MRS

LAB 07

MRSLab07 – Metodologija Razvoja Softvera – Vežba 07

Dijagram objekata
Dijagram komponenti
Dijagram rasporeda

1. Dijagram objekata

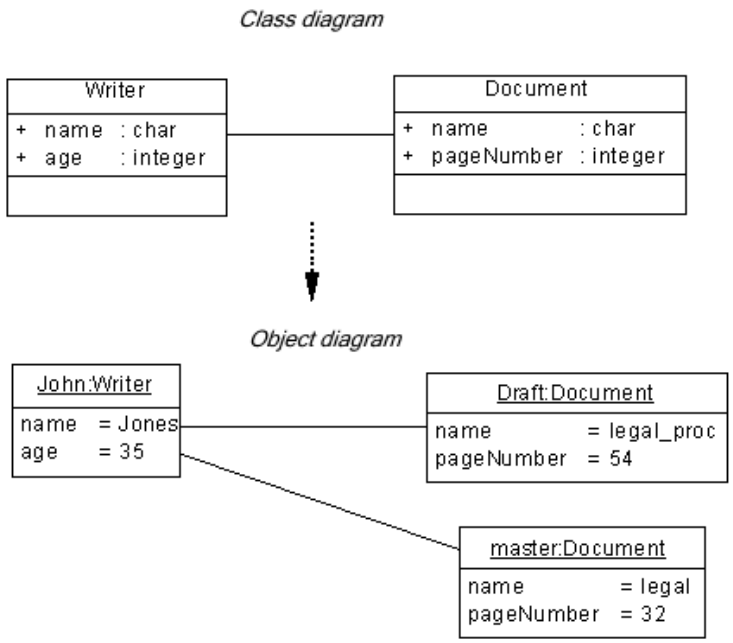
Dijagram objekata je UML dijagram koji opisuje strukturu elemenata koji se modeluju. On predstavlja instance klasa (objekte), instance asocijacija (linkovi instanci) i zavosnosti. Ne prikazuje relacije nasleđivanja.

Dijagram objekata prikazuje primer strukture podataka sa vrednostima podataka koje odgovaraju situaciji u sistemu u datom vremenskom momentu.

On se može koristiti za analizu, i to najčešće za prikaz organičenja među klasama koja nisu predstavljena u klasnom dijagramu.

Instance obično imaju više značenja od klase jer klasa predstavlja određeni nivo apstrakcije. Zadavanjem nekoliko instanci određene klase pomaže se razumeti svrha klase. Čak i kod apstraktne analize, dijagram objekata može pomoći da se razumeju neka strukturalna ograničenja koja se ne mogu lako grafički prikazati pomoću klasnih dijagrama.

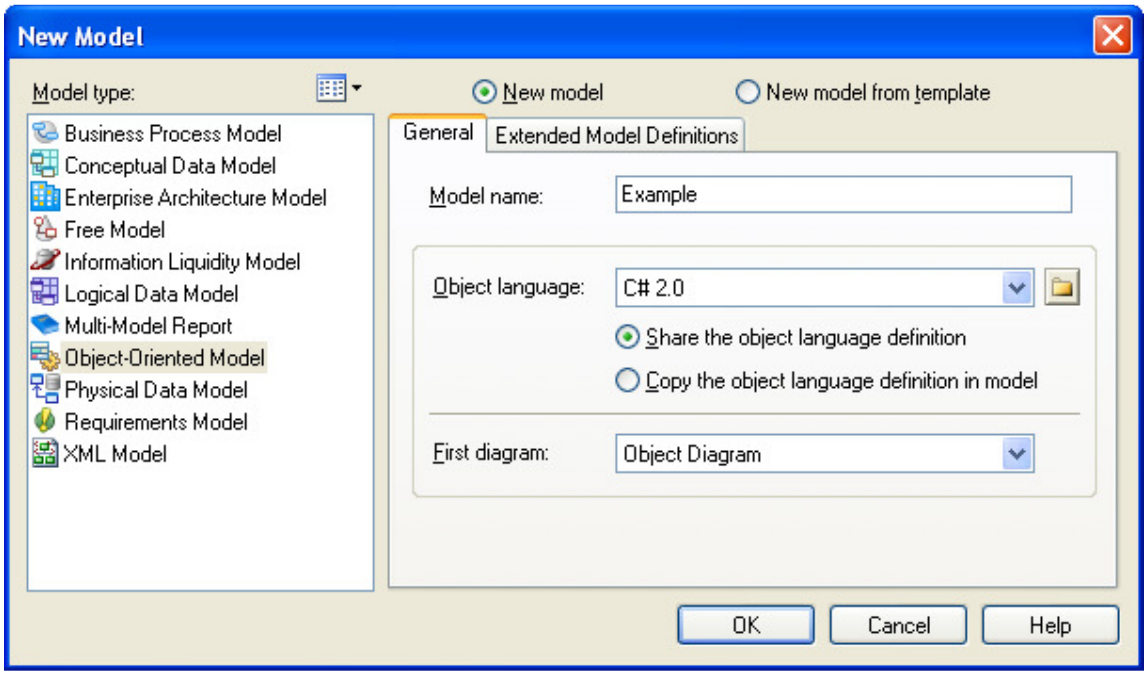
Na osnovu svega ovoga, može se reći da dijagrami objekata predstavljaju deo upotrebe dijagrama klasa. Npr. sledeća slika prikazuje dijagram klasa i dijagram objekata. Dijagram klasa specificira klasu *Writer* koja je povezana sa klasom *Document*. Dijagram objekata, izveden iz datog dijagrama klasa, naglašava sledeće detalje: objekat sa imenom *John* koji predstavlja instancu klase *Writer* je povezan sa dva različita objekta *Draft* i *Master* koja su oba instance klase *Document*.



