第五部分 软件编码、测试与质量保障

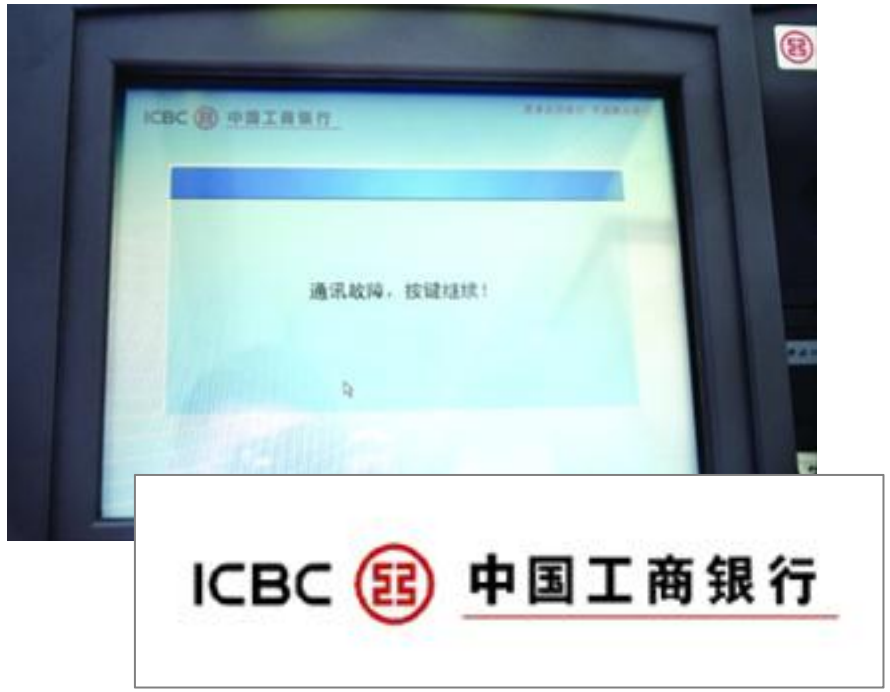
- 5.1 软件编程
- 5.2 软件测试
- 5.3 白盒测试
- 5.4 黑盒测试
- 5.5 变异测试
- 5.6 性能测试

5.2 软件测试

- 软件测试概述
  - 软件缺陷术语
  - 软件测试概念
  - 软件测试基本原则
  - 软件测试团队
- 软件测试策略

关于软件质量

The average software product released on the market is not error free.



软件缺陷

比尔·盖茨曾说：“如果通用汽车能用计算机产业的速度来发展技术，那我们今天的汽车就只要25美元，每加仑汽油可以跑1000英里。

1. 没毛病。但是你会想要你的汽车每天毫无理由地撞车两次吗？
2. 你想要安全气囊弹出之前问：你确定吗？
3. 车子熄火的时候唯一的解决办法只有重新安装引擎？

。。。

术语解释

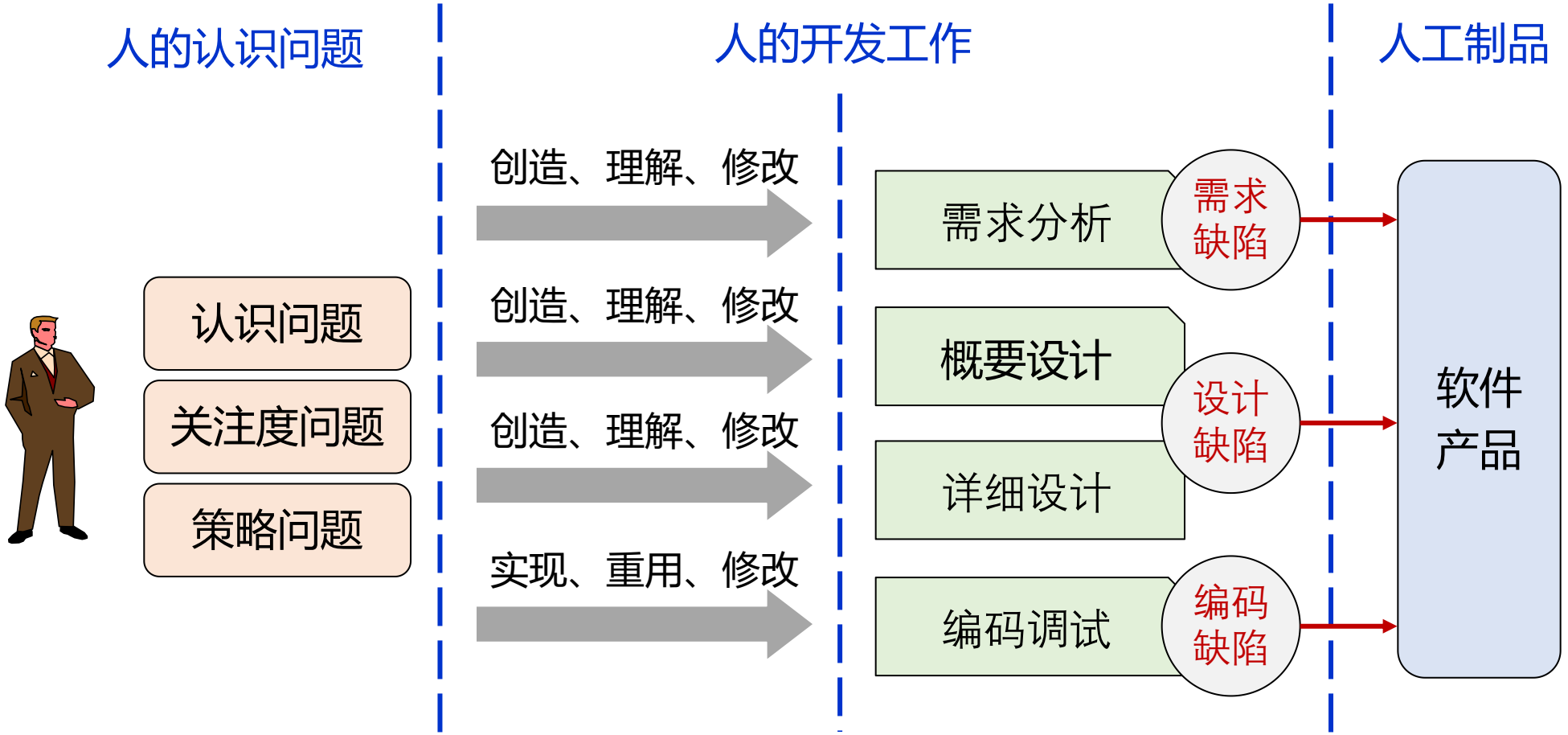
错误（Error）：在软件生存期内的不希望或不可接受的人为错误，其结果是导致软件缺陷的产生。

缺陷（Defect）：软件缺陷是存在于软件产品之中的那些不希望或不可接受的偏差，其结果是软件运行于某一特定条件时出现故障。

故障（Fault）：软件运行过程中出现的一种不希望或不可接受的内部状态，若无适当措施（容错）加以及时处理，便产生软件失效。

失效（Failure）：软件运行时产生的一种不希望或不可接受的外部行为结果。

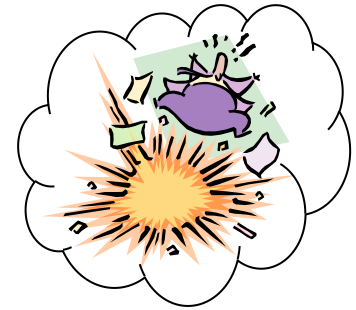
软件缺陷的产生


人的认识问题
人的开发工作
人工制品
认识问题
关注度问题
策略问题
创造、理解、修改
创造、理解、修改
创造、理解、修改
实现、重用、修改
需求分析
概要设计
详细设计
编码调试
需求缺陷
设计缺陷
编码缺陷
软件产品

软件缺陷的演化

举例：爱国者导弹

1991年2月，一次拦截失利
造成28名美国士兵丧生



软件系统

计时用系统时钟（1/10秒）
并以整数表达

缺少防范措施

0.1按十进制等于1/10，按二进制计算再转回来就成了209715/2097152。

缺陷 —激活→ 故障 —演变→ 失效

寄存器存储导致误差
$(0.000000095)_{10}$

$0.000000095 \times 100h \times 60 \times 60 \times 10 = 0.34s$

爱国者导弹的飞行速度约为2000米/秒，爱国者导弹与飞毛腿导弹的距离误差就是2000*0.3433=687米，而爱国者的杀伤力半径只有20米，这怎么可能拦截成功呢?

5.2 软件测试

- 软件测试概述
  - 软件缺陷术语
  - 软件测试概念
  - 软件测试基本原则
  - 软件测试团队
- 软件测试策略

软件测试

测试 —— 发现问题 —— 修复

什么是软件测试?

软件测试的目的是什么?

软件测试的概念

- **IEEE：** 测试是使用人工和自动手段来运行或检测某个系统的过程，其目的在于检验系统是否满足规定的需求或弄清预期结果与实际结果之间的差别。

该定义明确提出了软件测试以“检验是否满足需求”为目标。

软件测试的两种思维

正向思维：验证软件正常工作

- 评价一个程序或系统的特性或能力并确定是否达到预期的结果。
- 在设计规定的环境下运行软件的所有功能，直至全部通过。

开发者

系统的构建者，理解系统
测试是“温和”的，试图证明系统的正确性

软件测试的两种思维

逆向思维：假定软件有缺陷

- 测试是为了发现错误而针对某个程序或系统的执行过程。
- 寻找容易犯错地方和系统薄弱环节，试图破坏系统直至找不出问题。

独立测试组

必须学会系统
测试是破坏性的，试图证明系统存在错误

软件测试的两种思维

你完成了一个能够对5个数字从小到大的排序算法。你会怎么测试？

软件测试的目的

直接目标：发现软件错误

- 以最少的人力、物力和时间找出软件中潜在的各种错误和缺陷，通过修正这些错误和缺陷提高软件质量，回避软件发布后由于潜在的软件错误和缺陷造成的隐患所带来的商业风险。

期望目标：检查系统是否满足需求

- 测试是对软件质量的度量和评估，以验证软件的质量满足客户需求的程度，为用户选择和接受软件提供有力的依据。

附带目标：改进软件过程

- 通过分析错误产生的原因，可以帮助发现当前开发所采用的软件过程缺陷，从而进行软件过程改进；通过分析整理测试结果，可以修正软件开发规则，为软件可靠性分析提供依据。

