

queri

Dokument softverske arhitekture

Verzija 1.0

Istorija revizija

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 03/10/2023 | 1.0 | Prva verzija dokumenta. | Milan Vlaški |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Architectural Representation 4

3. Architectural Goals and Constraints 4

4. Use-Case View 4

4.1 Use-Case Realizations 5

5. Logical View 5

5.1 Overview 5

5.2 Architecturally Significant Design Packages 5

6. Process View 5

7. Deployment View 5

8. Implementation View 5

8.1 Overview 5

8.2 Layers 5

9. Data View (optional) 6

10. Size and Performance 6

11. Quality 6

Dokument softverske arhitekture

# Uvod

## Svrha

Ovaj dokument pruža sveobuhvatni arhitekturni pregled sistema, koristeći nekoliko različitih pogleda za izražavanje različitih aspekata sistema. Cilj mu je da predoči važnije arhitekturne odluke na sistemu.

## Obim

Dokument predstavlja bazu za arhitekturu koju će imati *queri*, projekat Grupe 1. Samim tim, utiče na *Smjernice za dizajn,* kaoi čitav pristup projektovanju i implementaciji softverskog rješenja.

## Definicije, akronimi i skraćenice

Za sve nejasnoće pogledati *Rječnik.*

## Reference

[GUI Architectures](https://www.martinfowler.com/eaaDev/uiArchs.html), Martin Fowler

*Projektovanje i dizajn softvera: koncepti, principi i projektni obrasci*, prof. dr Vladimir Vujović

## Pregled

# Predstavljanje arhitekture

[This section describes what software architecture is for the current system, and how it is represented. Of the **Use-Case**, **Logical**, **Process**, **Deployment**, and **Implementation Views**, it enumerates the views that are necessary, and for each view, explains what types of model elements it contains.]

Arhitektura softverskog alat *queri* predstavljena je pogledima: Use-case, Logički i Implementacioni. Koristiće se postojeći modeli, modelovani u PowerDesigner-u i to: Klasni dijagram, Dijagrami sekvence, Stanja i Use-case dijagrami.

Korištena je MVC arhitektura, gdje je thin controller realizovan preko command šablona, dok su Model i View povezani observer šablonom. Primjenom ove arhitekture postiže se razdvajanje strukture podataka i osnovnih funkcionalnosti sistema od njegovog prikaza i interakcije. Thin controller sadrži osnovne funkcionalnosti sistema, dok se pomoću observer šablona postiže održavanje konzistentnisti prikaza podataka i modela putem notifikacija koje se implementiraju oslanjajući se na ovaj šablon.

# Arhitekturni ciljevi i ograničenja

[This section describes the software requirements and objectives that have some significant impact on the architecture; for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]

Zbog kompleksnosti sistema neophodno je razdvojiti odgovornosti sistema i zaduženja pojedinačnih klasa, kako bi se zadovoljili osnovni principi objektno orijentisanog dizajna. Primjenom gorenavedene arhitekture u kombinaciji sa navedenim šablonima postiže se traženo. Takođe, ovim je kreiran sistem koji je pogodan za dalje proširenje sa minimalnom cijenom, samim tim smanjuje se obim klasa u kojima se mogu pojaviti potencijalne greške i koje se moraju ponovo testirati, što uzrokuje uštedu vremena i povećanje kvaliteta softvera. Upotrebom određenih šablona smanjena je mogućnost dupliranja koda što direktno utiče na brzinu softvera i smanjenje upotrijebljenog memorijskog prostora.

# Use-case pogled

[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements or if they stress or illustrate a specific, delicate point of the architecture.]

Dodavanje objekta (scenario)

Kreiranje novog modela (scenario)

Otvaranje dijagrama (scenario)

## Use-case realizacije

[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations, and explains how the various design model elements contribute to their functionality.]

# Logički pogled

[This section describes the architecturally significant parts of the design model, such as its decomposition into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

Implementacija softvera je razdvojena na programske pakete koji su odgovarajuće povezani. Svaki paket se statoji od odgovarajućeg broja klasa čija saradnja realizuje slučajeve korištenja. Izvršena je dekompozicija u četiri različita paketa, koji sadrže arhitektonski značajne klase.

