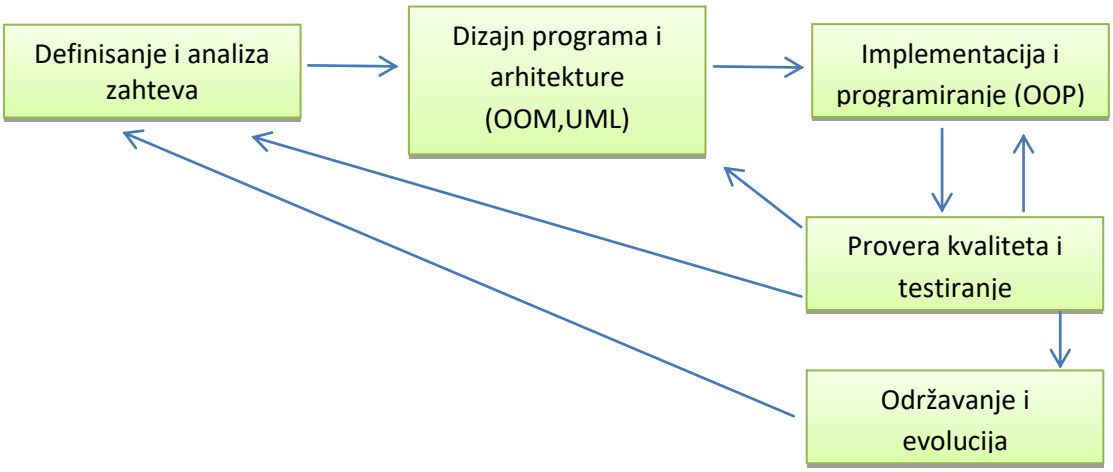
15. OBJEKTNO-ORIJENTISANI RAZVOJ

Pristup je postao popularan u kasnim 80.-tim u 90.-tim godinama 20. veka, prvo motivisan razvojem grafičkih korisničkih interfejsa, nastao kao rezultat ograničenja strukturnog pristupa u programiranju, kao i postojećih metoda razvoja softvera za vreme velike softverske krize. **Danas je ovo uobičajeni način razvoja, projektovanja, implementacije i oblikovanja softvera.** Ideja objektne orijentacije potiče iz šezdesetih godina prošlog veka kada je stvoren prvi objektno-orijentisani programski jezik, SIMULA. Tek sa pojavom programskog jezika SMALLTALK, početkom osamdesetih, šire je postao poznat objektno-orijentisani model razmišljanja u IT svetu. Posle toga su nastali mnogi objektno-orijentisani programski jezici: Objective C, C++, Eiffel, Clos, Object Pascal, Object Cobol, Ada 9, JAVA, Visual Basic i dr.

Objektno-orijentisano modelovanje (OOM) i dizajn, **koje je u suštini koncepcionalna osnova objektno-orijentisanog programiranja (OOP)**, nastalo je dosta kasnije. OOM se pojavio tek u kasnim 90.-tim godinama 20. veka. 1997. godine je nastao jedinstveni **UML (Engl. "Unified Modeling Language") jezik**, udruživanjem različitih pristupa objektno-orjentisanom modelovanju. UML predstavlja prvenstveno grafički jezik modelovanja u svim fazama razvoja softvera, počev od specifikacije zahteva, analize sistema, dizajna, dokumentovanja itd. **Autori su: Grady Booch, James Rambaugh i Ivor Jacobsen.**



Objektno-orijentisani (OO) razvoj softvera se zasniva na objektno-orijentisanoj paradigmi koja realan svet posmatra kao organizovan skup objekata. Objekti mogu biti predmeti, pojave, osobe i razni drugi koncepti iz realnog sveta koji okružuju čoveka.

Osnovne premise OO razvoja softvera su:

- Objekti imaju utvrđeno stanje i ponašanje;
- Objekti se nalaze u međusobnoj interakciji;
- Cilj interakcije je ostvarenje unapred zadatih ciljeva.

