Software Architecture in Practice
(Second Edition)

Len Bass
Paul Clements
Rick Kazman

Part One – Envisioning the Architecture

Where do architectures come from? They spring from the minds of the architects, of course, but how?

The architect is faced with a swarm of competing, if not conflicting, influences and demands, surprisingly few of which are concerned with getting the system to work correctly. The organizational and technical environment brings to bear a weighty set of sometimes implicit demands, and in practice these are as important as any of the explicit requirements for the software even though they are almost never written down.

Chapter 1 – The Architecture Business Cycle

Requirements beget design, which begets system. But they still make the implicit assumption that design is a product of the system's technical requirements period.

The architectural view of a system is abstract, distilling away details of implementation, algorithm, and data representation and concentrating on the behavior and interaction of "black box" elements. A software architecture is developed as the first step toward designing a system that has a collection of desired properties.

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationship among them.

The focusing question is this: What is the relationship of a system's software architecture to the environment in which the system will be constructed and exist? The answer to this question is the organizing motif of this book. Software architecture is a result of technical, business and social influences. Its existence in turn affects the technical, business and social environments that subsequently influence future architecture. We call this cycle of influences, from the environment to the architecture and back to the environment, the Architecture Business Cycle (ABC).

Where do architectures come from?

In any development effort, the requirements make explicit some – but only some – of the desired properties of the final system. Not all requirements are concerned directly with those properties; a development process or the use of a particular tool may be mandated by them. But the requirements specification only begins to tell the story. Failure to satisfy other constraints may render the system just as problematic as if it functioned poorly.

- Architectures are influenced by system stakeholders
- Architectures are influenced by the developing organization
- Architectures are influenced by the background and experience of the architects
- Architectures are influenced by the technical environment
- The architectures affect the factors that influence them

Software Process and the Architecture Business Cycle

Software process in the term given to the organization, ritualization, and management of software development activities.

Architecture Activities

- Creating Business Case for the System
- Understanding the Requirements: Regardless of the technique used to elicit the requirements, the desired qualities of the system to be constructed determine the shape of its architecture.
- Creating or Selecting the Architecture
- Communicating the Architecture: For the architecture to be effective as the backbone of the project's design, it must be communicated clearly and unambiguously to all of the stakeholders.
- Analyzing or Evaluating the Architecture
- Implementing Based on the Architecture
- Ensuring Conformance to an Architecture

What makes a "Good" Architecture?

- The architecture should be the product of a single architect or a small group of architects with an indentified leader.
- The architect should have the functional requirements for the system and an articulated, prioritized list of quality attributes (such as security or modifiability) that the architecture is expected to satisfy.
- The Architecture should be well documented, with at least one static view and one dynamic view, using an agreed-on notation that all stakeholders can understand with a minimum of effort.
- The Architecture should be circulated to the system's stakeholders, who should be actively involved with its review.
- The architecture should be analyzed for the applicable quantitative measures (such as maximum throughput) and formally evaluated for quality attributes before It is too late to make changes to it.
- The architecture should lend itself to incremental implementation via the creation of a "skeletal" system which the communication paths are exercised but which at first has minimal functionality. This skeletal system can then be used to "grow" the system incrementally, easing the integration and testing efforts.
- The architecture should result in a specific (and small) set of resource contention areas, the resolution of which is clearly specified circulated and maintained.

- The architecture should feature well-defined modules whose functional responsibilities are allocated on the principles of information hiding and separation of concerns.
- Each module should have a well-defined interface that encapsulate or “hides” changeable aspects (such as implementations strategies and data structures choices) from other software that uses it facilities.
- Quality attributes should be achieved using well-known architectural tactics specific to each attribute.
- The architecture should never depend on a particular version of a commercial product or tool.
- For parallel-processing system, the architecture should feature well-defined processes or tasks that do not necessarily mirror the module decomposition structure.
- Every task or process should be written so that its assignment to a specific processor can be easily changed, perhaps even at runtime,
- The architecture should feature a small number of simple interactions patterns. That is, the system should do the same things in the same way throughout.