Projektovanje softvera

Projektni uzorci

Uvod

- Materijali pripremljeni prema knjizi:
  Gama, Helm, Johnson, Vlissides, “Design Patterns”

- Christopher Alexander govori o uzorcima u građevinskoj arhitekturi:
  - “Svaki uzorak opisuje neki problem koji se ponavlja u našem okruženju
    i tada opisuje jezgro rešenja tog problema na takav način
    da se to rešenje može koristiti milion puta,
    a da se ne uradi ni dva puta na isti način”
- Definicija je primenljiva i na uzorke u objektno-orijentisanim sistemima

- Projektni uzorak predstavlja zabeleženo široko primenljivo iskustvo
  u projektovanju OO sistema
- Termini: projektni uzorak, uzor, obrazac, šablon (engl. *Design Pattern*)

O projektnim uzorcima

- Projektni uzorci su opisi komunicirajućih objekata i klasa koji su prilagođeni da reše neki opšti projektni problem u posebnom kontekstu
- Projektni uzorak
  - identifikuje učestvujuće klase i objekte, njihove relacije, njihove uloge u saradnji i raspodelu odgovornosti
  - sistematično imenuje, objašnjava i ocenjuje važno projektno rešenje ponavljajućeg problema u OO sistemima
  - korisan je za kreiranje ponovo upotrebljivog OO projektnog rešenja
- Svaki projektni uzorak ima 4 esencijalna elementa:
  - naziv uzorka
  - postavku problema
  - opis rešenja
  - diskusiju posledica

Naziv uzorka

- Koristi se da opiše projektni problem, njegovo rešenje i posledice primene u par reči
- Imenovanje uzorka proširuje projektni rečnik
  - omogućava projektovanje na višem nivou apstrakcije
  - pojednostavljuje komunikaciju u timu
  - olakšava dokumentovanje projekta
- Ponekad se u literaturi sreće i više imena za isti uzorak

Postavka problema

- Objašnjava opšti problem
- Navodi motivaciju za primenu uzorka
  - na primeru u nekom posebnom kontekstu
- Opisuje:
  - uopšten projektni problem,
    npr. kako reprezentovati algoritme kao objekte
  - strukture klasa ili objekata
    simptomatične za nefleksibilni dizajn
  - uslove koji se moraju ispuniti za primenu uzorka

Opis rešenja

- Opisuje elemente koji predstavljaju jezgro rešenja
  - njihove uloge, relacije, odgovornosti i saradnje
- Rešenje ne opisuje konkretan dizajn ili implementaciju
- Rešenje treba da posluži kao šablon
  - da se može primeniti u konkretnim slučajevima
- Rešenje predstavlja apstraktni opis načina aranžiranja elemenata (klasa i/ili objekata)

Posledice

- Posledice su rezultati primene uzorka
- Kritične su za razmatranje projektnih alternativa i razumevanje cene i dobiti primenom uzorka
- Posledice se često odnose na vreme i prostor
- Ponekad se odnose i na jezik i implementacione stvari
- Posledice najčešće uključuju uticaj na:
  - fleksibilnost (prilagodljivost) sistema
  - proširivost sistema
  - prenosivost (portabilnost) sistema

Primer primene uzoraka: MVC

- MVC je arhitekturni okvir za razvoj aplikacija
  - skraćeni naziv za trijadu klasa *Model*, *View* i *Controller*
- Poreklo – realizacija korisničkih interfejsa u *Smalltalk* jeziku
- *Model* – konkretan objekat domenske (poslovne) logike
- *View* – ekranska prezentacija objekta
- *Controller* – opis reakcije korisničkog interfejsa na ulaz korisnika
- Interakcija (ponašanje):
  - *Model-View*
  - *View-Controller*
- Strukturiranje: *View*
- Primeri za 3 projektna uzorka

Model – View komunikacija (1)