测试的局限性

测试的不彻底性

- 测试只能说明错误的存在，但不能说明错误不存在
- 经过测试后的软件不能保证没有缺陷和错误

测试的不完备性

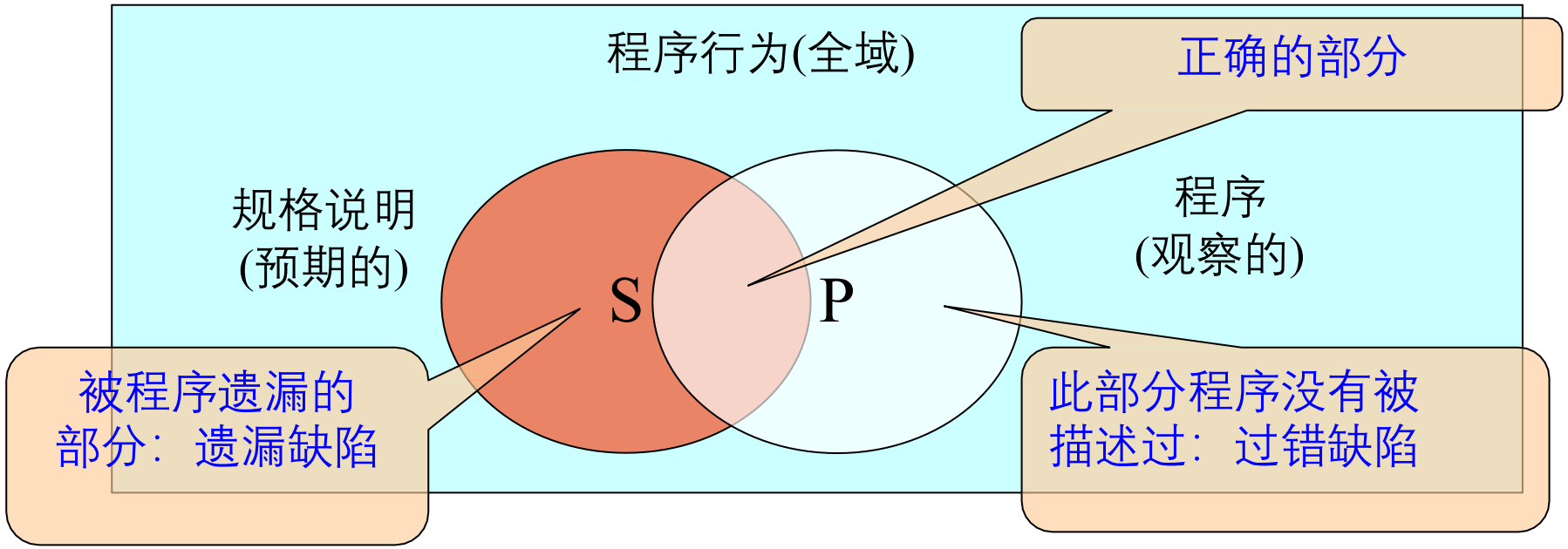
- 测试无法覆盖到每个应该测试的内容
- 不可能测试到软件的全部输入与响应
- 不可能测试到全部的程序分支的执行路径

