Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

Software engineering practice is a broad array of principles, concepts, methods, and tools that you must consider as software is planned and developed. Principles that guide practice establish a foundation from which software engineering is conducted.

The software process provides everyone involved in the creation of a computer-based system or product with a road map for getting to a successful destination. Practice provides you with the detail you'll need to drive along the road. It tells you where the bridges, the roadblocks, and the forks are located. It helps you understand the concepts and principles that must be understood and followed to drive safely and rapidly. It instructs you on how to drive, where to slow down, and where to speed up. In the context of software engineering, practice is what you do day in and day out as software evolves from an idea to a reality.

In this chapter, our focus is on principles and concepts that guide software engineering practice in general.

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

7.1 SOFTWARE ENGINEERING KNOWLEDGE

You often hear people say that software development knowledge has a 3-year half life:half of what you need to know today will be obsolete within 3 years. In the domain of technology-related knowledge, that's probably about right. But there is another kind of software development knowledge—a kind that I think of as "software engineering principles"—that does not have a three-year half-life. These software engineering principles are likely to serve a professional programmer throughout his or her career.

7.2 CORE PRINCIPLES

At the process level, core principles establish a philosophical foundation that guides a software team as it performs framework and umbrella activities, navigates the process flow, and produces a set of software engineering work products.

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

At the level of practice, core principles establish a collection of values and rules that serve as a guide as you analyze a problem, design a solution, implement and test the solution, and ultimately deploy the software.

7.2.1 Principles That Guide Process

The following set of core principles can be applied to the framework, and by extension, to every software process:

① **Principle 1. Be agile.**

② **Principle 2. Focus on quality at every step.**

③ **Principle 3. Be ready to adapt.**

④ **Principle 4. Build an effective team.**

⑤ **Principle 5. Establish mechanisms for communication and coordination.**

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

⑥ Principle 6. Manage change.

⑦ Principle 7. Assess risk.

⑧ Principle 8. Create work products that provide value for others.

7.2.2 Principles That Guide Practice

Software engineering practice has a single overriding goal—to deliver on-time,high-quality, operational software that contains functions and features that meet the needs of all stakeholders. The following set of core principles are fundamental to the practice of software engineering:

① Principle 1. Divide and conquer.

② Principle 2. Understand the use of abstraction.

③ Principle 3. Strive for consistency.

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

④ Principle 4. Focus on the transfer of information.

⑤ Principle 5. Build software that exhibits effective modularity.

⑥ Principle 6. Look for patterns.

⑦ Principle 7. When possible, represent the problem and its solution from a number of different perspectives.

⑧ Principle 8. Remember that someone will maintain the software.

7.3 PRINCIPLES THAT GUIDE EACH FRAMEWORK ACTIVITY

7.3.1 Communication Principles

Before customer requirements can be analyzed, modeled, or specified they must be gathered through the communication activity.But the road from communication to understanding is often full of potholes.

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

Effective communication (among technical peers, with the customer and other stakeholders, and with project managers) is among the most challenging activities that you will confront. we discuss communication principles as they apply to customer communication. However, many of the principles apply equally to all forms of communication that occur within a software project:

① **Principle 1. Listen.**

② **Principle 2. Prepare before you communicate.**

③ **Principle 3. Someone should facilitate the activity.**

④ **Principle 4. Face-to-face communication is best.**

⑤ **Principle 5. Take notes and document decisions.**

⑥ **Principle 6. Strive for collaboration.**

⑦ **Principle 7. Stay focused; modularize your discussion.**

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

- ⑧ Principle 8. If something is unclear, draw a picture.
- ⑨ Principle 9. (a) Once you agree to something, move on. (b) If you can’t agree to something, move on. (c) If a feature or function is unclear and cannot be clarifified at the moment, move on.
- ⑩ Principle 10. Negotiation is not a contest or a game. It works best when both parties win.

7.3.2 Planning Principles

The planning activity encompasses a set of management and technical practices that enable the software team to define a road map as it travels toward its strategic goal and tactical objectives.On many projects, overplanning is time consuming and fruitless(too many things change), but underplanning is a recipe for chaos. Like most things in life, planning should be conducted in moderation,

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

enough to provide most things in life, planning should be conducted in moderation, enough to provide useful guidance for the team—no more, no less. Regardless of the rigor with which planning is conducted, the following principles always apply:

① Principle 1. Understand the scope of the project.