- Prikaz mora da obezbedi da reflektuje aktuelno stanje modela
- Kad se podaci modela promene
  - model signalizira promenu prikazima koji zavise od njega
  - svaki prikaz dobija priliku da se ažurira
- Pristup omogućava više uzajamno nezavisnih pogleda na model
  - svaki pogled – prezentacija posebnog aspeka modela
- Mogu se jednostavno dodavati novi prikazi
  - ne menja se model
  - ne menjaju se drugi prikazi
- Primer:
  - model sadrži neke statističke podatke
  - postoje tri prikaza: tabelarni, histogram, pita

Model – *View* komunikacija (2)

- Protokol između *View* i *Model*-a je tzv. *subscribe-notify*
  - prikazi se pretplaćuju kod modela na obaveštavanje
  - model obaveštava pretplaćene prikaze kada se dogodi promena
  - kada dobiju obaveštenje (signal) prikazi čitaju novo stanje modela
  - prikazi prilagođavaju svoj izgled novom stanju modela
- Generalizacija problema obaveštavanja "pretplatnika"
  - promene jednog objekta treba da utiču na proizvoljan broj drugih
  - nema potrebe da menjani objekat zna za detalje drugih
- Rešenje generalnog problema
  - opisano uzorkom Posmatrač (*Observer*)
  - uloge: subjekat (MVC Model) i posmatrač (MVC View)

Komponovanje prikaza

- Jedna od mogućnosti MVC arhitekture
  - prikazi (*Views*) mogu biti ugnežđeni
- Primer:
  - kontrolni panel je prikaz koji sadrži ugnežđene prikaze dugmadi
- Klasa *CompositeView* izvedena iz *View*
  - predstavlja vrstu prikaza koji sadrži druge prikaze
  - može se pojaviti gde god se očekuje objekat osnovne klase *View*
  - dobija se objektna hijerarhija (stablo) prikaza
- Generalizacija problema strukturno složenih objekata (sklopova)
  - grupisanje objekata gde je grupa istog (nad)tipa kao i pojedini objekat
- Rešenje generalnog problema
  - opisano uzorkom Kompozicija (Sastav, Sklop, engl. *Composite*)
- Sličan je odnos klasa *Container* i *Component* u Javi

View – Controller komunikacija (1)

- MVC omogućava promenu načina reakcije *View* na ulaz korisnika
- Primer:
  - moguće je redefinisati odgovore na događaje sa tastature/miša ili
  - zameniti komandne tastere *pop-up* menijem
- MVC kapsulira mehanizam odgovora u objekat tipa *Controller*
- Postoji hijerarhija klasa kontrolera koja olakšava kreiranje novog
  - novi kontroler se kreira kao varijacija nekog postojećeg
- *View* koristi objekat potklase *Controller*
  - objekat kontrolera specificira strategiju odgovora
- Za promenu strategije odgovora
  - samo se zamenjuje objekat kontrolera drugom vrstom kontrolera

View – Controller komunikacija (2)

- Izmenu strategije je moguće vršiti i u vreme izvršenja
- Primer:
  - *View* može da onemogući interakciju (*disabled* stanje) tako što mu se dodeli kontroler koji ignoriše ulaze
- Generalizacija problema promenljivog algoritma
  - izmena (statička/dinamička) algoritma koji definiše ponašanje objekta
- Rešenje generalnog problema
  - opisano uzorkom Strategija (*Strategy*)
  - objekti strategije apstrahuju i kapsuliraju algoritam ponašanja
  - lako se zamenjuju u nekom kontekstu (promena pokazivača)

Klasifikacija projektnih uzoraka

- Namena i nivo apstrakcije varira kod različitih uzoraka
- Kao i svaka druga klasifikacija, ova klasifikacija
  - doprinosi boljem i bržem snalaženju pri traženju odgovarajućeg uzorka
  - usmerava napore ka otkrivanju novih uzoraka
- Klasifikacija koristi dva kriterijuma za razvrstavanje uzoraka
  - kriterijum namene (šta uzorak opisuje)
  - kriterijum domena (na koji nivo apstrakcije se odnosi uzorak)
- Klasifikacija nije strogo formalna, pa tako ni sasvim precizna
  - zasniva se u dobroj meri na intuitivnoj proceni
  - ipak, pomaže razumevanju prirode projektnog uzorka

