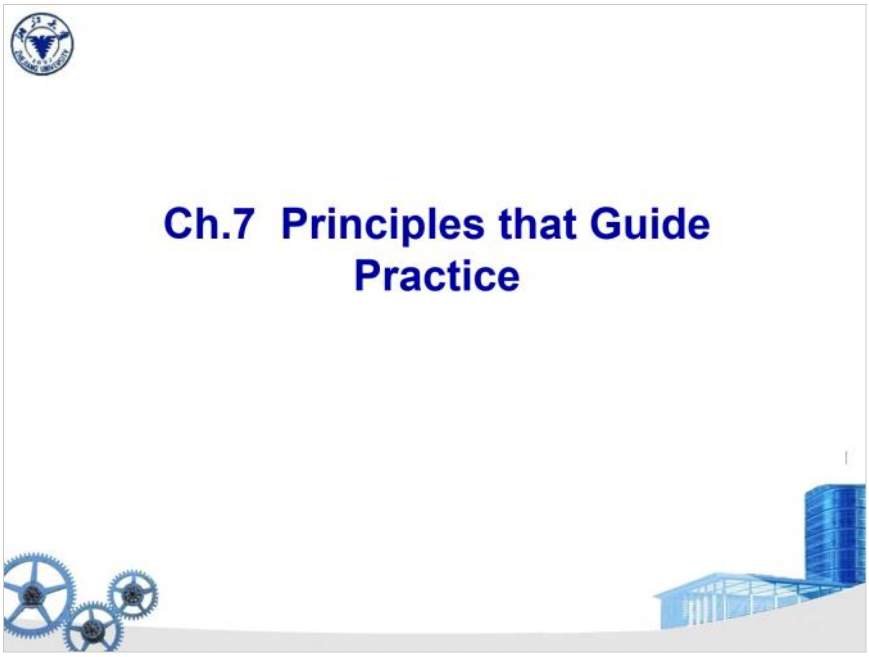
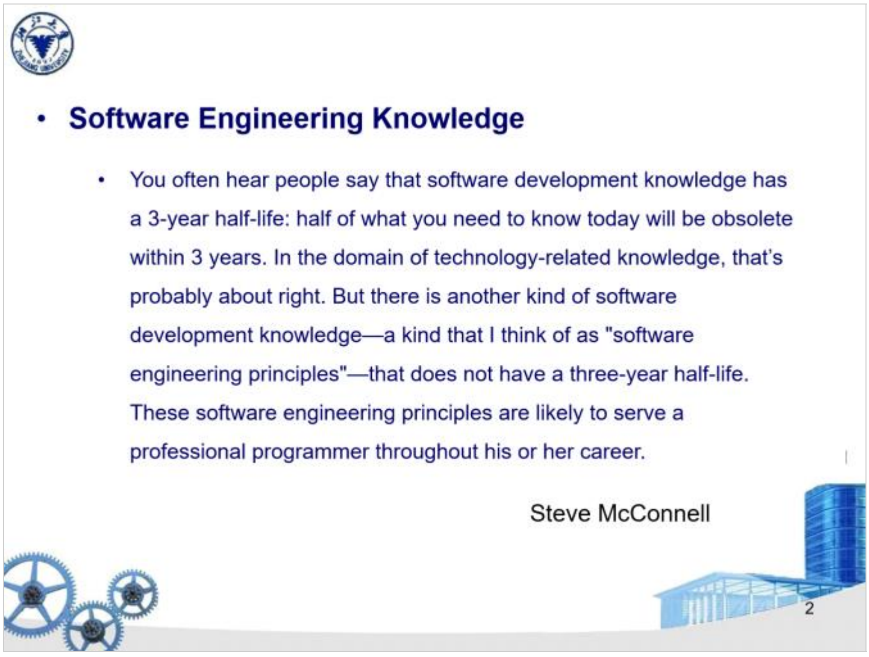
Lecture Note 7

2024年3月8日 20:20



Ch.7 Principles that Guide Practice



Software Engineering Knowledge

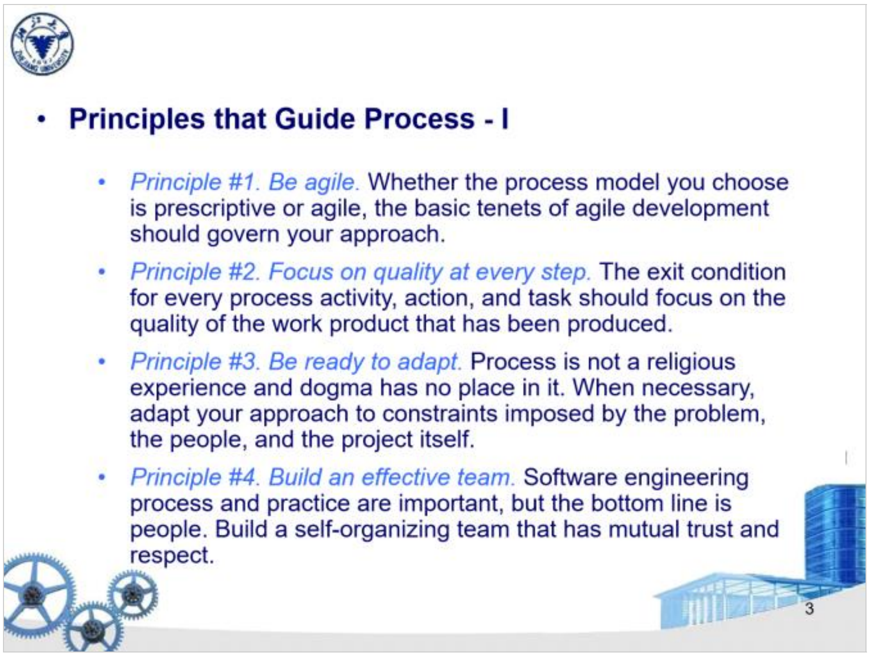
- You often hear people say that software development knowledge has a 3-year half-life: half of what you need to know today will be obsolete within 3 years. In the domain of technology-related knowledge, that's probably about right. But there is another kind of software development knowledge—a kind that 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.

Steve McConnell

2

三年只剩半条命

三年之后有一半的知识都没有价值了



Principles that Guide Process - I

- *Principle #1. Be agile.* Whether the process model you choose is prescriptive or agile, the basic tenets of agile development should govern your approach.
- *Principle #2. Focus on quality at every step.* The exit condition for every process activity, action, and task should focus on the quality of the work product that has been produced.
- *Principle #3. Be ready to adapt.* Process is not a religious experience and dogma has no place in it. When necessary, adapt your approach to constraints imposed by the problem, the people, and the project itself.
- *Principle #4. Build an effective team.* Software engineering process and practice are important, but the bottom line is people. Build a self-organizing team that has mutual trust and respect.

