Projektovanje softvera

Jezik UML

Izvori

- Materijal prireman na osnovu:
  - Booch,G., Rumbaugh,J., Jacobson,I., *The Unified Modeling Language User Guide*, 2nd Edition, Addison Wesley, May 2005
  - Rumbaugh,J., Jacobson,I., Booch,G., *Unified Modeling Language Reference Manual*, 2nd Edition, Addison-Wesley, 2004
  - Fowler,M., *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3rd Edition, Pearson Education, 2004
  - *OMG® Unified Modeling Language*, OMG, https://www.omg.org/spec/UML/2.5.1/About-UML
  - *The Unified Modeling Language*, https://www.uml-diagrams.org/

Standardni jezik za modelovanje UML

- UML (*Unified Modeling Language*) je grafički jezik za:
  - vizuelizaciju
  - specifikaciju
  - konstruisanje i
  - dokumentovanje

  softverski-intenzivnih sistema
- UML omogućava konstruisanje šema koje modeluju sistem opisujući:
  - konceptualne stvari
    - npr. proces poslovanja, funkcionalnosti sistema
  - konkretne stvari
    - npr. klasne tipove, šeme baza podataka, softverske komponente
- http://www.uml.org/

Korisnici UML-a

- Sledeće kategorije korisnika UML-a se uočavaju:
  - sistem-analitičari i krajnji korisnici:
    specificiraju zahtevanu strukturu i ponašanje sistema
  - arhitekti - projektanti:
    projektuju sistem koji zadovoljava zahteve
  - razvojni inženjeri - programeri (*developers*):
    transformišu arhitekturu u izvršni kod
  - kontrolori kvaliteta (*quality assurance personel*):
    proveravaju strukturu i ponašanje sistema
  - bibliotekari (*librarians*):
    kreiraju i katalogiziraju komponente
  - rukovodioci projekata (*managers*):
    vode i usmeravaju kadrove i upravljaju resursima

Praistorija UML-a

- Jezici za OO modelovanje se pojavljuju još od sredine 70ih
  - uzrok njihove pojave je pojava nove generacije OO jezika i povećana složenost softverskih sistema
- U periodu 1989-1994:
  - broj OO metoda je porastao sa manje od 10 na više od 50
- Metodi koji su ostvarili najveći uticaj na oblast OO modelovanja su:
  - *Booch* metod
  - OMT (*Object Modeling Technique*, *Rumbaugh*)
  - OOSE (*Object Oriented Software Engineering*, *Jacobson*)
  - *Fusion* metod
  - *Shlaer-Mellor* metod
  - *Coad-Yourdon* metod

Istorija UML-a

- 1994: početak rada na UML-u - Rumbaugh se pridružio Booch-u u firmi Rational
- Oktobar 1995: pojavila se verzija 0.8 drafta UM-a (*Unified Method*)
- Jesen 1995: Jacobson se pridružio Rational-u – rad na objedinjenju UM sa OOSE
- Jun 1996: pojavila se verzija 0.9 UML-a
- Važniji partneri (učestvovali u definisanju verzije 1.0):
  - DEC, HP, IBM, Microsoft, Oracle, Rational, TI, ...
- U januaru 1997: OMG-u (*Object Management Group*) podnet predlog std. UML 1.0
- Grupa partnera je proširena drugim podnosiocima predloga:
  - npr. ObjecTime, Ericsson,...
- Jul 1997: podnet predlog **UML 1.1 koji je prihvaćen od OMG 14.11.1997.**
- Jun 1998: verzija UML 1.2, 2000: verzija UML 1.3
- 2001: v1.4 (v. 1.4.2 => **ISO std.** 19501:2005), 2003: v1.5 (akciona semantika)
- 2005: v 2.0, 2007: v 2.1, 2009: v 2.2, 2010: v 2.3
- 2011.: v2.4 (v. 2.4.1 => **ISO std.** 19505:2012), 2013.: v2.5
- **2017.: 2.5.1**
- http://www.omg.org/spec/UML/