测试作用的间接性

- 测试不能直接提高软件质量，软件质量的提高要依靠开发
- 测试通过早期发现缺陷并督促修正缺陷来间接地提高软件质量

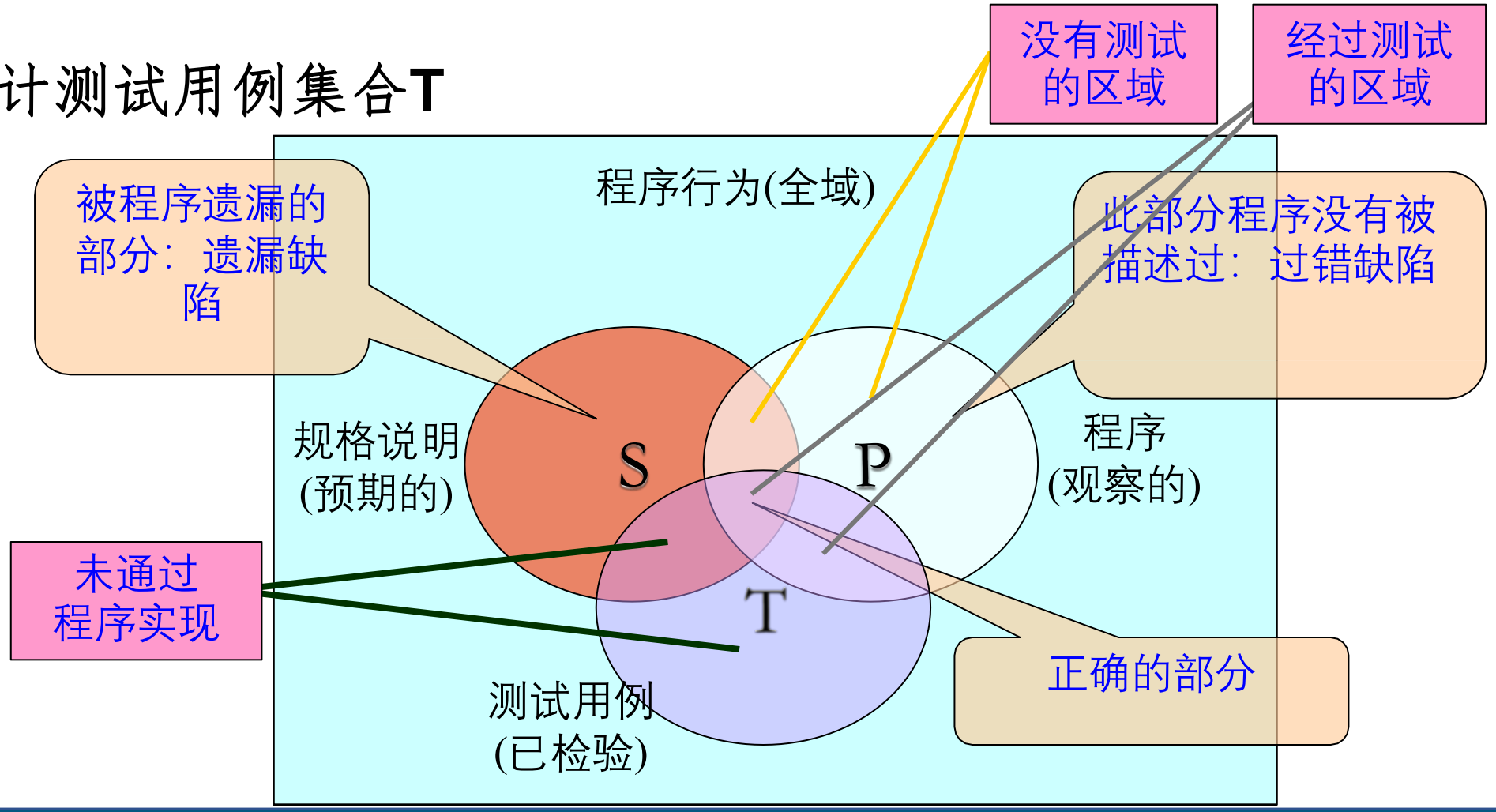
用Venn Diagram来理解测试

- 考虑一个程序行为全域，给定一段程序及其规格说明
  - 集合S是所描述的行为；
  - 集合P是用程序实现的行为；



用Venn Diagram来理解测试

设计测试用例集合T



5.2 软件测试

- 软件测试概述
  - 软件缺陷术语
  - 软件测试概念
  - 软件测试基本原则
  - 软件测试团队
- 软件测试策略

软件测试原则

原则1： 测试用例中一个必需部分是对预期输出或结果的定义。

原则2： 编程人员应当避免测试自己编写的程序。

原则3： 编写软件组织不应测试自己编写的软件。

原则4： 应当彻底检查每个测试的执行结果。

原则5： 测试用例不仅根据有效的和预期的输入情况，也应根据无效的和未预料到的输入情况。

原则6： 检查程序是否“未做应该做的”仅是测试的一半，另一半是检查是否“做了不应该做的”。

原则7： 应避免测试用例用后即弃，除非是一次性软件。

原则8： 计划测试工作时不应默许假定不会发现错误。

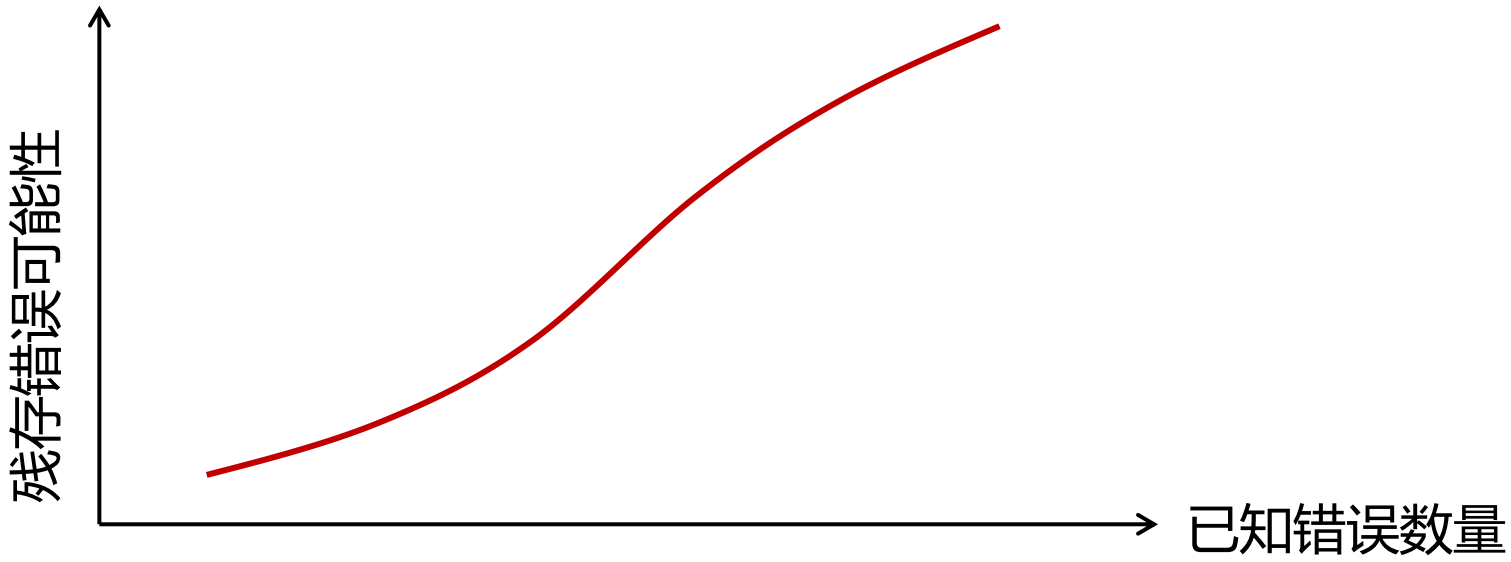
原则9： 程序某部分存在更多错误的可能性，与该部分已发现错误的数量成正比。

原则10： 软件测试是一项极富创造性、极具智力挑战性的工作。

缺陷的集群性

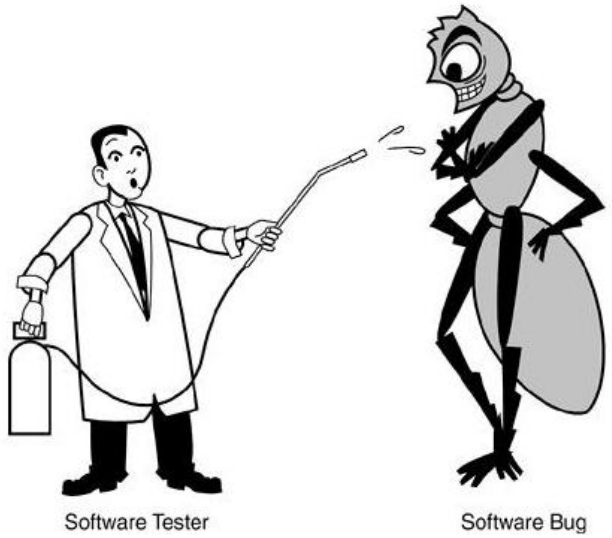
软件错误具有聚集性，对存在错误的部分应重点测试。

- 80/20原则：80%的软件错误存在于20%的代码行中
- 经验表明：测试后程序中残留的错误数目与该程序中已检出的错误成正比。



杀虫剂悖论

用同样的测试用例多次重复进行测试，最后将不再能够发现新的缺陷。



如同给果树喷撒农药，为了杀灭害虫只打一种杀虫药，虫子就会有抗体而变得适应，于是杀虫剂将不再发挥作用。

测试用例需要定期评审和修改，同时要不断增加新的不同测试用例来测试软件的不同部分，从而发现更多潜在的缺陷。

5.2 软件测试

- 软件测试概述
  - 软件缺陷术语
  - 软件测试概念
  - 软件测试基本原则
  - 软件测试团队
- 软件测试策略

软件测试团队的任务

软件测试与质量保证合二为一

① 发现软件程序、系统或产品中所有的问题

② 尽早地发现问题

③ 督促开发人员尽快地解决程序中的缺陷

④ 帮助项目管理人员制定合理的开发计划

⑤ 对问题进行分析、分类总结和跟踪

⑥ 帮助改善开发流程，提高产品开发效率

⑦ 提高程序编写的规范性、易读性、可维护性等

测试人员角色

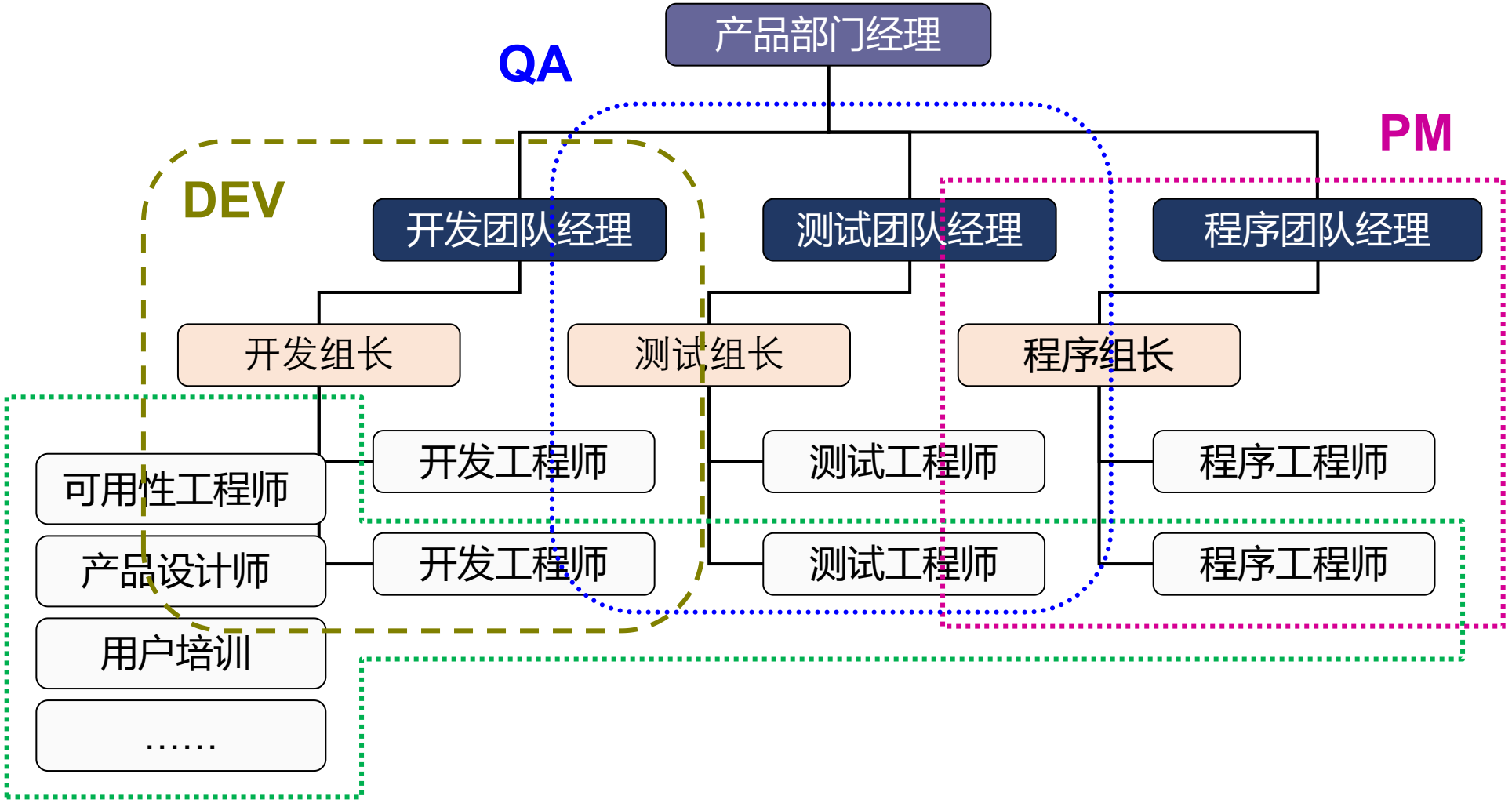
- 测试经理：建立和完善测试流程以及部门管理体制，审核测试项目并分配资源，监控和协调各项目的测试工作，负责与其他部门的协调和沟通工作。

- 测试组长：制定测试项目计划（包括人员、进度、软硬件环境和流程等），实施软件测试，跟踪和报告计划执行情况，负责测试用例质量，管理测试小组并提供技术指导。

- 测试工程师：理解软件产品的要求，对其进行测试以便发现软件中的错误，验证软件是否满足规格说明所描述的需求，编写相应的测试方案和测试用例。

- 测试工具工程师：编写软件测试工具，并利用所编写的测试工具对软件进行测试，或者为测试工程师开发测试工具。

举例：微软研发团队


产品部门经理
QA
PM
DEV
开发团队经理
测试团队经理
程序团队经理
开发组长
测试组长
程序组长
可用性工程师
产品设计师
用户培训
……
开发工程师
开发工程师
测试工程师
测试工程师
程序工程师
程序工程师