Pojam objekta - **objekat je fizička ili konceptualna jedinica posmatranja** koja poseduje: identitet, osobine, tj. obeležja i operacije. Objekat može biti kreiran, korišten tako što funkcioniše u nekom periodu vremena i na kraju treba da bude uništen. **Stanje objekta je skup obeležja** koji određuje strukturu objekta. Vrednosti obeležja definišu stanje objekta kao i veze između tog objekta i drugih objekata u nekom sistemu. **Operacije definišu ponašanje objekta** - izvršene operacije menjaju stanje objekta, tj. prevode objekat iz jednog u drugo stanje i daju informacije o stanju objekta.

Definicija klase - klasa, u OO pristupu, definiše identitet, relevantna obeležja i operacije objekta. To je tzv. "kalup" koji definiše strukturu objekta. Objekat je tzv. "odlivak" kreiran iz "kalupa", tj. jedna instanca (pojava) klase koja ima strukturu i ponašanje definisano u klasi. Objekt je konkretan entitet, za koji se vezuje skup atributa i skup operacija, koje menjaju vrednosti atributa objekta, te na

taj način deluju na stanje tog objekta. Kaže se da je klasa je skup objekata iste vrste. Definicija klase je obrazac za stvaranje objekata, sadrži deklaracije svih atributa i operacija vezanih uz objekt iz te klase.

Definicija klase:

Naziv klase
Modifikatori pristupa1 Naziv obeležja1: Tip podatka1, Ograničenja1; Modifikatori pristupa2 Naziv obeležja2: Tip podatka2, Ograničenja2; ... Modifikatori pristupaN Naziv obeležjaN: Tip podatkaN, OgraničenjaN;
Modifikatori pristupa1 Operacija1 (Parametar1, Parametar2, … ParametarN):Tip; Modifikatori pristupa2 Operacija2 (Parametar1, Parametar2, … ParametarN):Tip; … Modifikatori pristupaN OperacijaN (Parametar1, Parametar2, … ParametarN):Tip;

Primer klase i objekta:

NAZIV KLASE	**KLASA STUDENT**	**OBJEKAT KLASE STUDENT**
Atribut 1 Atribut 2 ... Atribut n	Broj indeksa Ime Prezime Smer	Broj indeksa: IT 1/16 Ime: Pera Prezime: Perić Smer: IT
Operacija 1 Operacija 2 ... Operacija n	Upis studenta(Parametri) Polaganje ispita(Parametri) Overa semestra(Parametri) Diplomiranje(Parametri)	Upis studenta (IT 1/16, Pera, Perić, IT) Polaganje ispita (IT 1/16, 10, BP) Overa semestra(IT 1/16, 1) Diplomiranje(IT 1/16, »1.1.2017«)

Osnovni koncepti OO paradigme su:

- **Klasifikacija**
  - grupisanje objekata koji imaju isti skup osobina i operacija,
  - definiše klasu objekata kao skup osobina i operacija,
  - klasa se nikada ne definiše u odnosu na vrednosti osobina.
- **Specijalizacija**
  - identifikacija podskupa nekog skupa objekata,
  - definiše podklase neke klase objekata,
  - objekti podklase imaju sve osobine i operacije nadklase, kao i dodatne osobine i operacije.
- **Polimorfizam**
  - sposobnost različitih klasa da realizuju iste operacije na različite načine;
  - različito ponašanje klasa u zavisnosti od konteksta, tj. konkretnog objekta čija se operacija (metoda) poziva.
- **Enkapsulacija**
  - sakrivanje implementacije.
- **Nasleđivanje**
  - preuzimanje atributa i metoda jedne klase u drugu klasu, uz mogućnost proširenja i redefinisanja. U objektnom modelu podataka postoje i dve vrste **nasleđivanja**: nasleđivanje ponašanja i nasleđivanje stanja. Za nasleđivanje ponašanja se koristi **generalizacija**, tj. **specijalizacija** (veza nadtip-podtip). Prilikom nasleđivanja stanja podređena klasa nasleđuje celokupno stanje i ponašanje klase koju proširuje.

