阅读笔记

# CH1 What is Software Architecture

## 1.1 What Software Architecture Is and What It Isn't

**definition：**The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both.

### Architecture Is a Set of Software Structures

**three categories of architectural structures：**

·First, some structures partition systems into implementation units, which in this book we call modules.

·Other structures are dynamic, meaning that they focus on the way the elements interact with each other at runtime to carry out the system’s functions.

·A third kind of structure describes the mapping from software structures to the system’s organizational, developmental, installation, and execution environments.

### Architecture Is an Abstraction

Because architecture consists of structures and structures consist of elements and relations, it follows that an architecture comprises software elements and how the elements relate to each other.

### Every Software System Has a Software Architecture

Every system can be shown to comprise elements and relations among them to support some type of reasoning. In the most trivial case, a system is itself a single element—an uninteresting and probably non-useful architecture, but an architecture nevertheless.

### Architecture Includes Behavior

The behavior of each element is part of the architecture insofar as that behavior can be used to reason about the system. This behavior embodies how elements interact with each other, which is clearly part of our definition of architecture.

### Not All Architectures Are Good Architectures

The definition is indifferent as to whether the architecture for a system is a good one or a bad one. An architecture may permit or preclude a system’s achievement of its behavioral, quality attribute, and life-cycle requirements.

## 1.2 Architectural Structures and Views

### Structures and Views

·A view is a representation of a coherent set of architectural elements, as written by and read by system stakeholders. It consists of a representation of a set of elements and the relations among them.

·A structure is the set of elements itself, as they exist in software or hardware.

In short, a view is a representation of a structure.

Architects design structures. They document views of those structures.

### Three Kinds of Structures

**three broad kinds of decisions that architectural design involves:**

·**Module structures** embody decisions as to how the system is to be structured as a set of code or data units that have to be constructed or procured.

·**Component-and-connector structures** embody decisions as to how the system is to be structured as a set of elements that have runtime behavior (components) and interactions (connectors).

·**Allocation structures** embody decisions as to how the system will relate to nonsoftware structures in its environment (such as CPUs, file systems, networks, development teams, etc.).

### Structures Provide Insight

### Some Useful Module Structures

Useful module structures include the following:

·**Decomposition structure**

·**Uses structure**

·**Layer structure**

·**Class (or generalization) structure**

·**Data model**

### Some Useful C&C Structures

Useful C&C structures include the following:

·**Service structure**

·**Concurrency structure**

### Some Useful Allocation Structures

Useful allocation structures include these:

**·Deployment structure**

**·Implementation structure**

**·Work assignment structure**

### Relating Structures to Each Other

### Fewer Is Better

### Which Structures to Choose?

## 1.3 Architectural Patterns

**common module type pattern：**Layered pattern

**Common component-and-connector type patterns：**

·Shared-data (or repository) pattern

·Client-server pattern

**Common allocation patterns：**

·Multi-tier pattern

·Competence center and platform

## 1.4 What Makes a "Good" Architecture

**process recommendations：**

·The architecture should be the product of a single architect or a small group of architects with an identified technical leader.

·The architect (or architecture team) should, on an ongoing basis, base the architecture on a prioritized list of well-specified quality attribute requirements.

·The architecture should be documented using views.

·The architecture should be evaluated for its ability to deliver the system’s important quality attributes.

·The architecture should lend itself to incremental implementation, to avoid having to integrate everything at once (which almost never works) as well as to discover problems early

**structural rules of thumb：**

·The architecture should feature well-defined modules whose functional responsibilities are assigned on the principles of information hiding and separation of concerns.

·Unless your requirements are unprecedented—possible, but unlikely—your quality attributes should be achieved using well-known architectural patterns and tactics specific to each attribute.

·The architecture should never depend on a particular version of a commercial product or tool.

·Modules that produce data should be separate from modules that consume data.

·Don’t expect a one-to-one correspondence between modules and components.

·Every 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 ways for components to interact.

·The architecture should contain a specific (and small) set of resource contention areas, the resolution of which is clearly specified and maintained.

# CH2 Why Is Software Architecture Important

## 2.1 Inhibiting or Enabling a System's Quality Attributes

## 2.2 Reasoning About and Managing Change

Every architecture partitions possible changes into three categories: local, nonlocal, and architectural.

·A local change can be accomplished by modifying a single element.

·A nonlocal change requires multiple element modifications but leaves the underlying architectural approach intact.

·An architectural change affects the fundamental ways in which the elements interact with each other and will probably require changes all over the system.

## 2.3 Predicting System Qualities

## 2.4 Enhancing Communication among Stakeholders

## 2.5 Carrying Early Design Decisions

## 2.14 Summary

Software architecture is important for a wide variety of technical and nontechnical reasons. Our list includes the following:

1. An architecture will inhibit or enable a system’s driving quality attributes.

2. The decisions made in an architecture allow you to reason about and manage change as the system evolves.

3. The analysis of an architecture enables early prediction of a system’s qualities.

4. A documented architecture enhances communication among stakeholders.

5. The architecture is a carrier of the earliest and hence most fundamental, hardest-to-change design decisions.

6. An architecture defines a set of constraints on subsequent implementation.

7. The architecture dictates the structure of an organization, or vice versa.

8. An architecture can provide the basis for evolutionary prototyping.

9. An architecture is the key artifact that allows the architect and project manager to reason about cost and schedule.

10. An architecture can be created as a transferable, reusable model that forms the heart of a product line.

11. Architecture-based development focuses attention on the assembly of components, rather than simply on their creation.

12. An architecture channels the creativity of developers, reducing design and system complexity.

13. An architecture can be the foundation for training of a new team member

# CH3 Software Architecture

There are four dominant software development processes, which we describe in roughly the order in which they came to prominence:

**Waterfall**

**Iterative**

**Agile**

**Model-driven development**

The process you use will determine how often and when you revisit and elaborate each of these activities. These activities include:

1. Making a business case for the system

2. Understanding the architecturally significant requirements

3. Creating or selecting the architecture

4. Documenting and communicating the architecture

5. Analyzing or evaluating the architecture

6. Implementing and testing the system based on the architecture

7. Ensuring that the implementation conforms to the architecture

Architectures exist in four different contexts:

**Technical**

**Project life cycle**

**Business**

**Professional**