软件测试与质量保证

软件测试与质量保证

- 第一部分 软件测试概述
- 第二部分 黑盒测试
- 第三部分 白盒测试
- 第四部分 软件测试的级别
- 第五部分 软件测试工具
- 第六部分 软件质量与软件质量保证
- 第七部分 程序分析技术（Optional）

第一部分 软件测试概述

1. 软件测试的背景
2. 软件测试的含义
3. 软件测试的过程
4. 错误与缺陷分类
5. 测试级别
6. 软件测试的分类
7. 对测试人员的要求

2. 软件测试的含义

- 2.1 什么是软件测试
- 2.2 软件测试的目的
- 2.3 软件测试的对象
- 2.4 测试≠调试
- 2.5 软件测试的特征

back

2.5 软件测试的特征

- 软件测试的风险性
- 软件测试的不修复原则
- 软件测试的群集现象 – *(Pareto原则)*
- 软件缺陷的寄生虫性

软件测试是有风险的行为

彻底测试程序是不可能的

- **原因：**
  - 输入量太大
  - 输出结果太多
  - 软件运行的路径太多
  - 软件说明书没有客观标准

- **如果决定不去测试所有的情况，那就是选择了风险。**

软件测试的不修复原则

并非所有软件缺陷都需要修复

原因：

- **没有足够的时间**
- **不算真正的软件缺陷**
- **修复的风险太大**
- **不值得修复**

软件测试的群集现象– *(Pareto原则)*

测试发现的错误中的80%很可能起源于程序模块中的20%。

- 例如: IBM OS/370操作系统中，47%的错误仅与该系统中的4%的程序模块有关。

- 软件缺陷的寄生虫性：
  找到的软件缺陷越多，就表明残存的软件缺陷越多

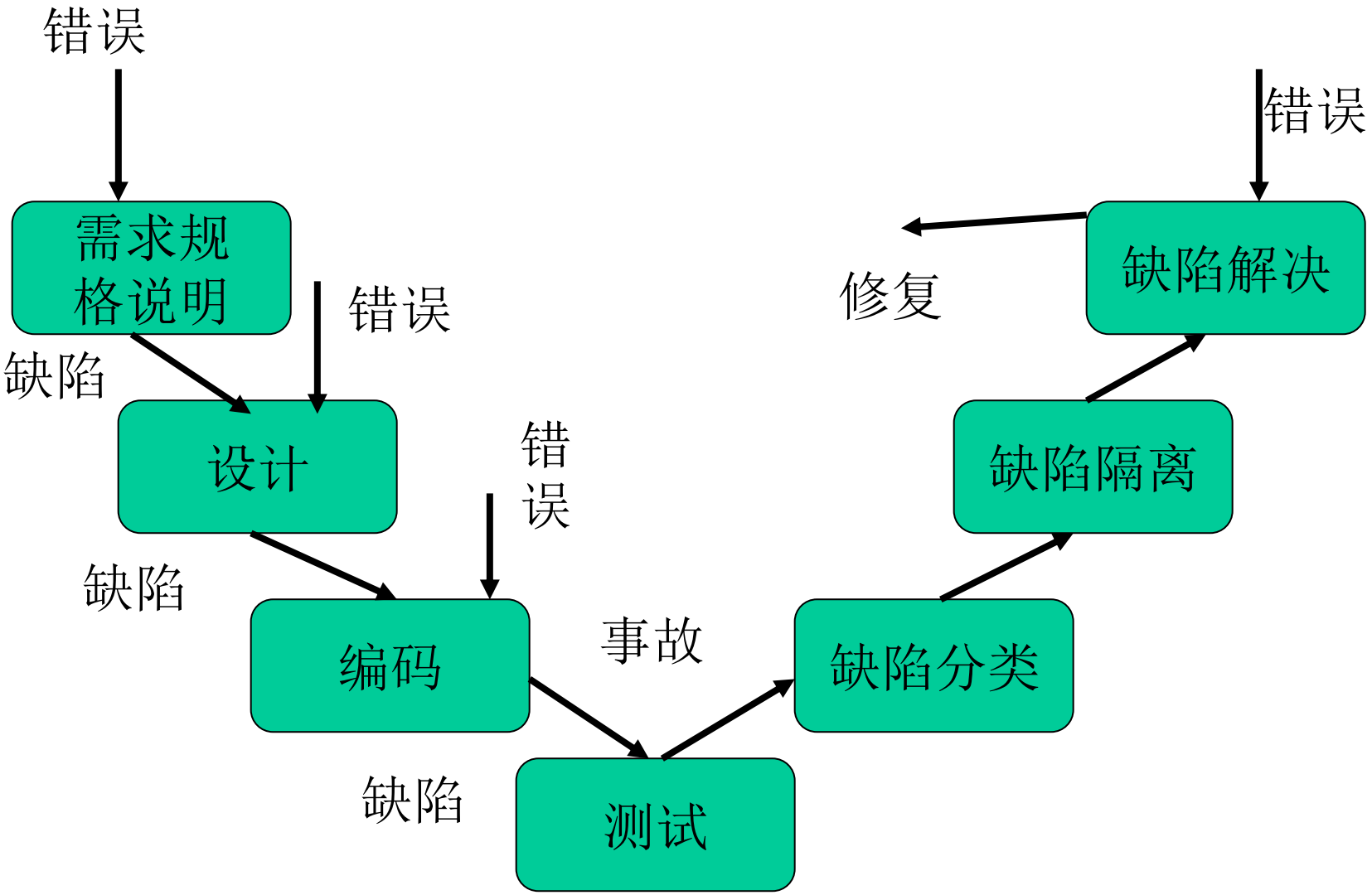
  – 原因：程序员的疲倦；程序员往往犯同样的错误

3. 软件测试的过程

- 3.1 软件测试的生命周期
- 3.2 软件测试的步骤
- 3.3 测试用例
- 3.4 通过维恩图理解测试用例
- 3.5 测试用例的设计
  - 3.5.1 功能性测试
  - 3.5.2 结构性测试