Primeri nasleđivanja:

```
interface Osoba {…specifikacija tipa Osoba…};
class Student : Osoba {…specifikacija tipa Student…};
class Zaposleni : Osoba {…specifikacija tipa Zaposleni…};
class Nastavnik : Zaposleni {…specifikacija tipa Nastavnik…};
```

Podtip nasleđuje sve operacije nadtipa, ali se u podtipu mogu specificirati i neke njegove specifične operacije. Takođe, se u podtipu može redefinisati ponašanje definisano u nadtipu. Ova osobina modela se zove **prepisivanje** (“overriding”). U OOM se operacije specificiraju samo u okviru tipova (klasa, interfejsa). Ne postoje specifikacije van tipova, niti operacije koje se odnose na dva ili više tipova. Ime operacije je jedinstveno u okviru tipa. Različiti tipovi mogu imati različite operacije sa istim imenom. Ovaj koncept se zove **preopterećenje** (“overloading”). Overriding i overloading su koncept koji se zove **polimorfizam**, koji nam omogućava da redefinišemo kako objekat radi, tj. kako se ponaša. U okviru specifikacije operacije se može definisati **izuzetak** (“exception”) koji se izvršava kada nastupi neka greška u izvršavanju operacije. Nasleđivanje stanja se postiže konceptom veze **extends**. Podređena klasa nasleđuje celokupno stanje (ali i ponašanje) klase koju proširuje.

OOM se projektuje i dizajnira uz pomoć UML jezika. To je standardni jezik za specifikaciju, projektovanje i dokumentovanje softverskih elemenata, komponenti, softvera, informacionih sistema, hardvera. UML je izuzetno semantički bogat jezik sa preko deset vrsta modela koji se u njemu mogu kreirati i velikim brojem koncepata između kojih se može uspostaviti velik broj različitih vrsta veza. **U OOM postoji grafički prikaz tipova, strukture i veza između njih u formi dijagrama. Osim dijagrama, OOM uključuje objektnu analizu (kroz specifikaciju).** OOR podrazumeva objektno programiranje. I analiza i dizajn se bave uočavanjem objekata i klasa te razvijanjem objektnih modela.

Načini upotrebe sistema definišu se se dinamičkim modelima ponašanja, koji prikazuju interakciju sistema sa okolinom. Oblikovanje arhitekture sastoji se od identifikovanja podsistema i veza između podsistema. Pritom se podsistemi u UML notaciji prikazuju kao “package”, tj. skupovi objekata u dijagramu razmeštaja. Ponašanje objekata se modeluje modelom promene stanja objekta. Komunikacija objekata se sastoji od operacija koje su dostupne drugim objektima u dijagramu objekata.

Konačni rezultat OO projektovanja i dizajna jeste sistem koji se sastoji od objekata pripadaju nekoj klasi i koji su u međusobnoj interakciji. Interakcija objekata odvija se na način da jedan objekt (klijent) pokrene operaciju drugog objekta (poslužitelja). U tom smislu, operacije se mogu tumačiti kao servisi, a svaki objekat može se pojaviti u ulozi klijenta i u ulozi poslužitelja. Prilikom poziva operacija, objekti razmjenjuju podatke (parametre i rezultate). U objektno-oblikovanom sistemu ne postoje globalne funkcije koje bi se izvršavale nezavisno od objekata, već je funkcionalnost sistema izražena isključivo u terminima operacija vezanih uz objekte.

Objekti unutar sistema, u principu postoje i funkcionišu istovremeno. To daje mogućnost da se sistem distribuira na više računara ili da se istovremeni rad objekata simulira na jednom računaru. Novi objektno-oblikovani sistemi mogu biti razvijeni uz upotrebu objekata koji su bili stvoreni za prethodne sisteme. To smanjuje cenu razvoja softvera i vodi ka upotrebi standardnih objekata. Ipak, pokazuje se da su za ponovnu upotrebu pogodnije veće celine od pojedinih objekata. Važno svojstvo objektno-oblikovanog sistema je da on omogućuje laganu evoluciju. Naime, sistem se sastoji od objekata, dakle delova s jakom unutrašnjom kohezijom i labavim vezama prema spolja. Evolucija sistema zahtevaće promenu unutrašnje građe pojedinog objekta ili dodavanje novih objekata.