Kriterijum namene

- Prvi kriterijum deli uzorke prema nameni
  - odražava čime se uzorak bavi (šta opisuje)
- Namena uzorka može biti da opiše
  - kreiranje (*creational patterns*)
    - uzorci tretiraju kreiranje objekata
  - strukturu (*structural patterns*)
    - uzorci tretiraju kompoziciju klasa ili objekata
  - ponašanje (*behavioral patterns*)
    - uzorci tretiraju načine na koje objekti ili klase interaguju

Kriterijum domena

- Drugi kriterijum deli uzorke prema nivou apstrakcije
  - da li se uzorak primarno primenjuje na klase ili na objekte
- Klasni uzorci (*class patterns*)
  - fokusiraju se na relacije između klasa i potklasa
  - ove relacije su generalizacije/specijalizacije
  - one su statičke – fiksirane u vreme prevođenja
- Objektni uzorci (*object patterns*)
  - fokusiraju se na relacije između objekata
  - ove relacije su uglavnom primerci asocijacija (veze)
  - one se mogu menjati u vreme izvršenja, pa su dinamičnije
- Većina uzoraka je u objektnom domenu

Karakteristike vrsta uzoraka

- Klasni uzorci kreiranja
  - odlažu neke delove kreiranja objekata da budu implementirani u potklasama
- Objektni uzorci kreiranja
  - delegiraju neke delove kreiranja objekata drugim objektima
- Klasni uzorci strukturiranja
  - koriste nasleđivanje da komponuju klase
- Objektni uzorci strukturiranja
  - opisuju načine asembliranja (sklapanja celina od nekih) objekata
- Klasni uzorci ponašanja
  - koriste nasleđivanje da opišu algoritme i tok kontrole
- Objektni uzorci ponašanja
  - opisuju kako grupa objekata sarađuje da obavi neki zadatak

Prostor projektnih uzoraka

- Tabela reflektuje klasifikaciju uzoraka po dva kriterijuma
  - kolone sadrže vrste uzoraka po nameni
  - vrste sadrže vrste uzoraka po domenu

		Namena		
		uzorci kreiranja	**uzorci strukture**	**uzorci ponašanja**
Domen	**klasni uzorci**	Fabrički metod	Adapter (klasni)	Interpreter Šablonski metod
	objektni uzorci	Apstraktna fabrika Graditelj Prototip Unikat	Adapter (objektni) Most Sastav Dekorater Fasada Muva Zastupnik	Lanac odgovornosti Komanda Iterator Posrednik Podsetnik Posmatrač Stanje Strategija Posetilac

Drugi odnosi između uzoraka

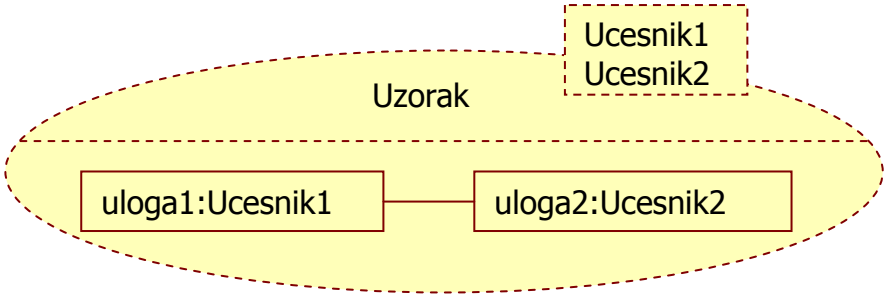
- Postoje i drugi načini za organizovanje uzoraka:
  - neki uzorci se često koriste zajedno
    - npr. Kompozicija (*Composite*) i Iterator (*Iterator*) ili Posetilac (*Visitor*)
  - neki uzorci su alternative jedni drugima
    - npr. Prototip (*Prototype*) i Apstraktna fabrika (*Abstract Factory*)
  - neki uzorci različitih namena rezultuju u sličnoj implementaciji
    - npr. Kompozicija (*Composite*) i Dekorater (*Decorator*)