back

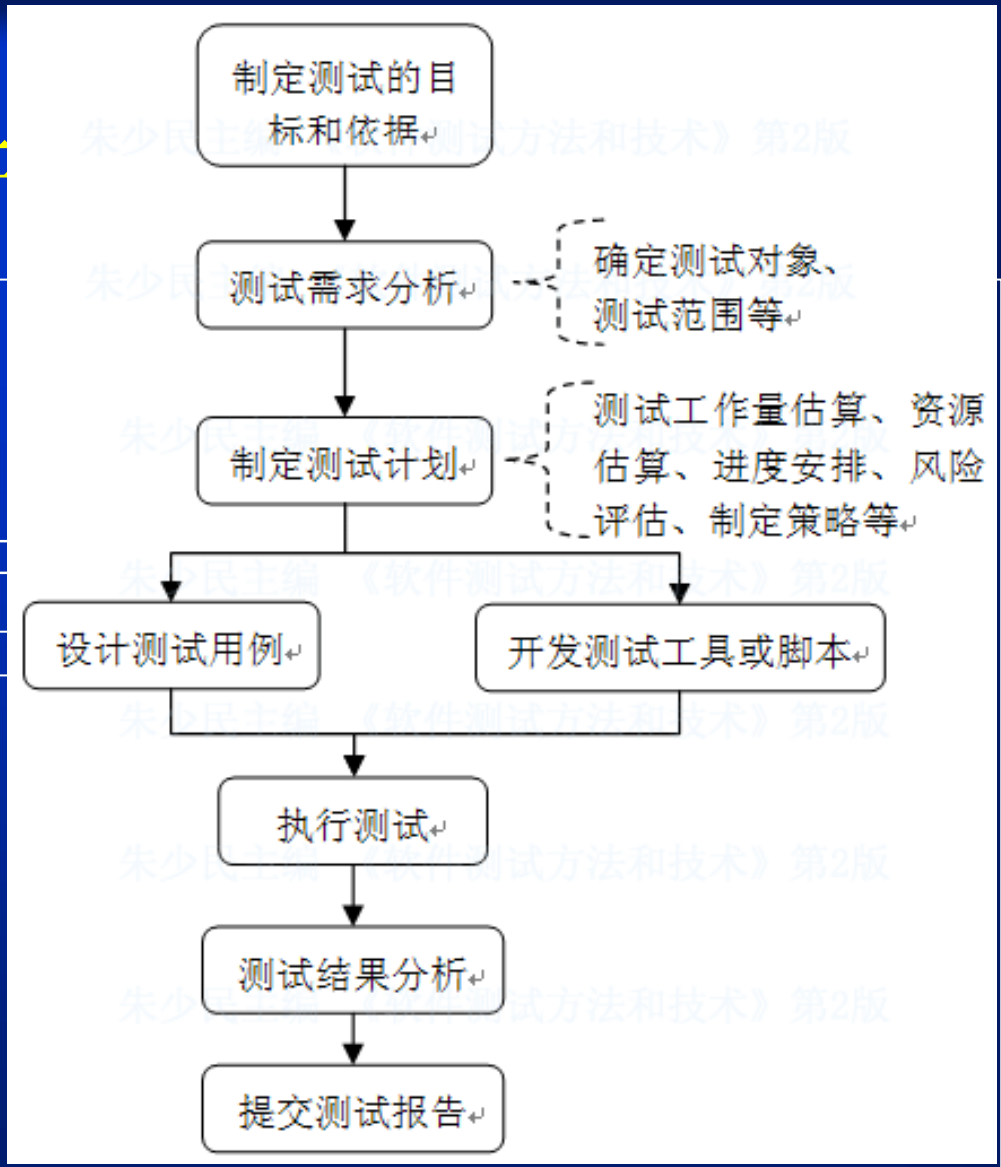
3.1 软件测试的生命周期



3.2 软件

四步:

- 1. 制定测试计划
  - 需求, 人员, 成本, 资源, 工
- 2. 设计测试用例和
- 3. 运行测试用例
- 4. 评估测试结果



- 核心: **测试用例**
  - 软件测试最重要的工作: 针对要测试的内容设计一组测试用例

3.3 测试用例（Test case)

- 测试用例 = 输入 + 预期输出
  - 输入：输入和前提 (前置条件,在测试用例执行之前已经存在的环境)
  - 预期输出：输出和后果 (前置条件,在测试用例执行之后将产生的环境)
- 测试结果：实际执行结果与预期结果是否一致

3.3 测试用例（Test case)

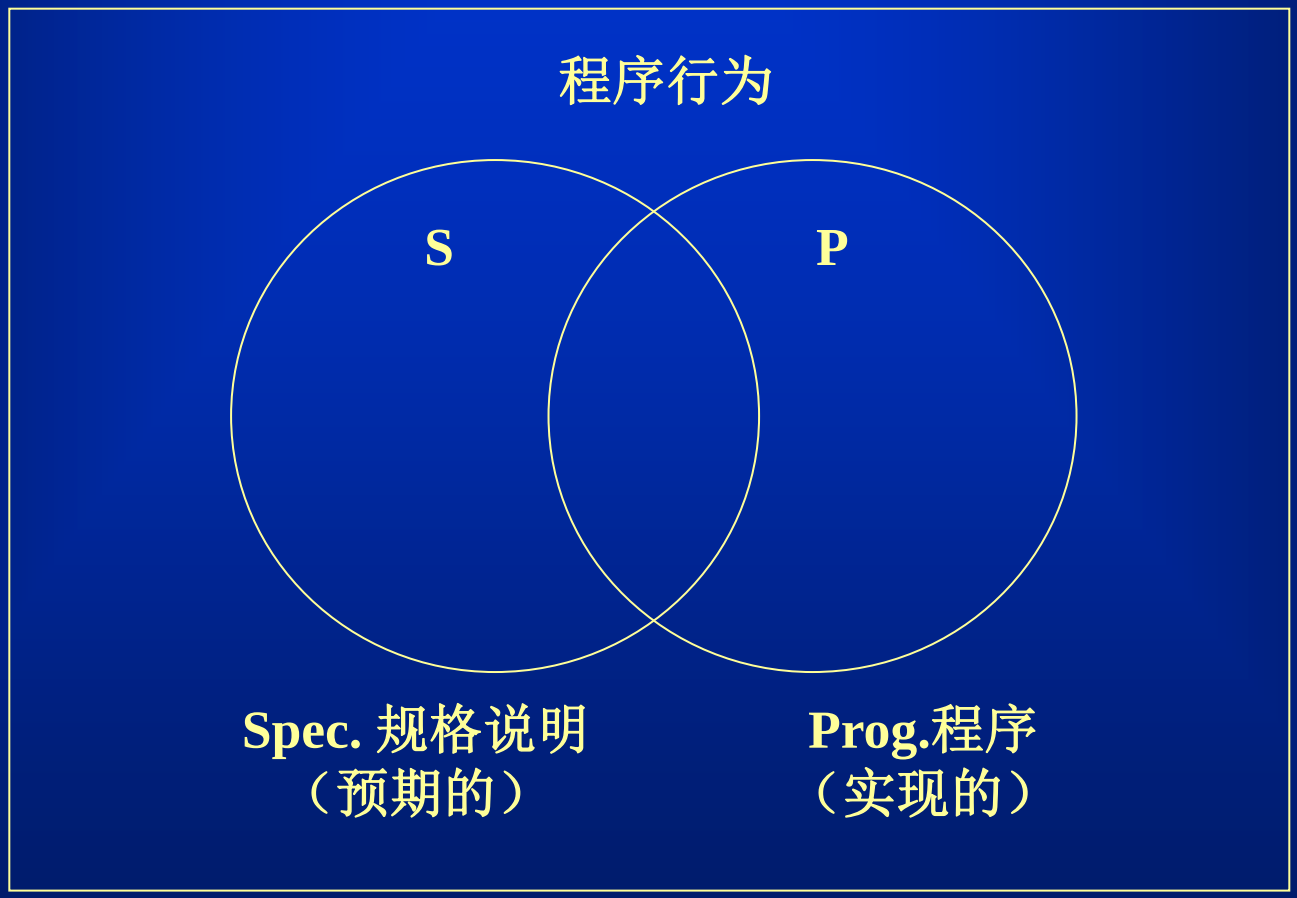
- 测试用例的要素与术语：
  - 输入： **Input； Test data**
  - 预期输出： 预言**Oracle**； **Expected output**
  - 其它：**Environment**