3

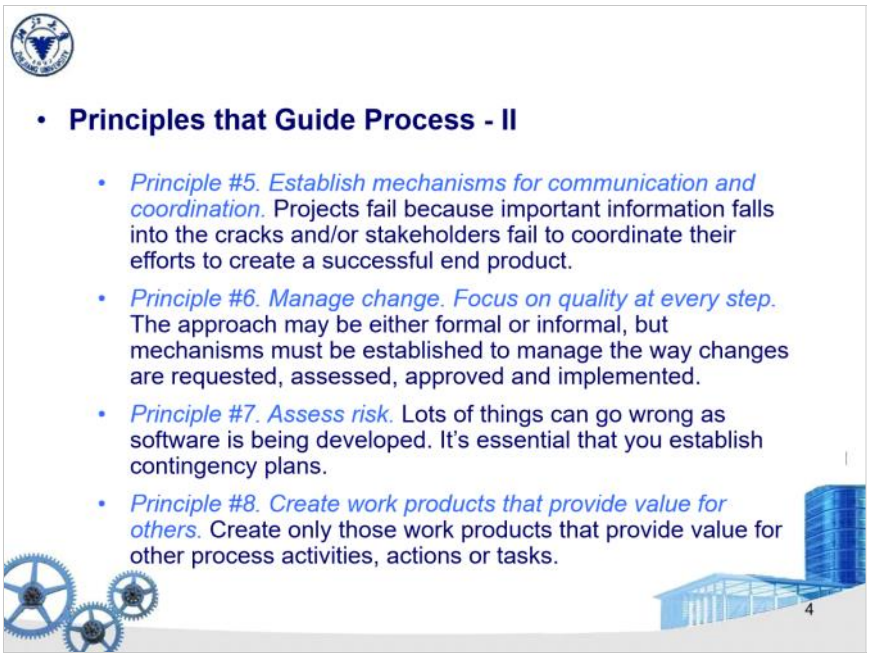
过程原则

1、敏捷化，去除一切冗余的东西

2、设计过程中每一步都要关注质量

3、做适配，不能生搬硬套

4、建立高效的团队

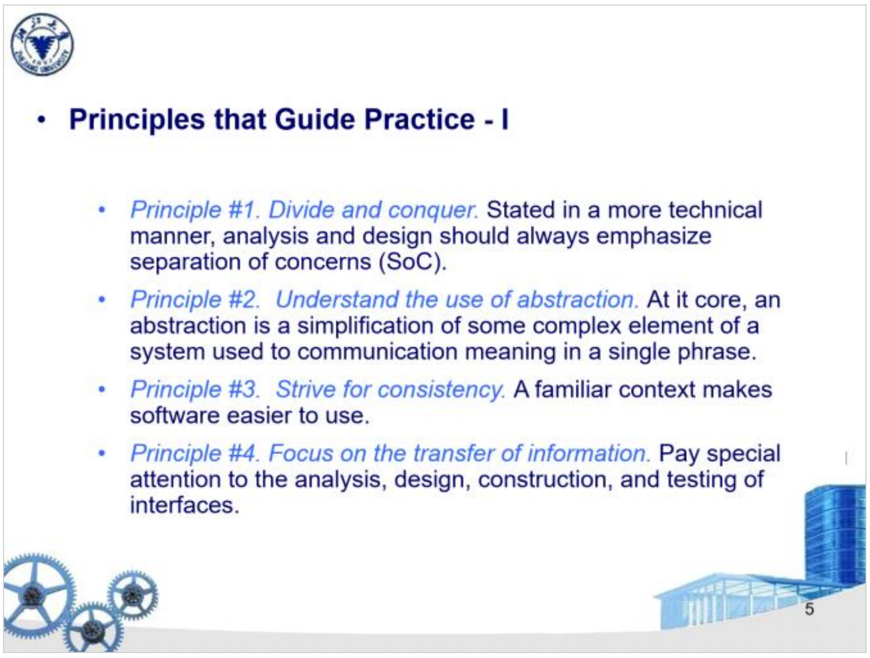


Principles that Guide Process - II

- *Principle #5. Establish mechanisms for communication and coordination.* Projects fail because important information falls into the cracks and/or stakeholders fail to coordinate their efforts to create a successful end product.
- *Principle #6. Manage change. Focus on quality at every step.* The approach may be either formal or informal, but mechanisms must be established to manage the way changes are requested, assessed, approved and implemented.
- *Principle #7. Assess risk.* Lots of things can go wrong as software is being developed. It's essential that you establish contingency plans.
- *Principle #8. Create work products that provide value for others.* Create only those work products that provide value for other process activities, actions or tasks.

4

5、沟通协调机制
6、管理变化
7、评价风险
8、能够给使用者提供价值

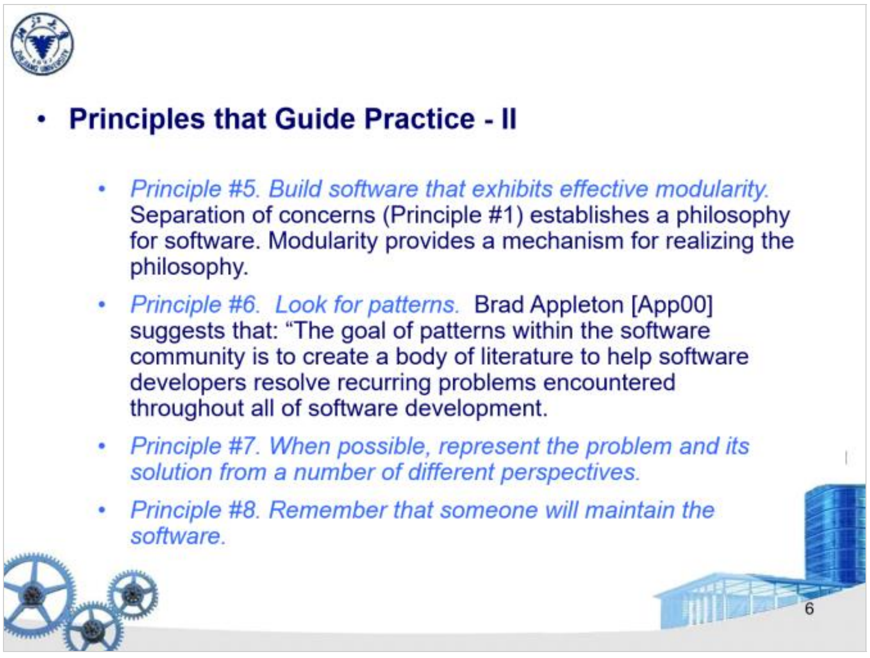


Principles that Guide Practice - I

- *Principle #1. Divide and conquer.* Stated in a more technical manner, analysis and design should always emphasize separation of concerns (SoC).
- *Principle #2. Understand the use of abstraction.* At it core, an abstraction is a simplification of some complex element of a system used to communication meaning in a single phrase.
- *Principle #3. Strive for consistency.* A familiar context makes software easier to use.
- *Principle #4. Focus on the transfer of information.* Pay special attention to the analysis, design, construction, and testing of interfaces.

