III

Architecture

The architecture of a software system is the shape given to that system by those who build it. The form of that shape is in the division of that system into components, the arrangement of those components, and the ways in which those components communicate with each other.

The purpose of that shape is to facilitate the development, deployment, operation, and maintenance of the software system contained within it.

The strategy behind that facilitation is to leave as many options open as possible, for as long as possible.

Perhaps this statement has surprised you. Perhaps you thought that the goal of software architecture was to make the system work properly. Certainly we want the system to work properly; and certainly the architecture of the system must support that as one of its highest priorities.

However, the architecture of a system has very little bearing on whether that system works. There are many systems out there, with terrible architectures, that work just fine. Their troubles are not in their operation; they are in deployment, maintenance, and ongoing development.

This is not to say that architecture plays no role in supporting the proper behavior of the system. It certainly does, and the role is critical. But the role is passive and cosmetic, not active or essential. There are few, if any, behavioral options that the architecture of a system can leave open.

The primary purpose of architecture is to support the lifecycle of the system. Good architecture makes the system easy to understand, easy to develop, easy to maintain, and easy to deploy. The ultimate goal is to minimize the lifetime cost of the system and maximize programmer productivity.

Note: This chapter is an overview. For a more complete treatment of the topic, see [Clean Arch].