Konceptualni model UML-a

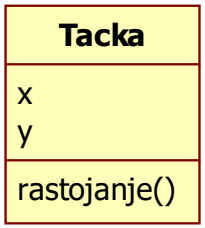
- Elementi UML-a su:
  - Gradivni blokovi
  - Pravila za povezivanje gradivnih blokova
  - Opšti mehanizmi koji se primenjuju u UML-u
- Gradivni blokovi UML-a
  - Stvari (*things*)
    - apstrakcije koje su "građani prvog reda" u modelu
  - Relacije (*relationships*)
    - povezuju stvari
  - Dijagrami (*diagrams*)
    - grupišu interesantne skupove povezanih stvari

Stvari

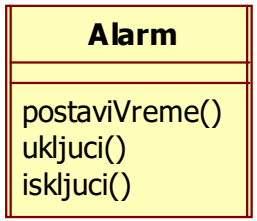
- Stvari strukture - statički delovi modela, predstavljaju logičke ili fizičke elemente (imenice)
- Stvari ponašanja - dinamički delovi modela, predstavljaju ponašanje kroz prostor i vreme (glagoli)
- Stvari grupisanja - organizacioni delovi modela, kutije u koje model može da se dekomponuje
- Stvari anotacije - objašnjavajući delovi modela, komentari koji se primenjuju na bilo koji element

Stvari strukture – klasa

- Klasa je opis skupa objekata koji dele zajedničke odlike (atribute i operacije), ograničenja i semantiku

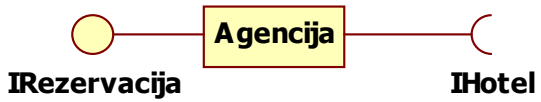


- Aktivna klasa je klasa čiji objekti imaju vlastitu nit kontrole i tako mogu da započnu neku upravljačku aktivnost
  - ponašanje objekta aktivne klase je konkurentno sa drugim aktivnim objektima



Stvari strukture – interfejs

- Interfejs je skup operacija
  koje specificiraju uslugu klase ili komponente
  - opisuje ponašanje elementa koje je spolja vidljivo (ugovor)
  - interfejs predstavlja skup operacija (deklaracija metoda)
    ali ne i njihove implementacije (metode)
  - klasa i komponenta mogu da implementiraju više interfejsa
  - razlikuju se implementirani i zahtevani interfejs

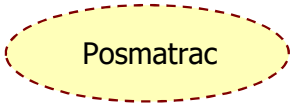


Stvari strukture – slučaj korišćenja

- Slučaj korišćenja (*use-case*) je opis skupa sekvenci akcija koje obavlja sistem da bi proizveo vidljiv rezultat vredan za pojedinog aktera
- Sekvenca akcija
  - primerak slučaja korišćenja (scenario)
  - može da se opiše, ne primer, nekim dijagramom ponašanja
- Slučaj korišćenja
  - predstavlja funkcionalnost sistema
  - koirsti se da bi se strukturirale stvari ponašanja u modelu
  - realizuje se kroz saradnju (kolaboraciju)

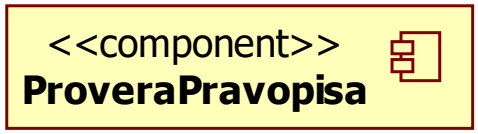
Stvari strukture – saradnja

- Saradnja (*collaboration*) opisuje strukturu skupa uloga koje imaju specifične funkcije da bi zajedno ostvarile ciljnu funkcionalnost
  - ima strukturalnu, kao i dimenziju ponašanja
    - ponašanje se opisuje posebnim dijagramima
  - klasa ili objekat može da učestvuje u više saradnji
- Projektni uzorci (*design patterns*)
  - predstavljaju se kao saradnje u kojima učesnici igraju odgovarajuće uloge
  - učesnici mogu da budu klase ili objekti klasa



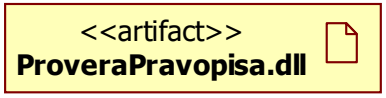
Stvari strukture – komponenta

- Komponenta je modularni deo sistema koji kapsulira neki sadržaj
  - ostvaruje realizaciju skupa interfejsa i sakriva implementaciju
  - ima stanje i ponašanje
  - ima ponuđene (realizovane) i zahtevane interfejse
  - implementira operacije ponuđenog interfejsa
  - nema sopstvene korisnički definisane atribute
  - može da se zameni drugom koja realizuje iste interfejse
  - ista komponenta može da se koristiti u raznim sistemima
  - u UML-u 1 – fizička stvar, ali u UML-u 2 – logička