Slika broj 1. Primer Dijagrama objekata

1.1. Kreiranje dijagrama objekata

Da bi kreirali novi dijagram klasa iz menija u PowerDesigner-u biramo File → New Model da bi se prikazao prozor New Model kao na sledecoj slici.



Slika broj 2. Kreiranje Dijagrama objekta

U listi tipova modela na levoj strani prozora biramo "Object-Oriented Model".

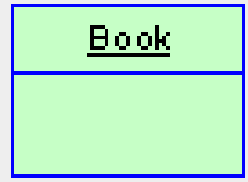
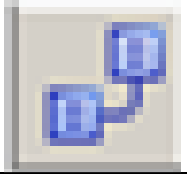
U polju "Model name:" upisujemo ime novog modela koji kreiramo.

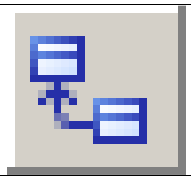
Sledeci korak je odabir objektnog jezika iz liste i odabir jedne od dve ponudene opcije za vidljivost promena u objektnom jeziku. Prva opcija podrazumeva da se promene u jeziku vide u svim povezanim objektno orjentisanim modelima, dok druga opcija govori da su promene vidljive samo u okviru datog modela.

Nakon ovoga se bira kog tipa ce biti prvi dijagram u modelu. Za kreiranje dijagrama objekata biramo "Object Diagram".

Potvrđujemo odabir klikom na taster "OK".

Prilikom kreiranja dijagrama objekata najčešće se koriste sledeći elementi:

Objekat	Alat	Simbol	Opis
Objekat			Instanca klase
Link Instance			Komunikacioni link između dva objekta

Zavisnost			Relacija između dva elementa koja se modeluju, pri čemu će promena u jednom elementu uticati na drugi element.

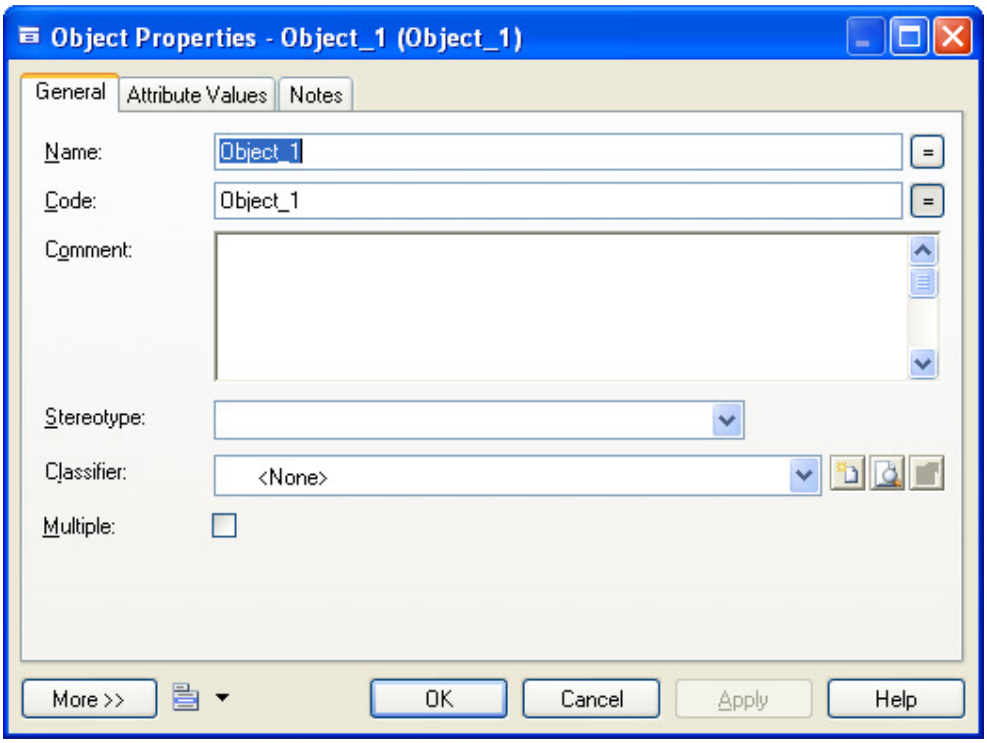
Slika broj 3. Elementi Dijagrama objekata

1.2. Objekat

Objekat u dijagramu objekata predstavlja instancu klase. On može prikazati vrednosti atributa definisanih u klasi. Kada se klasa izbriše, pridruženi objekti se ne brišu.

