Costa Rica Institute of Technology

Computer Engineering

Software Specification and Design

Assignment 6

Student:
Malcolm Davis

Professor:
Daniel Madriz

December 21, 2017

Abstract

The first chapter of the Patterns of Enterprise Application Architecture book is about the layering technique to divide complex software into more maintainable parts. This abstract define, list some common uses and applications of this technique.

What is Layering?

Layering is a technique to divide a complex software system. A layered system can be visualized as a stack books, each book have his propose in that stack, handles something and cannot communicate or see any other book that is not atop or beneath it. Some of its benefits are that you can understand whats on a book(layer) without having to know whats on the others. You can exchange the books if they have the same content(the implementation can be changed if it does the same). You can reuse the books to make another stack. But as all the architectures there are some drawbacks, not all things can be well encapsulated. Extra layers can harm the performance since overhead is added.

Typical Layers

There are three principal layers that are the ones the ones that normally every system at least has. This layers are presentation, data source and domain logic. The **presentation** is where the code designated to make the interface between the user and the software. This can be a simple interface as it is a console that expects orders or a advance user interface. This layer must display the information and get the user input. The **domain logic**, this is also called business logic. In this layer the calculations with the data are made, validation from presentation layer's data. The **data source** logic is the layer that communicates the system with other systems that makes task on behalf of the system, sometimes this communications are with databases to store data.
This are the most common layers, but there are another layers that can be implemented, as data mapping, controller, client, integration, etc... And you as a software architect have to choose when separate layers and when is not possible/required or if is unnecessarily harming the system performance.