

Dragan Milićev

Objektno orijentisano
modelovanje na jeziku
UML

Skripta

Skripta

Glava 1

Uvod

Glava 2

Uvod u objektno modelovanje

Glava 3

Proces

Glava 4

Projektni zahtevi

Glava 5

Analiza

Glava 6

Projektovanje

Glava 7

Implementacija



Glava 1

Uvod

- Razvijanje softvera samo primenom tehnika objektno orijentisanog programiranja (OOP) na nekom klasičnom, tekstualnom OO programskom jeziku (npr. C++ ili Java) pokazalo se kao nedovoljno efikasno iz sledećih razloga:
 - Softverski sistemi su, po pravilu, jako složeni, a OOP ne pruža dovoljno efikasna sredstva za specifikaciju i dokumentovanje složenih softverskih sistema. Potrebna su efikasnija sredstva za hijerarhijsku dekompoziciju i organizaciju složenih softverskih sistema.
 - Koncepti koje nudi OOP su dobri, ali i dalje su nedovoljno apstraktni, tj. nedovoljno bliski visoko apstraktnom načinu razmišljanja neophodnom za rešavanje složenih problema. Na primer, pokazalo se da između klasa, odnosno objekata, postoje i neke druge relacije osim onih koje direktno podržavaju OO programski jezici. Dalje, scenariji po kojima objekti međusobno saraduju da bi obezbedili neko ponašanje sistema teško se uočavaju u tekstualnom programskom kodu jer su njihovi delovi rasuti po kodu mnogih klasa i njihovih operacija. Osim toga, izmišljeni su i mnogi koncepti pogodni za efikasan razvoj softvera koji su potpuno nepoznati nekim ili svim OO programskim jezicima.
 - Tekstualni način specifikacije manje je efikasan od vizuelnog, grafičkog načina (notacije).
 - OOP ne pruža dobru podršku za sve faze razvoja softvera (specifikacija zahteva, analiza, projektovanje, implementacija, testiranje) i ne omogućava lagan prelaz između njih.
 - OOP ne pruža dovoljno efikasna sredstva za dobru dokumentaciju softvera bez previše dodatnih napora.
- Zbog ovoga su formirane metode za *objektno orijentisano modelovanje* (engl. *object-oriented modeling*, OOM). Te metode podrazumevaju:
 - razvoj *modela* softvera na višem nivou apstrakcije, korišćenjem apstraktnijih koncepata od onih koje nudi OOP (ali i nekih koje direktno podržava OOP);
 - specifikaciju modela pomoću vizuelnih, grafičkih notacija;
 - transformaciju takvih visoko apstraktnih, vizuelnih modela u implementacione forme; implementaciona forma je tipično programski kôd na nekom klasičnom, tekstualnom OO programskom jeziku (npr. C++ ili Java) koji se dobija iz OO modela softvera.
- Posle mnogih predloženih metoda modelovanja, usvojen je jedinstven, standardni jezik za OOM koji se naziva UML (*The Unified Modeling Language*).

- Prema tome, OOM je viši tehnološki nivo razvoja softvera od OOP-a, koji se oslanja na (nadograđuje) OOP. To znači da svi principi OOP-a i dalje važe, ali se koriste na nižem nivou apstrakcije, prilikom same implementacije OO modela. Principi OOM-a se koriste u ranijim fazama razvoja softvera u cilju efikasnije specifikacije i analize zahteva i projektovanja rešenja. Principi OOP-a se koriste u fazi implementacije, tj. preslikavanja modela u izvršni oblik.
- UML je standardni jezik koji podržava OOM.
- OOM i UML su glavna tema ovog teksta.