Stvari strukture – artefakt i čvor

- Artefakt je fizički i zamenljivi deo sistema koji sadrži informacije
  - predstavlja fizičku manifestaciju elementa modela (npr. izvršni artefakt predstavlja manifestaciju komponente)
  - može biti fajl sa izvornim ili izvršnim kodom, dokument i sl.
- Čvor (*node*) je fizička stvar koja postoji u vreme izvršenja i predstavlja resurs obrade
  - svakako poseduje neku memoriju
  - često poseduje i mogućnost procesiranja
- Skup artefakata može biti u čvoru, a može i migrirati sa čvora na čvor
- Artefakt i čvor predstavljaju fizičke stvari



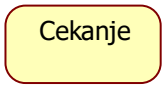
Stvari ponašanja – interakcija

- Interakcija je ponašanje
  koje specificira sekvence poruka
  koje se razmenjuju između skupa uloga
  unutar posebnog konteksta
  da se ostvari specifična svrha

- Interakcija uključuje određen broj elemenata:
  - poruke (priložen grafički simbol predstavlja sinhronu poruku)
  - aktivnosti na strani prijema (ponašanje izazvano porukom)
  - konektore (komunikacione putanje između uloga)
- Poruka je stimulus koji izaziva pokretanje aktivnosti na strani prijema

Stvari ponašanja – automat stanja

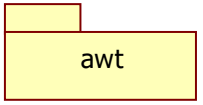
- Automat stanja je ponašanje
  koje specificira sekvence stanja
  kroz koje prolazi jedan objekat ili jedna interakcija
  za vreme svog životnog veka,
  sa prelazima kao posledicama događaja,
  zajedno sa odgovorima na te događaje



- Automat stanja uključuje određen broj elemenata:
  - stanja (priložen grafički simbol)
  - tranzicije (prelaze između stanja)
  - događaje (stimuluse koji izazivaju prelaze između stanja)
  - akcije (odgovore na prelaze ili ulaske/izlaske iz stanja)

Stvari organizacije i anotacije

- Paket je opštenamenski mehanizam za organizovanje elemenata u grupe

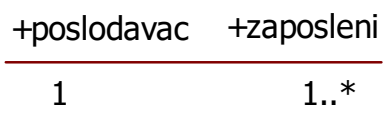


- Stvari strukture, ponašanja i druge stvari grupisanja mogu biti smeštene u paket
- Za razliku od komponente, paket postoji samo u vreme razvoja
- Pored paketa postoje i sledeće stvari grupisanja:
  - radni okviri (*frameworks*), modeli
- Napomena (*note*) je simbol za prikazivanje komentara pridruženih jednom elementu ili kolekciji elemenata

Relacije

- Zavisnost je semantička relacija između dve stvari u kojoj izmena jedne (nezavisne) stvari može uticati na semantiku druge (zavisne) stvari

- Asocijacija je strukturna relacija koja opisuje skup veza između objekata
  - sadržanje je specijalna vrsta asocijacije koja predstavlja strukturnu relaciju između celine i njenih delova
  - često grafički simbol sadrži ukrase kao što su multiplikativnost i imena uloga



Relacije (2)

- Generalizacija je relacija
  u kojoj su objekti specijalizovanog elementa (deca)
  zamene za objekte generalizovanog elementa (roditelja)
  - dete deli strukturu i ponašanje roditelja
- Realizacija je semantička relacija između stvari
  u kojoj jedna stvar propisuje ugovor
  a druga stvar ostvaruje taj ugovor
- Realizacija se sreće:
  - između interfejsa i klase koja ga realizuje
  - Između interfejsa i komponente koja ga realizuje
  - između slučajeva korišćenja i saradnji koje ih realizuju

Dijagrami

- Dijagram je grafička predstava skupa povezanih elemenata
  - najčešće se pojavljuje u obliku grafa temena (stvari) povezanih granama (relacijama)
- Dijagrami se crtaju da bi se sistem vizualizovao iz različitih perspektiva
- Svaki dijagram predstavlja jedan pogled na model
- Vrste dijagrama u UML-u:
  - dijagrami za prikaz strukturnih aspekata sistema
    - strukturni aspekti odgovaraju statičkim aspektima
  - dijagrami za prikaz aspekata ponašanja sistema
    - aspekti ponašanja (uglavnom) odgovaraju dinamičkim aspektima
    - izuzetak su d. slučajeva korišćenja koji strukturiraju ponašanje

