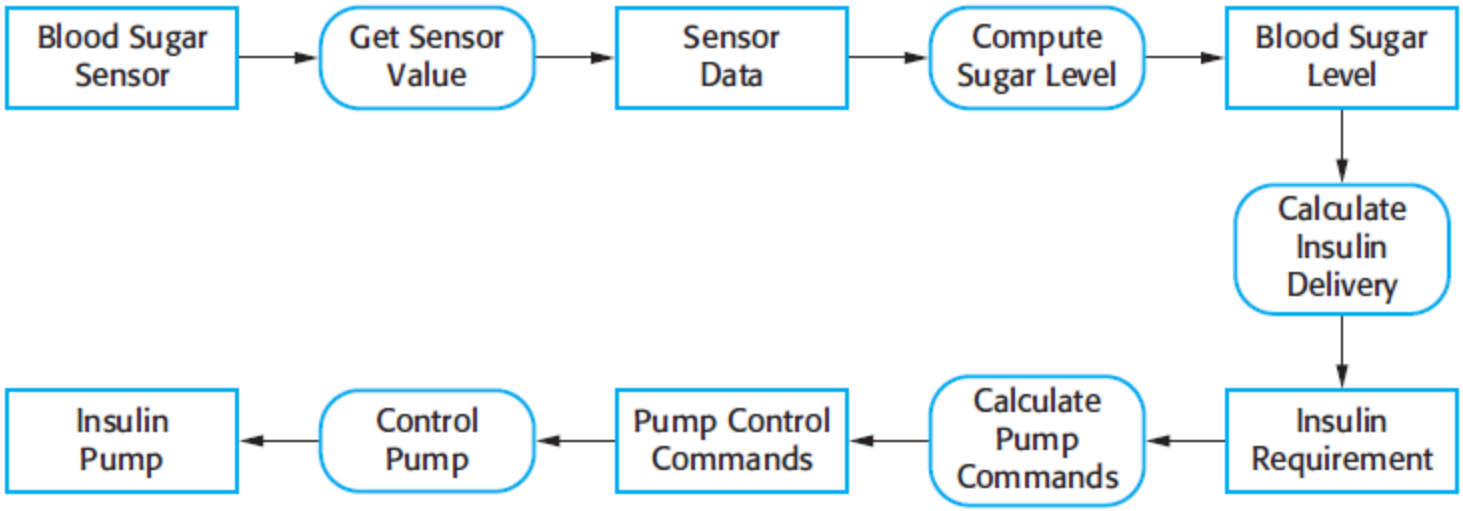
- Prikazuju niz aktivnosti koji obrađuju podatke od ulaza do njegovog izlaza
- Korisni su za analizu celog sistema i njegovo razumevanje
- Među prvim grafičkim modelima su bili dijagrami toka podataka

Dijagrami toka podataka

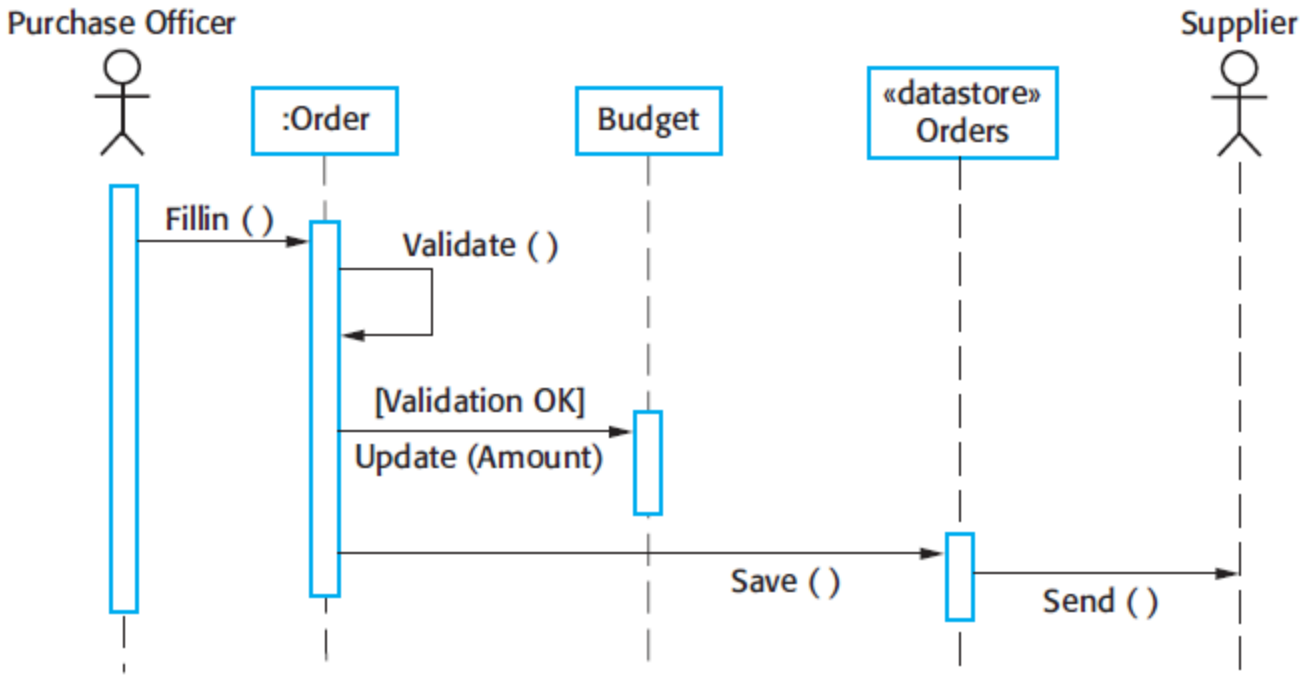
- Prikazuju funkcionala pogled sistema
- Sastoji se od niza transformacija ili funkcija kroz koje podaci prolaze
- Mogu biti različitog nivoa
- Naglasak je na funkcijama sistema a ne na objektima koji učestvuju u procesu

