Principles that guide practice

Core principles:

- **Principles that guide process:**
    1. Be agile.
    2. Focus on quality at every step.
    3. Be ready to adapt.
    4. Build an effective team.
    5. Establish mechanisms for communication and coordination. 建立沟通协调机制。
    6. Manage change.
    7. Assess risk. 评估风险。
    8. Create work products that provide value for others.
- **Principles that guide practice:**
    1. Divide and conquer. 分而治之。
    2. Understand the use of abstraction.
    3. Strive for consistency. 争取一致性。
    4. Focus on the transfer of information.
    5. Build software that exhibits effective modularity.
    6. Look for patterns.
    7. When possible, represent the problem and its solution from a number of different perspectives.
    8. Remember that someone will maintain the software.

Principles that guide each framework activities:

- **Communication principles:**
    1. Listen.
    2. Prepare before you communicate.
    3. Someone should facilitate（促使） the activity.
    4. Face-to-face communication is best.
    5. Take notes and document decisions.
    6. Strive for collaboration. 努力合作。
    7. Stay focus; modularize your discussion.
    8. If something is unclear, draw a picture.
    9. Once you agree to something, move on.

       If you can't agree to something, move on.

       If a feature or function is unclear and can not be clarified at the moment, move on.
    10. Negotiation（谈判） is not a contest or a game. It works best when both parites win.

- **Planning principles:**
  1. Understand the scope of the project.
  2. Involve stakeholders in the planning activity. 使利益相关者参与计划活动。
  3. Recognize that planning is iterative.
  4. Estimate（估计） based on what you know.
  5. Consider risk as you define the plan.
  6. Be realistic.
  7. Adjust granularity as you define the plan. 在定义计划时调整粒度。
  8. Define how you intend to ensure quality.
  9. Describe how you intend to accommodate change.(适应变化)
  10. Track the plan frequently and make adjustments as required.
- **Modeling principles:**

  requirements models & design models

  3 domains: the information domain, the functional domain, and the behavioral domain

  1. The primary goal of the software team is to build software, not create models.
  2. Travel light — don't create more models than you need.
  3. Strive to produce the simplest model that will describe the problem or the software.
  4. Build models in a way that makes them amenable to change.
  5. Be able to state an explicit（明确的） purpose for each model that is created.
  6. Adapt the models you develop to the system at hand.
  7. Try to build useful models, but forget about building perfect models.
  8. Don't become dogmatic（教条的） about the syntax of the model.
  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. Get feedback as soon as you can.

  - **Requirements modeling principles:**
    1. The information domain of a problem must be represented and understood.
    2. The functions that the software performs must be defined.
    3. The behavior of the software must be represented.
    4. The models that depict（描绘） information, function, and behavior must be partitioned（分割） in a manner that uncovers detail in a layered fashion.
    5. The analysis task should move from essential information towards implementation detail. 分析任务应该从基本信息转向实现细节。
  - **Design modeling principles:**
    1. Design should be traceable to the requirements model.
    2. Always consider the architecture of the system to be built.
    3. Design of data is as important as design of processing functions.
    4. Interfaces (both internal and external) must be designed with care.
    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. Component-level design should be functionally independent.

7. Components should be loosely coupled to one another and to the external environment.
8. Design representations (models) should be easily understandable.
9. The design should be developed iteratively.
10. Creation of a design model does not preclude an agile approach. 设计模型的创建并不排除敏捷方法。

- **Construction principles**
  - **coding principles**
    1. Preparation Principles: *Before you write one line of code, be sure you*
       - Understand of the problem you're trying to solve.
       - Understand basic design principles and concepts.
       - 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.
    2. Coding Principles: *As you begin writing code, be sure you*
       - Constrain your algorithms by following structured programming [Boh00] practice. 结构化编程
       - Consider the use of pair programming. 结对编程
       - Select data structures that will meet the needs of the design. 数据结构
       - Understand the software architecture and create interfaces that are consistent with it. 软件架构+接口
       - Keep conditional logic as simple as possible. 条件逻辑
       - Create nested loops in a way that makes them easily testable. 嵌套循环
       - Select meaningful variable names and follow other local coding standards. 变量名
       - Write code that is self-documenting.
       - Create a visual layout (eg., indentation and blank lines) that aids understanding 可视化布局
    3. Validation Principles: *After you've completed your first coding pass, be sure you*
       - Conduct a code walkthrough when appropriate.
       - Perform unit tests and correct errors you've uncovered.
       - **Refactor the code.** 优化、提高质量、便于复用（标准化）、可读性

         重命名、抽取代码、封装字段、抽取接口、提升方法内的局部变量变为方法的参数、删除参数、重排参数
  - **testing principles**
    1. All tests should be traceable to customer requirements.
    2. Tests should be planned long before testing begins.
    3. **The Pareto principle applies to software testing.(80-20, focus 20)**
    4. Testing should begin "in the small" and progress toward testing "in the large."
    5. Exhaustive testing is not possible.
    6. Apply to each module in the system a testing effort commensurate with its expected fault density. 对系统中的每个模块 应用与其预期故障密度相称的测试工作。

7. Static testing techniques can yield（产生） high results.
8. Track defects(缺陷) and look for patterns in defects uncovered by testing.
9. Include test cases that demonstrate software is behaving correctly.

- **deployment principles**

1. Customer expectations for the software must be managed.
2. A complete delivery package should be assembled(收集) and tested.
3. A support regime(管理体制) must be established before the software is delivered.
4. Appropriate instructional materials must be provided to end users.
5. Buggy software should be fixed first, delivered later. 有缺陷的软件