软件测试人员

核心素质：责任心

测试技术能力

- 掌握理论、方法、工具
- 开发测试工具、自动化测试框架、测试脚本等
- 掌握操作系统、网络、数据库、中间件等知识

非技术能力

- 沟通能力，协作精神
- 自信心、耐心、专心
- 洞察力、记忆力
- 怀疑精神，创造性
- 反向思维与发散思维
- 自我督促、幽默感

专业领域经验

- 了解业务知识
- 积累实践经验

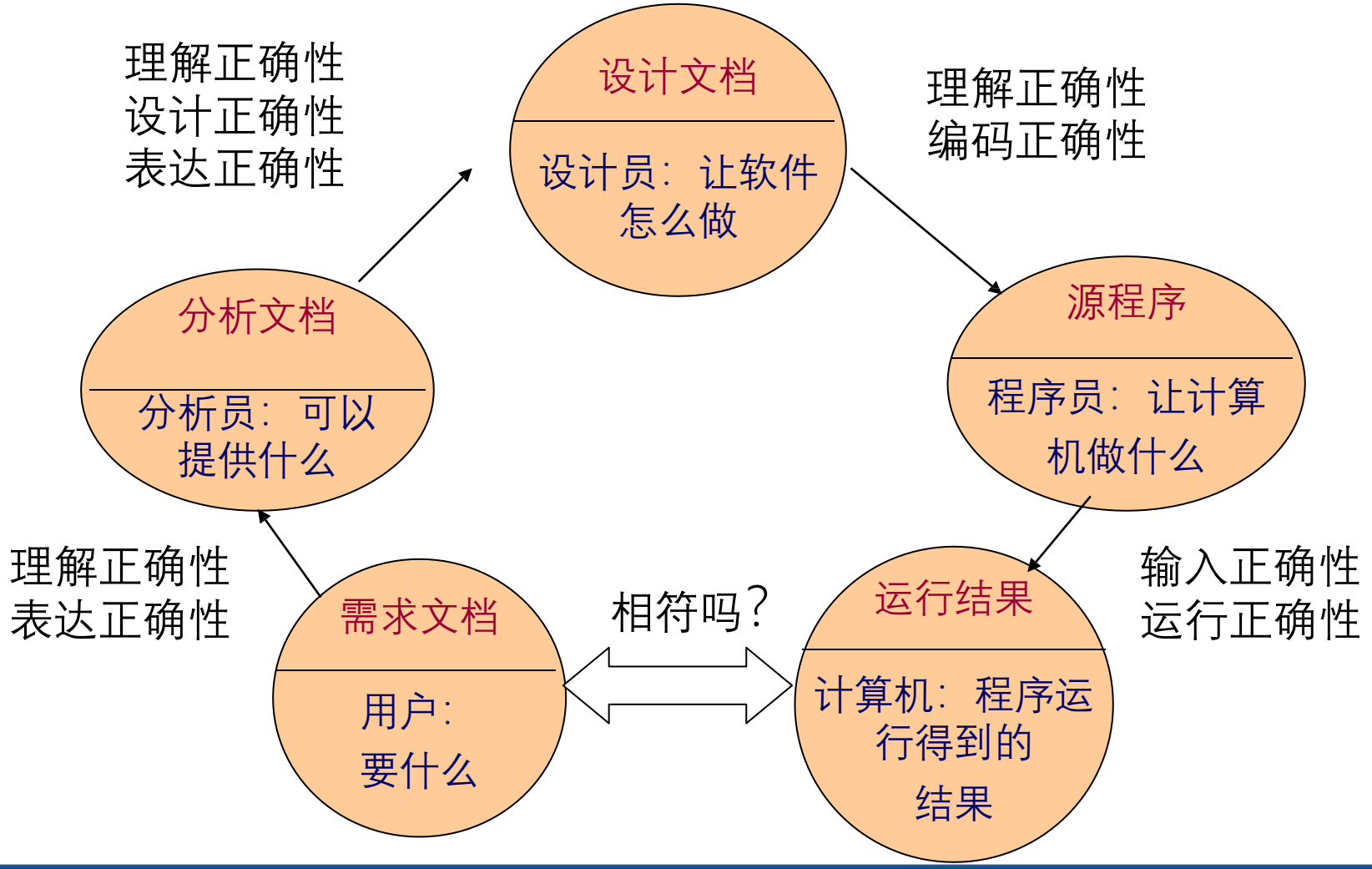
5.2 软件测试

- 软件测试概述
- 软件测试策略
  - 软件测试对象
  - 软件测试过程
  - 软件测试类型
  - 测试用例

软件测试的对象

- 软件测试并不等于程序测试，应贯穿于软件定义与开发的各个阶段。
- 测试对象包括：
  - 需求规格说明
  - 设计规格说明
  - 源程序

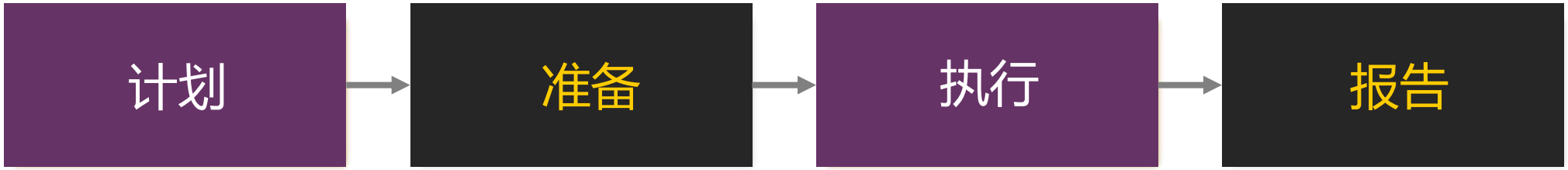
软件测试对象之间关系


理解正确性
设计正确性
表达正确性
设计文档
设计员：让软件怎么做
理解正确性
编码正确性
分析文档
分析员：可以提供什么
源程序
程序员：让计算机做什么
理解正确性
表达正确性
需求文档
用户：要什么
相符吗？
运行结果
计算机：程序运行得到的结果
输入正确性
运行正确性

5.2 软件测试

- 软件测试概述
- 软件测试策略
  - 软件测试对象
  - 软件测试过程
  - 软件测试类型
  - 测试用例

软件测试过程



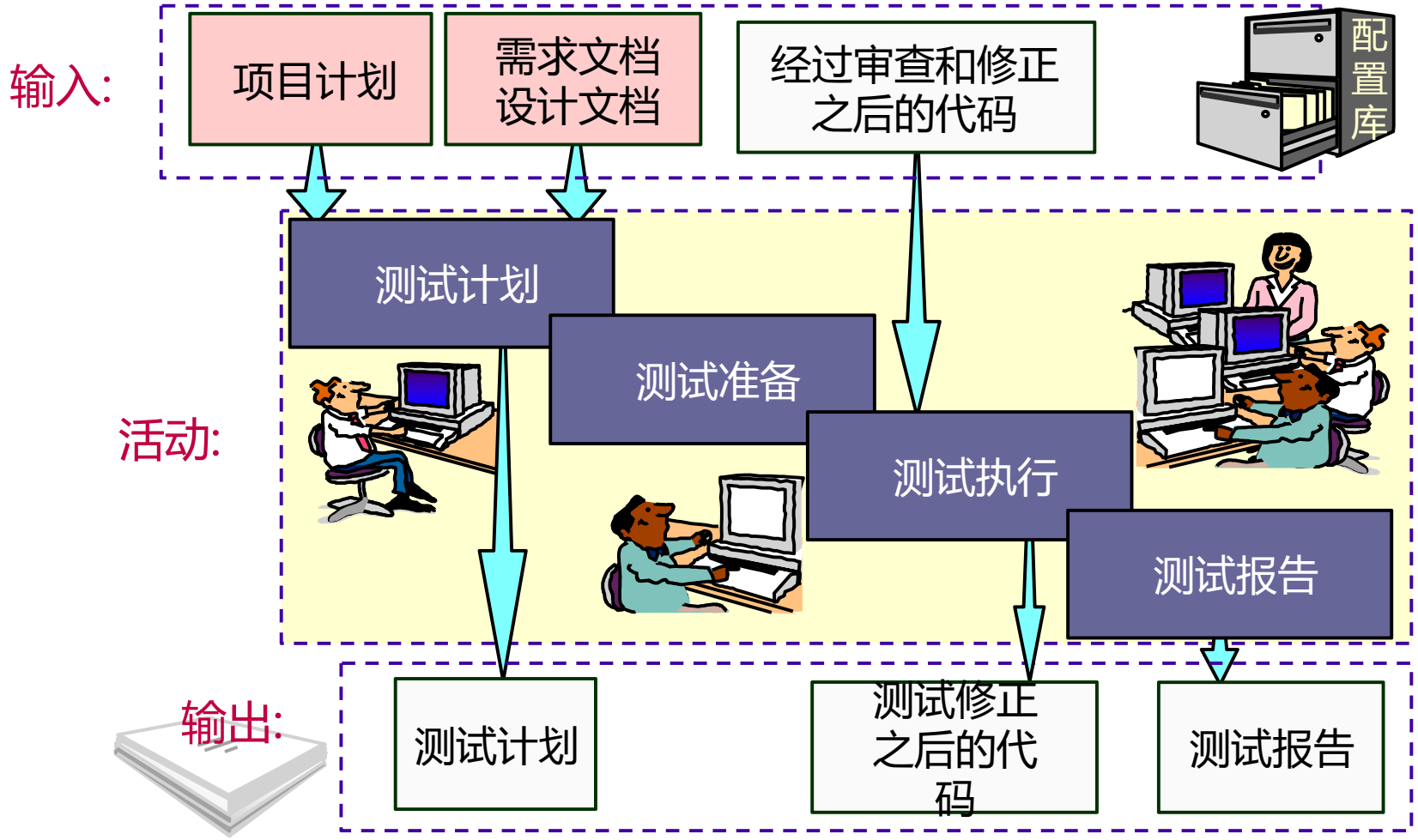
- 识别测试需求
- 分析质量风险
- 拟定测试方案
- 制定测试计划

- 组织测试团队
- 设计测试用例
- 开发工具和脚本
- 准备测试数据

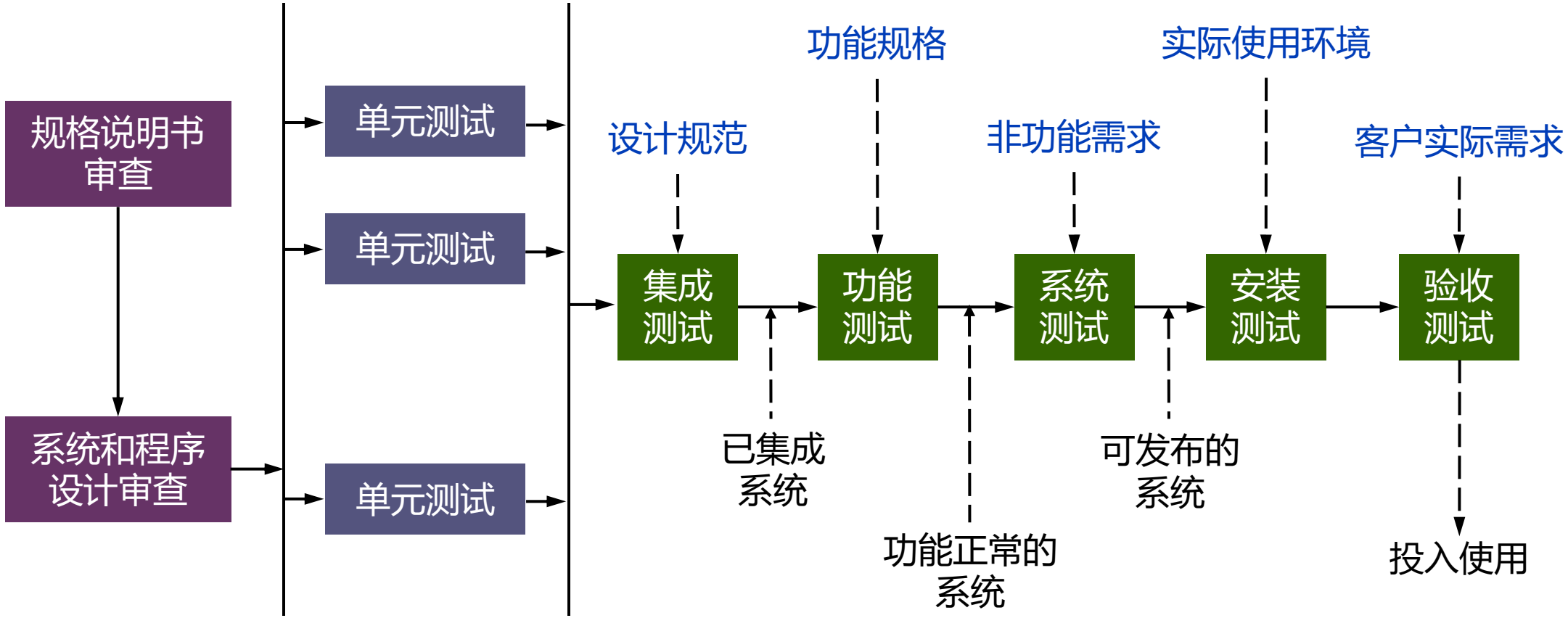
- 获得测试版本
- 执行和实施测试
- 记录测试结果
- 跟踪和管理缺陷

- 分析测试结果
- 评价测试工作
- 提交测试报告

软件测试活动


输入:
项目计划
需求文档
设计文档
经过审查和修正之后的代码
配置库
活动:
测试计划
测试准备
测试执行
测试报告
输出:
测试计划
测试修正之后的代码
测试报告

软件测试阶段


规格说明书审查
系统和程序设计审查
单元测试
单元测试
单元测试
设计规范
功能规格
非功能需求
实际使用环境
客户实际需求
集成测试
功能测试
系统测试
安装测试
验收测试
已集成系统
功能正常的系统
可发布的系统
投入使用

5.2 软件测试

- 软件测试概述
- 软件测试策略
  - 软件测试对象
  - 软件测试过程
  - 软件测试类型
  - 测试用例

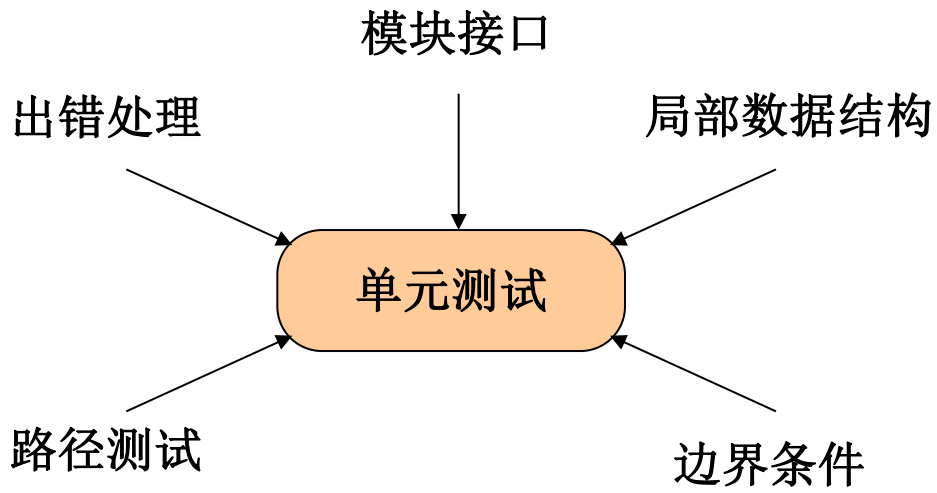
软件测试类型

测试对象角度	单元测试、集成测试、系统测试、验收测试
测试技术角度	黑盒测试（功能测试）、白盒测试（结构测试）
程序执行角度	静态测试、动态测试
人工干预角度	手工测试、自动化测试