Da bi dodali novi objekat u radni prostor jednostavno kliknemo na ikonicu koja predstavlja objekat u paleti sa alatkami a zatim kliknemo na radni prostor.

Osobine objekta možemo menjati iz njegove stranice sa osobinama. Da bi otvorili stranicu sa osobinama za jedan objekat, potrebno je da dvokliknemo levim tasterom miša na ikonicu koja predstavlja taj objekat na radnoj površini. Nakon toga se otvara prozor sa osobinama prikazan na sledećoj slici.



Slika broj 4. Svojstva objekta

Prvi tab koji se otvara je tab General. Najčešće korišćene osobine iz ovog taba su sledeće:

- Name - određuje ime objekta koje treba da bude jasno i da upućuje na svrhu objekta. Ime objekta nije obavezno pošto možemo imati objekat koji predstavlja neimenovanu instancu klase ili interfejsa. Ime mora biti jedinstveno u okviru klase. Dva objekta mogu imati isto ime samo ako pripadaju različitim klasama.
- Code - određuje tehničko ime objekta koje se koristi za generisanje koda ili skripti

- Classifier - određuje ime klase za koju je dati objekat instanca. Može se povezati objekat sa postojećom klasom, ili se može kreirati nova klasa koristeći dugme *Create Class* koje se nalazi pored polja.
- Multiple - objekat može predstavljati više instanci (npr. činovnik može raditi sa više dokumenata, pri čemu ovi dokumenti predstavljaju višestruki objekat).

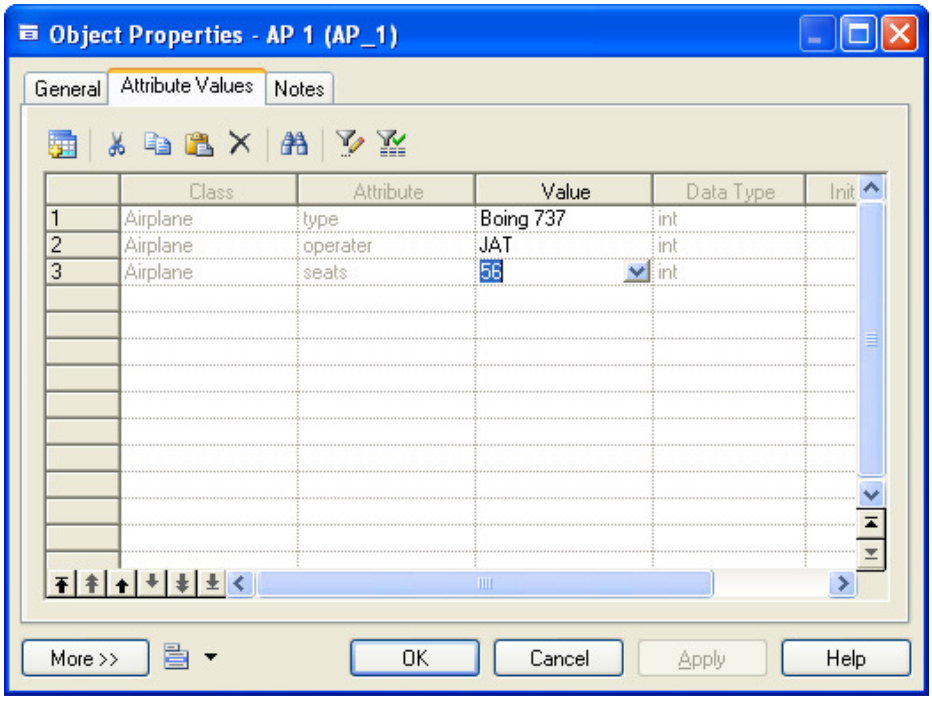
Kada objekat predstavlja instancu klase, atributi klase se mogu dodati u objekat iz taba Attribute Values u prozoru sa osobinama objekta. Kada se doda atribut moguće je dodeliti vrednost atributu u istom tabu.

Vrednost atributa predstavlja instancu atributa klase. On se odnosi na atribut klase za koju je dati objekat instanca, ili se može odnositi na atribut nasleđen iz klase roditelja date klase. Vrednost atributa je uskladištena kao tekst.

Dodeljivanje vrednosti atributima se vrši kroz tab Attribute Values. Dugme Add Attribute Values otvara prozor koji prikazuje sve atribute klase kojoj pripada objekat, uključujući i nasleđene objekte iz klase koja predstavlja klasu pretka za datu klasu. Kada se atribut doda objektu, moguće mu je dodeliti vrednost u koloni Value.

Jednom atributu klase se dodeljuje samo jedna vrednost. Vrednosti atributa se prikazuju u okviru simbola za objekat u dijagramu objekata.