Modelovanje

- Modelovanje je centralna aktivnost u izgradnji dobrog softvera.
- Model je pojednostavljen prikaz realnosti. Model se pravi da bi se bolje razumeo sistem koji se gradi. Model kompleksnog sistema pravi se zato što se takav sistem ne može razumeti kao kompaktna celina.
- Modelovanjem se postižu sledeći ciljevi:
 - model služi da se sistem vizuelno prikaže kakav jeste ili kakav želimo da bude (*vizualizacija*);
 - modelom se definiše struktura i ponašanje sistema (*specifikacija*);
 - model predstavlja uzor koji pokazuje kako sistem treba konstruisati (*konstrukcija*);
 - model predstavlja dokumentaciju projektnih odluka (*dokumentacija*).
- Principi modelovanja:
 - Izbor modela koji se prave ima ključan uticaj na to kako se problem rešava i kako se oblikuje rešenje.
 - Svaki model može imati različite nivoe detalja.
 - Najbolji modeli su povezani sa realnim svetom.
 - Nijedan model nije dovoljan sam za sebe; svaki netrivialni sistem najbolje se opisuje malim skupom skoro nezavisnih modela.
- Objektivno orijentisano modelovanje predstavlja alternativu tradicionalnom, algoritamskom modelovanju.

UML

- UML (*The Unified Modeling Language*) je standardni vizuelni jezik za modelovanje dominantno, ali ne isključivo, složenih softverskih sistema.
- Kratak istorijat jezika UML:
 - Od sredine sedamdesetih do kasnih osamdesetih pojavljuju se jezici za objektivno orijentisano modelovanje. U to vreme istraživači počinju da eksperimentišu sa alternativnim načinima analize i projektovanja složenih sistema implementiranih na OO programskim jezicima.

- Od 1989. do 1994. broj OO metoda raste od desetak do više od 50. Nijedna od njih ne zadovoljava uvek sve potrebe korisnika. Pojavljuje se potreba za unifikacijom i standardizacijom.
- Ističu se tri metode: OO Analysis and Design (autor je Booch), Object Modeling Technique (OMT, autor je Rumbaugh) i OO Software Engineering (OOSE, autor je Jacobson).
- Sredinom devedesetih Booch, Rumbaugh i Jacobson razmenjuju svoje ideje i počinju da razmišljaju o jedinstvenom metodu.
- Oktobra 1994. Rumbaugh se pridružuje Boochu u korporaciji Rational. Tada zvanično počinje rad na jedinstvenoj metodologiji koju najpre nazivaju Unified Method i koja obuhvata koncepte metoda Booch i OMT.
- Oktobra 1995. izlazi specifikacija UM, verzija 0.8.
- Otprilike u isto vreme, korporaciji Rational pridružuje se i Jacobson i u jedinstveni pristup ugrađuje principe iz metode OOSE.
- Autori shvataju da ne treba da propisuju celu metodologiju, koja osim jezika (notacije i njene semantike) uključuje i proces razvoja softvera. Oni ostavljaju da svaka organizacija proces prilagodi sebi, a dalje rade samo na jedinstvenom jeziku koga nazivaju UML.
- Juna 1996. izlazi specifikacija UML 0.8.
- Tokom 1996. autori dobijaju komentare iz industrije na nacrt jezika UML. Organizuje se UML konzorcijum. U izgradnji UML 1.0 učestvuju mnoge značajne kompanije.
- Januara 1997, predlaže se organizaciji OMG (*Object Management Group*) da usvoji UML 1.0 kao standard.
- Jula 1997, organizaciji OMG podnosi se UML 1.1, kao dopuna prethodne verzije.
- 14. novembra 1997. OMG usvaja UML 1.1 kao standard.
- Od tada se nadležnost nad definicijom UML-a ostavlja grupi za revizije RTF (*Revision Task Force*) organizacije OMG. U jesen 1998. RTF izdaje UML 1.3.
- Opširniji pregled jezika UML potražite u izvornoj knjizi: Booch, G., Rumbaugh, J., Jacobson, I., *The Unified Modeling Language User Guide*, Addison-Wesley, 1999.