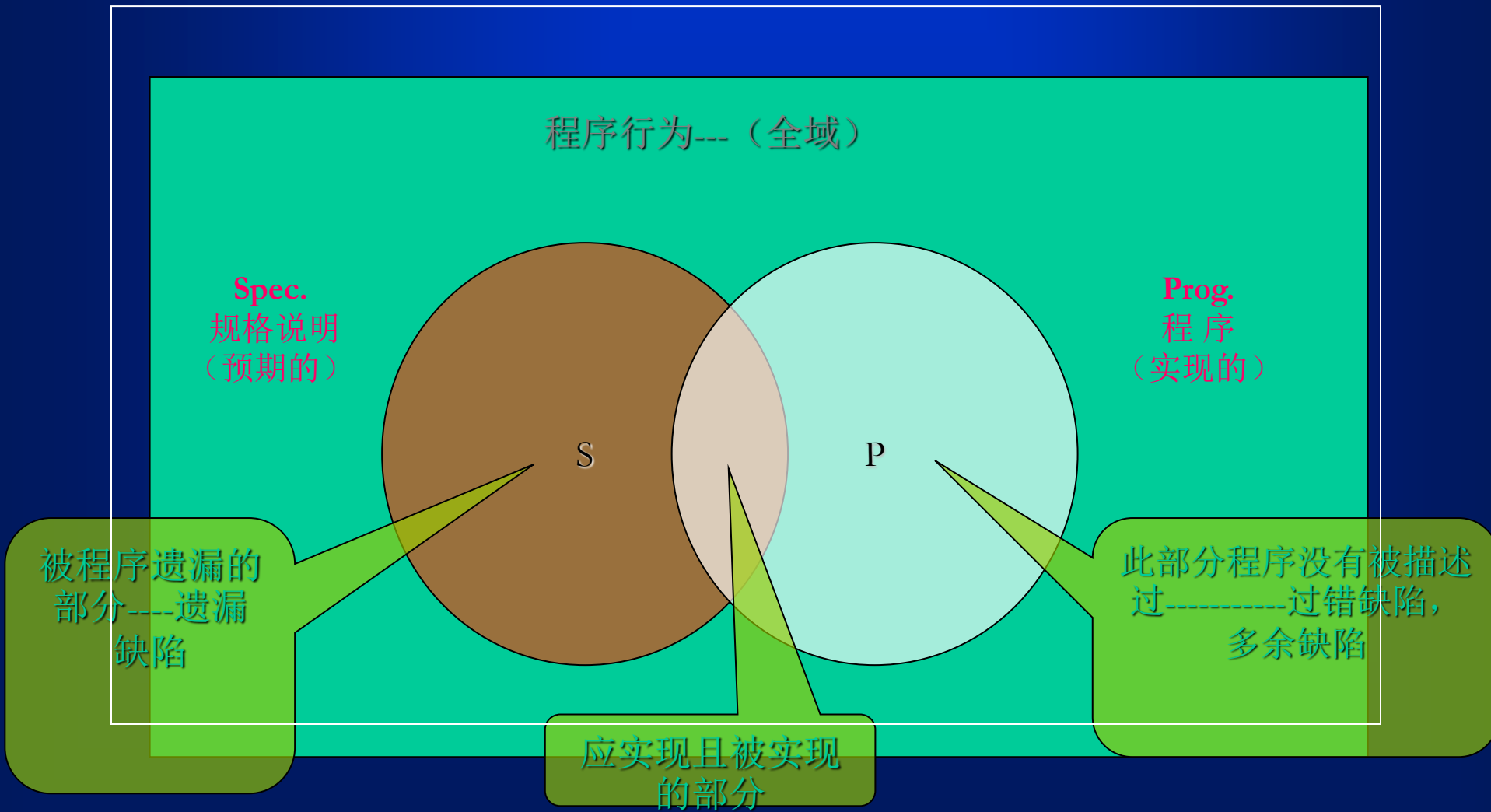
典型的测试用例

测试用例ID
目的
前提
输入
预期输出
后果
执行历史
日期　　结果　　版本　　执行人

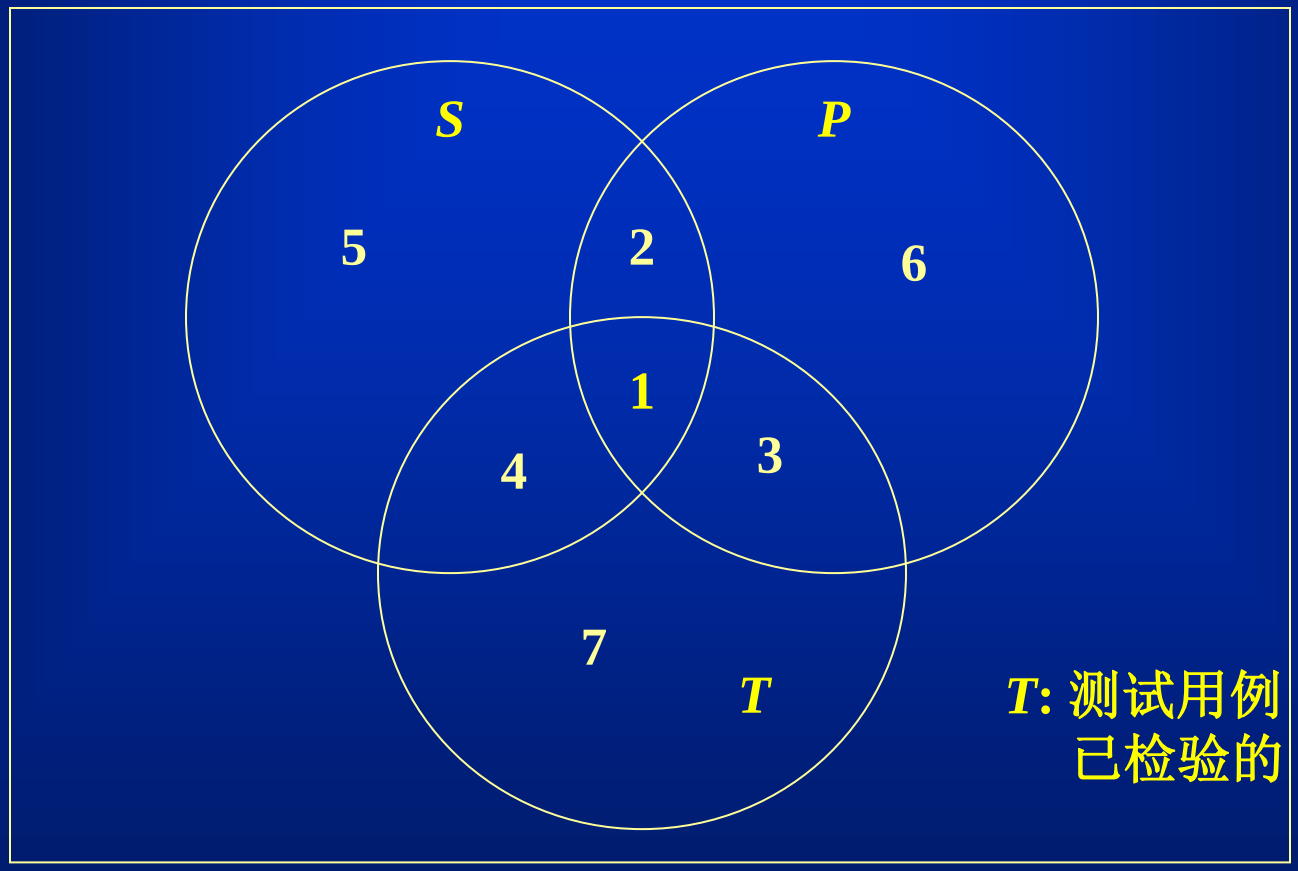
注: 其它信息用于支持 测试管理

3.4 通过维恩图理解测试




程序行为---（全域）
Spec.
规格说明
（预期的）
Prog.
程 序
（实现的）
S
P
被程序遗漏的部分----遗漏缺陷
应实现且被实现的部分
此部分程序没有被描述过----------过错缺陷，多余缺陷

通过维恩图理解测试



一个Project的最终目的：
使S、P、T都相交的区域，即区域1 尽可能的大

3.5 测试用例的设计

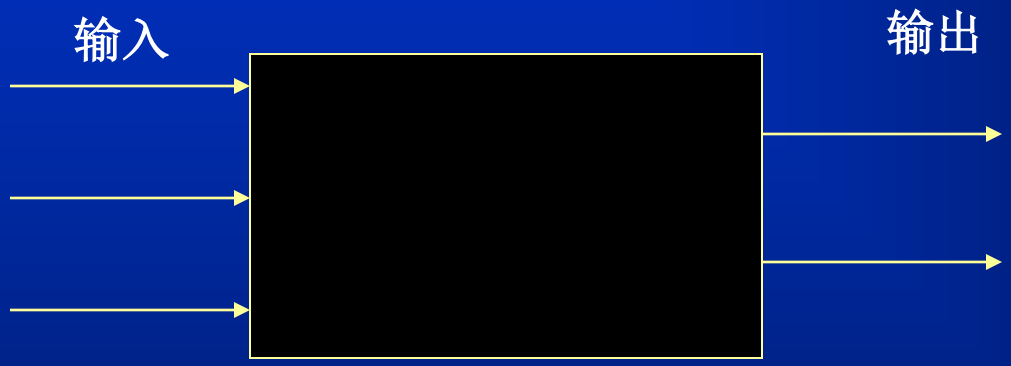
两种基本的测试技术:

- **黑盒测试**
  - **功能性测试**, 基于规格说明的测试
- **白盒测试**
  - **结构性测试**，透明盒测试, 玻璃盒测试, 基于程序的测试