5

实践原则
1、分而治之
2、理解抽象的使用
3、争取一致性
4、注意信息的转移

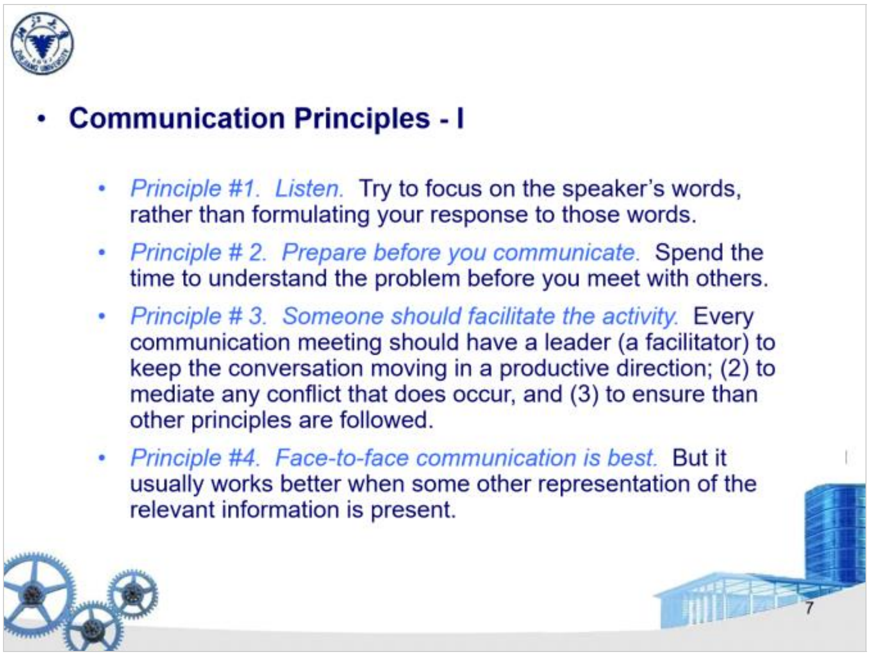


Principles that Guide Practice - II

- *Principle #5. Build software that exhibits effective modularity.* Separation of concerns (Principle #1) establishes a philosophy for software. Modularity provides a mechanism for realizing the philosophy.
- *Principle #6. Look for patterns.* Brad Appleton [App00] suggests that: "The goal of patterns within the software community is to create a body of literature to help software developers resolve recurring problems encountered throughout all of software development.
- *Principle #7. When possible, represent the problem and its solution from a number of different perspectives.*
- *Principle #8. Remember that someone will maintain the software.*

6

5、模块化
6、模式化
7、多角度思考问题
8、软件维护



Communication Principles - I

- *Principle #1. Listen.* Try to focus on the speaker's words, rather than formulating your response to those words.
- *Principle # 2. Prepare before you communicate.* Spend the time to understand the problem before you meet with others.
- *Principle # 3. Someone should facilitate the activity.* Every communication meeting should have a leader (a facilitator) to keep the conversation moving in a productive direction; (2) to mediate any conflict that does occur, and (3) to ensure than other principles are followed.
- *Principle #4. Face-to-face communication is best.* But it usually works better when some other representation of the relevant information is present.

7

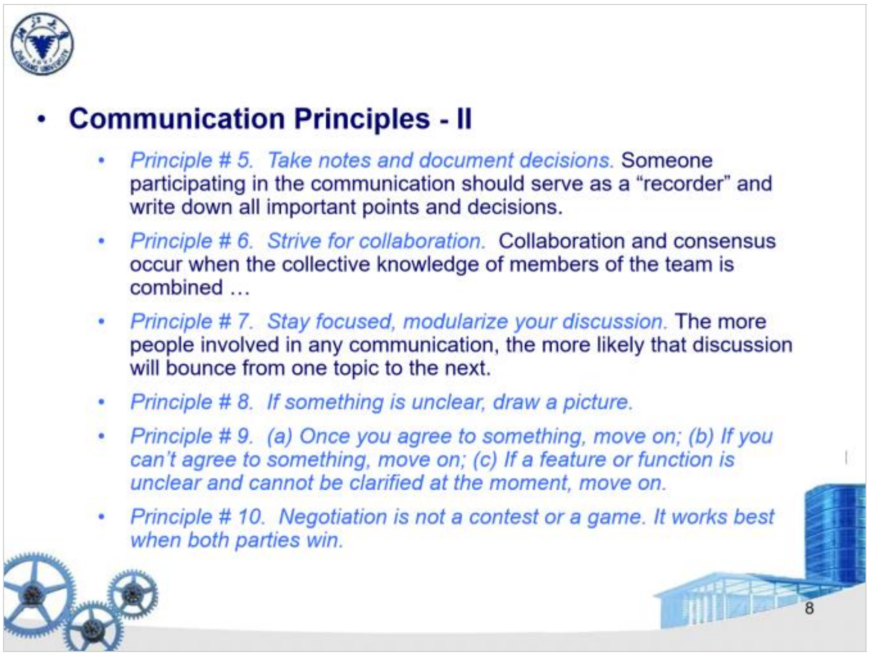
沟通原则

1、听

2、做好准备

3、要有推进者

4、面对面沟通最好



Communication Principles - II

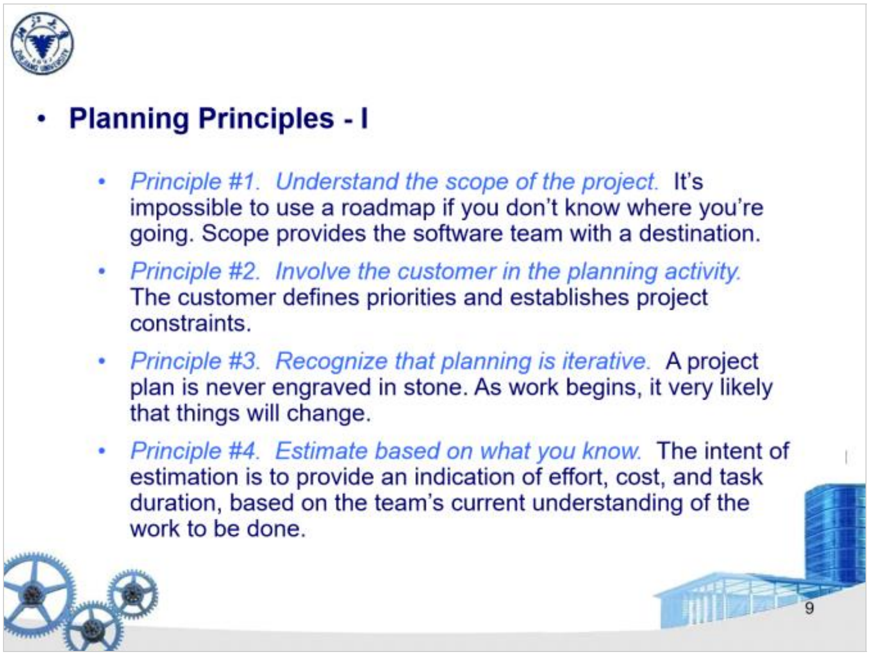
- *Principle # 5. Take notes and document decisions.* Someone participating in the communication should serve as a "recorder" and write down all important points and decisions.
- *Principle # 6. Strive for collaboration.* Collaboration and consensus occur when the collective knowledge of members of the team is combined …
- *Principle # 7. Stay focused, modularize your discussion.* The more people involved in any communication, the more likely that discussion will bounce from one topic to the next.
- *Principle # 8. If something is unclear, draw a picture.*
- *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 clarified at the moment, move on.*
- *Principle # 10. Negotiation is not a contest or a game. It works best when both parties win.*

8

5、记录重要信息

6、合作

7、保持专注，使讨论模块化（啥玩意?



Planning Principles - I

- *Principle #1. Understand the scope of the project.* It's impossible to use a roadmap if you don't know where you're going. Scope provides the software team with a destination.
- *Principle #2. Involve the customer in the planning activity.* The customer defines priorities and establishes project constraints.
- *Principle #3. Recognize that planning is iterative.* A project plan is never engraved in stone. As work begins, it very likely that things will change.
- *Principle #4. Estimate based on what you know.* The intent of estimation is to provide an indication of effort, cost, and task duration, based on the team's current understanding of the work to be done.