② Principle 2. Involve stakeholders in the planning activity.

③ Principle 3. Recognize that planning is iterative.

④ Principle 4. Estimate based on what you know.

⑤ Principle 5. Consider risk as you define the plan.

⑥ Principle 6. Be realistic.

⑦ Principle 7. Adjust granularity as you define the plan.

⑧ Principle 8. Define how you intend to ensure quality.

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

⑨ Principle 9. Describe how you intend to accommodate change.

⑩ Principle 10. Track the plan frequently and make adjustments as required.

To be most effective, everyone on the software team should participate in the planning activity. Only then will team members “sign up” to the plan.

7.3.3 Modeling Principles

In software engineering work, two classes of models can be created: requirements models and design models. Requirements models (also called analysis models) represent customer requirements by depicting the software in three different domains: the information domain, the functional domain, and the behavioral domain. Design models represent characteristics of the software that help

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

practitioners to construct it effectively: the architecture, the user interface, component-level detail and so on.

On agile modeling, Scott Ambler and Ron Jeffries define a set of modeling principles that are intended for those who use the agile process model but are appropriate for all software engineers who perform modeling action and tasks:

① Principle 1. The primary goal of the software team is to build software, not create models.

② Principle 2. Travel light—don’t create more models than you need.

③ Principle 3. Strive to produce the simplest model that will describe the problem or the software.

④ Principle 4. Build models in a way that makes them amenable to change.

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

- ⑤ **Principle 5. Be able to state an explicit purpose for each model that is created.**
- ⑥ **Principle 6. Adapt the models you develop to the system at hand.**
- ⑦ **Principle 7. Try to build useful models, but forget about building perfect models.**
- ⑧ **Principle 8. Don't become dogmatic about the syntax of the model. If it communicates content successfully, representation is secondary.**
- ⑨ **Principle 9. If your instincts tell you a model isn't right even though it seems okay on paper, you probably have reason to be concerned.**
- ⑩ **Principle 10. Get feedback as soon as you can.**

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

Requirements modeling principles. Over the past three decades, a large number of requirements modeling methods have been developed. Investigators have identified requirements analysis problems and their causes and have developed a variety of modeling notations and corresponding sets of heuristics to overcome them. Each analysis method has a unique point of view. However, all analysis methods are related by a set of operational principles:

① **Principle 1. The information domain of a problem must be represented and understood.**

② **Principle 2. The functions that the software performs must be defined.**

③ **Principle 3. The behavior of the software (as a consequence of external events) must be represented.**

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

④ Principle 4. The models that depict information, function, and behavior must be partitioned in a manner that uncovers detail in a layered (or hierarchical) fashion.

⑤ Principle 5. The analysis task should move from essential information toward implementation detail.

Design modeling principles. The software design model is the equivalent of an architect's plans for a house. It begins by representing the totality of the thing to be built (e.g., a three-dimensional rendering of the house) and slowly refines the thing to provide guidance for constructing each detail (e.g., the plumbing layout).Similarly, the design model that is created for software provides a variety of different views of the system. a set of design principles that can be applied regardless of the method that is used:

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

① **Principle 1. Design should be traceable to the requirements model.**

② **Principle 2. Always consider the architecture of the system to be built.**

③ **Principle 3. Design of data is as important as design of processing functions.**

④ **Principle 4. Interfaces (both internal and external) must be designed with care.**

⑤ **Principle 5. User interface design should be tuned to the needs of the end user. However, in every case, it should stress ease of use.**

⑥ **Principle 6. Component-level design should be functionally independent .**

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

⑦ Principle 7. Components should be loosely coupled to one another and to the external environment .

⑧ Principle 8. Design representations (models) should be easily understandable.

⑨ Principle 9. The design should be developed iteratively .

⑩ Principle 10. Creation of a design model does not preclude an agile approach

7.3.4 Construction Principles

The construction activity encompasses a set of coding and testing tasks that lead to operational software that is ready for delivery to the customer or end user.

The initial focus of testing is at the component level, often called

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

unit testing. Other levels of testing include (1) integration testing (conducted as the system is constructed), (2) validation testing that assesses whether requirements have been met for the complete system (or software increment), and (3) acceptance testing that is conducted by the customer in an effort to exercise all required eatures and functions.