3.5.1 黑盒测试

任何程序都可以看作是将输入定义域取值映射到输出值域的函数。将系统看成“黑盒子”。

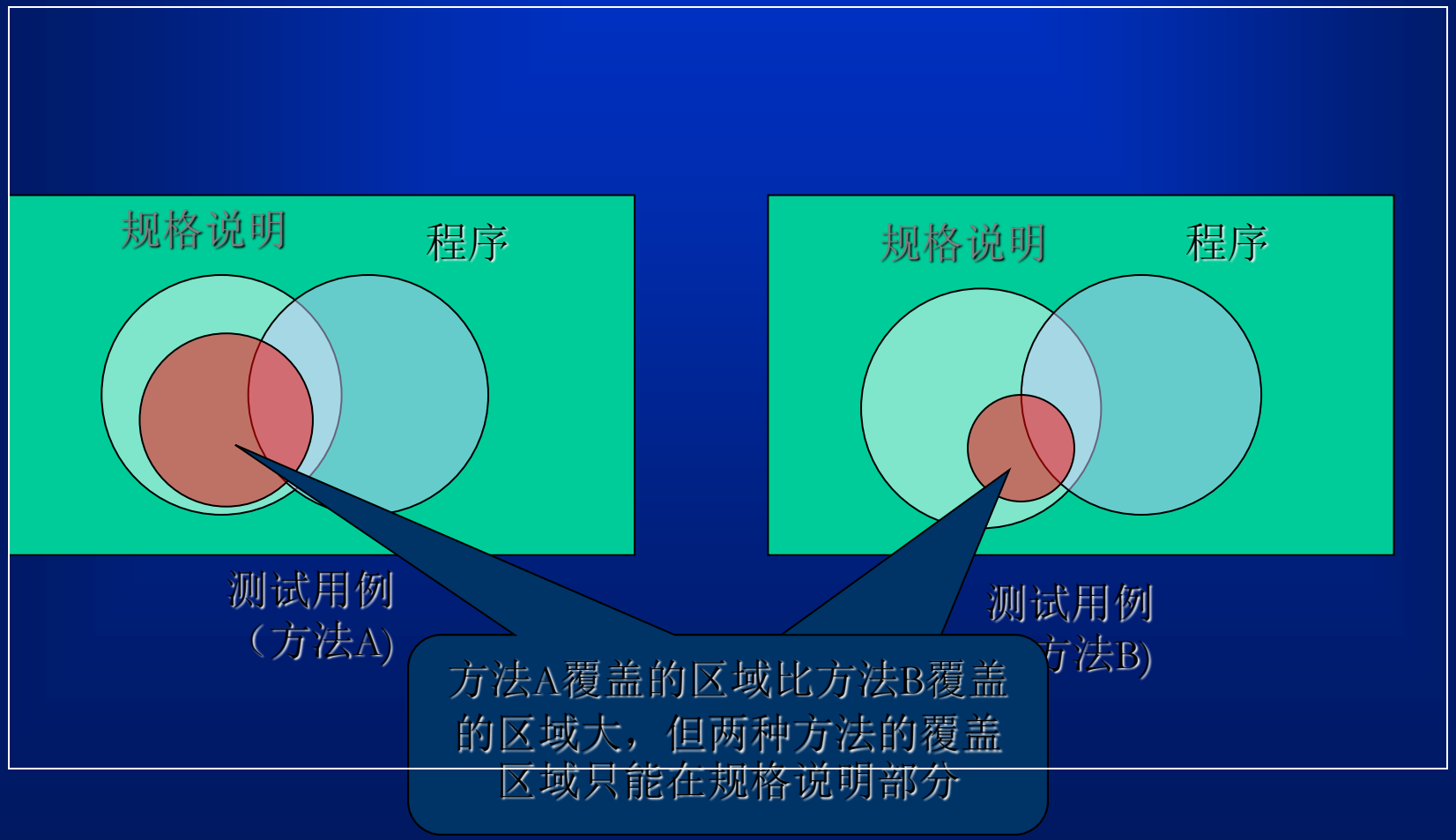
。



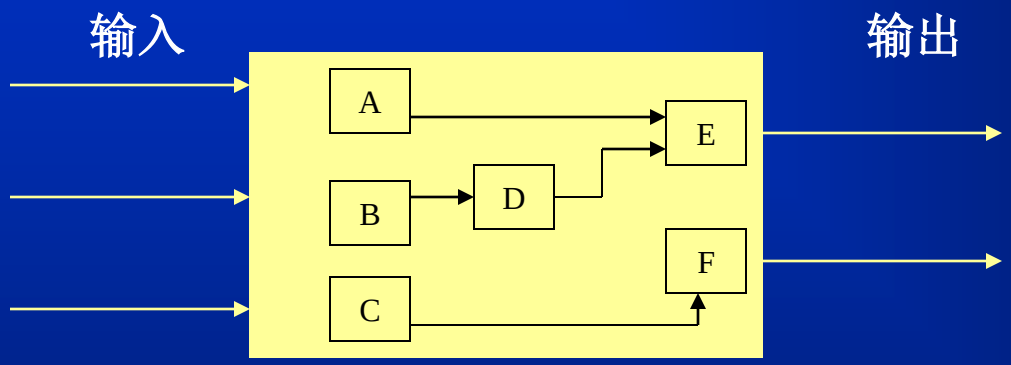
黑盒测试的依据

- 设计测试用例的唯一依据是软件的**规格说明**。
- 与软件的具体实现无关
  - 优点:
    - 如果实现发生变化，测试用例仍然有用
    - 测试用例的开发可以与实现并行进行
  - 缺点: (可用下页维恩图理解)
    - 测试用例之间可能冗余度大;
    - 测试有漏洞: 程序实现的某部分未被测试到
    - 不能发现: 多余缺陷(即:程序实现了未描述的行为)

黑盒测试


规格说明
程序
规格说明
程序
测试用例
（方法A)
测试用例
方法B)
方法A覆盖的区域比方法B覆盖的区域大，但两种方法的覆盖区域只能在规格说明部分

3.5.2 白盒测试

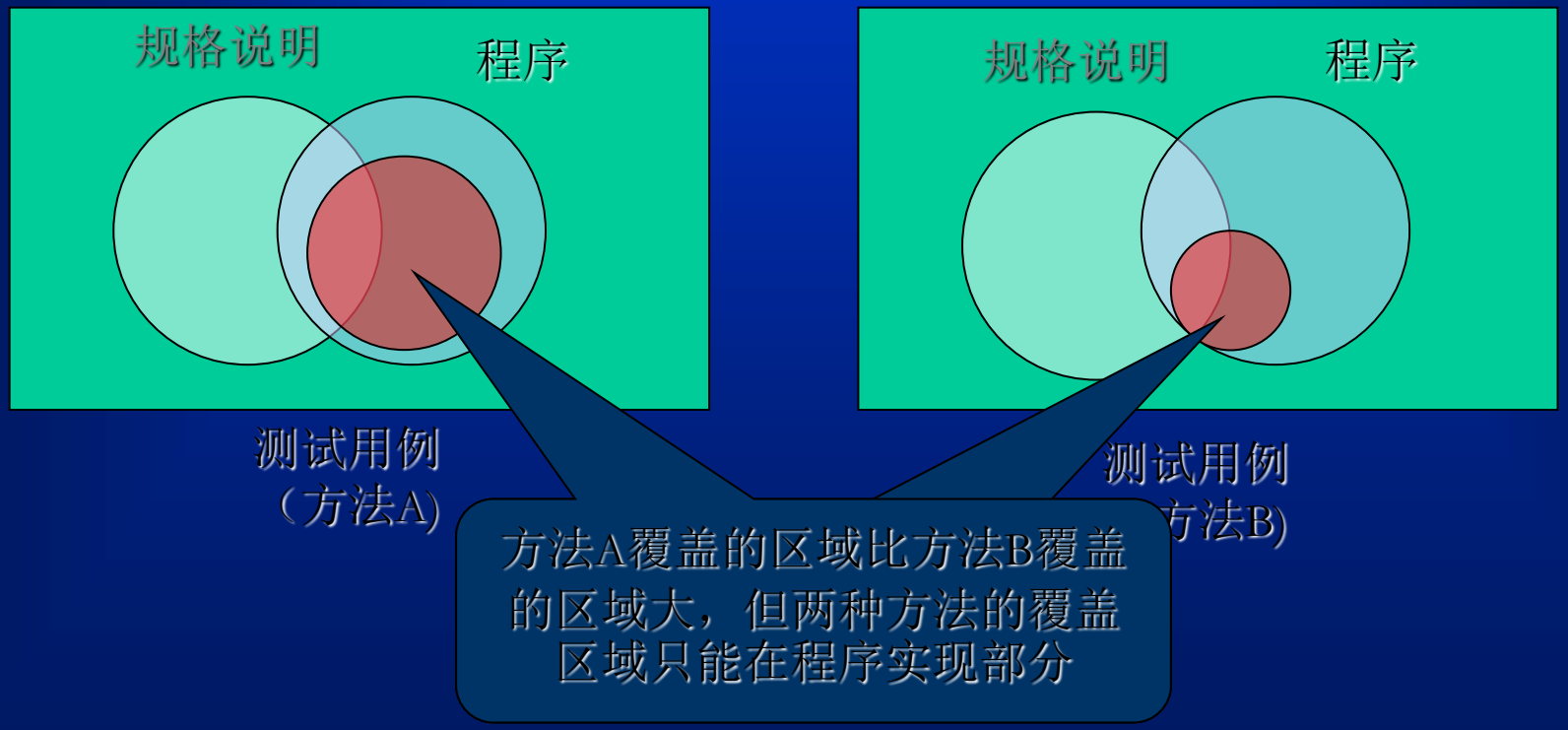


程序实现是已知的：把系统看做一个透明的盒子，利用程序内部的逻辑结构及有关信息，设计或选择测试用例.