单元测试

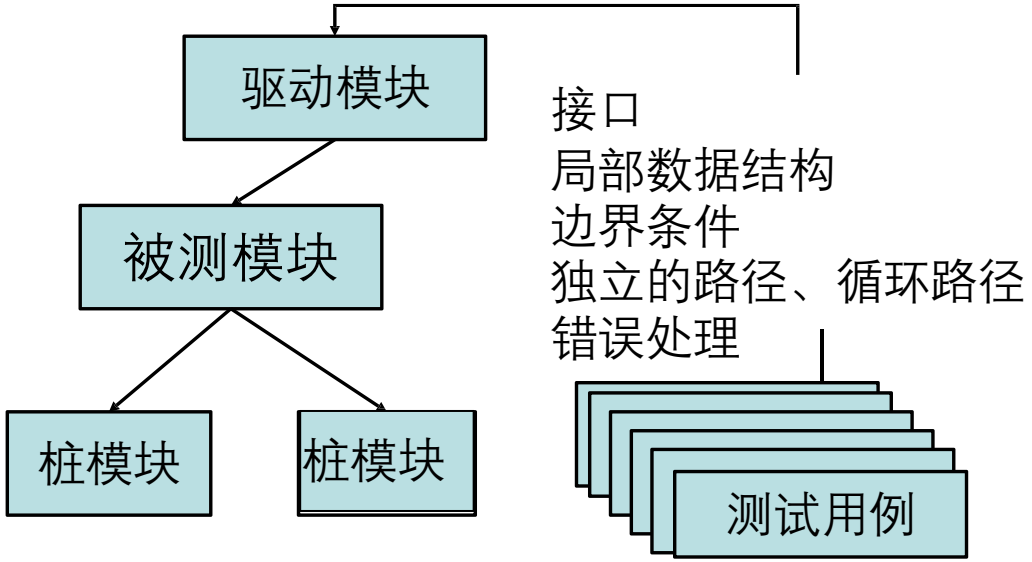
- 单元测试(Unit Testing)
  - 单元测试是对软件基本组成单元进行的测试，有时也称“组件测试”
  - 单元测试一般由编写该单元代码的开发人员执行，该人员负责设计和运行一系列的测试以确保该单元符合需求
- 单元测试的目的
  - 验证开发人员所书写的代码是否可以按照其所设想的方式执行而产出符合预期值的结果，确保产生符合需求的可靠程序单元

单元测试



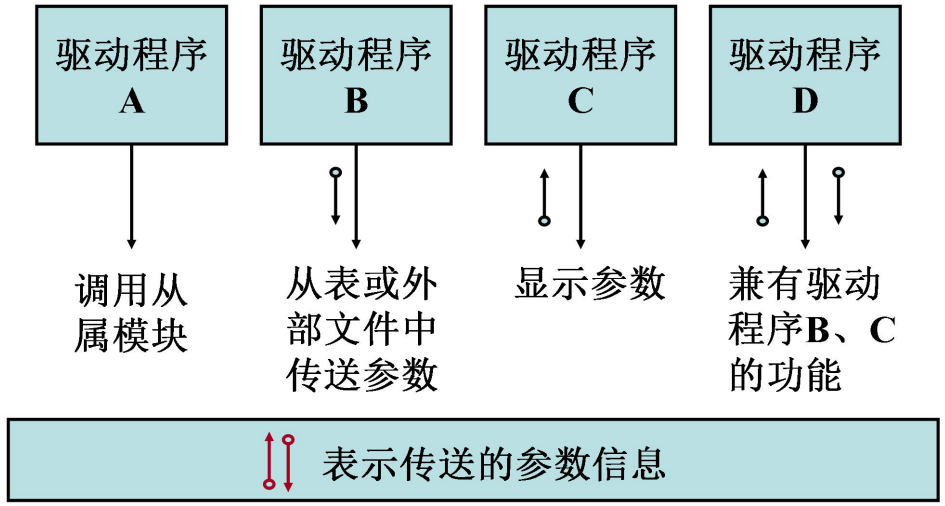
单元测试

- **单元测试环境**
  - 驱动模块(driver)：模拟被测模块的上一级模块，接收测试数据，把这些数据传送给所测模块，最后再输出实际测试结果；
  - 桩模块(stub)：模拟被测单元需调用的其他函数接口，模拟实现子函数的某些功能。

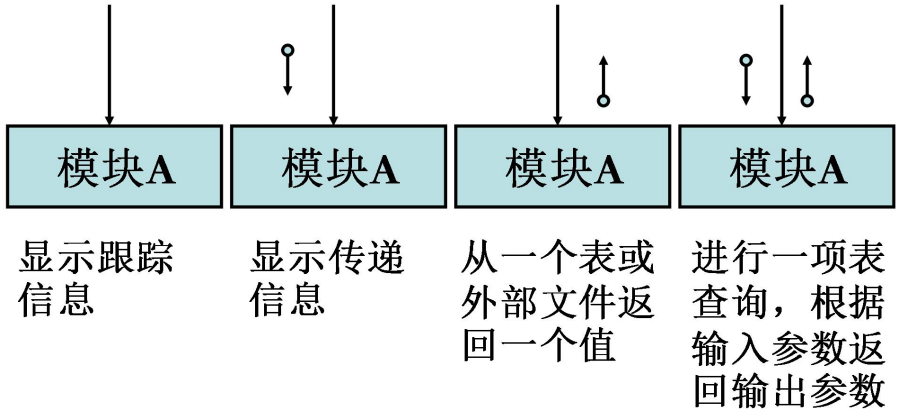


单元测试

- 单元测试的驱动模块/桩模块



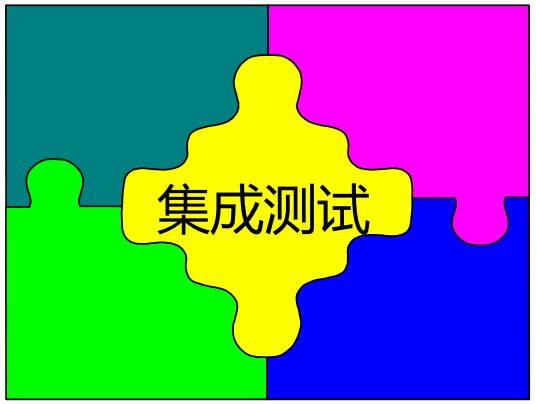
驱动模块：为被测模块提供数据



桩模块：只做少量的数据操作

集成测试

集成测试（Integration Testing）是在单元测试的基础上，将所有模块按照总体设计的要求组装成为子系统或系统进行的测试。



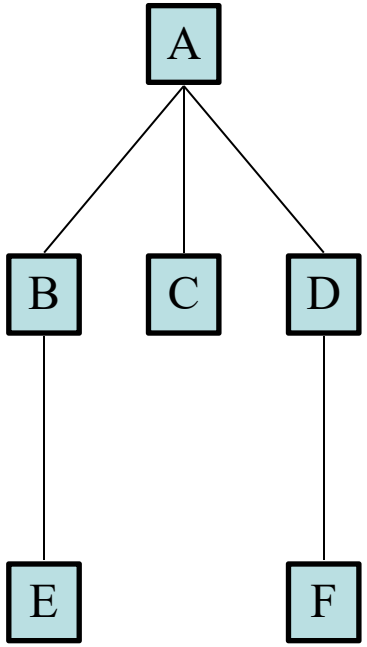
- 一次性集成方式：分别测试每个单元，再一次性将所有单元组装在一起进行测试。
- 渐增式集成方式：先对某几个单元进行测试，然后将这些单元逐步组装成较大的系统，在组装过程中边连接边测试。

集成测试的对象是模块间的接口，其目的是找出在模块接口上，包括系统体系结构上的问题。

集成测试的方法：整体集成

- **整体集成方式(非增量式集成)**
  - 把所有模块按设计要求一次全部组装起来，然后进行整体测试

- 例如：
  - Test A (with stubs for B, C, D)
  - Test B (with driver for A and stub for E)
  - Test C (with driver for A)
  - Test D (with driver for A and stub for F)
  - Test E (with driver for B)
  - Test F (with driver for D)
  - Test (A, B, C, D, E, F)



集成测试的方法：整体集成

- **优点：**
  - 效率高，所需人力资源少；
  - 测试用例数目少，工作量低；
  - 简单，易行；

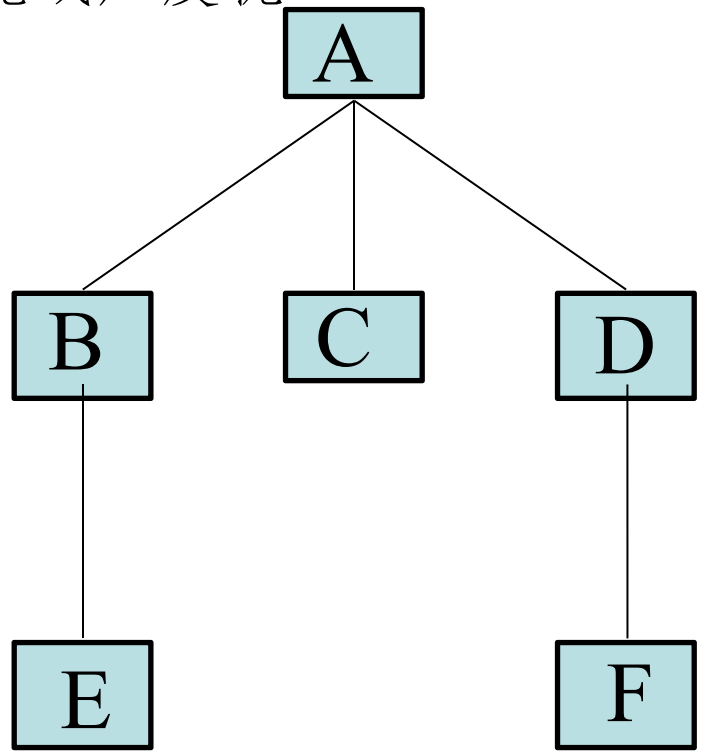
- **缺点：**
  - 可能发现大量的错误，难以进行错误定位和修改；
  - 即使测试通过，也会遗漏很多错误；
  - 测试和修改过程中，新旧错误混杂，带来调试困难；

