Coding an Architecture Style

A practical guide to learn **Software Architecture** by coding in Java

Learn fundamental concepts of Software Architecture by coding a **Layered**, **Hexagonal**, **Modular** and **Microservice** Architecture Style

+ A pragmatic study of Monolith and Distributed Structures, Architectural Views and the Availability and Scalability Quality Attributes

Enrique Pablo Molinari

Coding an Architecture Style

A practical guide to learn Software Architecture by coding in Java

Enrique Pablo Molinari

This book is for sale at http://leanpub.com/codinganarchitecturestyle

This version was published on 2020-12-06

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

1. Feedback

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

I Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

2. Introduction

In 1986 Frederick Brooks wrote "No Silver Bullet"[1] where he explains that there are no solutions, silver bullets, to deal with the essential difficulties of software development such as complexity. He described software projects as werewolves, because at the beginning they look familiar but eventually they become into something horrible. But as he describes, for werewolves you have silver bullets, but unfortunately solutions like that do not exist for software projects. However, he mentions **great designers** as one of the promising solutions. He said, "the very best designers produce structures that are faster, smaller, simpler, cleaner, and produced with less effort". Today, 34 years from that paper, building **simple** and **clean** software structures is still a hard goal to achieve for software developers.

Over the years, different terms became popular, reflecting the frustration generated by software developers with the tedious work of updating and maintaining code as new requirements are added over time. In 1997 the expression "Big Ball of Mud"[2] became popular and even earlier the term "spaghetti-code" was used. Both referring to software systems without a clear definition of their structure and where all the elements are connected to each other, without an order or clear rules. This obviously makes it extremely difficult to modify them.

In 1999 a concept emerged that would define these insights: software entropy[3]. Meaning that software systems, after several years of maintenance and new added requirements, become something very complex to change. As a general rule, when entropy reaches the software system, that is when frustrated programmers begin to evaluate other job offers, since making changes there is so complicated that it involves allocating overtime, outside of working hours, neglecting other aspects of everyday life. On the other hand, in cases where there is a budget the company decides to start in parallel the re-implementation of the same system with "best practices" or modern technology and thus begins a new iteration towards the (almost always) inevitable entropy.

Years go by, new terms emerge, but the problems and complications remain the same. How to structure a software system in such a way that it allows us to modify it without great complications after several years?

Throughout this book, different practices, concepts, and implementations that seek to answer this question will be presented.

I believe that producing a software system that does not become complex to modify over time is a highly sought after goal (almost unattainable) for every software programmer, designer or architect. Much of what is exposed in this book is dedicated especially to trying to prevent software entropy from winning over us.

Another Book about Software Architecture?

In 1994 David Garlan and Mary Shaw[4] wrote a technical report on the importance of software architecture as an emerging discipline. They said:

"As the size and complexity of software systems increases, the design problem goes beyond the algorithms and data structures of the computation: designing and specifying the overall system structure emerges as a new kind of problem."

Since that technical report to today dozens of books on software architecture have been published. However, most of them lack code examples and implementation. It is usual to find in these books the explanation of the layered architecture style with a beautiful diagram, but no mention is made of how to represent the layers in source code, that is, how they are implemented. From my point of view, this is not a minor omission. I think it is necessary to clearly express and exemplify the set of syntactical constructions that can be used to define a layer and its limits. Today, mainstream programming languages that belong to the same paradigm have a very similar set of syntactical constructions that we can use, with which it is possible to explain these concepts using any language and it will be applicable in an analogous way to any other.

What Will I Learn?

When Professor Garlan and Professor Shaw talk about the difficulties of designing and specifying the general structure of a software system, they mention protocols for communication, synchronization, and data access. Assignment of functionality to design elements, physical distribution, composition of design elements, scaling and performance. This is the next level of design: the **software architecture**.

In line with this definition, the following image shows four diagrams of the *same application* with the intention of visualizing that next level of design, the architecture:

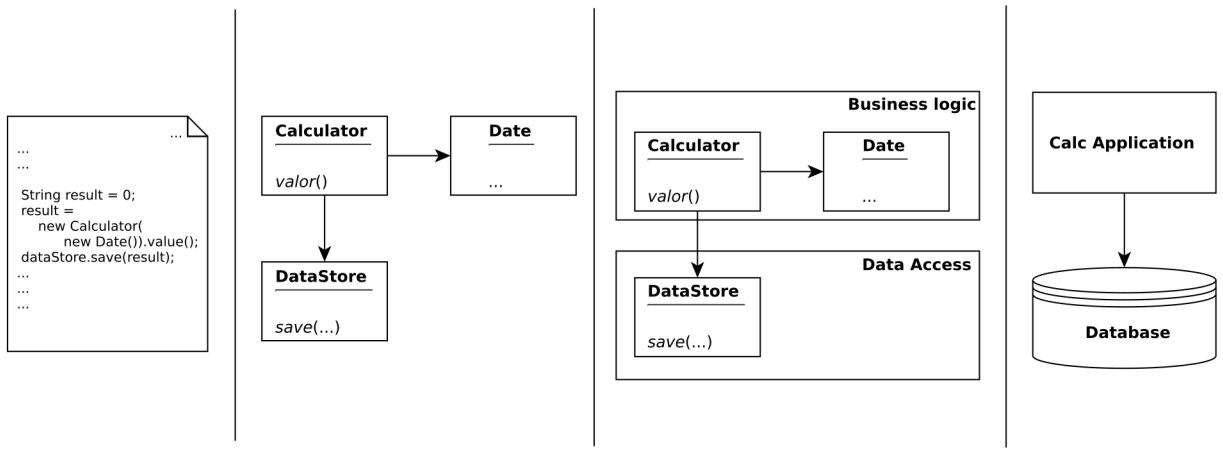

Figure 1.1: Code, Design and Architecture

The first level is clear, software is written using programming languages with specific syntax and semantics, within a specific paradigm. In the next level of abstraction, an object-oriented design is observed, expressed in a class diagram. Here again, no doubt arises, we are talking about design. The third level of abstraction is where some questions begin to appear: is this architecture, or is it design? What exactly does this application diagram describe? Can we have a syntactical construction to map the *Business Logic* and *Data Access* boxes into source code? For now we say that it represents the layered structure of the application, where classes are grouped in each layer according to whether they are related to business logic or data access. Where in addition, the classes of the business logic layer depend on some classes of the data access layer, and not the other way around. In the fourth and last diagram it is possible to observe a higher level of abstraction than the previous one. In this case, *Calc Application* represents the application at *runtime* and *Database* represents the persistent storage at *runtime* too. The arrow indicates an inter-process communication between them.

This book is dedicated to understand, describe, and implement software structures that belong to the last two levels of abstraction illustrated in figure 1.1.

In addition, in this book we will answer these questions:

What is software architecture? Where does design end and architecture begin? How to draw the architecture of a system? How to design a system with clean architecture concepts? What are layers? What about modules? What is physical and logical separation? Am I really implementing a layered architecture? How can I verify if the architecture rules are accomplished? Are monolithic architectures a bad practice? How can I build a modular architecture? How can I build a Microservice architecture? What is high availability? Single point of failure? How do you scale an application to better support growing demand?

This book does not include detailed discussions of all the different styles and patterns of architecture

that exists. It is dedicated to studying and **implementing** in the Java language those styles that I consider most used in enterprise applications. On the other hand, once the exposed concepts are understood, the reader will be able to handle without inconvenience any other style of architecture not included in this book.

Who Is This Book For?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

3. Software Architecture: Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

What is Software Architecture?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Visualizing Software Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

4+1 View Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Structures and Views (SEI)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

UML Diagrams

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

C4 Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Software Architecture Styles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Design vs Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

II Fundamentals and Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

4. Modules, Layers and their Architecture Styles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Namespaces, Packages, Layers and Modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Logical vs Physical Separation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Java Module System

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Horizontal vs Vertical Partitions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Why Do We Split an Application?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Modules vs Layers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Layered Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Definitions and Style

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Hexagonal Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Definitions and Style

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Inverted Layered Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Style and Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Modular Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Definitions and Style

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

5. Monolithic and Distributed Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Monolithic Architectures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Definitions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Modular Monolith

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Distributed Architectures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Reasons to Distribute

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Partial Failures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Transactions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Distributed Monolith

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Microservices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Definitions and Style

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Some Comments about the Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Modular Architecture vs Microservices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

III Quality Attributes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

6. Availability

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Calculating the Availability

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Other Issues Affecting Availability

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Availability in Microservices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

7. Scalability

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Scaling Web Applications

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Scaling Databases

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

CAP and PACELC Theorems

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Scaling Reads

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Sharding: Scaling Writes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/codinganarchitecturestyle.

Notes

1 No Silver Bullet, Essence and Accident in Software Engineering. Frederick P. Brooks. 1986.

2 Big Ball of Mud. Brian Foote and Joseph Yoder, Fourth Conference on Patterns Languages of Programs Monticello, Illinois, September 1997.

3 The Pragmatic Programmer: From Journeyman to Master. Andrew Hunt, David Thomas. 1999.

4 An Introduction to Software Architecture. David Garlan, Mary Shaw. CMU Software Engineering Institute Technical Report. 1994.