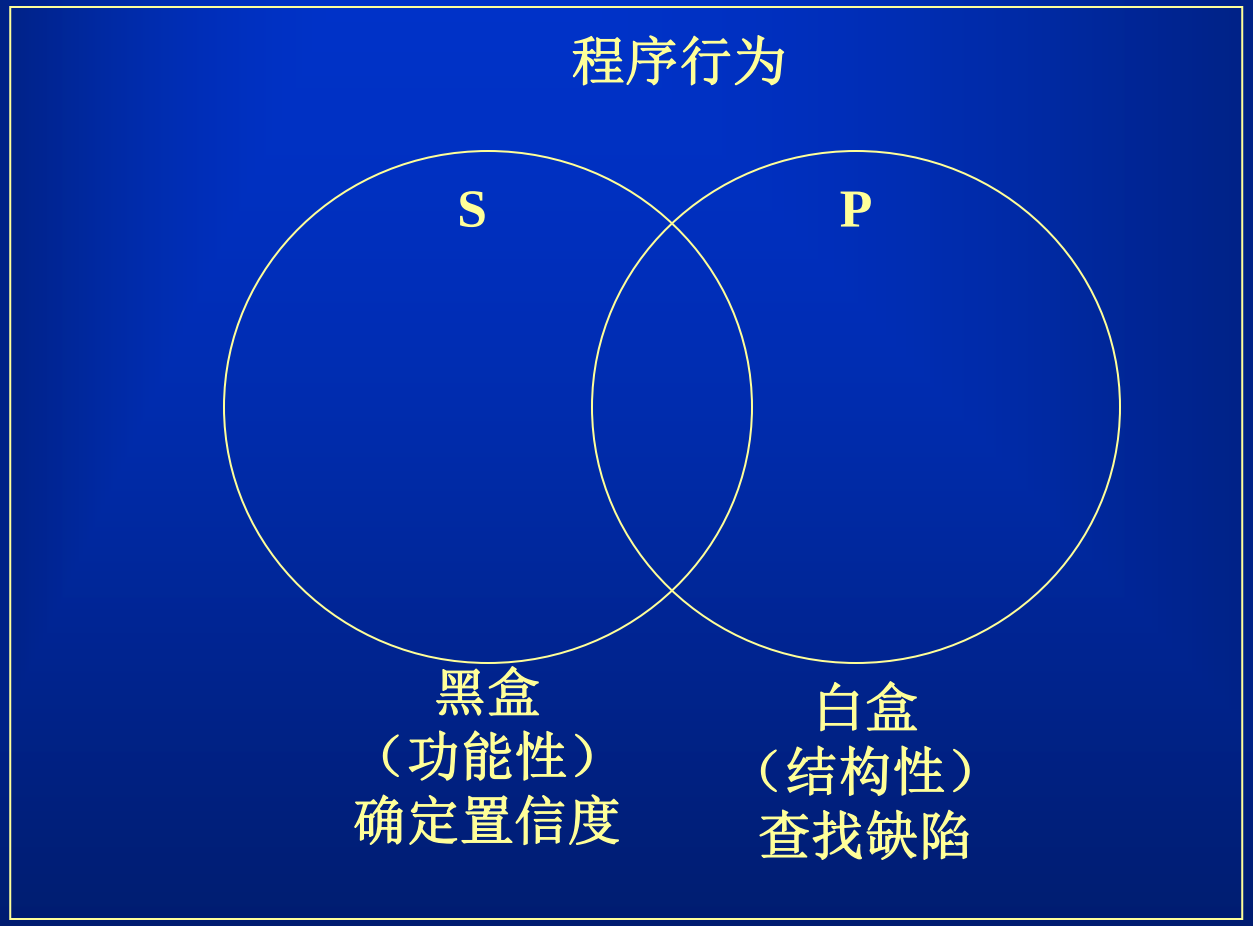
白盒测试的依据

- 设计测试用例的唯一依据是**程序实现**(程序源代码)
  - 优点:
    - 具有覆盖率指标
  - 缺点: (可用下页维恩图理解)
    - 不能发现: 遗漏缺陷(即:程序未能实现已描述的行为)

白盒测试


规格说明
程序
规格说明
程序
测试用例
（方法A)
测试用例
方法B)
方法A覆盖的区域比方法B覆盖的区域大，但两种方法的覆盖区域只能在程序实现部分

黑盒测试与白盒测试比较



单独使用都有其局限性；

最好的方法：结合两种方法。

3.5 测试用例的设计

测试技术:

- 黑盒测试
- 白盒测试
- 灰盒测试：
  - 通常指在无法直接获得源代码的情况下，通过一些软件制品、或者通过反编译反汇编等手段，获得代码的部分结构信息，从而进行测试。
  - 例如：对java、android等写的程序进行反编译，对嵌入式程序进行反汇编等。

4 错误与缺陷分类

缺陷**分类的方法**:

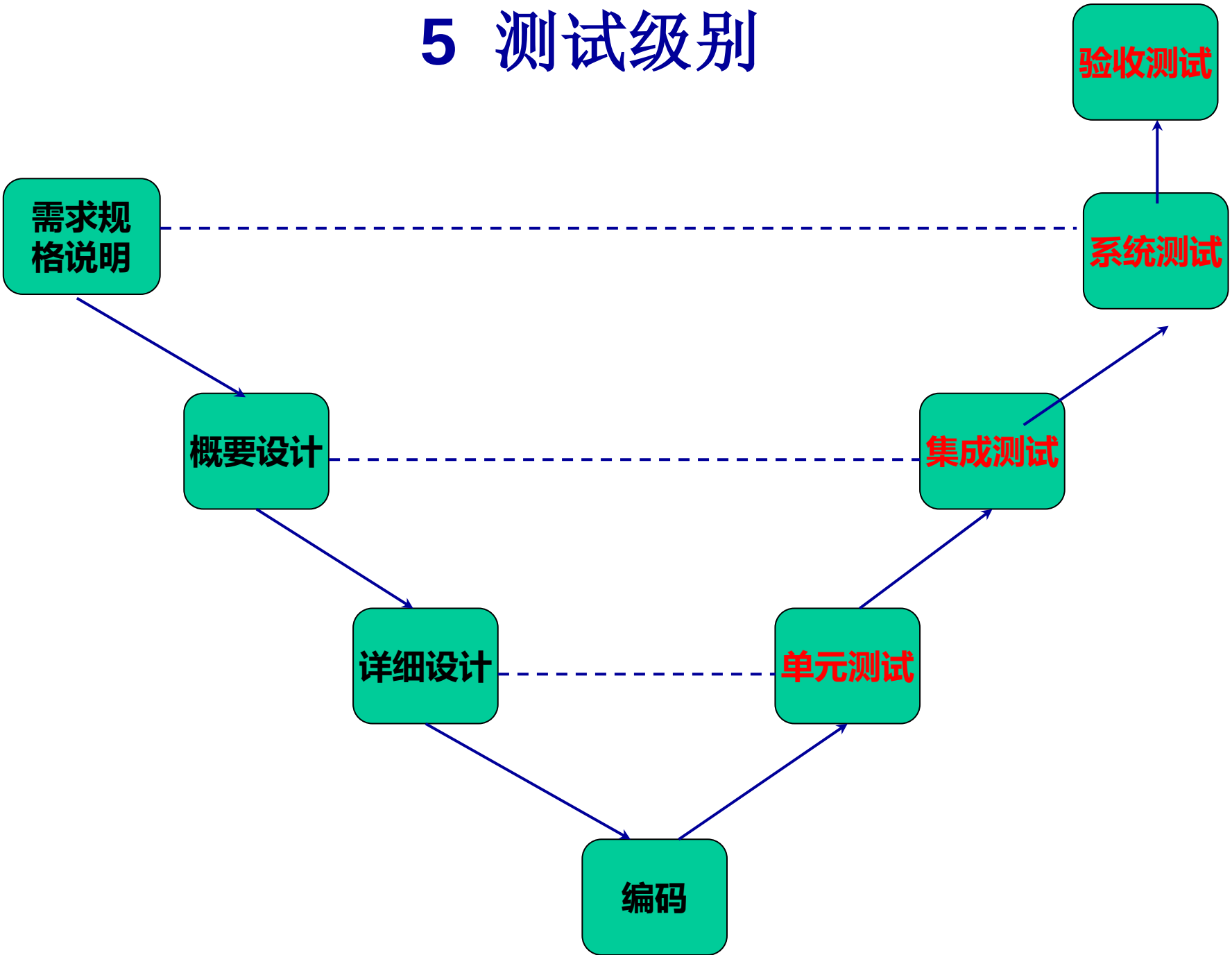
- 以出现相应错误的开发阶段来划分;
- 以相应失效产生的后果来划分;
- 以解决难度来划分;
- 以不解决会产生的风险来划分;