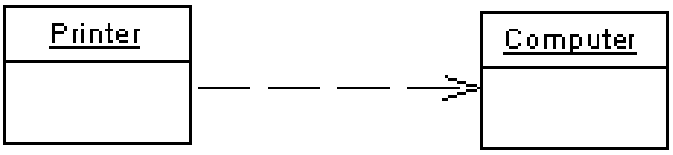
Da bi se atributu dodelila vrednost on mora pripadati klasi čiji je dati objekat instanca. Veza između klase objekata, atributa i vrednosti atributa se reguliše pomoću PowerDesigner-a. Npr. ako se promeni nasleđivanje, vrednost atributa datog objekta se automatski uklanja ako nije povezana sa atributima u novoj klasi ili njenoj klasi pretku.



Slika broj 6. Dodeljivanje vrednosti atributu

1.3. **Zavisnost**

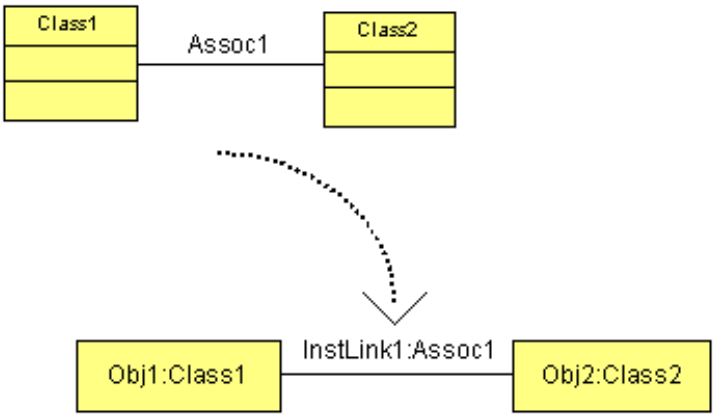
U dijagramu objekata zavisnost između dva objekta se može prikazati pomoću isprekidane linije. To je semantička relacija između dva objekta u kojoj promena u jednom objektu može uticati na značenje drugog objekta. Relacija zavisnosti ukazuje da jedan objekat na dijagramu koristi usluge drugog objekta.



Slika boj 7. Primer veze zavisnosti

1.4. **Linkovi instanci**

Linkovi instanci predstavljaju konekcije između dva objekta. On se crta kao puna linija. Linkovi instanci imaju jaku povezanost sa asocijacijama u djagramima klasa: asocijacije između klasa ili asocijacije između klasa i interfejsa mogu postati linkovi instanci između objekata u dijagramu objekata. Čak je i simbol za link instance u dijagramu objekata sličan sa simbolom za asocijaciju u dijagramu klasa, sa razlikom da link instance nema kardinalitete.



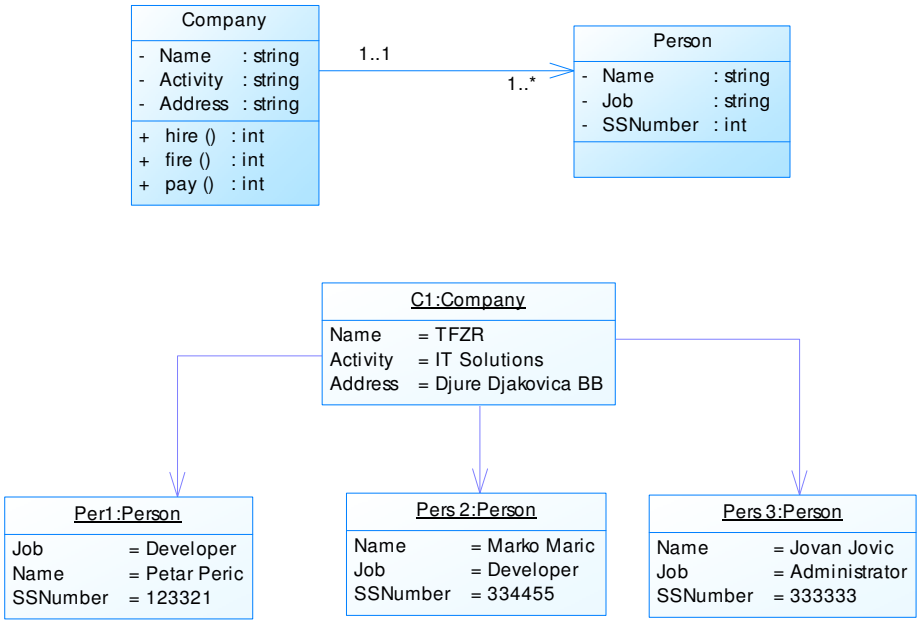
Slika broj 8. Primer Linkova instanci

Uloga linkova instanci je duplirana sa ulogom asocijacija. Zbog toga link instance može biti agregacija ili kompozicija, baš kao i kod asocijacije u dijagramima klasa. U datim slučajevima, simbol za agregaciju ili kompoziciju je prikazan u simbolu za link instance.