9

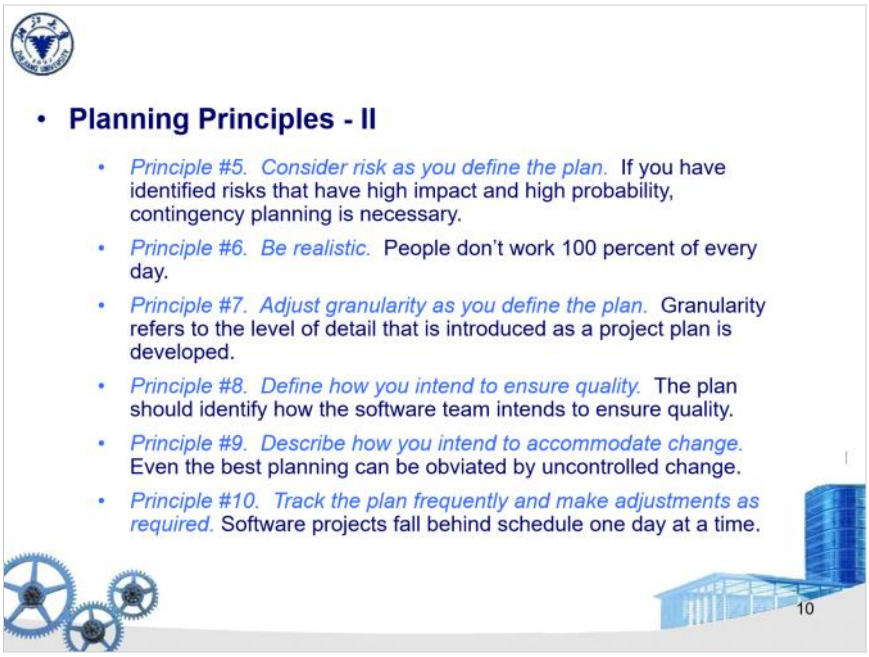
计划原则

1、理解项目范围

2、让用户参与进来

3、计划需要不断更新

4、做好评估

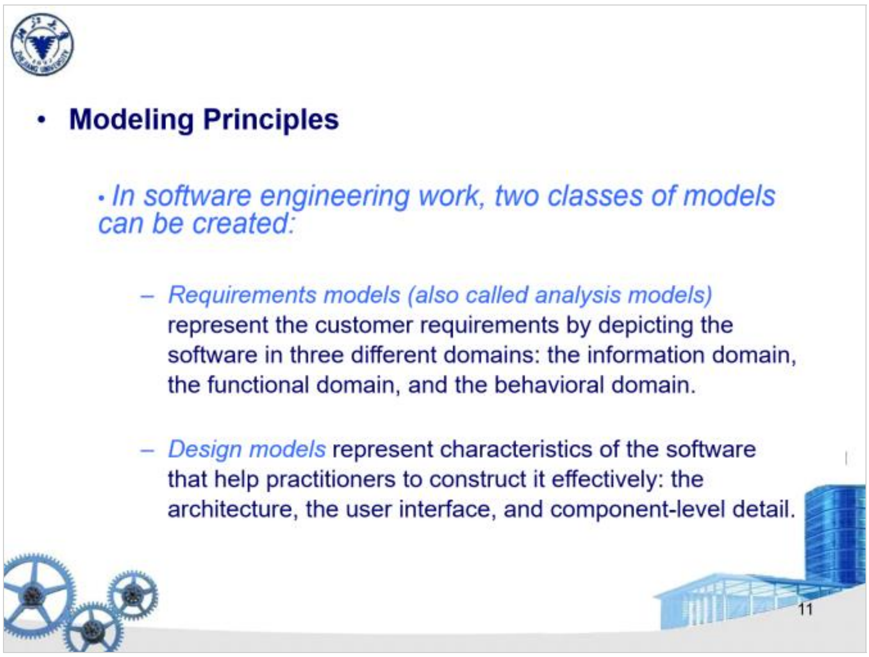


Planning Principles - II

- *Principle #5. Consider risk as you define the plan.* If you have identified risks that have high impact and high probability, contingency planning is necessary.
- *Principle #6. Be realistic.* People don't work 100 percent of every day.
- *Principle #7. Adjust granularity as you define the plan.* Granularity refers to the level of detail that is introduced as a project plan is developed.
- *Principle #8. Define how you intend to ensure quality.* The plan should identify how the software team intends to ensure quality.
- *Principle #9. Describe how you intend to accommodate change.* Even the best planning can be obviated by uncontrolled change.
- *Principle #10. Track the plan frequently and make adjustments as required.* Software projects fall behind schedule one day at a time.

10

5、风险评估
6、实际一些
7、调整计划的粒度（细节程度?
8、你打算怎么保证质量
9、你打算怎么适应变化
10、持续跟踪并根据需求进行调整

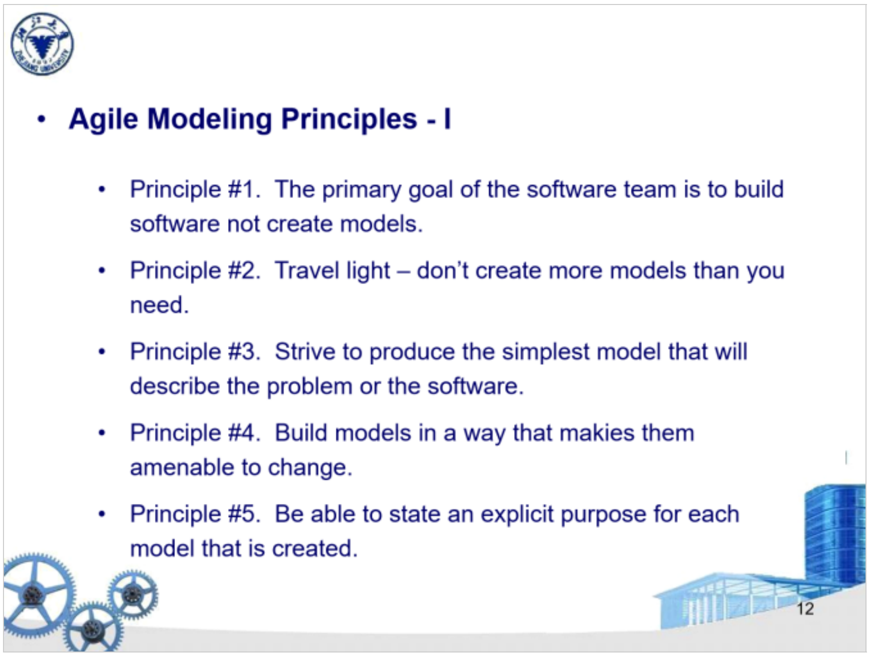


Modeling Principles

- *In software engineering work, two classes of models can be created:*
  - *Requirements models (also called analysis models)* represent the 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 practitioners to construct it effectively: the architecture, the user interface, and component-level detail.