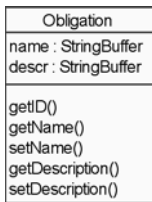
Osnovni elementi jezika UML

- UML je jezik za OO modelovanje koji omogućuje:
 - vizualizaciju: UML je vizuelni, grafički jezik;
 - specifikaciju: pomoću UML jezika formiraju se precizni, nedvosmisleni i potpuni modeli;
 - konstrukciju: pomoću jezika UML konstruiše se softverski sistem koji se posle može implementirati;
 - dokumentovanje: pomoću jezika UML mogu se dokumentovati zahtevi, arhitektura, projekat, izvorni kôd itd.

- Osnovni gradivni elementi jezika UML su: *stvari* (engl. *things*), *relacije* (engl. *relationships*) i *dijagrami* (engl. *diagrams*).
- Ove elemente ćemo samo nabrojati i formalno kratko definisati, jer cilj ovog uvoda nije da čitalac razume značenje i način upotrebe ovih elemenata. Oni će biti detaljno objašnjeni kasnije, kako se bude ukazivala potreba za njima. Ovo je samo okvirni uvodni pregled elemenata jezika UML.

Stvari

- Postoje sledeće grupe stvari:
 - strukturne stvari (engl. *structural things*);
 - stvari ponašanja (engl. *behavioral things*);
 - stvari grupisanja (engl. *grouping things*);
 - stvari označavanja (engl. *annotation things*).
- Strukturne stvari su apstrakcije koje čine strukturni, uglavnom statički deo modela. To su:
 - *Klasa* (engl. *class*) je opis skupa objekata koji imaju iste atribute, operacije, relacije i semantiku:



- *Interfejs* (engl. *interface*) je skup operacija koje definišu usluge klase ili komponente:



ISpelling

- *Kolaboracija* (engl. *collaboration*) je skup uloga i drugih elemenata koji saraduju da bi ispunili kooperativno ponašanje složenije od proste sume pojedinačnih ponašanja elemenata:



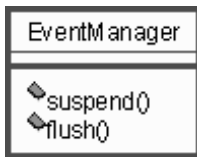
Chain of responsibility

- *Slučaj upotrebe* (engl. *use case*) je opis skupa sekvenci akcija koje sistem izvršava da bi proizveo spolja uočljivo ponašanje bitno za nekog aktera:



Place order

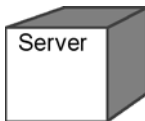
- *Aktivna klasa* (engl. *active class*) je klasa čiji objekti poseduju jedan ili više procesa ili niti toka kontrole:



- *Komponenta* (engl. *component*) je fizički i zamenljiv deo sistema koji zadovoljava i realizuje skup interfejsa:



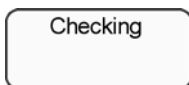
- *Čvor* (engl. *node*) je fizički element koji postoji u vreme izvršavanja i predstavlja računarski resurs koji, u principu, poseduje memoriju i najčešće mogućnost obrade:



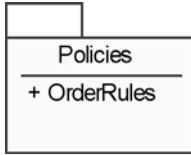
- Stvari ponašanja opisuju ponašanje sistema, tj. čine dinamički deo modela:
 - *Interakcija* (engl. *interaction*) je ponašanje koje uključuje skup poruka koje se razmenjuju između objekata u određenom kontekstu, da bi se ostvarila određena svrha. Poruka:



- *Mašina stanja* (engl. *state machine*) je ponašanje koje definiše sekvencu stanja kroz koje objekat ili interakcija prolazi tokom svog života, kao posledica događaja, zajedno sa reakcijama na te događaje. Stanje:



- Stvari za grupisanje su organizacioni delovi jezika. Postoji samo jedna stvar za grupisanje – *paket* (engl. *package*). Paket je čisto konceptualna stvar u koju se grupišu svi ostali elementi jezika. Paket:

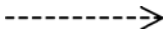


- Stvari označavanja predstavljaju komentare (engl. *note*). Komentar:



Relacije

- Postoje četiri vrste relacija u jeziku UML:
 - *Zavisnost* (engl. *dependency*) je semantička relacija između dve stvari koja označava da se promena jedne (nezavisne) stvari odlikava na drugu (zavisnu) stvar:



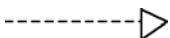
- *Asocijacija* (engl. *association*) je strukturna relacija koja opisuje skup veza, pri čemu veza povezuje instance:



