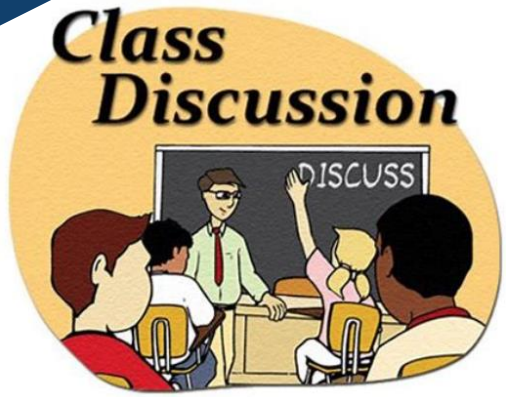
CS2001
软件工程

16. 软件测试
—软件测试概述

课件来源　　复旦大学软件学院

课堂讨论：测试是什么



课堂讨论：用一句话概括软件测试是什么或者软件测试是什么样的？

测试是什么

使用人工或自动的手段来运行或测量软件系统的过程，目的是检验软件系统是否满足规定的要求，并找出与预期结果之间的差异。

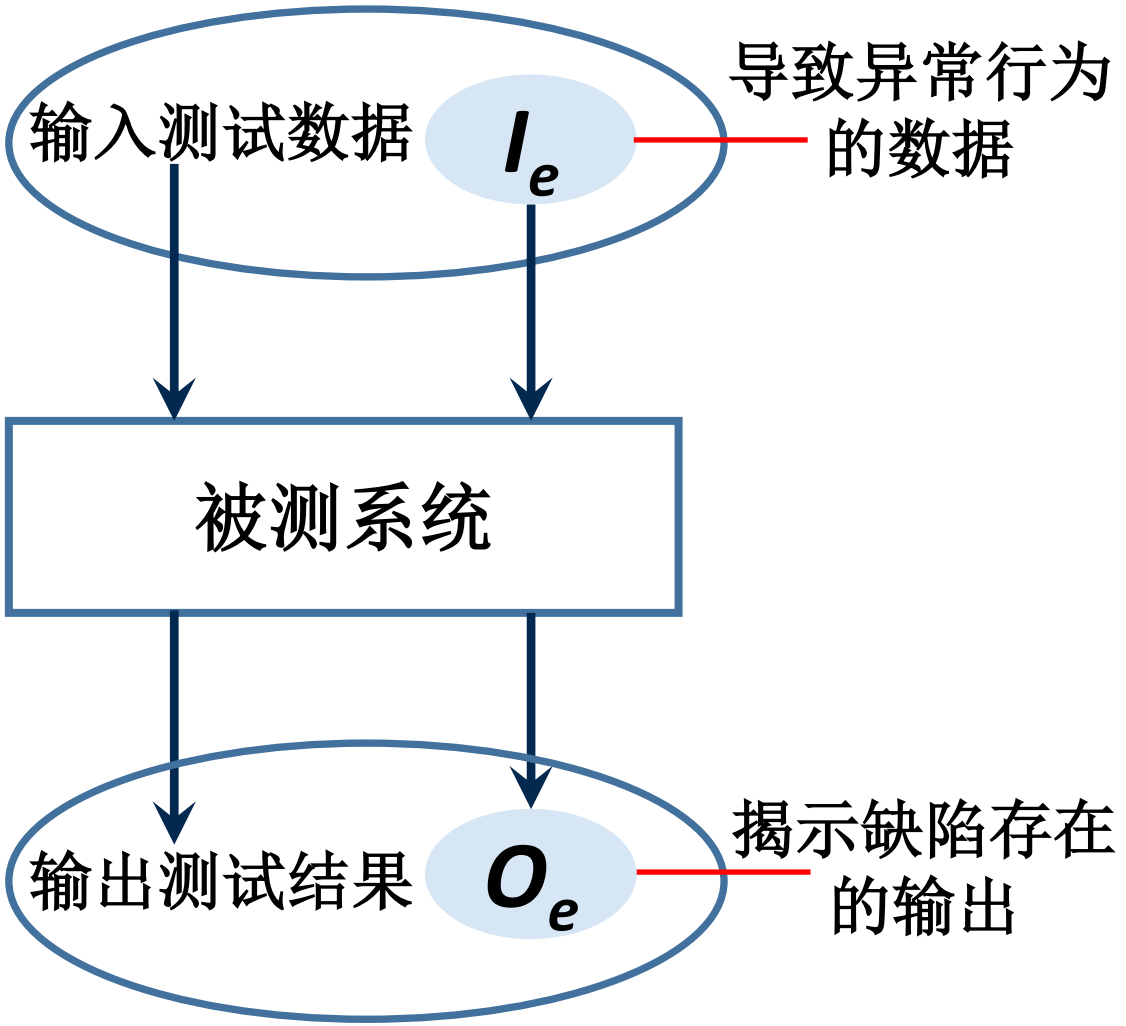
IEEE 729-1983: IEEE Standard Glossary of Software Engineering Terminology