Dijagrami strukture

- Dijagram klasa (*class diagram*) prikazuje logičku strukturu apstrakcija: skup klasa, interfejsa, saradnji i njihovih relacija
- Dijagram paketa (*package diagram*) [UML 2] prikazuje statičku strukturu grupisanja elemenata modela u pakete
- Dijagram objekata (*object diagram*) prikazuje logičku strukturu primeraka: skup objekata (primeraka klasa) i njihovih veza u nekom trenutku
- Dijagram složene strukture (*composite structure diagram*) [UML 2] prikazuje hijerarhijsko razlaganje primerka klase, komponente ili saradnje na delove
- Dijagram komponenata (*component diagram*) prikazuje komponente, njihovu unutrašnju strukturu i zavisnosti između komponenata
- Dijagram raspoređivanja (*deployment diagram*) prikazuje topologiju čvorova obrade i artefakata koji se raspoređuju na njih

Dijagrami ponašanja

- Dijagram slučajeva korišćenja (*use case diagram*) prikazuje skup slučajeva korišćenja, aktera (specijalne vrste klasa) i njihovih relacija
- Dijagram interakcije (*interaction diagram*) prikazuje interakciju koju čine skup uloga i njihovih veza preko kojih razmenjuju poruke
  - Dijagram sekvence (*sequence diagram*) je dijagram interakcije koji naglašava vremenski redosled poruka
  - Dijagram komunikacije (*communication diagram*) je dijagram interakcije koji naglašava strukturnu organizaciju povezanih uloga koje razmenjuju poruke
  - Dijagram pregleda interakcije (*interaction overview diagram*) [UML 2] je dijagram interakcije koji definiše interakcije kroz vrstu dijagrama aktivnosti
  - Vremenski dijagram (*timing diagram*) [UML 2] je dijagram interakcije koji prikazuje promenu stanja uloge u vremenu i poruke koje se razmenjuju
- Dijagram stanja (*statechart diagram*) prikazuje konačni automat koji obuhvata stanja, prelaze, događaje i pokrenute akcije
- Dijagram aktivnosti (*activity diagram*) prikazuje tok između aktivnosti (nije specijalna vrsta dijagrama stanja u UML 2)

Pravila UML-a

- Specifikacija jezika UML propisuje apstraktnu sintaksu (UML dijagramima) koja određuje semantiku elemenata modela i pravila njihovog povezivanja
- UML pravila definišu kako izgleda dobro formiran model – on je:
  - semantički konzistentan
  - u harmoniji sa korelisanim modelima
- UML ima (semantička) pravila za:
  - imena - kako se nazivaju stvari, relacije i dijagrami
  - doseg - koji kontekst daje specifično značenje imenu
  - vidljivost - gde se imena mogu koristiti od strane drugih (rešava pravo pristupa)
  - integritet - kako se stvari propisno i konzistentno povezuju sa drugim stvarima
  - izvršenje - šta nešto znači za izvršenje ili simulaciju dinamičkog modela
- Tokom razvoja se ne prave samo modeli koji su dobro formirani, već mogu biti i:
  - nekompletni - izvesni elementi nedostaju
  - nekonzistentni - integritet modela nije garantovan
- Pravila UML-a vode kroz vreme ovakve modele prema dobro formiranim
- Skraćeni (*elided*) model - izvesni elementi su sakriveni da se pojednostavi izgled
  - ti elementi postoje u specifikaciji modela, ali su potisnuti iz grafičkog prikaza

Opšti mehanizmi UML-a

- Gradnja je jednostavnija i harmoničnija ako se poštuju opšti obrasci
- Postoje četiri opšta mehanizma koja se primenjuju konzistentno kroz jezik:
  - specifikacije
  - ukrasi
  - opšte podele
  - mehanizmi proširivosti

Specifikacije (1)

- Iza elementa grafičke notacije UML-a leži specifikacija
  - tekstualni iskaz sa sintaksom i semantikom gradivnog bloka
- Iza grafičkog simbola (sličice, ikone) klase stoji specifikacija koja navodi:
  - potpun skup atributa
  - potpun skup operacija (uključujući kompletne deklaracije)
  - odgovornosti klase
- Grafički simbol može prikazivati samo jedan deo potpune specifikacije
  - alati podržavaju suspendovanje pojedinih detalja