集成测试的方法：增量式集成

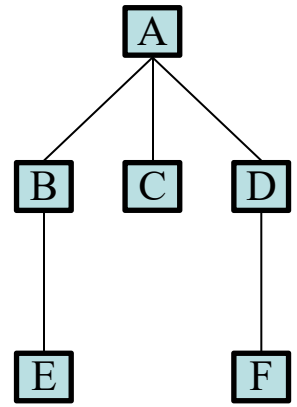
- **自顶向下的增量集成：**
  - 从主控模块开始，按软件的控制层次结构，以深度优先或广度优先的策略，逐步把各个模块集成在一起。

广度优先的测试过程：

- Test A (with stubs for B,C,D)
- Test A;B (with stubs for E,C,D)
- Test A;B;C (with stubs for E,D)
- Test A;B;C;D (with stubs for E,F)
- Test A;B;C;D;E (with stubs for F)
- Test A;B;C;D;E;F

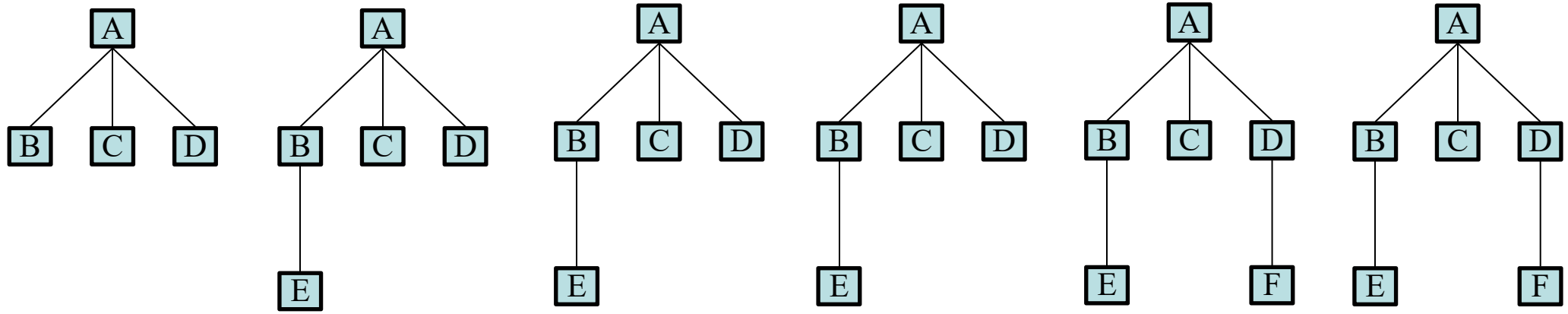


集成测试的方法：增量式集成



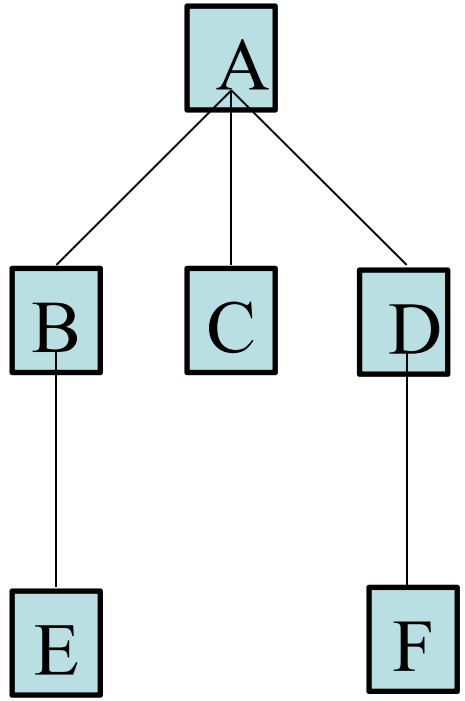
- **自顶向下的增量集成：**
  - 从主控模块开始，按软件的控制层次结构，以深度优先或广度优先的策略，逐步把各个模块集成在一起。

深度优先的测试过程：



集成测试的方法：增量式集成

- **自底向上的集成测试：**
  - 从软件结构最底层的模块开始组装测试。
  - Test E (with driver for B)
  - Test C (with driver for A)
  - Test F (with driver for D)
  - Test B;E (with driver for A)
  - Test D;F (with driver for A)
  - Test (A;B;C;D;E;F)



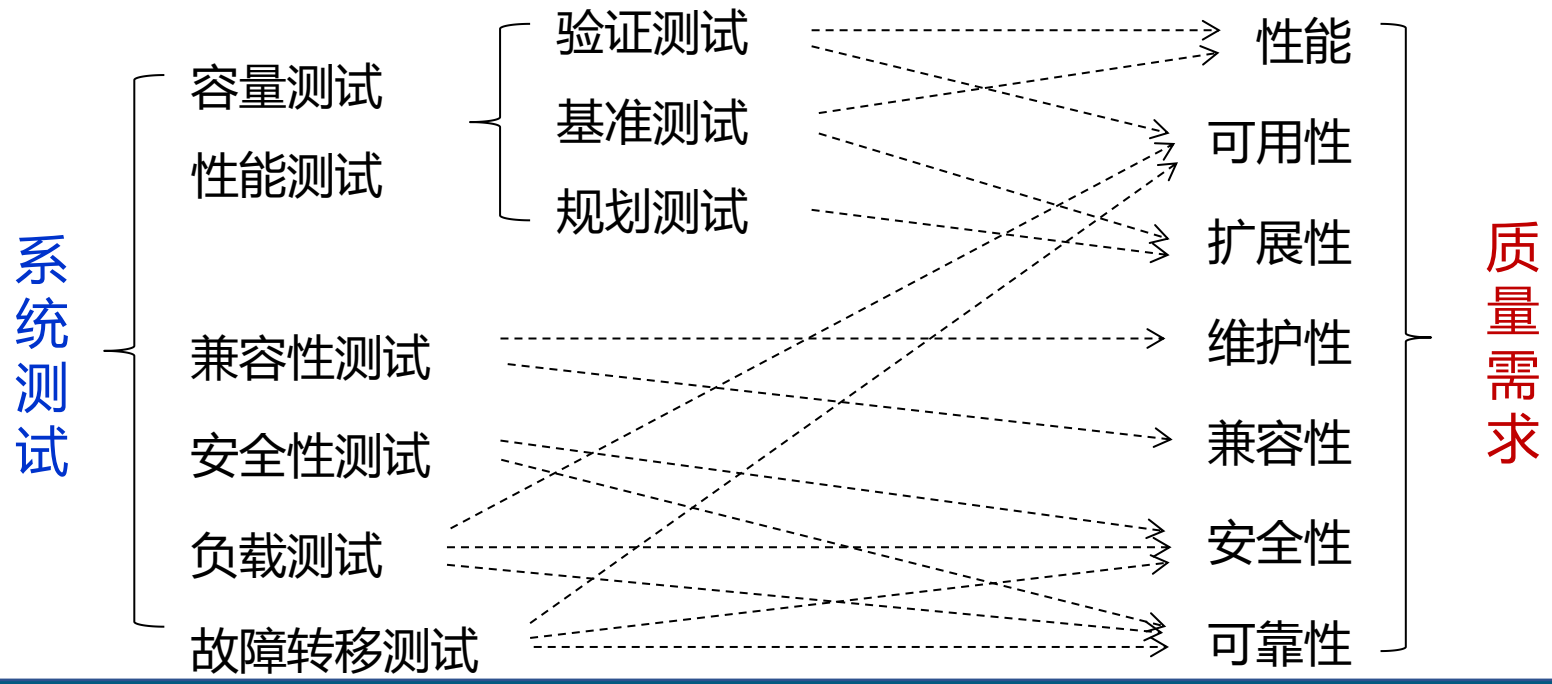
两种集成测试的优缺点

- **自顶向下集成：**
  - 优点：能尽早地对程序的主要控制和决策机制进行检验，因此较早地发现错误；较少需要驱动模块；
  - 缺点：所需的桩模块数量巨大；在测试较高层模块时，低层处理采用桩模块替代，不能反映真实情况，重要数据不能及时回送到上层模块，因此测试并不充分；

- **自底向上集成：**
  - 优点：不用桩模块，测试用例的设计亦相对简单；
  - 缺点：程序最后一个模块加入时才具有整体形象，难以尽早建立信心。

系统测试

系统测试（System Testing）是在实际运行环境或模拟实际运行环境下，针对系统的非功能特性所进行的测试，包括负载测试、性能测试、压力测试、恢复测试、安全测试和可靠性测试等。



系统测试：恢复测试

- **恢复测试(Recovery Testing)**
  - 恢复测试是检验系统从软件或者硬件失败中恢复的能力，即采用各种人工干预方式使软件出错，而不能正常工作，从而检验系统的恢复能力。
  - 恢复性测试的例子
    - 当供电出现问题时的恢复
    - 恢复程序的执行
    - 对选择的文件和数据进行恢复
    - 恢复处理日志方面的能力
    - 通过切换到一个并行系统来进行恢复

系统测试：安全性测试

- **安全性测试(Security Testing)**
  - 安全性测试检查系统对非法侵入的防范能力。
  - 安全性测试期间，测试人员假扮非法入侵者，采用各种办法试图突破防线。
  - 安全性测试的例子
    - 想方设法截取或破译口令
    - 专门定做软件破坏系统的保护机制
    - 故意导致系统失败，企图趁恢复之机非法进入
    - 试图通过浏览非保密数据，推导所需信息

系统测试

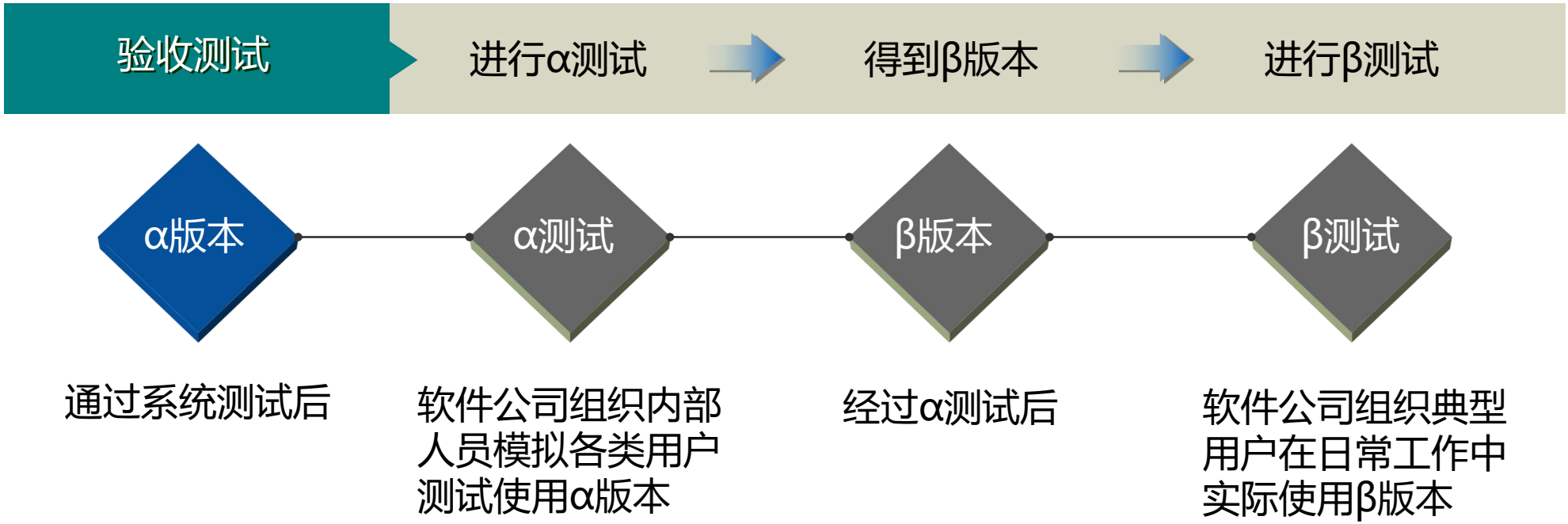
- **压力测试(Press Testing)**
  - 压力测试是检查系统在资源超负荷情况下的表现，特别是对系统的处理时间有什么影响。
  - 压力测试的例子
    - 对于一个固定输入速率(如每分钟120 个单词)的单词处理响应时间
    - 在一个非常短的时间内引入超负荷的数据容量
    - 成千上万的用户在同一时间从网上登录到系统
    - 引入需要大量内存资源的操作
  - 压力测试采用边界值和错误猜测方法，且需要工具的支持。