测试只能显示问题的存在，而不是没有存在

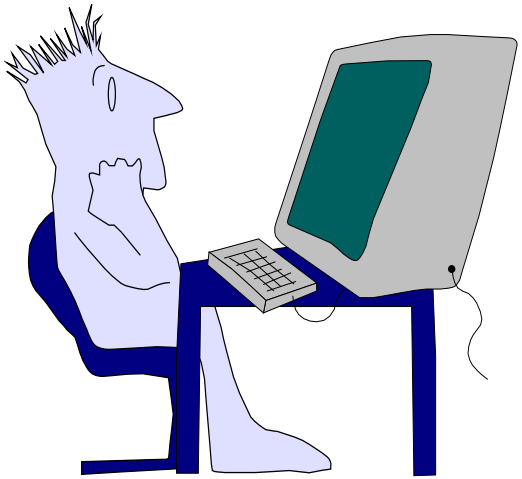
两种测试出发点

缺陷驱动的测试：目的是发现缺陷，因此优先考虑找到I_e中的输入

确认驱动的测试：目的是确认系统满足需要，因此全面考虑各种输入



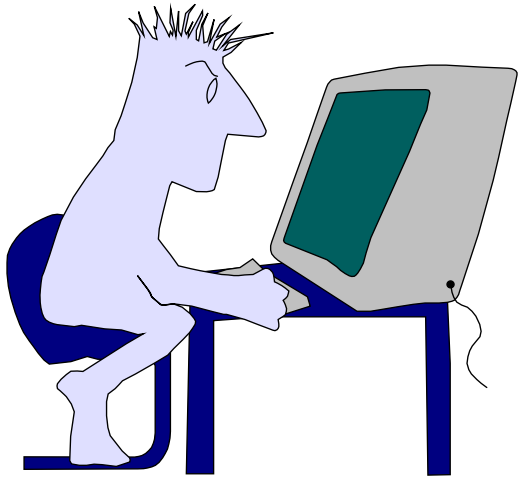
谁来测试?