11

建模原则
两种模型：
1、需求模型
从三个域（信息、功能、表现）展示用户需求
2、设计模型
展示软件的特点（架构、用户界面、以及成分级【成分设计】的细节）

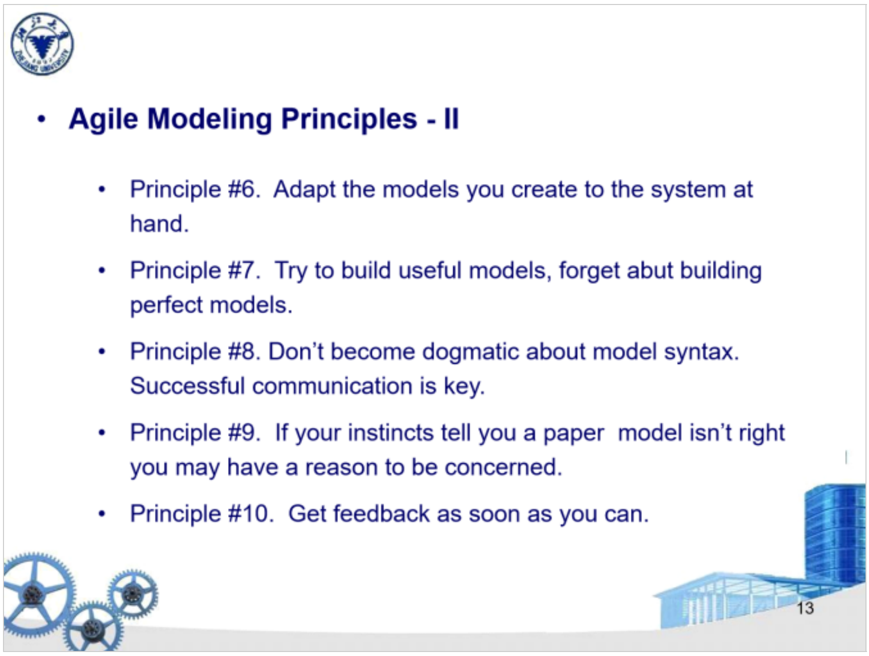


Agile Modeling Principles - I

- Principle #1. The primary goal of the software team is to build software not create models.
- Principle #2. Travel light – don't create more models than you need.
- Principle #3. Strive to produce the simplest model that will describe the problem or the software.
- Principle #4. Build models in a way that makies them amenable to change.
- Principle #5. Be able to state an explicit purpose for each model that is created.

12

敏捷模型原则
1、目标是构造软件不是模型
2、模型够用就行
3、简单化
4、能适应变化
5、能够对于每个模型给出清晰的目的

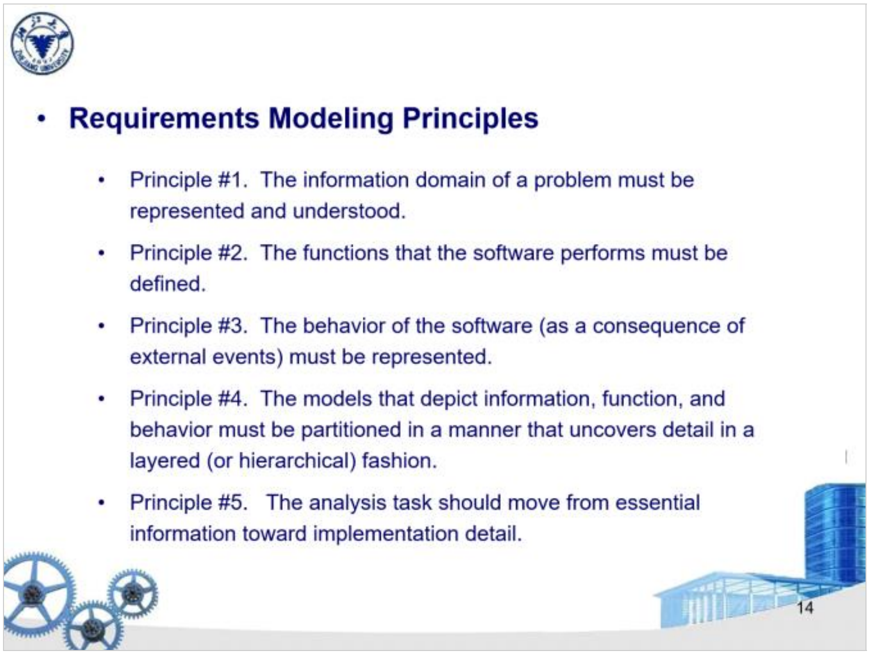


Agile Modeling Principles - II

- Principle #6. Adapt the models you create to the system at hand.
- Principle #7. Try to build useful models, forget abut building perfect models.
- Principle #8. Don't become dogmatic about model syntax. Successful communication is key.
- Principle #9. If your instincts tell you a paper model isn't right you may have a reason to be concerned.
- Principle #10. Get feedback as soon as you can.

13

6、让模型能够很好地嵌入系统
7、做有用的
8、不要教条主义，要沟通
9、关注本能?
10、及时反馈

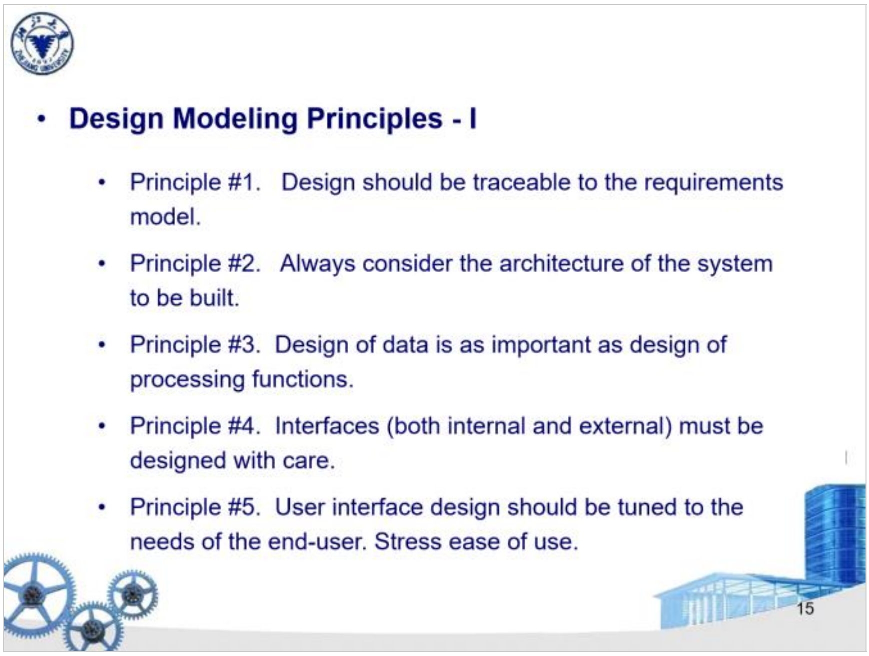


Requirements Modeling Principles

- Principle #1. The information domain of a problem must be represented and understood.
- Principle #2. The functions that the software performs must be defined.
- Principle #3. The behavior of the software (as a consequence of external events) must be represented.
- 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.
- Principle #5. The analysis task should move from essential information toward implementation detail.