back

根据严重程度分类的缺陷

1. 轻微　　词语拼写错误
2. 中等　　误导或重复信息
3. 使人不悦　　被截断的名称
4. 影响使用　　有些交易没有处理
5. 严重　　丢失交易
6. 非常严重　　不正确的交易处理
7. 极为严重　　经常出现“非常严重”的错误
8. 无法忍受　　数据库破坏
9. 灾难性　　系统停机
10. 容易传染　　扩展到其它系统的系统停机

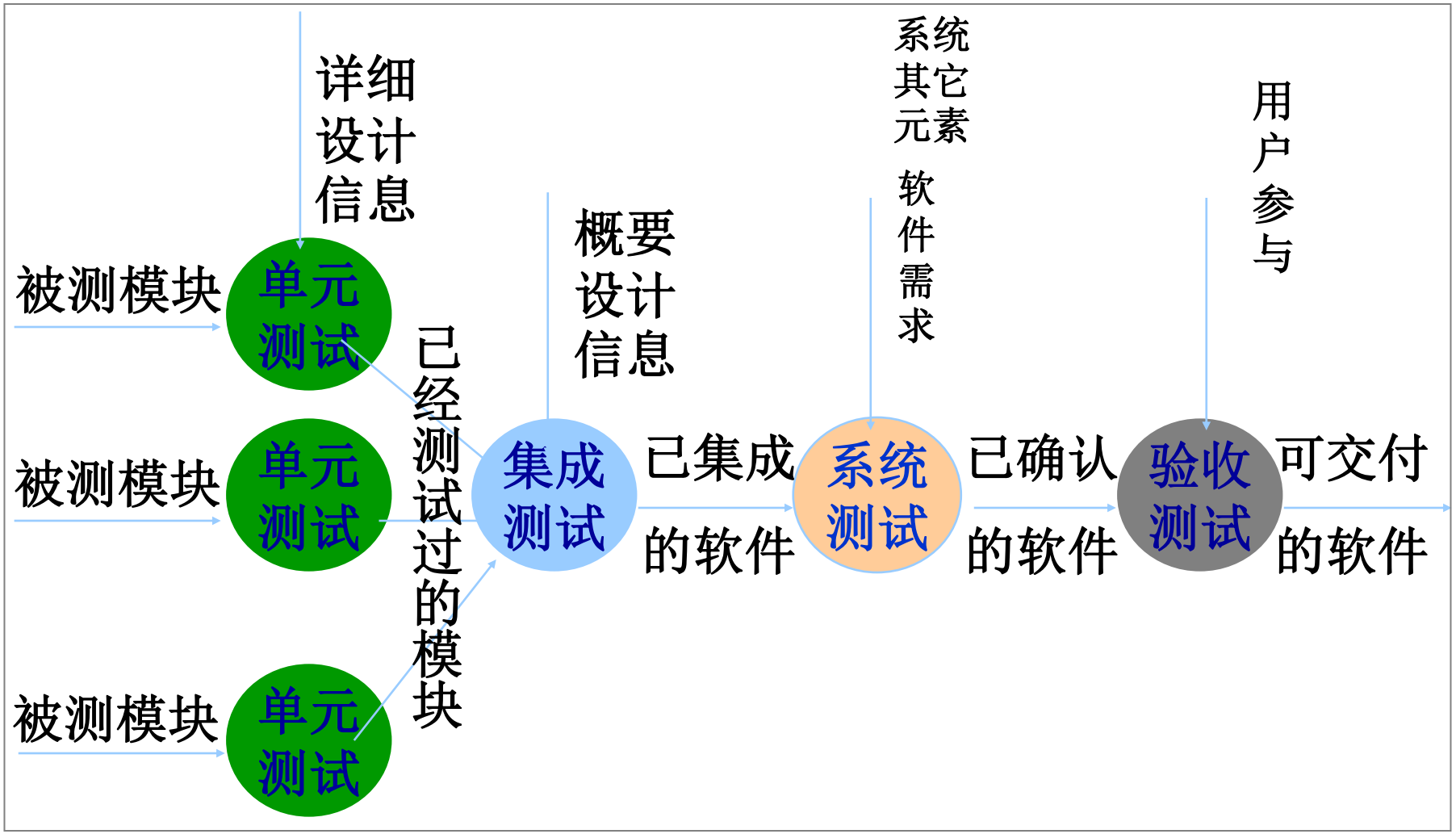
5 测试级别

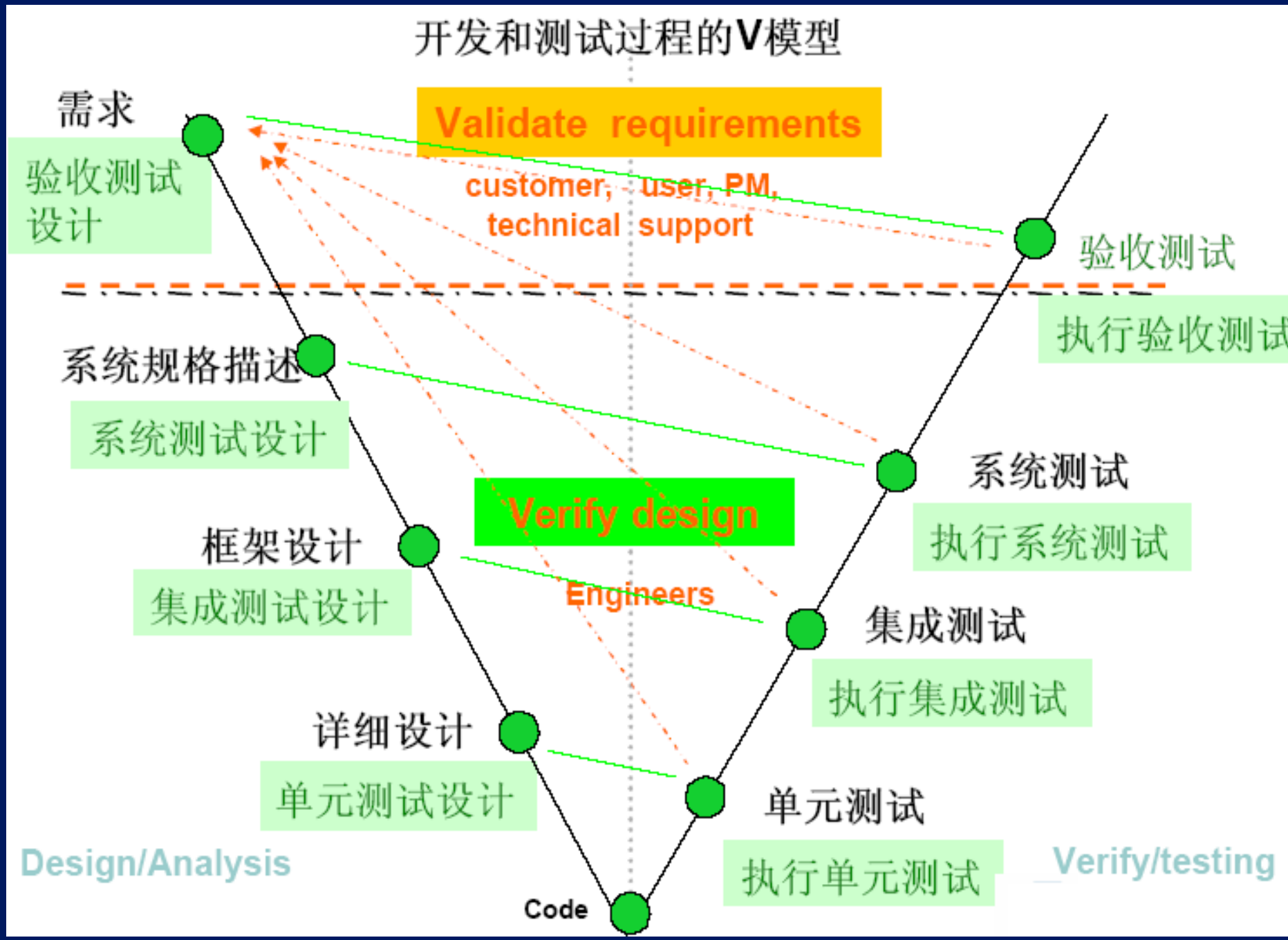


- **单元测试**（Unit test）
  - 正确，规范；白盒测试为主
  - 模块测试 （模块的粒度）
- **集成测试**（Integration test）
  - 将通过单元测试的多个模块组合,形成更大的模块或子系统或产品，然后进行测试，如:模块接口、代码和界面的规范等。
- **系统测试**(System test):
  - 测试软件系统和其它系统元素（硬件、数据库和人机交互信息）的配合, 以及整个系统的测试规范，如：安装测试，配置测试，界面测试，可用性测试，安全性测试等。

- **确认测试**: (Validation test)
  - 根据需求规格说明书，测试系统对用户需求的满足程度。
- **验收测试**（Acceptance test）
  - 主要由用户执行,检查是否符合验收标准。
  - 黑盒测试

软件测试的级别


详细
设计
信息
被测模块
单元
测试
被测模块
单元
测试
被测模块
单元
测试
已经
测试过的
模块
概要
设计
信息
集成
测试
已集成
的软件
系统
其它
元素
软件
需求
系统
测试
已确认
的软件
用户
参与
验收
测试
可交付
的软件


开发和测试过程的V模型
需求
验收测试
设计
Validate requirements
customer, user, PM,
technical support
验收测试
执行验收测试
系统规格描述
系统测试设计
系统测试
执行系统测试
Verify design
框架设计