开发者

优势：理解系统

劣势：“温柔”地测试，“交付”驱动而不是“质量”驱动



独立测试人员

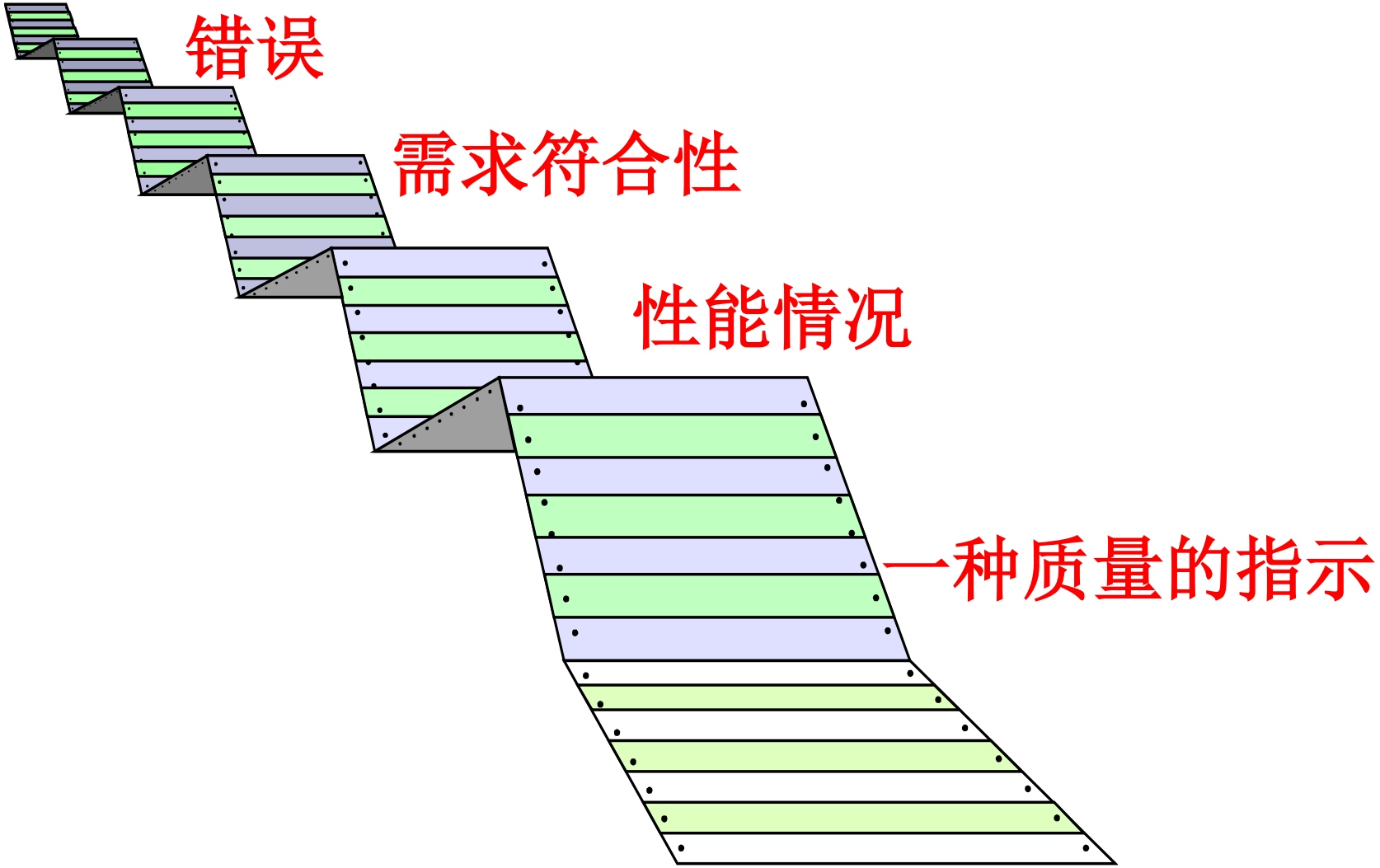
优势：竭尽全力暴露系统问题，“质量”驱动

劣势：需学习系统

测试用例（Test Case）

- 一组条件或（输入）变量，用来让测试人员判断被测试的软件系统是否正常工作（期望输出）
- 经常被称为测试脚本，特别是当被写下来之后
- 确定一个软件系统是否可以正常工作往往需要很多测试用例