## Pregled

[This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.]

* Paket *model* cjelokupno predstavlja dio sistema koji sadrži klase odgovorne za stanje samog sistema i njihovu prezentaciju u operativnoj memoriji računara.
* Paket *controller* odgovoran za obradu akcija korisnika i događaja sa ulaznih jedinica, koje preslikava u komandu ili niz komandi koje izvršavaju operacije modela i pogleda u svrhu odgovarajuće primjene.
* Paket *observer*
* Paket *view* sadrži klase koje su odgovorne za prezentaciju sistema i prikaz podataka korisniku koji se kreiraju na osnovu informacija sadržanih u modelu.

## Arhitekturno značajni dizajn paketi

[For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

For each significant class in the package, include its name, brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.]

* Paket *model* sačinjen je od podpaketa *appmodel* i *datamodel.* Podpaket *appmodel* sadrži centralnu klasu ApplicationModel pomoću koje se predstavlja osnova i fundamentalne funkcionalnosti aplikacije. Nadovezujući se na centralnu klasu pomoću šablona State realizovana su moguća stanja aplikacije, šablon Strategy upotrijebljen je za predstavljanje mogućnosti importa i exporta. Pomoću apstraktne fabrike predstavljeni su dostupni tipovi modela. Podpaket *datamodel* sarži sve podatke o workspace komponentama, uključujući projekat, modele, dijagrame i objekte, gdje su dijelovi predstavljeni uz pomoć State i Strategy šablona. Objekti su dodatno predstavljeni u podpaketu *objects* gdje se navode ključni objekti; elementi i linkovi i njihove varijacije. Za ovaj podpaket karakteristična je upotreba šablona Decorator.
* Paket *controller* sačinjen je od podpaketa *executablecommand* i *undoablecommand.* U oba podpaketa iskorišten je šablon Command za predstavljanje pojedinačnih operacija koje su dostupne korisniku za izvršenje.

# Pogled na implementaciju

[This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.]

Implementaciona arhitektura odgovara dizajnu pri čemu se MVC dizajn šablon razlaže u odgovarajuće pakete koji odgovaraju pojedinim funkcijama i koje sadrže odgovarajuće klase potrebne za realizaciju tih funkcionalnosti.

## Overview

[This subsection names and defines the various layers and their contents, the rules that govern the inclusion to a given layer, and the boundaries between layers. Include a component diagram that shows the relations between layers. ]

* Sloj podataka sadrži informacije o podacima i osnovnim dijelovima softvera koji su u operativnoj memoriji računara. Implementacija ovog sloja vrši se pomoću klasa smještenih u programskom paketu *model*.
* Prezentacioni sloj zadužen je za grafičku reprezentaciju podataka. Implementacija ovog sloja izvrašava se paketom *view.* Ovaj sloj ima neposrednu svijest o sloju podataka, jer koristi njegove resusrse u procesu grafičkog predstavljanja.
* Paket *observer* sadrži obrazac ponašanja. Observer obrascom se definišu načini pretplate za obavještavanje više objekata o nastalim događajima na objektu koji se posmatra, pri čemu se prate promjene na dijagramu i aplikacionom modelu o kojima se obavještava prezentacioni sloj.

## Layers

[For each layer, include a subsection with its name, an enumeration of the subsystems located in the layer, and a component diagram.]

# Veličina i performanse

[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

Izabrana softverska arhitektura podržava ključne zahtjeve po pitanju veličine i performansi, kao što je navedeno u dokumentu *Dopunska specifikacija.* Veličina softvera je relativno mala, sa ukupno oko 100 klasa koje uz primjenu odgovarajućih šablona rezultuju dobrim karakteristikama u pogledu sikorištenja memorijskog prostora

# Kvalitet

[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.]

Arhitektura softvera podržava zahtjeve kvaliteta navedene u dokumentu *Dopunska specifikacija.*