Katalog projektnih uzoraka

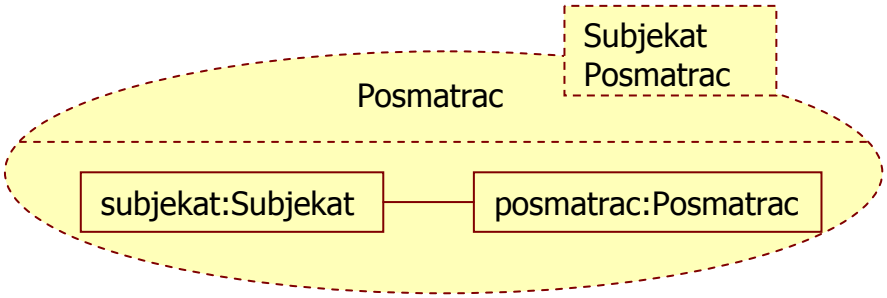
- Katalog sadrži za svaki uzorak sledeće stavke:
  - ime uzorka i klasifikacija
  - namena – odgovori na pitanja "šta radi? čemu služi? koji problem rešava?"
  - drugi nazivi – isti uzorak može imati više naziva
  - motivacija – scenario koji ilustruje projektni problem
  - primenljivost – situacije u kojima se uzorak može primeniti (da se izbegne loš dizajn)
  - struktura – klasni (eventualno i objektni) dijagram koji opisuje uzorak
  - učesnici – klase i objekti koji učestvuju u uzorku i njihove odgovornosti
  - saradnje – kako učesnici sarađuju da ispolje svoje odgovornosti (eventualno d. interakcije)
  - posledice – diskusija dobrih i loših strana primene uzorka
  - implementacija – zamke, preporuke i tehnike kojih treba biti svesan pri implementaciji
  - primer koda – fragment koda koji ilustruje kako se uzorak može implementirati
  - poznate primene – primeri primene uzorka u realnim sistemima (barem po dva)
  - povezani uzorci – koji uzorci su bliski sa datim, koje su razlike, sa kojima se često koristi
  - UML notacija – grafički simbol za saradnju koja realizuje uzorak

Definicija uzoraka u UML notaciji

- Definicija uzorka (strukturirana parametrizovana saradnja):

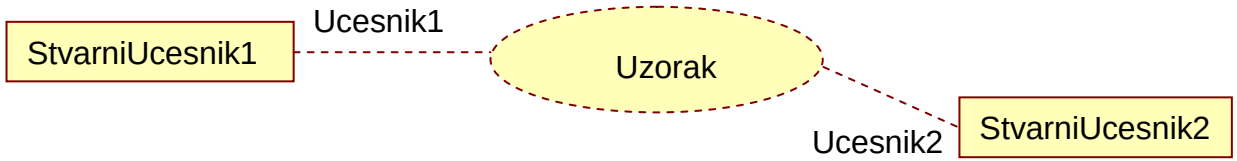


- Formalni parametri parametrizovane saradnje:
  - tipovi uloga učesnika u definiciji projektnog uzorka
- Primer:



Primena uzoraka u UML notaciji

- Primena uzorka:



- Stvarni parametri parametrizovane saradnje:
  - tipovi stvarnih učesnika – konkretne klase u modelu
- Relacija između saradnje i stvarnih učesnika je "binding“
  - isprekidana linija na kojoj se navodi formalni tip učesnika iz definicije
- Primer: