

Architectural Patterns and Styles - Notes

Priscilla Madigoe

08 March 2016

1 Architectural Patterns

1.1 Introduction

Often times source code needs to be organised so that clear roles, responsibilities and relationships of different modules of the code can be well defined. Architectural patterns help in making this possible and numerous types will be discussed in this section. Using these patterns, code becomes easier to maintain, manage and visualise. They also make understanding of how each component works in a system easier. They are reusable solutions to a commonly occurring problem in Software Architecture within a given context.

1.2 Types of Architectural Patterns

- Layered Pattern
- Client-server
- Microkernel
- Master-slave
- Pipe-filter
- Broker Pattern
- Model View Controller
- Blackboard Pattern

1.3 Architectural Patterns Proposed for PAPERS

1.3.1 Model View Controller

- Background

In the Model-View-Controller Pattern, an interactive application is divided into three parts: the Model is an object representing the data and activities, the View displays information to the user and the Controller offers a way to change the state of the Model.

- Motivation

Any form of application that uses a request-and-response logic needs to be decoupled. Separation of responsibilities is useful because functions can work normally when certain subsystems are changed. The Model View Controller makes this a possibility and hence it is chosen for use in this context.

- Components of the Model View Controller Pattern

1. Model

The Model consists of the Database that will be used to store all the relevant information used by the system. For this particular system, Django will be used to handle the Database. For the Android Application, an appropriate API that will implement REST or Representational State Transfer will be used to communicate with the main Model (Database).

2. View

- (a) Web Application

For the Web Application, the browser will be the View because it will display the HTML (with or without CSS and JQuery) that the user will be able to use to interact with the application logic. The Controller will send data to the viewer so that the user can see it. In addition, Django provides a way for data to be sent to the View.

- (b) Android Application

For the Android Application, the Android Graphic User Interface, or GUI will be the View and appropriate APIs, such

as REST, will be used that will enable this application to manipulate the Model via the Controller.

3. Controller

The Django Framework itself will be the Controller as it will provide appropriate functionality for data manipulation.

1.3.2 Layered Pattern

- Background

The Layered Pattern assists in structuring applications that can be decomposed into subtasks. Each subtask provides services to the layer above it and services in that layer are implemented using those from the layer below it.

- Motivation

A system typically consists of subsystems that have a certain functionality. These functions need to be placed in layers so that their services can be used by other layers and a change in behaviour of one layer does not affect the layers below it. This assists in developing different layers independently.

- Components of the Layered System

This pattern will be used to separate the internal functions within each subsystem.

1.3.3 Concurrency - Master Slave

- Background

The Master-slave Pattern can be described as a model where one process (Master) has unidirectional control over one or more processes that are known as Slaves.

- Motivation

An interactive application typically consist of multiple users that use it at different rates. These users make requests that usually expect a

response to either use the data or edit it in some way and thus is imperative for the application have an ability to collect user information simultaneously.

1.4 Patterns Not Considered

1.4.1 Client Server Pattern

- Client Application provides access to humans, or systems (Fritz, 2012)
- Requests are typically handled in separate threads on the server.
- Inter-process communication causes overhead.
- Requests and data have to be transformed because they have different representation in Server and Client, and there is network traffic.

1.4.2 Microkernel

- Used for applications that must be adapted to changing system requirements.
- This pattern provides minimal functionality (Internal Servers) as a building block and customizable components that can be adapted accordingly (External Servers).
- It provides an Adapter that provides a single access point (Fritz, 2012)
- It has a more complex process management if not provided by architecture (Fritz, 2012)