Dijagram Aktivnosti

- Kako se dijagrami toka podataka fokusiraju na funkcije sistema i ne prepoznaju objekte, napravljeni su dijagrami aktivnosti koji su dosta slični njima.
- Takođe dijagrami sekvenci mogu predstavljati sekvencijalno izvršavanje procesa ako se kreću samo sa leva na desno.



Slika 12: Dijagram procesa za naručivanje insulinske pumpe



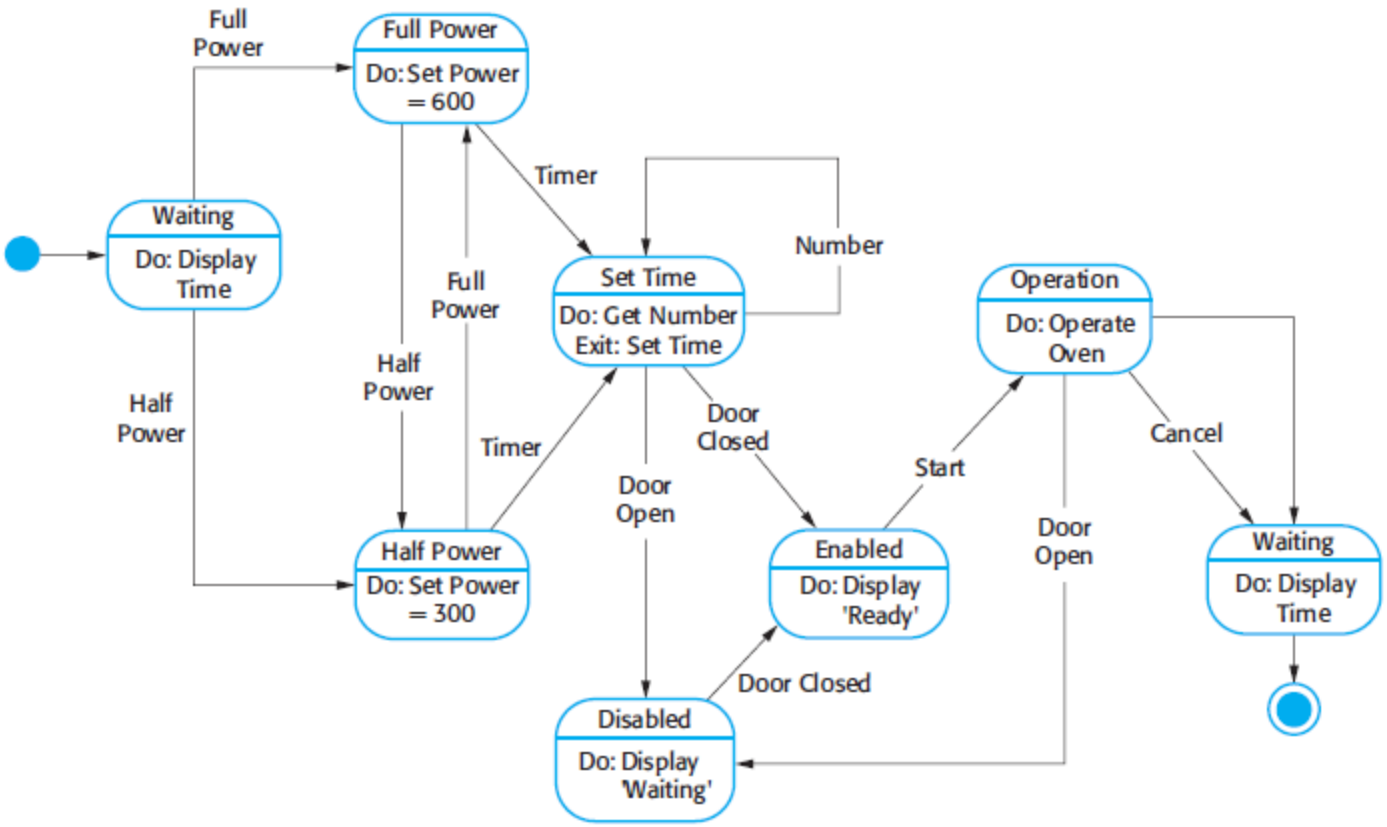
Slika 13: Dijagram sekvenci kao dijagram procesa za ugrađivanje insulinske pumpe