Sledeća pravila se primenjuju na linkove instanci:

- Kada asocijacija između klasa postane link instance, obe klase koje povezuje asocijacija i oba objekta koje povezuje link instance moraju odgovarati. Ovo važi i za asocijaciju između klase i interfejsa.
- Dva linka instance se mogu definisati između istih polaznih i odredišnih objekata (paralelni linkovi instanci).
- Mogu se koristiti refleksivni linkovi instanci (isti polazni i odredišni objekat).

Primer dijagrama klasa za kompaniju i zaposlenog kao i primer dijagrama objekata za isti premer je dat na sledećoj slici.



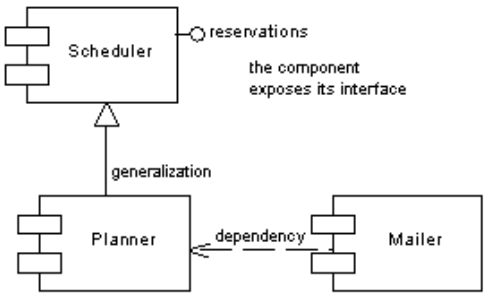
Slika broj 9. Primer dijagrama klasa i dijagram objekata za primer Kompanije za koju rade Osobe

2. Dijagram komponenti

Dijagrami komponenti su po svojoj prirodi i ponašanju drugačiji od klasnih dijagrama i dijagrama objekata. Oni se koriste za modelovanje fizičkih pogleda na sistem. Pitanje je šta su ovi fizički pogledi? Fizički pogledi su elementi poput izvršnih fajlova, biblioteka, dokumenata itd. koji predstavljaju sistem. Drugim rečima, dijagrami komponenti se koriste da prikažu organizaciju i odnose među komponentama u sistemu. Ovi dijagrami se koriste i za kreiranje izvršnih sistema.

Dijagram komponenti je UML dijagram koji predstavlja zavisnosti među softverskim komponentama, uključujući komponente koje sadrže izvorni kod aplikacije, komponente sa binarnim kodom i izvršne komponente. Komponente mogu biti povezane pomoću relacije generalizacije gde jedna komponenta nasleđuje drugu.

Sledeći primer prikazuje zavisnosti među komponentama u sistemu za rezervacije:



Slika broj 10. Primer zavisnosti mešu komponentama u sistemu za rezervacije

Dijagrami komponenti se koriste da definišu relacije i zavisnosti između objekata na višem nivou apstrakcije u odnosu na dijagram klasa.

Komponente bi trebale da budu dizajnirane tako da ih može koristiti više aplikacija, kao i da mogu biti proširene bez pada postojećih aplikacija.

Dijagram komponenti se koristi kako bi modelovali strukturu softvera i prikazali zavisnosti koje postoje između koda aplikacije, binarnog koda i izvršnih komponenti kako bi se mogao proceniti uticaj promena u nekoj od njih.

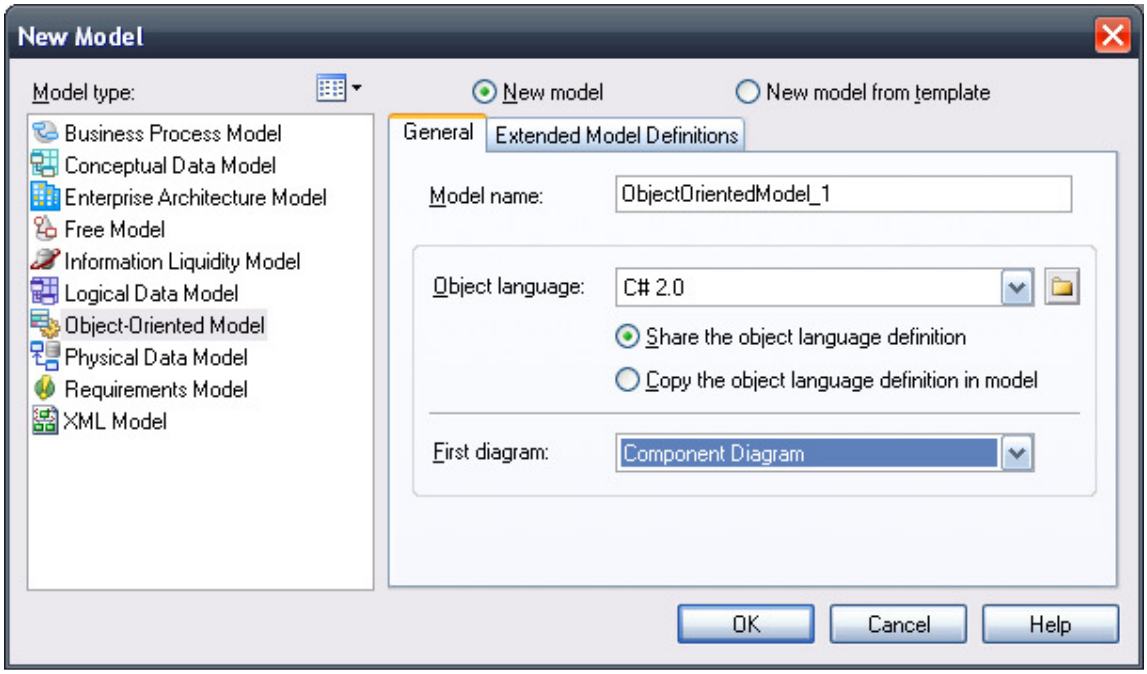
Ovi dijagrami su korisni kod analize i dizajna. Oni omogućavaju analitičarima i rukovodiocima projekata da odrede potrebne komponente pre nego što se krene sa njihovim razvojem i implementacijom. Dijagram komponenti daje pogled na komponente i omogućava njihov lakši dizajn, razvoj i održavanje.

