Stephan Valentan

Design patterns in practice: A case study

Bachelor's Thesis - Exposeé

Graz University of Technology

Knowledge Technologies Institute
Head: Stefanie Univ.-Prof. Dr. Stefanie Lindstaedt

Supervisor: Dipl.-Ing. Ass. Prof. Roman Kern

Graz, July 2017

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Contents

Abstract and Exposeé v

1 Introduction 1

2 Background 3
- 2.1 Fundamentals of Object-Oriented programming 3
  - 2.1.1 Single Responsibility Principle 3
  - 2.1.2 Open Closed Principle 3
- 2.2 Design Patterns 3
  - 2.2.1 Unit of Work 3
  - 2.2.2 Repository 3
  - 2.2.3 Model-View-Controller 3
  - 2.2.4 Model-View-Presenter 3
- 2.3 Architecture Patterns 3
  - 2.3.1 Multi-Layer Architecture 3
  - 2.3.2 Multi-Tier Architecture 3

3 Methods 5
- 3.1 Requirements of application 5
  - 3.1.1 Classes 5
- 3.2 Phase 1: One Databse and UI 5
  - 3.2.1 Monolithic Code Samples 5
  - 3.2.2 Best-Practice Code Samples 5
- 3.3 Phase 2: More Databases / UIs 5
  - 3.3.1 Monolithic Code Samples 5
  - 3.3.2 Best-Practice Code Samples 5

4 Results 7
- 4.1 Lines of Code 7

4.2 Programming Effort 7
4.3 Efficiency and Performance 7
4.4 Touched and Edited Files 7
4.5 Readability 7

5 Discussion **9**
5.1 Programming Overhead 9
5.2 Use of OOP in General 9

6 Conclusion **11**

7 Future Work **13**
7.1 Physical Seperation of Tiers 13

Abstract and Exposeé

There are many many different approaches to achieve good software design, however often it is not clear whether the benefits of the best practices overweight the additional effort of implementing them. Little research is done on comparisons between the designs in terms of efficiency and programming effort. Many books do show sample code of their proposed designs, however they do not cover how the code would look like if implemented in a more straight-forward and monolithic way.

This bachelor's thesis tries to cover this topic by implementing the same application two times: as a monolithic program and as a program that pays attention to the information provided about software design and architecture. In the first step, the program will use only one data source and one graphical interface (GUI). In the second step in each program another data source and GUI was added.

The sourcecode will be available on Github [1] under GPLv3 [2]. The application will be written in Java using Swing as a front-end and PostgreSQL as database. In phase two JavaFX should be supported as a front-end, XML should be supported as a database. The user will be able to choose between all options using parameters when starting the application.

[1] https://github.com/Vallant/design-patterns-study.git
[2] https://www.gnu.org/licenses/gpl-3.0.de.html

17.03.2017	Requirements, Complexity and Basic Design of Application
31.03.2017	Implementation of Best-Practice Approach Phase 1
14.04.2017	Implementation of Monolithic Approach Phase 1
21.04.2017	Summary of Findings Phase 1
12.05.2017	Implementation of Best-Practice Approach Phase 2
02.06.2017	Implementation of Monolithic Approach Phase 2
09.06.2017	Summary of Findings Phase 2
30.06.2016	Thesis finished

Table 1: Basic Timetable for Bachelor's thesis

1 Introduction

2 Background

2.1 Fundamentals of Object-Oriented programming

2.1.1 Single Responsibility Principle

2.1.2 Open Closed Principle

2.2 Design Patterns

2.2.1 Unit of Work

2.2.2 Repository

2.2.3 Model-View-Controller

2.2.4 Model-View-Presenter

2.3 Architecture Patterns

2.3.1 Multi-Layer Architecture

2.3.2 Multi-Tier Architecture

3 Methods

3.1 Requirements of application

3.1.1 Classes

3.2 Phase 1: One Databse and UI

3.2.1 Monolithic Code Samples

3.2.2 Best-Practice Code Samples

3.3 Phase 2: More Databases / UIs

3.3.1 Monolithic Code Samples

3.3.2 Best-Practice Code Samples

4 Results

4.1 Lines of Code

4.2 Programming Effort

4.3 Efficiency and Performance

4.4 Touched and Edited Files

4.5 Readability

5 Discussion

5.1 Programming Overhead

5.2 Use of OOP in General

6 Conclusion

7 Future Work

7.1 Physical Seperation of Tiers

Gof94 Hfd04 Aspect06 Composing06