14

需求模型原则
1、问题信息需要被理解
2、功能要被定义
3、软件的行为要被体现?
4、层级式架构?
5、分析任务需要从重要信息转移到实现细节

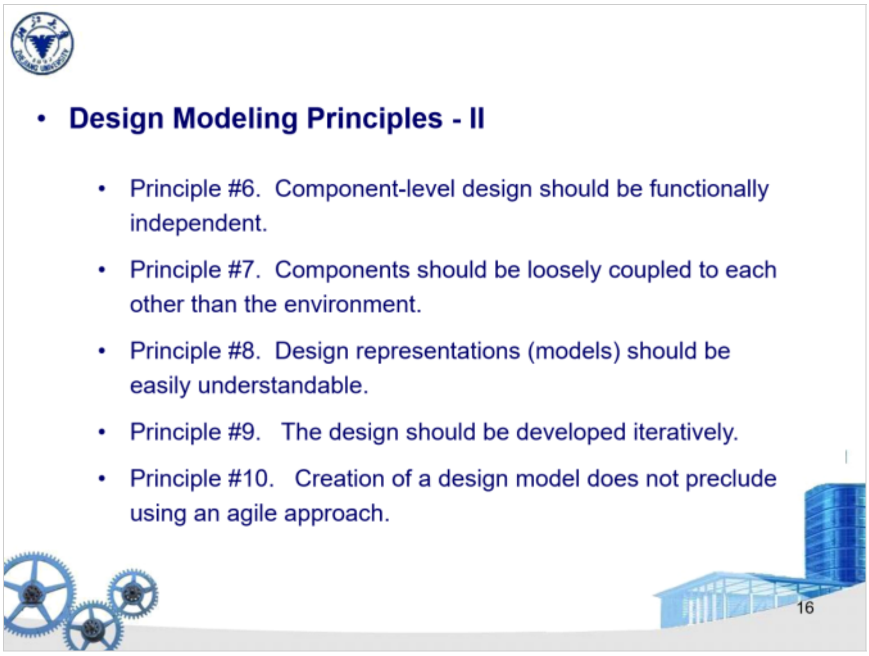


Design Modeling Principles - I

- Principle #1. Design should be traceable to the requirements model.
- Principle #2. Always consider the architecture of the system to be built.
- Principle #3. Design of data is as important as design of processing functions.
- Principle #4. Interfaces (both internal and external) must be designed with care.
- Principle #5. User interface design should be tuned to the needs of the end-user. Stress ease of use.

15

设计模型原则
1、设计应当根据需求进行
2、考虑架构
3、数据的设计也很重要
4、接口一定要人性化
5、用户接口要适应用户

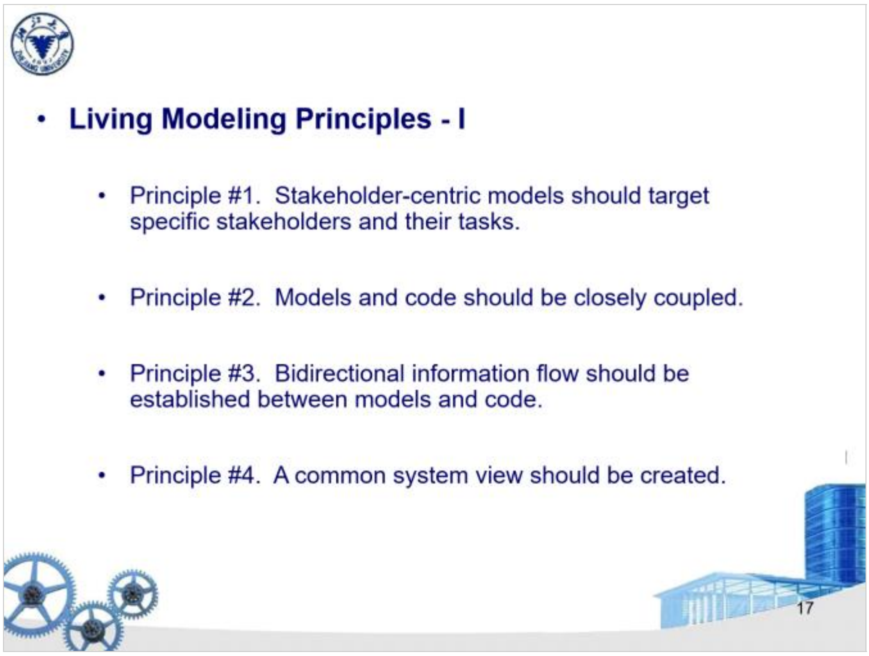


Design Modeling Principles - II

- Principle #6. Component-level design should be functionally independent.
- Principle #7. Components should be loosely coupled to each other than the environment.
- Principle #8. Design representations (models) should be easily understandable.
- Principle #9. The design should be developed iteratively.
- Principle #10. Creation of a design model does not preclude using an agile approach.

16

6、部分级要功能性独立
7、组成部分之间的羁绊少一点
8、容易理解
7、迭代式设计
8、设计模型和敏捷方法不冲突

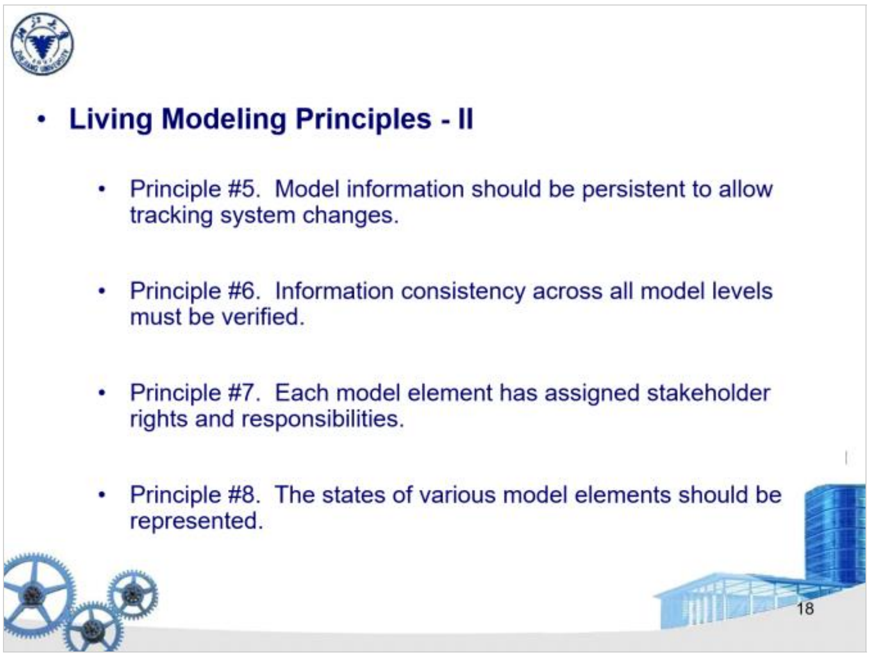


Living Modeling Principles - I

- Principle #1. Stakeholder-centric models should target specific stakeholders and their tasks.
- Principle #2. Models and code should be closely coupled.
- Principle #3. Bidirectional information flow should be established between models and code.
- Principle #4. A common system view should be created.

17

在使用模型原则
1、关注股东需求
2、模型和代码要紧密联系
3、模型和代码之间建立双向联系
4、构建通用模型方法



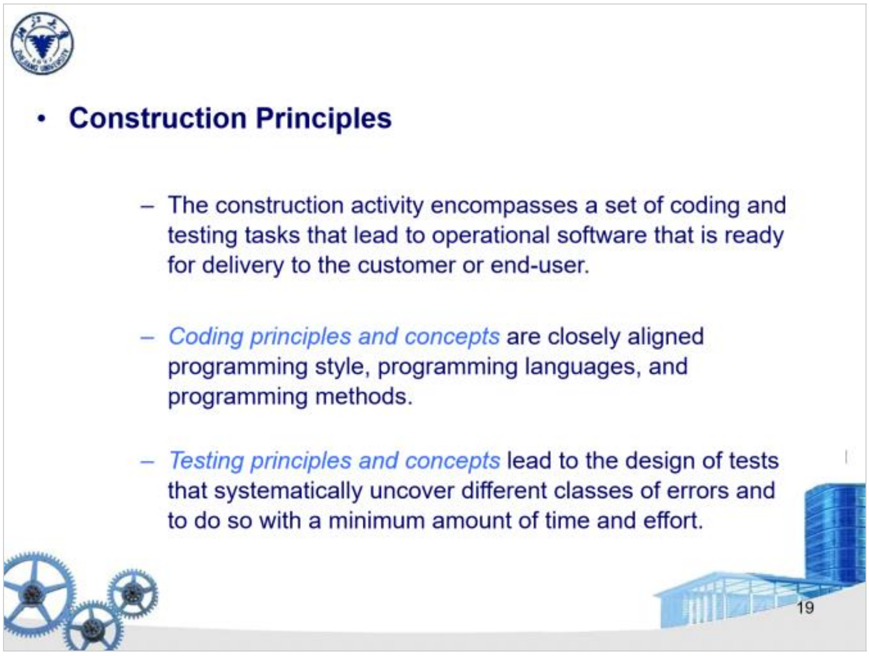
Living Modeling Principles - II

- Principle #5. Model information should be persistent to allow tracking system changes.
- Principle #6. Information consistency across all model levels must be verified.
- Principle #7. Each model element has assigned stakeholder rights and responsibilities.
- Principle #8. The states of various model elements should be represented.