Dijagrami komponenti se mogu opisati i kao statički prikaz implementacije sistema. Statička implementacija predstavlja organizaciju komponenti u određenom momentu.

Ovaj dijagram je veoma važan jer se pomoću njega omogućava efikasno kreiranje aplikacija. Dobro kreiran dijagram komponenti je takođe važan zbog drugih pogodnosti koje donosi a to su performanse aplikacije i lakše održavanje.

2.1. **Kreiranje dijagrama komponenti**

Da bi kreirali novi dijagram komponenti iz menija u PowerDesigner-u biramo File → New Model da bi se prikazao prozor New Model kao na sledećoj slici.



Slika broj 11. Kreiranje dijagrama komponenti

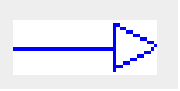
U listi tipova modela na levoj strani prozora biramo "Object-Oriented Model".

U polju "Model name:" upisujemo ime novog modela koji kreiramo. Kod modela koji može biti korišćen za generisanje koda, je izveden iz ovog imena u skladu sa konvencijama imenovanja modela.

Sledeći korak je odabir objektnog jezika iz liste i odabir jedne od dve ponuđene opcije za vidljivost promena u objektnom jeziku. Prva opcija podrazumeva da se promene u jeziku vide u svim povezanim objektno orjentisanim modelima, dok druga opcija govori da su promene vidljive samo u okviru datog modela.

Nakon ovoga se bira kog tipa će biti prvi dijagram u modelu. Za kreiranje dijagrama komponenti biramo "Component Diagram". Potvrđujemo podešavanje klikom na taster "OK".

U dijagramima komponenti se kreiraju sledeći objekti:

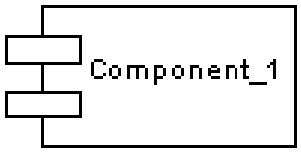
Objekat	Alat	Simbol	Opis
Komponenta			Komponenta predstavlja deljivi deo implementacije sistema.
Interfejs			Komponenta koja opisuje spolja vidljive operacije klasa, objekata ili drugih entiteta bez specifikacije unutrašnje strukture.
Generalizacija			Veza između opšte komponente i određenije komponente koja je nasleđuje i dodaje određene funkcionalnosti.
Realizacija			Semantička relacija u kojoj jedan klasifikator, najčešće interfejs, specificira sadržaj koji drugi klasifikator implementira.
Zavisnost			Relacija između dva elementa u kojoj promena u jednom elementu utiče na drugi.

Slika broj 12. Objekti dijagrama komponenti

2.2. **Komponente**

Komponente su fizički delovi sistema koji sadrže implementaciju i omogućavaju realizaciju za skup interfejsa. One mogu predstavljati fizički deo implementacije sistema kao što su softverski kod, skripte ili komandni fajlovi. Predstavljaju nezavisni deo softvera razvijen za specifičnu upotrebu. Mogu biti kreirane od klasnih dijagrama ili napisane od početka za novi sistem koji se modeluje.

Simbol za komponentu je:

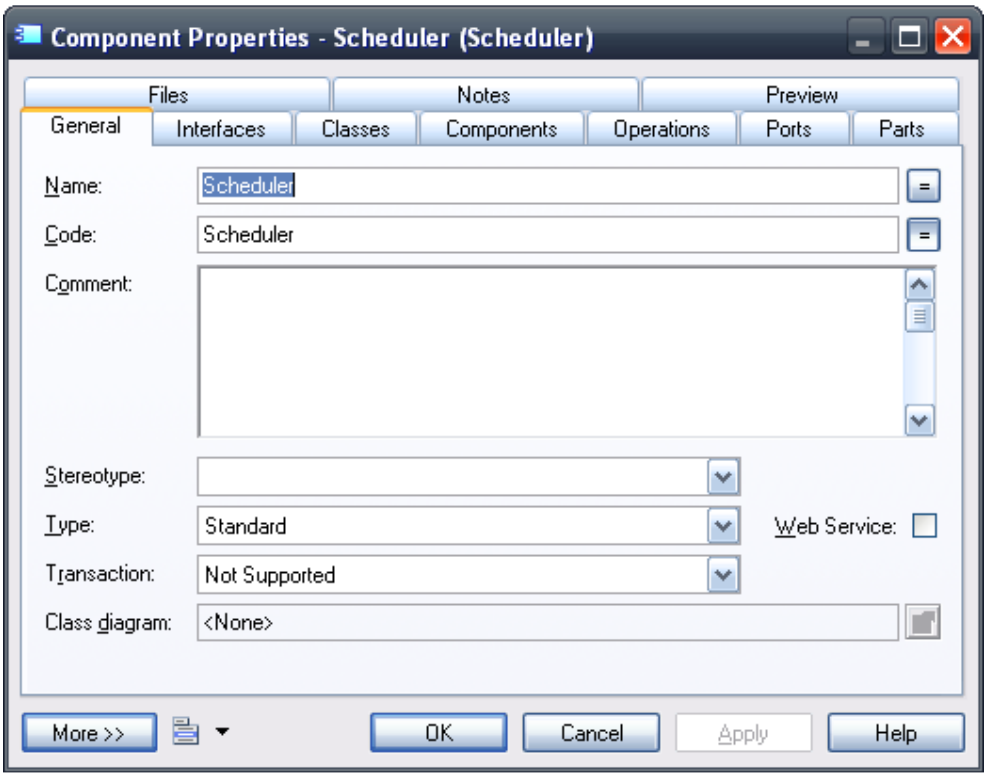


Slika broj 13. Simbol komponente

Komponenta predstavlja primer principa "crne kutije" u kreiranju softvera. Osoba koja kreira komponentu ima unutrašnji pogled u komponentu: njene interfejse i klase koje ih implementiraju, dok osoba koja povezuje komponente kako bi kreirala druge komponente ili aplikaciju ima samo spoljašnji pogled na komponente. Ona vidi samo njen interfejs.

Komponenta može biti implementirana u bilo kom programskom jeziku koji podržava PowerDesigner.

Osobine komponente možemo menjati iz njene stranice sa osobinama. Da bi otvorili stranicu sa osobinama za jednu komponentu, potrebno je da dvokliknemo na ikonicu koja predstavlja tu komponentu na radnoj površini. Nakon toga se otvara prozor sa osobinama prikazan na sledećoj slici.



Slika broj 14. Svojsta komponente

Prvi tab koji se otvara je tab General. Najčešće korišćene osobine iz ovog taba su sledeće:

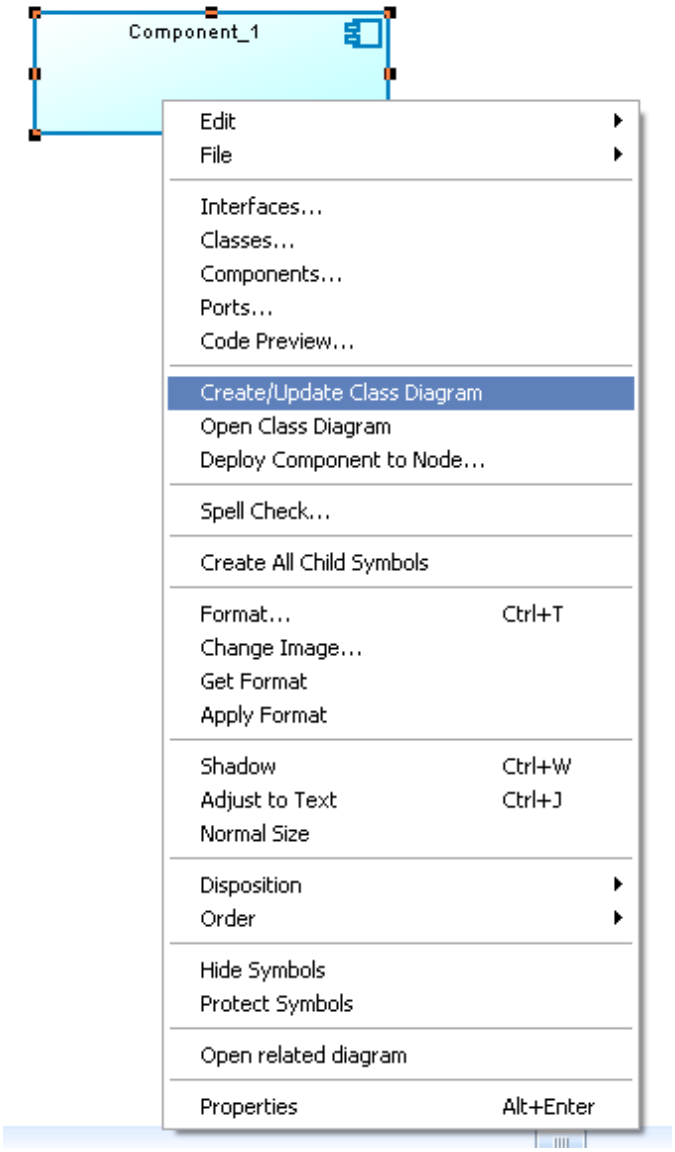
- Name - određuje ime komponente koje treba da bude jasno i da upućuje na svrhu.
- Code - određuje tehničko ime objekta koje se koristi za generisanje koda ili skripti.
- Class diagram - dijagram sa klasama i interfejsima koji je povezan sa komponentom.

