Chapter 5
Practice: A Generic View

Software Engineering: A Practitioner's Approach, 6th edition
by Roger S. Pressman

本章要点

- 什么是实践
- 软件实践的本质
- 核心软件工程原则
- 软件工程实践指导原则

What is “Practice”?

- Practice consists of the
  - concepts
  - principles
  - methods
  - tools

that must be considered as software is planned and developed.

- It represents the details—the how-to’s—of the software process.

The Essence of Practice

<How to Solve it>

- George Polya, in a book written in 1945 (!), describes the essence of software engineering practice …
    - Understand the problem (communication and analysis).
    - Plan a solution (modeling and software design).
    - Carry out the plan (code generation).
    - Examine the result for accuracy (testing and quality assurance).
- At its core, good practice is common-sense[通常意义下的] problem solving

Core Software Engineering Principles

David Hooker [1996] presented seven SE principles:

1. Provide Value to Your Users[存在价值]
2. KIS (Keep It Simple)[保持简洁]
3. Maintain the Vision[维护视图]
4. What You Produce, Others Will Consume[生产者要让消费者理解]
5. Be Open to the Future[面向未来]
6. Plan Ahead for Reuse[计划复用]
7. Think before You Do[认真思考]

Software Engineering Practices

- Consider the generic process framework
  - Communication
  - Planning
  - Modeling
    - Analysis Modeling
    - Design Modeling
  - Construction
    - Coding
    - Testing
  - Deployment

Communication Practices[沟通实践]

- **Principles**
  1. Listen effectively[倾听]
  2. Prepare before you communicate[有准备的沟通]
  3. Someone should facilitate the activity[需要有人推动]
  4. Face-to-face communication is best[最好当面沟通]
  5. Take notes and document decisions[记录所有决定]
  6. Strive for collaboration[保持通力协作]
  7. Stay focused, modularize your discussion[聚焦并协调话题]
  8. If something is unclear, draw a picture[采用图形表示]
  9. Know when to move on[继续前进原则]
  10. Negotiation works best when both parties win[谈判双赢原则]

Planning Practices

- **Principles**
  1. Understand the project scope[理解项目范围]
  2. Involve the customer in planning[客户参与策划]
  3. Recognize that planning is iterative[采用迭代计划]
  4. Estimate based on what you know[基于已知估计]
  5. Consider risk as you define the plan[计划时考虑风险]
  6. Be realistic[脚踏实地]
  7. Adjust granularity as you define the plan[调整计划粒度]
  8. Define how you intend to ensure quality[制定计划确保质量]
  9. Describe how you intend to accommodate changes[描述如何适应变化]
  10. Track the plan frequently and make necessary adjustments[经常跟踪/校正计划]

Modeling Practices

- **Analysis modeling principles**
  1. Represent the information domain[必须描述并理解问题的信息域]
  2. Represent software functions[必须确定软件所要实现的功能]
  3. Represent software behavior[必须描述软件的行为]
  4. Partition these representations[分层表示]
  5. Move from essence toward implementation[分析任务应该从本质信息转向实现细节]

Modeling Practices…

- **Design Modeling Principles**
  1. Design must be traceable to the analysis model[设计可追溯到分析模型]
  2. Always consider architecture[经常关注待建系统的架构]
  3. Focus on the design of data[数据设计和功能设计同等重要]
  4. Interfaces (both user and internal) must be designed[必须设计接口]
  5. User interface should consider the user first[用户界面设计必须否和最终用户需求]
  6. Components should exhibit functional independence[功能独立的构件级设计]
  7. Components should be loosely coupled[构件之间以及构件与外部环境之间的松散耦合]
  8. Design representations should be easily understood[设计表述要容易理解]
  9. The design model should be developed iteratively[设计应该迭代式进行]

Construction Practices

- **Coding Principles**
- Preparation. Before you write one line of code, be sure you:
  1. Understand of the problem you're trying to solve.[理解所要解决的问题]
  2. Understand basic design principles and concepts.[理解基本的设计原则和概念]
  3. Pick an appropriate programming language.[选择合适的编程语言]
  4. Select an appropriate programming environment and tools.[选择合适的编程环境]
  5. Create unit tests for each component you plan to create.[构件级编码完成后的单元测试]

Construction Practices…

- **Coding Principles**
- Coding. As you begin writing code, be sure you:

1. Follow structured programming practice.[遵循结构化编程方法约束算法]
2. Select data structures that will meet the needs of the design.[选择能满足设计要求的数据结构]
3. Create interfaces consistent with the software architecture.[理解软件架构并开发出与其相符的接口]
4. Keep conditional logic as simple as possible.[尽可能保持条件逻辑简单]
5. Create nested loops in a way that makes them easily testable.[用易于测试的方法开发嵌套循环]
6. Select meaningful variable names and follow other local coding standards.[选择有意义的变量名并符合相关编码标准]
7. Write code that is self-documenting.[编码注释]
8. Create a visual layout that aids understanding.[增强代码的可读性]

Construction Practices…

- **Coding Principles**
- **Validation**. After you've completed the first pass, be sure you:
  1. Conduct a code walkthrough when appropriate.[适当进行代码走查]
  2. Perform unit tests and correct errors you've uncovered.[进行单元测试并改正所发现的错误]
  3. Refactor the code.[重构代码]

Construction Practices…

- **Testing Principles**
  1. All tests should be traceable to requirements[所有的测试都应该可以追溯到用户需求]
  2. Tests should be planned[测试计划应该远在测试开始前就开始着手]
  3. The Pareto Principle* applies to testing[将Pareto原则应用于软件测试]
  4. Testing begins “in the small” and moves toward “in the large”[测试应该从”微观”开始,逐步转向”宏观”]
  5. Exhaustive testing is not possible[穷举测试是不可能的]

- Note: A successful test is one that uncovers an as-yet-undiscovered error.

*Pareto Principle (帕雷多原则 80/20原则) :80%的软件Bug集中在20%的软件模块中，体现了重点核心模块的重点开发和维护。

Deployment Practices

- **Principles**
  1. Manage customer expectations for each increment[客户对于软件的期望必须得到管理]
  2. A complete delivery package should be assembled and tested[完整的交付包应该经过安装和测试]
  3. A support regime should be established[技术支持必须在软件交付之前就确定下来]
  4. Instructional materials must be provided to end-users[必须为最终用户提供适当的说明材料]
  5. Buggy software should be fixed first, delivered later[有缺陷的软件应该改正再交付]