18

5、模型信息要持续跟踪系统变化
6、保证信息一致性
7、给股东分配权力和责任（啥玩意?
8、不同模型元素的状态应该被体现

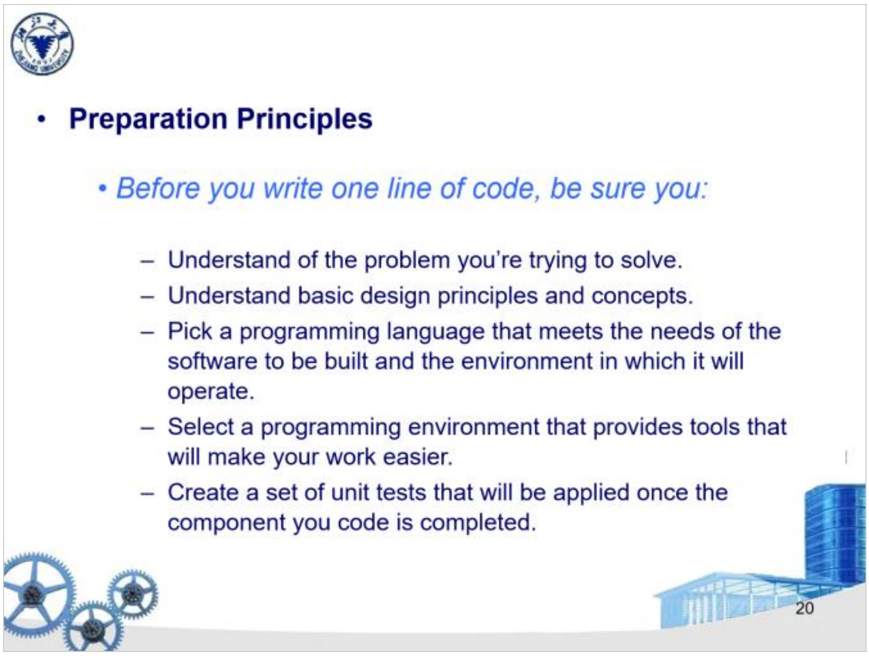
18



- **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.
  - *Coding principles and concepts* are closely aligned programming style, programming languages, and programming methods.
  - *Testing principles and concepts* lead to the design of tests that systematically uncover different classes of errors and to do so with a minimum amount of time and effort.

19

构建原则包含代码原则和设计原则】



- **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.

20

准备原则

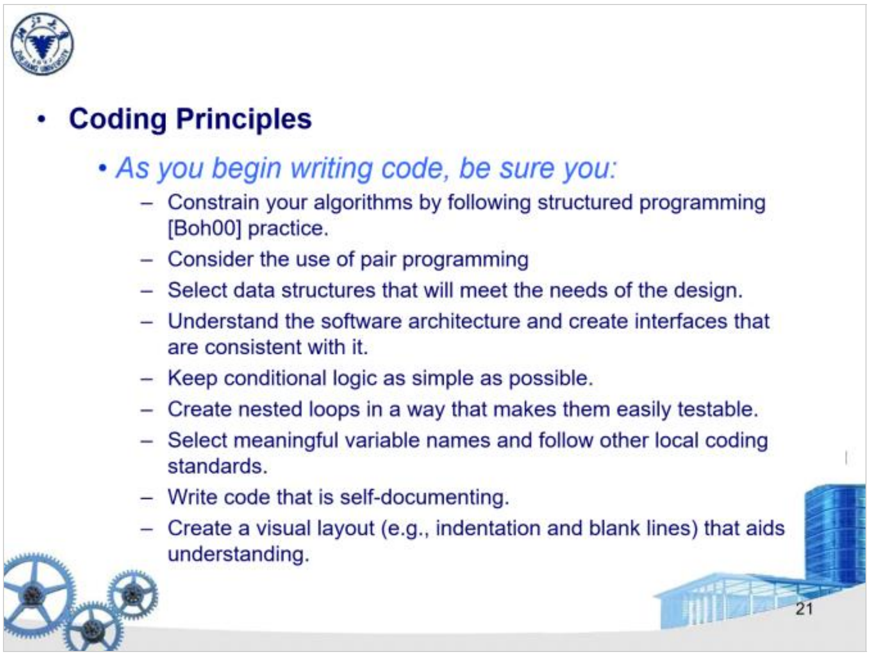
1、理解问题

2、理解基本设计理念

3、选个合适的语言

4、选个合适的编程环境

5、每个部分代码都要做测试



- **Coding Principles**
  - *As you begin writing code, be sure you:*
    - Constrain your algorithms by following structured programming [Boh00] practice.
    - Consider the use of pair programming
    - Select data structures that will meet the needs of the design.
    - 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.

21

写代码原则

1、算法要参照结构式编程

2、同伴式编程（一个人写代码一个人检查）

3、选择合适的数据结构

4、理解软件架构并构造合适的接口

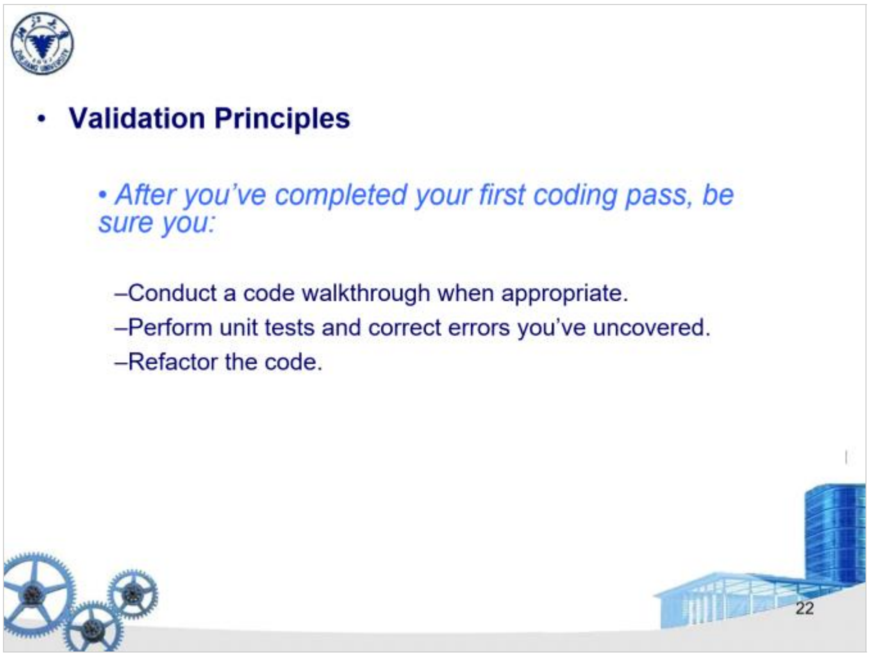
5、条件判断越简单越好

6、循环要方便测试

7、选择有意义的变量名

8、做好注释等方便理解和维护的工作

9、方便理解（排版、空行）



- **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.