系统测试

- **性能测试(Performance Testing)**
  - 在实际应用的环境下系统性能的表现
  - 常与压力测试一起进行

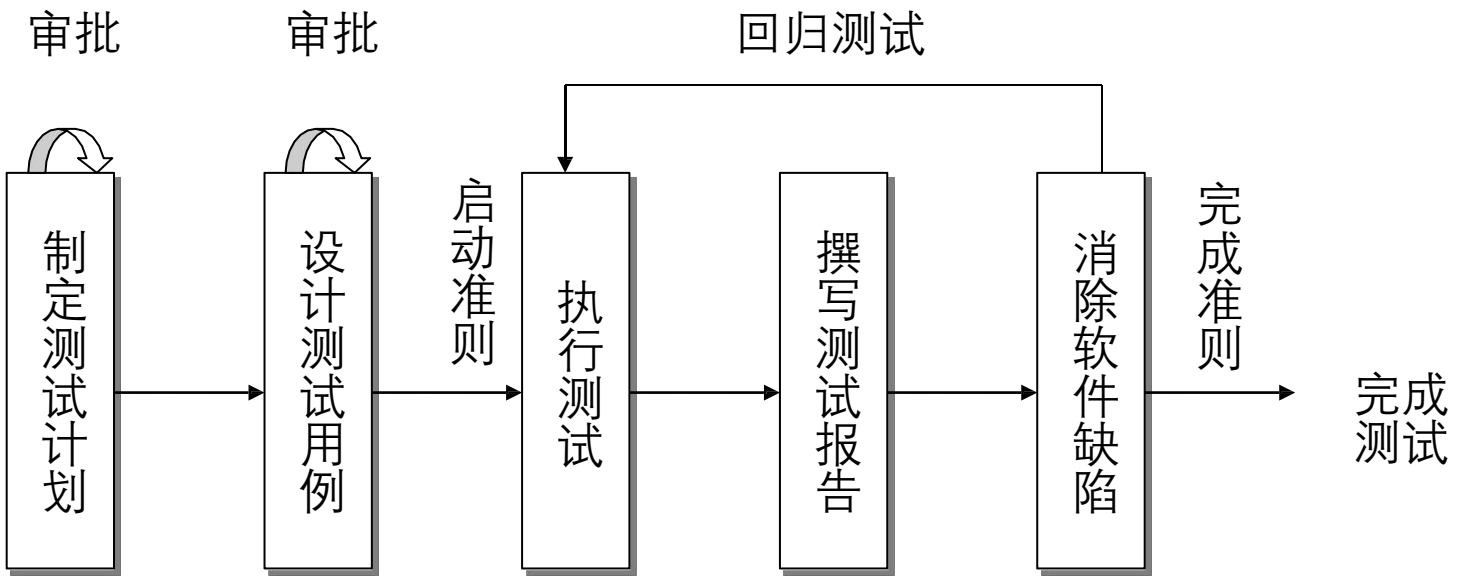
验收测试

验收测试是在软件产品完成了功能测试和系统测试之后、产品发布之前所进行的软件测试活动，其目的是验证软件的功能和性能是否能够满足用户所期望的要求。



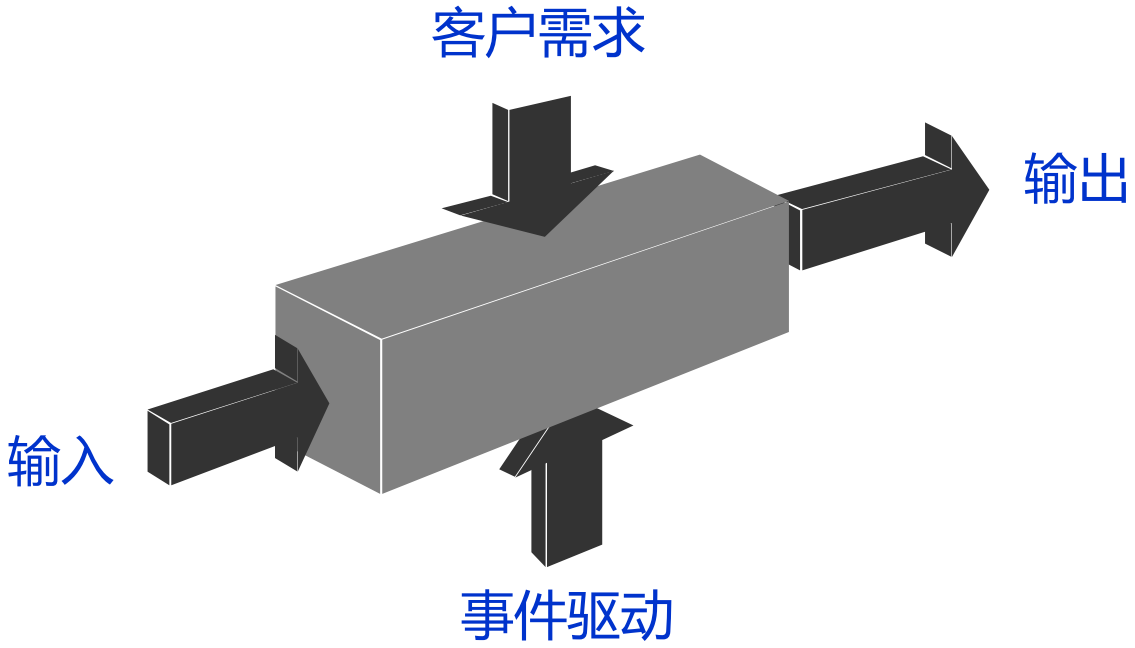
回归测试

- 在软件生命周期中的任何一个阶段，只要软件发生了改变，就可能给该软件带来问题。
- 为了验证修改的正确性及其影响就需要进行回归测试。



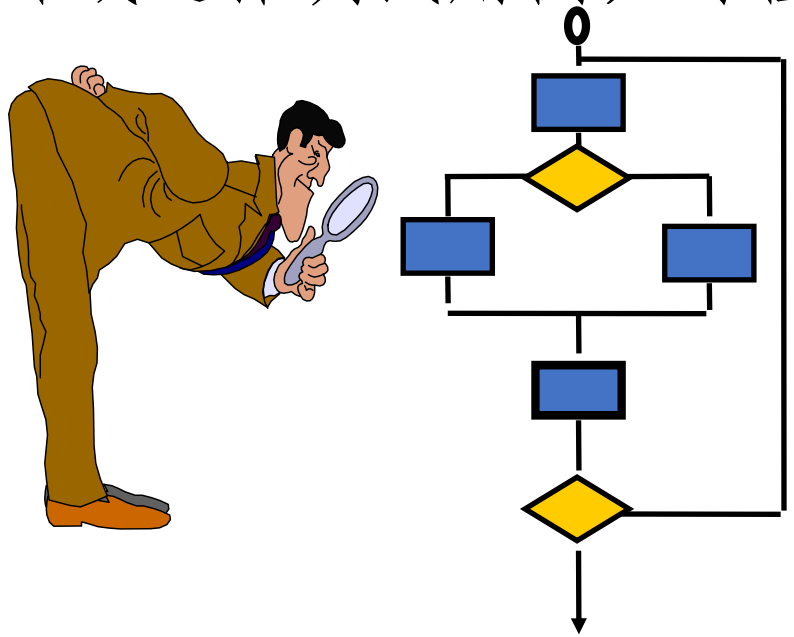
黑盒测试

黑盒测试（Black Box Testing）：又称功能测试，它将测试对象看做一个黑盒子，完全不考虑程序内部的逻辑结构和内部特性，只依据程序的需求规格说明书，检查程序的功能是否符合它的功能说明。



白盒测试

白盒测试（White Box Testing）：又称结构测试，它把测试对象看做一个透明的盒子，允许测试人员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，对程序所有逻辑路径进行测试。



静态测试与动态测试

静态测试：通过人工分析或程序正确性证明的方式来确认程序正确性。

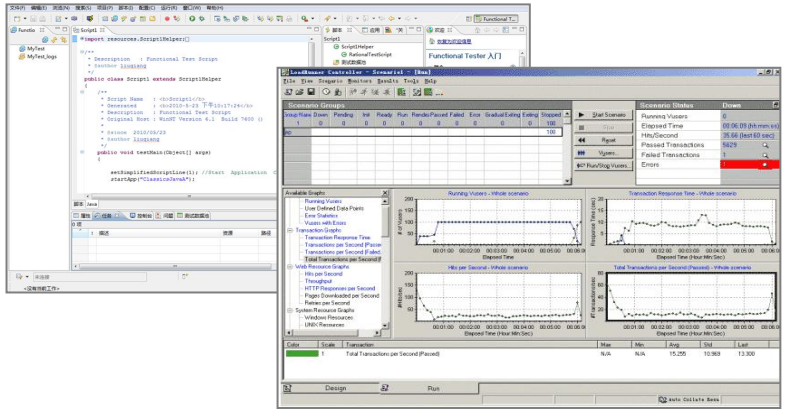
非正式 正式

互审 走查 审查会议

动态测试：通过动态分析和程序测试等方法来检查程序执行状态，以确认程序是否有问题。

手工测试与自动化测试

- 手工测试：测试人员根据测试大纲中所描述的测试步骤和方法，手工地输入测试数据并记录测试结果。
- 自动化测试：相对于手工测试而言，主要是通过所开发的软件测试工具、脚本等手段，按照测试工程师的预定计划对软件产品进行的自动测试。