Modeliranje ponašanja zasnovano na događajima

- Prikazuju kako sistem reaguje na spoljašnje i unutrašnje događaje
- Bazirana je na pretpostavci da postoji konačan broj stanja i događaja
- UML podržava ovakvo modeliranja dijagramima stanja
- Detaljna specifikacija sistema zahteva dodatne informacije o sistemu (slika 15)

Dijagrami stanja

- Prikazuju sva stanja u kojima sistem može da se nalazi
- Događaji aktiviraju prelazak iz jednog stanja u drugo
- Dozvoljavanju i dodatnu informaciju o akciji koja se izvršava u tom stanju

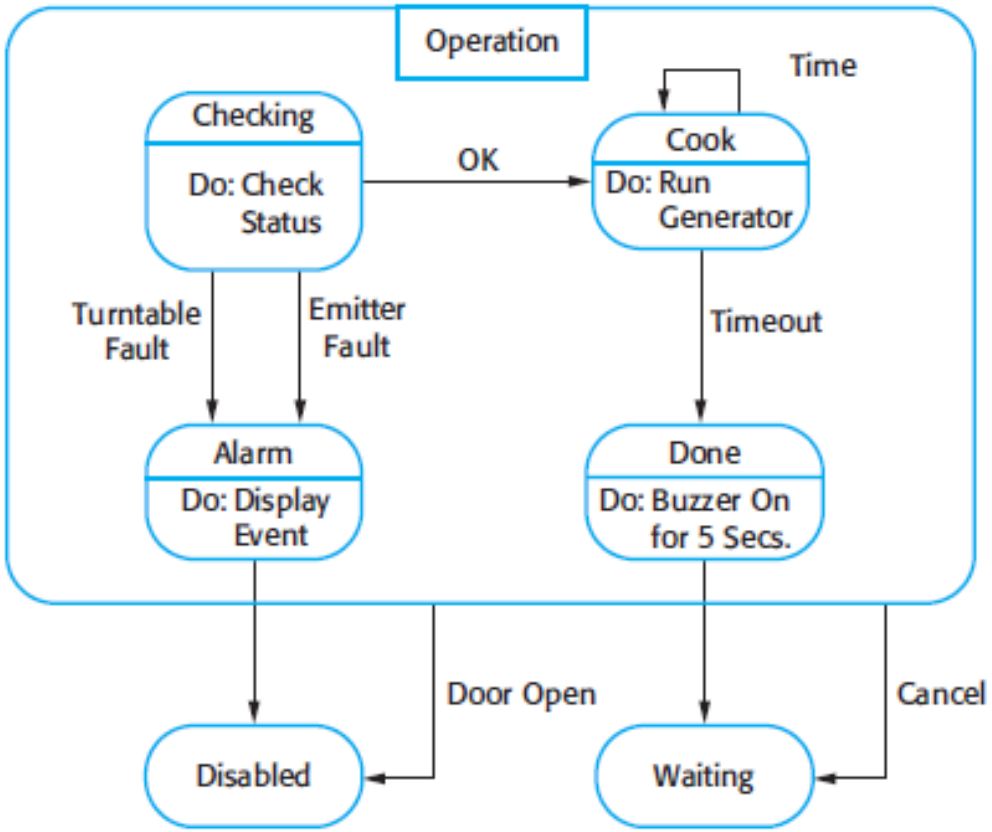


Slika 14: Dijagram stanja mikrotalasne peći

State	Description
Waiting	The oven is waiting for input. The display shows the current time.
Half power	The oven power is set to 300 watts. The display shows 'Half power'.
Full power	The oven power is set to 600 watts. The display shows 'Full power'.
Set time	The cooking time is set to the user's input value. The display shows the cooking time selected and is updated as the time is set.
Disabled	Oven operation is disabled for safety. Interior oven light is on. Display shows 'Not ready'.
Enabled	Oven operation is enabled. Interior oven light is off. Display shows 'Ready to cook'.
Operation	Oven in operation. Interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for five seconds. Oven light is on. Display shows 'Cooking complete' while buzzer is sounding.
Stimulus	**Description**
Half power	The user has pressed the half-power button.
Full power	The user has pressed the full-power button.
Timer	The user has pressed one of the timer buttons.
Number	The user has pressed a numeric key.
Door open	The oven door switch is not closed.
Door closed	The oven door switch is closed.
Start	The user has pressed the Start button.
Cancel	The user has pressed the Cancel button.

Slika 15: Stanja i dodatne informacije o stanjima mikrotalasne peći

- Postoji mogućnost enkapsulacije više stanja u jedno superstanje radi lakšeg pregleda sistema



Slika 16: Dijagram stanja rada mikrotalasne peći

Model-driven engineering

- Pristup razvoju softvera gde su modeli primarni cilj razvojnog planiranja projekta
- Iz modela se automatski generiše kod za određenu platformu
- Podiže nivo apstrakcije programiranja na viši nivo
- Zasniva se na model-driven architecture (MDA) koja je predstavlja noviju programsku paradigmu razvijenu od strane Object Management Group (OMG) 2001.

- Cilj MDE je povećavanje produktivnosti razvoja i smanji vreme isporuke, time što omogućava razvoj sistema na višem nivou apstrakcije, korišćenjem koncepata bliže domenu problema, umesto onih kojie nude programski jezici.
