Grupa 1

LightsOn
Dokument softverske arhitekture

Verzija 1.0

Istorija revizija

Datum	Verzija	Opis	Autor
10.01.2025	1.0	Odradjen je dokument u cijelini	Dušan Marilović

Sadržaj

Dokument softverske arhitekture

1. Uvod

Ovaj dokument pruža osnovne informacije o arhitekturi softvera. Uključuje namjenu, obim, definicije i reference. Arhitektura softvera predstavlja značajnu postavku u dizajnu i implementaciji softverskog proizvoda.

1.1 Namjena

Namjena ovog dokumenta jeste prikaz arhitektonske prirode sistema, uključujući različite prikaze radi opisa sistema. Osnovna funkcija mu je da indentifikuje značajne arhitektonske odluke koje se odnose na sistem.

1.2 Obim

Ovaj dokument obuhvata specificiranje arhitekturu sistema, i značajne arhitektonske odluke u dizajnu za projektovanje sistema za projekat iz predmeta Projektovanje softvera Elektrotehničkog fakulteta u Istočnom Sarajevu.

1.3 Definicije i skraćenice

Ovaj dio dokumenta objašnjava definicije i skraćenice korištene pri dizajnu arhitekture softvera i to:

i. **Model-Observer** - Arhitekturalni obrazac koji se zasniva na principu delegiranja obaveštenja od modela ka svim zainteresovanim posmatračima (view-ovima).
ii. **Observer (posmatrač)** - Obrazac koji omogućava objektima da se prijave na promene drugog objekta i reaguju na njih.

iii. **Pattern ili šablon** - predstavlja naziv za programsku strukturu softvera koji predstavlja dobru praksu u riješavanju nekog problema

1.4 Pregled

U nastavku ćemo dati pregled arhitekture softvera, njenih ciljeva, i različitih pogleda na samu arhitekturu sistema.

2. Arhitektonski prikaz

U dizajnu softvera korišćena je Model-Observer arhitektura. Ovaj pristup omogućava da model funkcioniše nezavisno od korisničkog interfejsa, dok posmatrači (observeri) reaguju na promene modela bez potrebe da model direktno zna za njih. Komunikacija izmedju komponenti sistema ostvaruje se putem obrazaca kao što su Observer, koji omogućava ažuriranje interfejsa kada se model promeni, Command, koji omogućava fleksibilnu obradu korisničkih akcija, State, koji se koristi za upravljanje različitim stanjima sistema.

3. Arhitektonski ciljevi I ogranicenja

Primena Model-Observer arhitekture omogućava jasno razdvajanje odgovornosti izmedju modela i interfejsa, poboljšanu modularnost koja olakšava zamenu komponenti sistema, jednostavno dodavanje novih funkcionalnosti bez velikih promena u postojećem kodu i fleksibilnost u prikazivanju podataka različitim korisnicima. Odabrani arhitektonski šablon je pogodan za kasniju nadogradnju softvera.

4. Logički pogled

Implementacija softvera je razdvojena na programske module koji su odgovarajuće povezani. Svaki modul se sastoji od odgovarajućeg broja klasa čija saradnja realizuje slučajeve korištenja.

4.1 Pregled

Ovaj dio dokumenta prikazuje način dekompozicije programa na programske module i definiše njihovu strukturu, odgovornosti i funkcije.

4.2 Arhitektonski značajni dizajn paketi

Model programski paket predstavlja dio sistema koji sadrži klase za stanje samog sistema i njihovu prezentaciju u operativnoj memoriji računara. On služi za čuvanje informacija o sistemu, trenutnim dokumentima i odgovoran je za konkretne podatke koji su potrebni radu softvera.

View programski paket sadrži klase koje definišu način predstavljanja podataka iz modela. On je zadužen za grafičko predstavljanje podataka korisnicima

Controller programski paket je dio sistema koji sadrži klase koje implementiraju logiku sistema i odgovorni su za prihvatanje korisničkih akcija i ažuriranje stanja modela.

5. Pogled prema procesima

Glavne korisničke akcije obradjuju se kroz delegat mehanizam, gde model obaveštava posmatrače o promenama, a posmatrači ažuriraju prikaz ili iniciraju dodatne promene u sistemu. Ovaj proces omogućava visok stepen fleksibilnosti u obradi korisničkih zahteva i minimizuje direktne zavisnosti izmedju komponenata sistema.

6. Pogled prema instalaciji

Ne postoje hardverska ograničenja u smislu pokretanja softvera na različitim računarskim sistemima. Softver je realizovan u Java programskom jeziku koji je multiplatformski.

7. Implementacioni pogled

Implementacija softvera je zasnovana na Model-Observer arhitekturi, pri čemu se sistem deli na logičke slojeve. Sloj modela implementira stanje i poslovnu logiku softvera, pri čemu koristi obrazac Observer za obaveštavanje interfejsa o promenama. Sloj posmatrača se prijavljuje na model i ažurira prikaz podataka u interfejsu ili preduzima druge akcije u zavisnosti od promena u modelu.

7.1 Pregled

Ovaj dio dokumenta definiše različite slojeve u sistemu, njihovu implementaciju i načine komunikacije.

7.2 Slojevi

Sloj podataka predstavlja sloj koji sadrži informacije o podacima i osnovnim dijelovima softvera koji su u operativnoj memoriji računara. Njegova implementacija se vrši pomoću klasa iz programskog paketa model.

Prezentacioni sloj predstavlja sloj koji služi za prezentaciju podataka prema korisniku, tj. njihovu grafičku reprezentaciju. Implementira se programskim paketom view. Kako on prikazuje podatke iz sloja podataka mora imati neposrednu svijest o njemu.

Sloj kontrole, tj. logike predstavlja sloj koji služi za implementaciju funkcionalnog dijela aplikacije.

On reaguje na korisničke instrukcije. Realizuje se programskim paketom observer. Kao takav, on ima svijest i o sloju podataka i o prezentacionom sloju.

8. Veličina i performance

Performanse sistema zavise od broja posmatrača i složenosti delegiranih zadataka. Optimizacija obaveštenja može smanjiti preopterećenje sistema i poboljšati odziv aplikacije. Efikasan način upravljanja događajima i obaveštenjima ključan je za održavanje dobre skalabilnosti sistema.

9. Kvalitet

Arhitektura softvera bi trebala rezultovati u poboljšanom kvalitetu softvera, prije svega na njegovu pouzdanost, prenosivost i proširivost.