Coding principles. The principles that guide the coding task are closely aligned with programming style, programming languages, and programming methods. However, there are a number of fundamental principles that can be stated:

Preparation Principles: Before you write one line of code, be sure you:

- Understand of the problem you're trying to solve.

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

- Understand the component-level design.
- Pick a programming language that meets the needs of the software to be built and the environment in which it will operate.
- Select a programming environment that provides tools that will make your work easier.
- Create a set of unit tests that will be applied once the component you code is completed.

Coding Principles: As you begin writing code, be sure you

- Constrain your algorithms by following structured programming practice.
- Consider the use of pair programming.
- Select data structures that will meet the needs of the design.

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

- Understand the software architecture and create interfaces that are consistent with it.
- Keep conditional logic as simple as possible.
- Create nested loops in a way that makes them easily testable.
- Select meaningful variable names and follow other local coding standards.
- Write code that is self-documenting.
- Create a visual layout (e.g., indentation and blank lines) that aids understanding.

Validation Principles: After you’ve completed your first coding pass, be sure you

- Conduct a code walkthrough when appropriate.

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

- **Perform unit tests and correct errors you've uncovered.**
- **Refactor the code.**

可以在以下面对代码进行重构：

1.重命名：对类，接口，方法，属性等重命名，以使得更易理解

2.抽取代码：将方法内的一段代码抽取为另一个方法，以使得该段代码可以被其他方法调用，这是重构中很重要很常用的，此举可以极大的精炼代码，减少方法的代码行数

3.封装字段：将类的某个字段转换成属性，可以更加合理的控制字段的访问(account里包含支局信息，这样应该进行拆分)

4.抽取接口：将类的某些属性，方法抽取组成个接口，该类自动实现该接口

5.提升方法内的局部变量为方法的参数：这主要是在写代码的过程中会使用到

6.删除参数：将方法的一个或多个参数删掉

7.重排参数：将方法的参数顺序重新排列

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

Testing principles. In a classic book on software testing, a number of rules that can serve well as testing objectives:

- **Testing is a process of executing a program with the intent of finding an error.**
- **A good test case is one that has a high probability of finding an as-yet undiscovered error.**
- **A successful test is one that uncovers an as-yet-undiscovered error.**

Everett and Meyer suggest additional principles:

① **Principle 1. All tests should be traceable to customer requirements.**

② **Principle 2. Tests should be planned long before testing begins.**

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

- ③ Principle 3. The Pareto principle applies to software testing.
- ④ Principle 4. Testing should begin "in the small" and progress toward testing "in the large."
- ⑤ Principle 5. Exhaustive testing is not possible.
- ⑥ Principle 6. Apply to each module in the system a testing effort commensurate with its expected fault density.
- ⑦ Principle 7. Static testing techniques can yield high results.
- ⑧ Principle 8. Track defects and look for patterns in defects uncovered by testing.
- ⑨ Principle 9. Include test cases that demonstrate software is behaving correctly.

7.3.5 Deployment Principles

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

the deployment activity encompasses three actions: delivery, support, and feedback. Because modern software process models are evolutionary or incremental in nature, deployment happens not once, but a number of times as software moves toward completion.

A number of key principles should be followed as the team prepares to deliver an increment:

① Principle 1. Customer expectations for the software must be managed.

② Principle 2. A complete delivery package should be assembled and tested.

③ Principle 3. A support regime must be established before the software is delivered.

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

④ Principle 4. Appropriate instructional materials must be provided to end users.

⑤ Principle 5. Buggy software should be fixed first, delivered later.

7.4 WORK PRACTICES

The human aspects of software engineering are as important as any other technology area. For that reason, it is interesting to examine the traits and work habits that seem to be shared among successful software engineers. Beyond basic traits and work habits, Isklod suggests 10 concepts that transcend programming languages and specific technologies. Some of these concepts form the prerequisite knowledge needed to appreciate the role of software engineering in the software process:

Chapter 7 PRINCIPLES THAT GUIDE PRACTICE

① Interfaces.

② Conventions and templates.

③ Layering.

④ Algorithmic complexity.

⑤ Hashing.

⑥ Caching.

⑦ Concurrency.

⑧ Cloud computing.

⑨ Relational databases.

A good software engineer must know what principles, practices, and tools to use, when to use them, and why they are needed.