- MDA podrazumeva dizajn i implementaciju softvera,dok MDE uključuje ceo proces analize i izgradnje sistema kao i testiranje objekata.

Dobre i loše strane MDE

- Dobre:
  - Dozvoljava programeru da misli na višem, apstraktnijem nivou što sprečava gomilu grešaka
  - Povećava brzinu izrade softvera
  - Kreira se kod kod koji je nezavisan od platforme
- Loše:
  - Neke autogenerisane implemenatcije ne moraju predstavljati najefikasniju implementaciju
  - Ova vrsta modelovanja dobra je za modelovanje velikih sistema kod kojih glavni problem nije implementacija nego dobra analiza sistema, bezbednost i nezavisnost, uklapanje sa starim sistemo kao i testiranje

Model-driven architecture

- Glvani cilj je dizajn i implementacija softvera
- Koriste podskup UML dijagrama za opisivanje softvera iz kojih se automatski generiše kod
- Modeli se klasifikuju u tri grupe:
  - Computation Independent Model (CIM)
  - Platform Independent Model(PIM)
  - Platform Specific Model(PSM)

Computation Independent Models

- Ovoj klasi pripadaju modeli koji opisuju sistem sa računarski nezavisnog stanovišta
- Ovo su modeli visokog nivoa apstrakcije i često se za njihovo kreiranje koriste koncepti koji su karakteristični za domen primene
- U kreiranju ovih modela poželjno je uključiti i domenske eksperte
- Ovi modeli nisu pogodni za proces transformacije jer su često u pitanju neformalni modeli, tada je apstrakcioni jaz prevelik

PIM i PSM

- PIM modeluje funkcionisanje sistema nezavisno od platforme.
- On opisuje strukturu ili modele sistema i kako se njima upravlja
- Obično je opisan UML dijagramima koji pokazuju statičku sliku sistema i kako sistem reaguje na spoljašnje i unutrašnje događaje
- PIM se prevodi u PSM (Primer generisanja Java Byte koda)

Izvršivi UML(Executable)

- Kako MDE zasniva na automatskom generisanju koda iz modela tako modeli moraju biti striktno definisani
- UML 2 dozvoljava dodavanje informacija o načinu izvršavanja nekih operacija modela kako one trebaju biti implementirane
- Dizajneri UML2 su više pažnje posvetili izražajnosti jezika nego njegovoj semantici

- Postoje tri ključna modela UML dijagrama koji podržavaju automatsko generisanje koda:
  - Modeli domena koji označavaju bitne karakteristike sistema. Definišu se pomoću dijagrama klasa koji uključuju druge objekte, atribute, metode i asocijaciju
  - Modeli klasa, kojima se definišu klase, sa njihovim atributima i metodama
  - Modeli stanja, u kojima dijagramima satanja predstavljaju životni ciklus svake klase
- Dinamičko ponašanje sistema se opisuje pomoću Object Constraint Language (OCL) ili može biti prikazano dijagramima akcije.