集成测试设计
Engineers
集成测试
执行集成测试
详细设计
单元测试设计
单元测试
执行单元测试
Design/Analysis
Code
Verify/testing

Software V & V

Validation确认

- **“Are we building the right product?”**
  - “Determination of the correctness of the final program or software produced from a development project with respect to the user needs and requirements”
  - 从外部用户的角度，强调需求

Verification验证

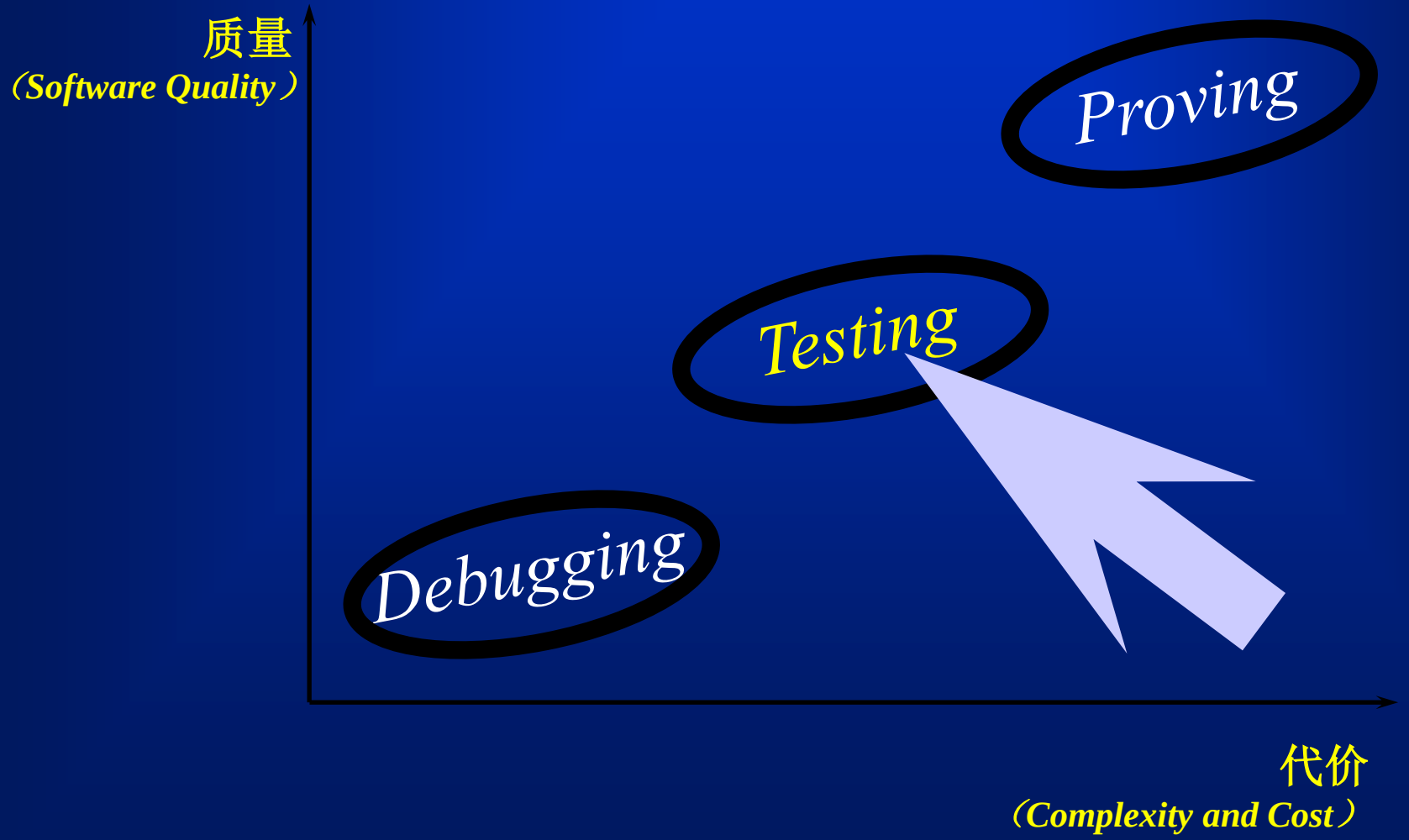
“Are we building the product right?”

- “The demonstration of consistency, completeness, and correctness of the software at each stage and between each stage of the development life cycle.”
- 从内部开发的角度，强调过程

Software Verification验证

- Software testing测试
  - Executing input cases to a program and checking the outputs to decide whether they are correct
- Software debugging
  - Diagnosing the precise nature of known errors and then correcting them.
- Proof of correctness
  - Mathematically proving the programs in question to be correct