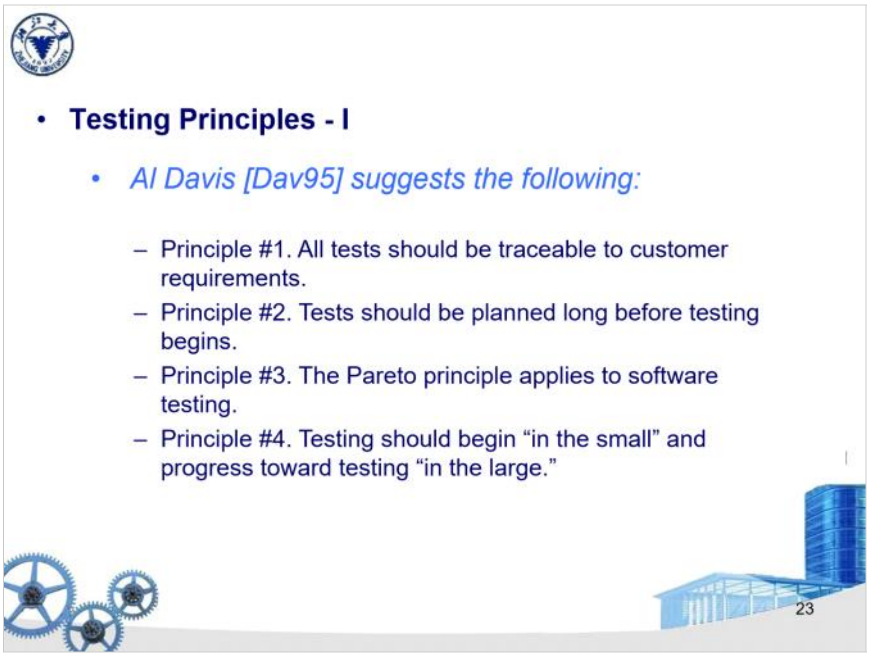
22

合法性原则

1、walkthrough整个代码

2、测试并修正错误

3、重构代码



- **Testing Principles - I**
  - *Al Davis [Dav95] suggests the following:*
    - Principle #1. All tests should be traceable to customer requirements.
    - Principle #2. Tests should be planned long before testing begins.
    - Principle #3. The Pareto principle applies to software testing.
    - Principle #4. Testing should begin "in the small" and progress toward testing "in the large."

23

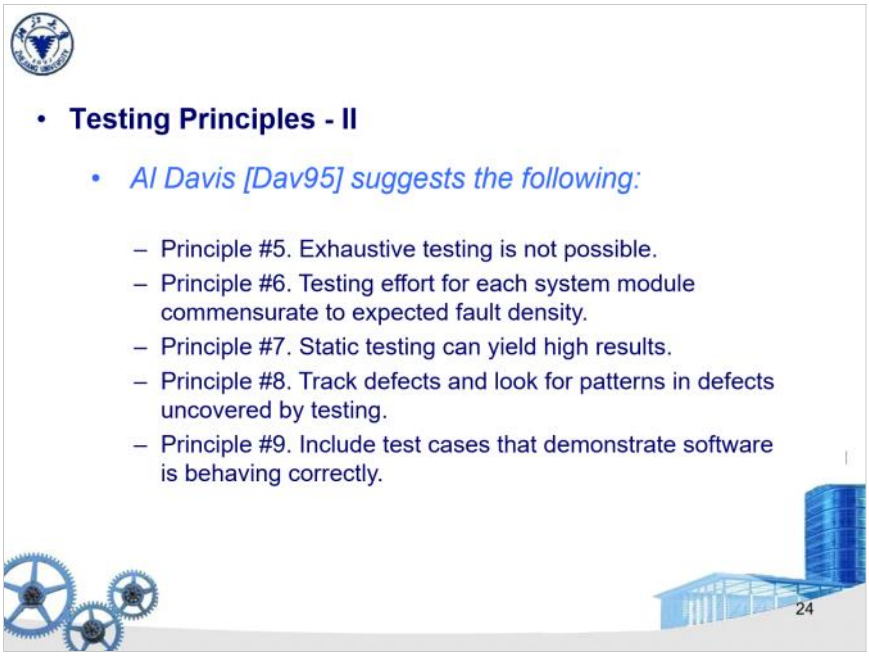
测试原则

1、依据用户需求

2、计划充分

3、Pareto原则（最重要的仅占20%）

4、由小到大



- **Testing Principles - II**
  - *Al Davis [Dav95] suggests the following:*
    - Principle #5. Exhaustive testing is not possible.
    - Principle #6. Testing effort for each system module commensurate to expected fault density.
    - Principle #7. Static testing can yield high results.
    - Principle #8. Track defects and look for patterns in defects uncovered by testing.
    - Principle #9. Include test cases that demonstrate software is behaving correctly.

24

5、彻底的测试没必要

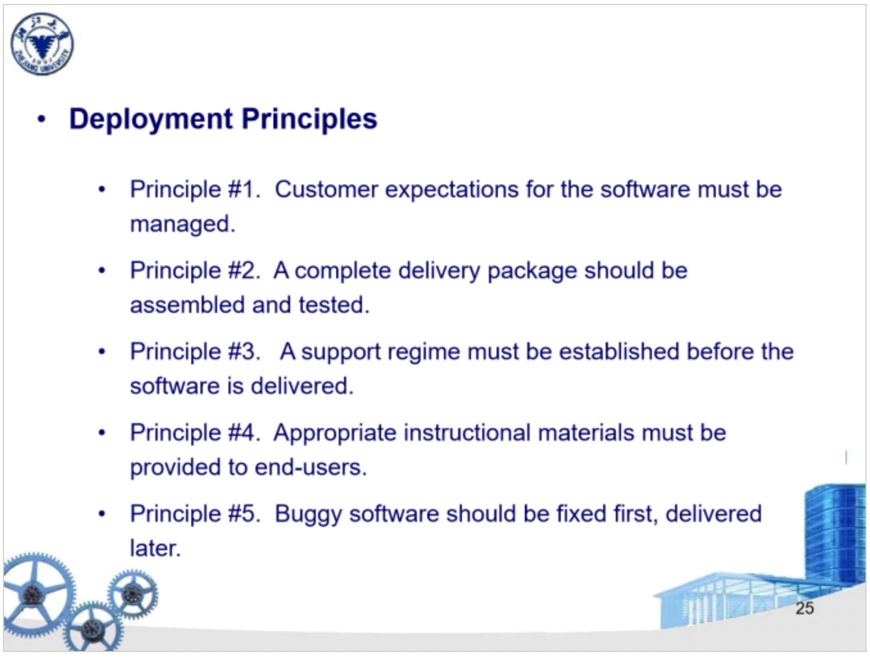
6、测试所花的努力和代码里的错误密度相当

7、静态测试更好?

8、跟踪代码缺陷，并寻找缺陷的模式

9、包含具有正确输出的测试

24



- **Deployment Principles**
  - Principle #1. Customer expectations for the software must be managed.
  - Principle #2. A complete delivery package should be assembled and tested.
  - Principle #3.   A support regime must be established before the software is delivered.
  - Principle #4. Appropriate instructional materials must be provided to end-users.
  - Principle #5. Buggy software should be fixed first, delivered later.

25

部署原则

1、满足用户期望

2、最终版要经过组装和测试

3、在传送软件之前建立组织方法

4、使用手册

5、有bug的先修再传