Interfejsi koji se nalaze u komponenti označavaju da komponenta vrši implementaciju datih interfejsa. Svaka komponenta koristi jedan ili više interfejsa. Ona takođe koristi interfejse iz drugih komponenti. Ovi interfejsi predstavljaju ulazne tačke i servise koje komponenta omogućava drugim komponentama ili klasama.

Klase koje se nalaze u komponenti su klase koje vrše implementaciju komponente. Komponenta ima jednu ili više klasa.

Za određenu komponentu možemo dodati dijagram klasa kako bi imali sveobuhvatan pregled klasa i interfejsa pridruženih komponenti. Za jednu komponentu se može kreirati samo jedan dijagram klasa.

Da bi dodali dijagram klasa za komponentu kliknemo desnim tasterom miša na komponentu i odaberemo: Create/Update Class Diagram.



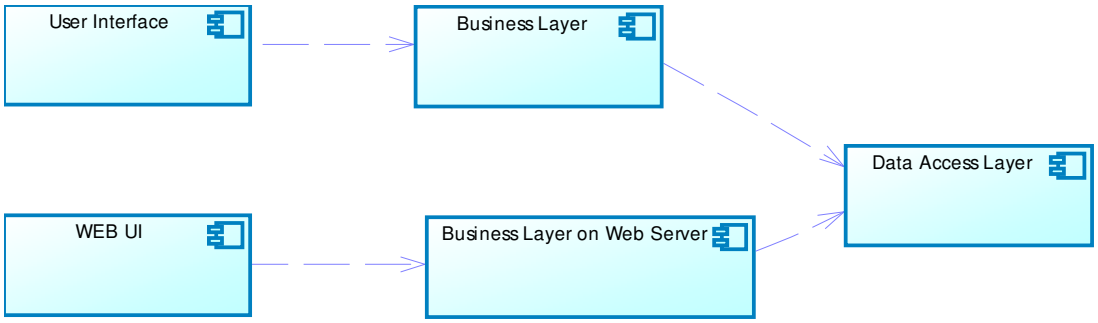
Slika broj 15. Kreiranje dijagrama klasa / povezivanje dijagrama klasa sa komponentom

Opcija Create/Update Class Diagram radi sledeće:

- Kreira dijagram klasa ukoliko on ne postoji
- Povezuje dijagram klasa sa komponentom

Opcija Open Calss Diagram otvara dijagram klasa koji je pridružen komponenti.

Primer dijagrama komponenti za troslojnu aplikaciju koja koristi istu bazu podataka i desktop i web pristup podacima je dat na sledećoj slici:



Slika broj 16. Primer dijagrama komponenti troslojne aplikacije

3. Dijagram rasporeda

Dijagrami rasporeda se koriste da prikažu topologiju fizičkih komponenti sistema, odnosno gde su razmeštene softverske komponente sistema. Oni se koriste da opišu statički prikaz rasporeda sistema. Ovi dijagrami se sastoje od čvorova i njihovih međusobnih odnosa.

Sam naziv *Raspored* opisuje svrhu ovih dijagrama. Dijagrami rasporeda se koriste da opišu hardverske komponente na kojima su raspoređene softverske komponente. Dijagrami komponenti i dijagrami rasporeda su blisko povezani. Dijagrami komponenti se koriste da opišu komponente a dijagrami rasporeda prikazuju kako su te komponente razmeštene po hardveru.

UML je dizajniran da se pretežno fokusira na softverske činioce sistema. Međutim, ova dva dijagrama su specijalni dijagrami koji se koriste za fokus na softverske komponente i hardverske komponente. Većina UML dijagrama se koristi za logičke komponente dok se dijagrami rasporeda koriste da bi se ukazalo na hardversku topologiju sistema.

Svrha dijagrama rasporeda se može predstaviti kao:

- Vizualizacija hardverske topologije sistema.
- Opis hardverskih komponenti koje se koriste za raspodele softverske komponente.
- Opis čvorova za obradu izvršavanja.

Dijagrami rasporeda se sastoje od čvorova. Čvorovi predstavljaju fizički hardver koji se koristi da se na njega smesti aplikacija. Ovi dijagrami su korisni za inženjere koji se bave projektovanjem sistema. Dobar dijagram rasporeda je veoma važan jer kontroliše sledeće parametre:

- Performanse.
- Skalabilnost
- Održavanje
- Prenosivost

Za vežbu na času / Domaći zadatak:

Kreirati dijagram klasa i dijagram objekata za aplikaciju koja sadrži podatke o studentima, katedri kojoj dati studenti pripadaju i položenim ispitima. (Napomena: student, katedra i ispit predstavljaju posebne klase).