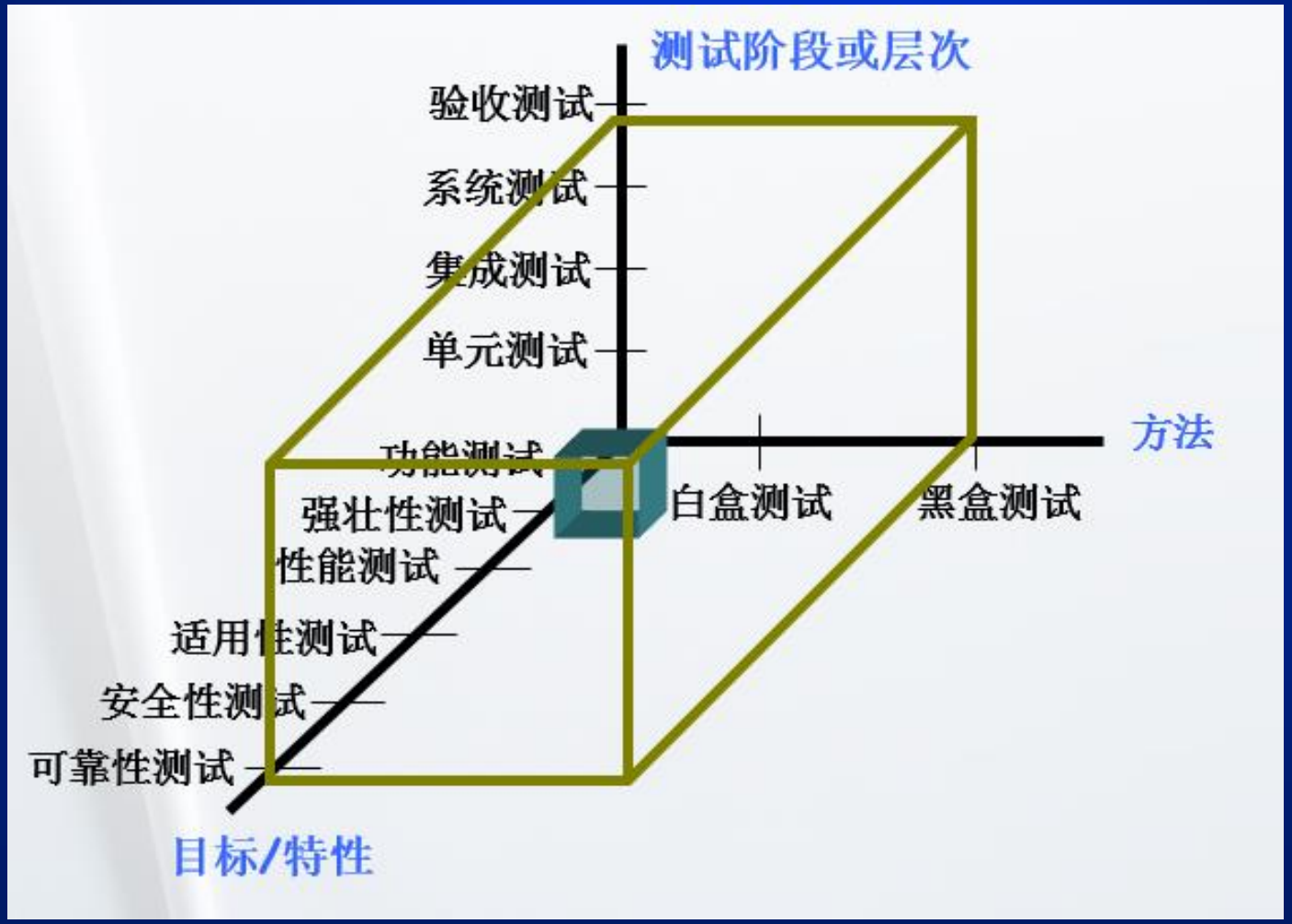
Software Verification



Software V & V
General References

- W.R. Adrion, M.A. Branstad, and J.C. Cherniavsky, “Validation, verification and testing of computer software”, *ACM Computing Surveys* **14** (2): 159-192 (1982),
- B. Hailpern and P. Santhanamp, “Software debugging, testing, and verification”, Special Issue on Software Testing and Verification, *IBM Systems Journal* **41** (1) (2002). Available at `http://www.research.ibm.com/journal/sj41-1.html`

6. 软件测试的分类



From 朱少民 slides

6. 软件测试的分类

常见分类:

- 6.1 静态测试与动态测试
- 6.2 人工测试与自动测试
- 6.3 主动测试与被动测试

back

6.1 动态测试与静态测试

- **动态测试** (Dynamic testing)
  - 通过**运行被测试程序**来达到测试的目的。
  - **本课程重点**
  - **动态黑盒；动态白盒**

- **静态测试** (Static testing)
  - **不运行**被测试程序，通过其它手段达到测试的目的，如：检查和审阅。
  - **静态审查**“需求规格说明书”
    - 从用户角度出发，依据行业、国家、公司的标准与规范，检查描述的属性和用语，填写“检查表”。
  - **静态扫描与分析**

6.2 自动测试与人工测试

- **自动测试：利用自动化测试工具**将大量的**重复性工作**交给计算机去完成。
  - 节约资源，提高效率，可重用

    节约人力、物力、资金、时间等资源，且测试脚本可重复利用（可以是不同项目）。

- **人工测试：**
  - 走查（Walk-through）
  - 互评（Peer Review）
  - 会审（Inspection）

    对照实验发现：人工走查和审查会平均能找出被测程序的38%错误。IBM代码审查会的查错效率高达80%

    - 1. 请程序员逐个语句地**讲述**程序的逻辑结构。其间大家**提出问题**加以追究，以断定错误是否存在。实际上，有许多错误是在叙述的过程中被程序员自己发现的。
    - 2. 根据**常见程序错误检验单**分析程序。

6.2 主动测试与被动测试

- **主动测试：** 测试人员设计和运行 测试用例。主动向被测试对象发送请求、用数据、事件驱动被测试对象的行为，从而验证被测试对象的反应或输出结果

- **被动测试：** 测试人员不设计测试用例。软件在线运行在实际环境中，测试者不干预其运行，而是被动地监控其运行，通过一定的被动机制来获得运行的数据，包括输入、输出数据.
  - 关键：建立监控程序、分析所得的数据
  - 用于：性能测试、在线测试

例：在线测试 （from Zhu’s Slide）

7 专业测试人员的责任与要求

- QA/测试经理：人员管理，资源调配、测试方法改进等；
- 实验室管理人员：设置、配置和维护实验室的测试环境
- 内审员：审查流程，建立测试模板，跟踪缺陷测试报告的质量等；
- 测试组长：负责项目的管理、测试计划、测试用例、任务安排等；
- 测试设计人员/资深测试工程师，产品设计规格说明书的审查、测试用例的设计、技术难题的解决、培训和指导、实际测试任务的执行；
- 一般（初级）测试工程师，执行测试用例和相关的测试任务。

对测试人员的要求

- **技术，编程能力**
- 责任感、耐力
- 沟通能力、理解能力
- 分析问题能力（批判性思维）
- 项目管理能力
- 组织能力
- ……

优秀测试工程师的素质

- 高度的责任感
- 非常好的沟通能力、幽默感
- 技术能力、自信心、耐心
- 怀疑一切的精神、勤奋精神
- 洞察力、适度的好奇心
- 反向思维和发散思维能力、记忆力
- 自我学习能力、创新能力等

Certified Software Test Professional (CSTP)

Body of Knowledge 知识体系

- **Principles of Software Testing**
  - Levels of Testing
  - Testing client/server applications
  - Testing Internet and web applications
  - Testing object-oriented applications
  - Testing embedded systems
  - The testing life cycle
- **Test Design**
  - Code-based test case design techniques
  - Requirement-based test case design techniques
  - Test design specification
- **Managing the Testing Process**
  - Planning
  - Scheduling
  - Reporting
  - Resources
  - Risk Management
  - Measuring and improving the test process
- **Test Executions and Defect Tracking**
  - Test scripting
  - Reporting
  - Defect tracking
- **Requirement Definitions, Refinement and Verification**
  - Writing testable requirements
  - Exploring requirements
  - Refining requirements
  - Defining requirements
  - Requirement verification
  - Requirement tractability
- **Test Automation**
  - Tool evaluation and selection
  - Architectures
  - Automation standards and guidelines
  - Planning the test automation process
  - Automation team roles
- **Static Testing (Inspections, Reviews, and Walkthroughs)**
  - Types of static testing
  - The process of static testing
  - Defect data analysis
  - Improving the process

小结

软件测试概述

1. 软件测试的背景
2. 软件测试的含义
3. 软件测试的过程
4. 错误与缺陷分类
5. 测试级别
6. 软件测试的分类
7. 对测试人员的要求

在若干被测模块中，已找出更多软件缺陷的模块，所残留的软件缺陷就更少。

软件测试只能发现错误，而不能保证测试后的软件没有错误。

程序员兼任测试员可以提高工作效率。

测试用例应由测试输入数据和对应的实际输出结果这两部分组成。

动态测试是通过运行被测程序来发现程序的错误。

软件测试是为了验证软件已正确地实现了用户的需求。