Specifikacije (2)

- Može da postoji i drugi izgled iste klase
  - prikazuje drugi skup delova iste klase
  - konzistentan je sa specifikacijom klase
- UML grafička notacija se koristi za vizuelizaciju
- UML specifikacija se koristi da se saopšte detalji
- Modeli se mogu graditi:
  - najpre pomoću crtanja dijagrama,
    a zatim dodavanjem detalja u specifikaciji
    - tipično za direktni inženjering pri kreiranju novog sistema
  - direktnim kreiranjem specifikacije,
    pa naknadnim kreiranjem dijagrama koji su njene projekcije
    - tipično za reverzni inženjering postojećeg sistema

Ukrasi

- Detalji specifikacije se prikazuju kao stilski, simbolički, grafički ili tekstualni ukras osnovnog grafičkog elementa
- Na primer:
  - za klasu se može naglasiti da je apstraktna tako što se ime piše *italic* slovima
  - vidljivost (pravo pristupa) atributa i operacija se može naglasiti pomoću znakovnog simbola:
    + (javni), # (zaštićeni), – (privatni) i ~(paketni)
  - agregacija se predstavlja dodatnim grafičkim simbolom na simbolu asocijacije 

Opšte podele

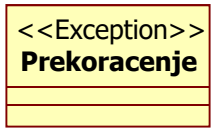
- Dve osnovne podele:
  - apstrakcije i primerci (instance)
  - interfejsi i implementacije (interfejs=ugovor, implementacija=realizacija ugovora)
- Primeri prve podele:
  - klasa/objekat (klasa je apstrakcija, a objekat primerak te apstrakcije)
  - slučaj korišćenja/scenario
  - čvor/primerak čvora
- Primeri druge podele:
  - interfejs/komponenta
  - slučaj korišćenja/saradnja
  - operacija/metod
- U UML-u se razlika između apstrakcije i primerka pravi tako što se imena primeraka podvlače
- Relacijom realizacije se uspostavlja veza između ugovora i implementacije

Mehanizmi proširivosti

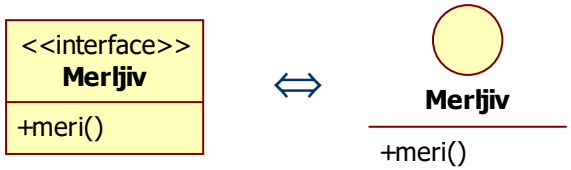
- UML je otvoren za proširenja jezika na kontrolisani način
- Mehanizmi proširivosti uključuju:
  - stereotipe
  - obeležene vrednosti
  - ograničenja

Stereotipi

- Stereotip proširuje rečnik UML-a dopuštajući kreiranje novih specifičnih vrsta gradivnih blokova
- Novi gradivni blokovi su izvedeni iz postojećih
- Stereotip se prikazuje kao ime uokvireno znacima << i >> smešteno iznad imena odgovarajućeg elementa
- Na primer, izuzeci su klase čiji se objekti mogu bacati i hvatati

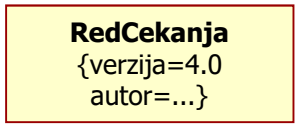


- Može se definisati i grafički simbol za određeni stereotip
  - napomena: <<interface>> više nije stereotip, već ključna reč



Obeležene vrednosti

- Obeležene vrednosti proširuju osobine UML gradivnog bloka dopuštajući dodavanje nove informacije
- Obeležene vrednosti se prikazuju kao niska okružena zagradama { i } ispod imena odgovarajućeg elementa
- Niska sadrži ime (*tag*), separator (simbol =) i vrednost
- Na primer, verzija i autor klase nisu koncepti UML-a, a mogu da se dodaju nekom gradivnom bloku, kao što je klasa



Ograničenja

- Ograničenja proširuju semantiku UML gradivnog bloka dopuštajući da se dodaju nova pravila ili promene postojeća
- Ograničenja se mogu pisati:
  - kao slobodan tekst u vitičastim zagradama
  - na jeziku OCL (*Object Constraint Language*)