- *Generalizacija* (engl. *generalization*) je relacija generalizacije/specijalizacije kod koje objekte generalizovanog elementa (roditelja) mogu zameniti objekti specijalizovanog elementa (potomka):



- *Realizacija* (engl. *realization*) je semantička veza u kojoj je jedan element ugovor koji drugi element ispunjava:

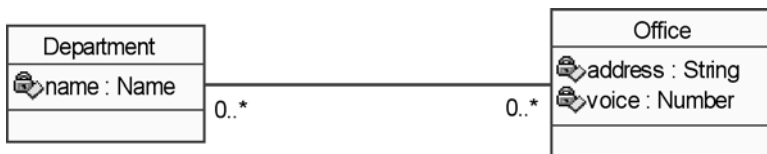


Dijagrami

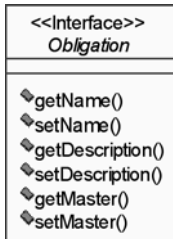
- *Dijagram* (engl. *diagram*) je grafička prezentacija skupa elemenata koji predstavlja samo jedan pogled na jedan deo modela.
- Dijagram ne nosi semantiku i ne sadrži elemente modela. Dijagram samo predstavlja prikaz nekih elemenata modela.
- Elementi modela su, inače, grupisani u pakete. Paket sadrži elemente, od kojih neki mogu biti i drugi ugrađeni paketi. Tako se model organizuje hijerarhijski.
- Paket može sadržati i dijagrame, koji samo prikazuju elemente tog, ali i drugih paketa.
- U jeziku UML postoje sledeće osnovne vrste dijagrama:
 - *dijagram klasa* (engl. *class diagram*)
 - *dijagram objekata* (engl. *object diagram*)
 - *dijagram slučajeva upotrebe* (engl. *use case diagram*)
 - *dijagram sekvence* (engl. *sequence diagram*)
 - *dijagram kolaboracije* (engl. *collaboration diagram*)
 - *dijagram stanja* (engl. *statechart diagram*)
 - *dijagram aktivnosti* (engl. *activity diagram*)
 - *dijagram komponentata* (engl. *component diagram*)
 - *dijagram raspoređivanja* (engl. *deployment diagram*).
- UML dozvoljava pravljenje i ostalih vrsta dijagrama, prema potrebi korisnika.

Opšti mehanizmi jezika UML

- *Specifikacija* (engl. *specification*) je precizan opis sadržaja svakog elementa modela. U specifikaciji se nalazi potpuni opis elementa. Vizuelni prikaz elementa ne mora sadržati ceo opis, već samo neke delove bitne za prikaz. Štaviše, na različitim dijagramima se jedan isti element može prikazati na različite načine. Specifikacija jedina sadrži kompletni sadržaj elementa.
- *Ukrasi* (engl. *adornment*) se mogu pridružiti elementima na vizuelnom prikazu da bi potpunije prikazali sadržaj elementa:



- *Mehanizmi proširivosti* (engl. *extensibility mechanisms*) jezika UML:
 - *Stereotip* (engl. *stereotype*) je proširenje rečnika jezika UML koji dozvoljava definisanje novih apstrakcija jezika. Element modela može se označiti stereotipom, koji kazuje da je taj element zapravo instanca specifične podvrste odgovarajuće apstrakcije definisane u osnovnom jeziku. Stereotip se označava rečju između simbola << i >>.



- *Označena vrednost* (engl. *tagged value*) proširuje svojstva elemenata jezika UML i dozvoljava da se u specifikaciju elementa dodaju nove informacije. Označene vrednosti se navode unutar vitičastih zagrada {}, kao lista specifikacija u formatu name=value razdvojenih zarezom.
- *Ograničenje* (engl. *constraint*) proširuje semantiku elementa jezika UML i dozvoljava da se uvode nova semantička pravila ili menjaju postojeća. Ograničenja se zadaju između vitičastih zagrada {} i pridružuju se nekom elementu modela.