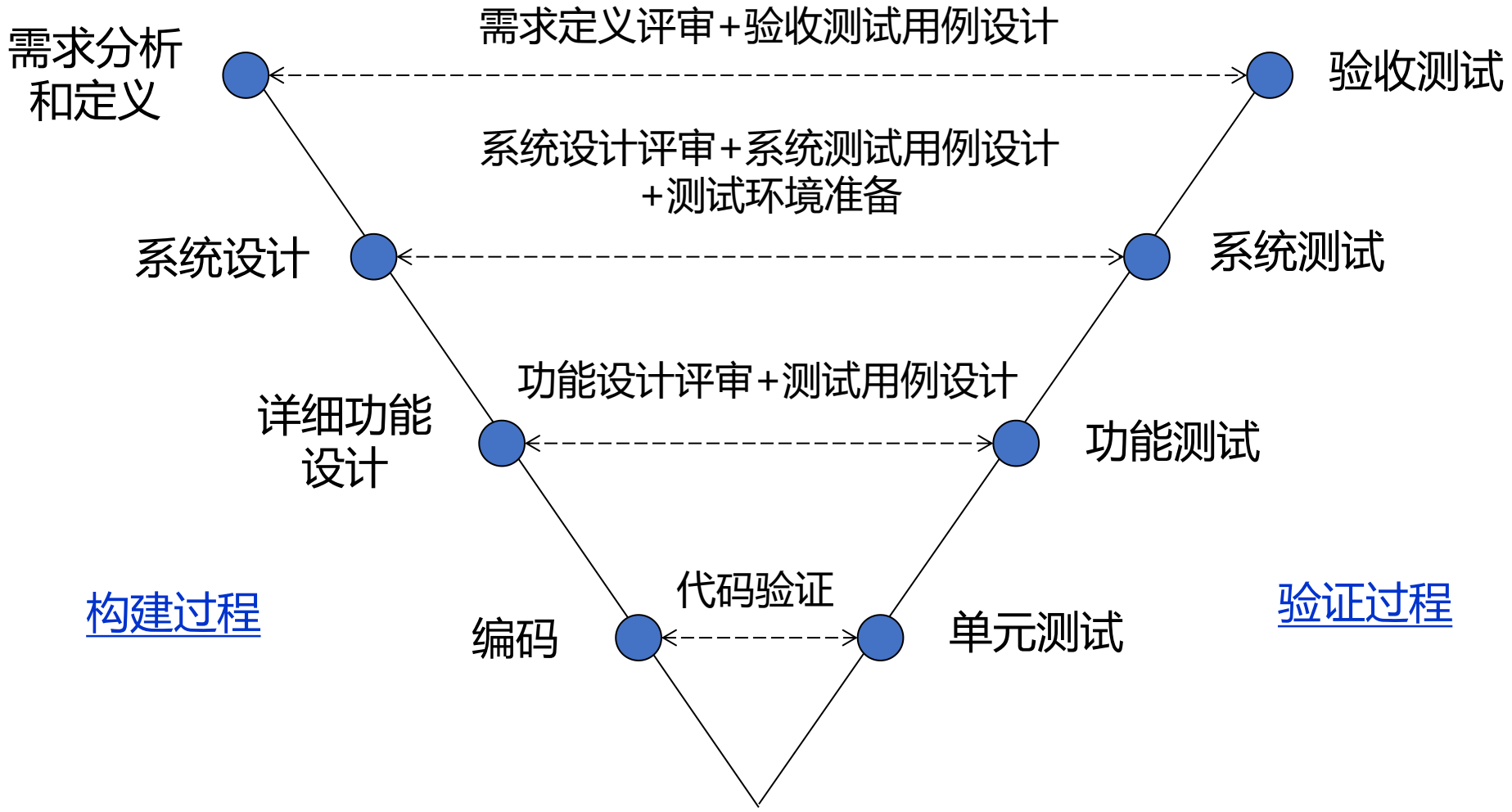


手工测试与自动化测试

自动化测试只是对手工测试的一种补充，但绝不能代替手工测试，二者有各自的特点。

- 在系统功能逻辑测试、验收测试、适用性测试、涉及物理交互性测试时，多采用黑盒测试的手工测试方法；
- 单元测试、集成测试、系统负载或性能、稳定性、可靠性测试等比较适合采用自动化测试；
- 对那种不稳定软件的测试、开发周期很短的软件、一次性的软件等不适合自动化测试；
- 工具本身并没有想象力和灵活性，一般自动化测试只能发现15~30%的缺陷，而手工测试可以发现70~85%的缺陷；自动化测试工具在进行功能测试时，其准确的含义是回归测试工具。

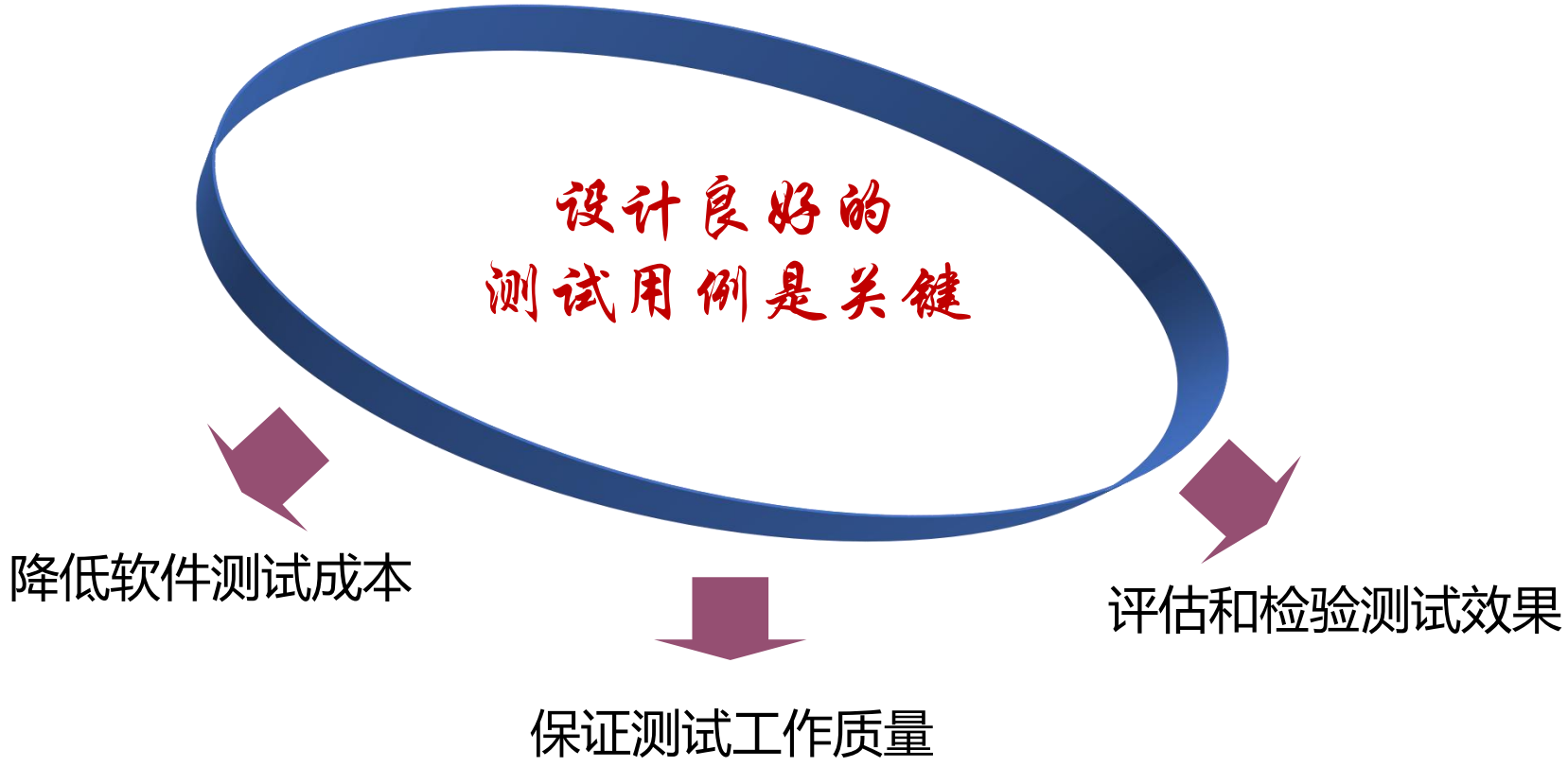
测试的V 模型


需求分析
和定义
需求定义评审+验收测试用例设计
验收测试
系统设计评审+系统测试用例设计
+测试环境准备
系统设计
系统测试
功能设计评审+测试用例设计
详细功能
设计
功能测试
代码验证
编码
单元测试
构建过程
验证过程

5.2 软件测试

- 软件测试概述
- 软件测试策略
  - 软件测试对象
  - 软件测试过程
  - 软件测试类型
  - 测试用例

测试用例的重要性



测试用例的定义与特征

- **测试用例(testing case):**
  - 测试用例是为特定的目的而设计的一组测试输入、执行条件和预期的结果。
  - 测试用例是执行的最小测试实体。
  - 测试用例就是设计一个场景，使软件程序在这种场景下，必须能够正常运行并且达到程序所设计的执行结果。
- **测试用例的特征：**
  - 最有可能抓住错误的；
  - 不是重复的、多余的；
  - 一组相似测试用例中最有效的；
  - 既不是太简单，也不是太复杂。

测试用例的重要性

指导人们系统地进行测试

- 临时性发挥也许会有灵感出现，但是多数情况下会感觉思维混乱，甚至一些功能根本没有测到而另一些功能已经重复测过几遍。
- 测试用例可以帮助你理清头绪，进行比较系统的测试，不会有太多的重复，也不会让你的测试工作产生遗漏。

有效发现缺陷，提高测试效率

- 测试不可能是完备的而且受到时间约束，测试用例可以帮助你分清先后主次，从而更有效地组织测试工作。
- 编写测试用例之后需要标识重要程度和优先级，以便在时间紧迫的情况下有重点地开展测试工作。

测试用例的重要性

作为评估和检验的度量标准

- 测试用例的通过率和软件缺陷的数量是检验软件质量的量化标准，通过对测试用例的分析和改进可以逐步完善软件质量，不断提高测试的水平。
- 测试用例也可以用于衡量测试人员的工作量、进度和效率，从而更有效地管理和规划测试工作。

积累和传递测试的经验与知识

- 测试用例不是简单地描述一种具体实现，而是描述处理具体问题的思路。设计和维护测试用例有助于人们不断积累经验和知识，通过复用测试用例可以做到任何人实现无品质差异的测试。

测试用例

测试用例

- 测试用例值：完成被测软件的某个执行所需的输入值
- 期望结果：当且仅当程序满足其期望行为，执行测试时产生的结果
- 前缀值：将软件置于合适的状态来接受测试用例值的任何必要的输入
- 后缀值：测试用例值被发送以后，需要被发送到软件的任何输入

测试集
测试用例值的集合

验证值：查看测试用例值结果所要用到的值
结束命令：终止程序或返回到稳定状态所要用到的值

测试用例设计要求

测试用例设计

- 具有代表性和典型性
- 寻求系统设计和功能设计的弱点
- 既有正确输入也有错误或异常输入
- 考虑用户实际的诸多使用场景