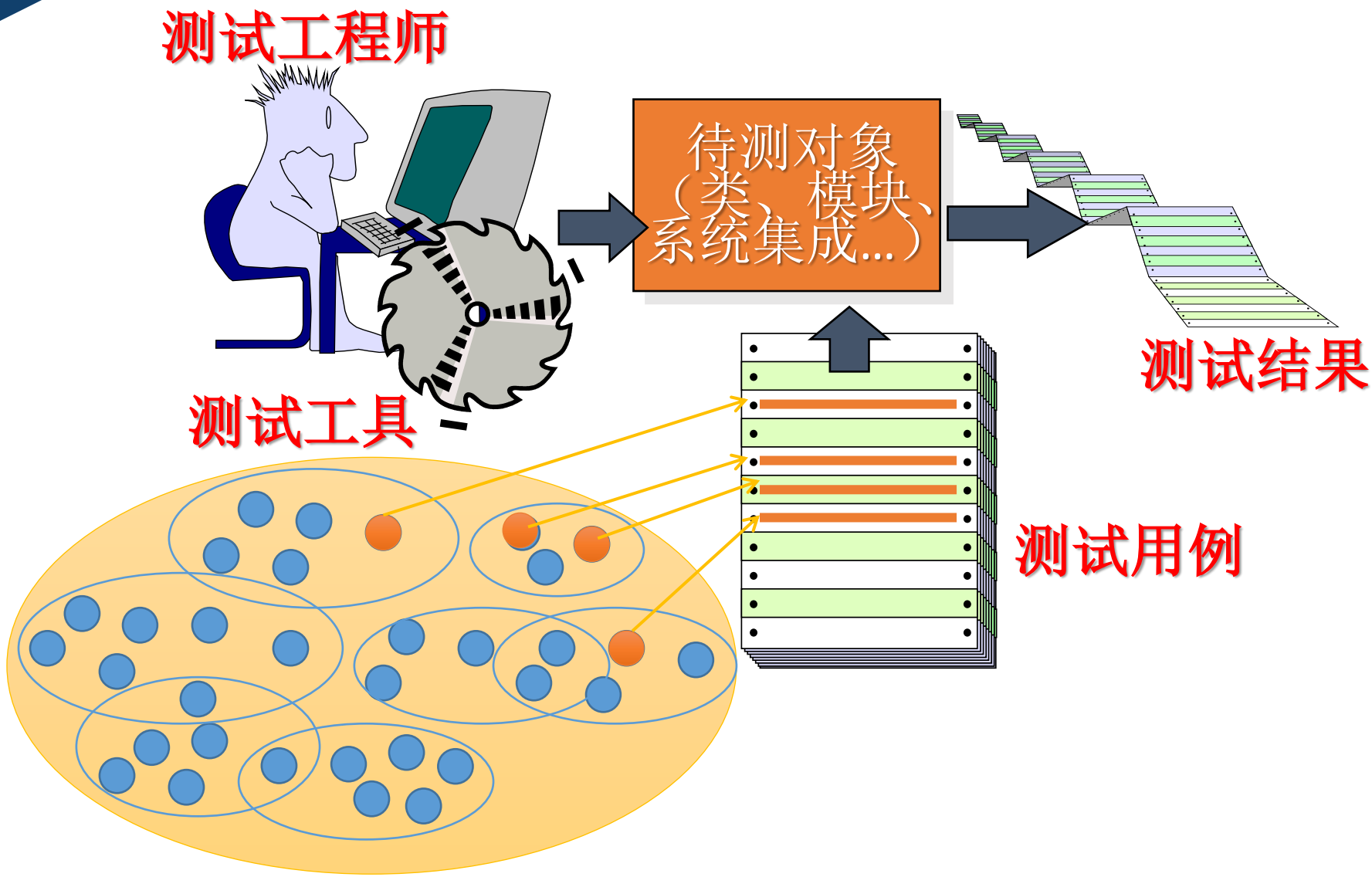
测试可以表明什么



关于软件测试的一些常识

- 测试是一种之于软件系统所有可能的使用和运行方式的采样方法
- 理论上讲，测试永远不可能完备
- 测试需要耗费大量的成本和时间
- 测试总是越多越好，但是我们不能一直等下去（考虑成本和交付时间）
- 虽然远非完美，但是遵循好的软件测试方法和实践仍然可以让我们获得满意的软件质量

测试：基于采样的过程



测试完备性与成本的权衡

根据质量准则和时间/成本约束
进行仔细的权衡决策

覆盖度、完备性、充分性

成本、交付时间

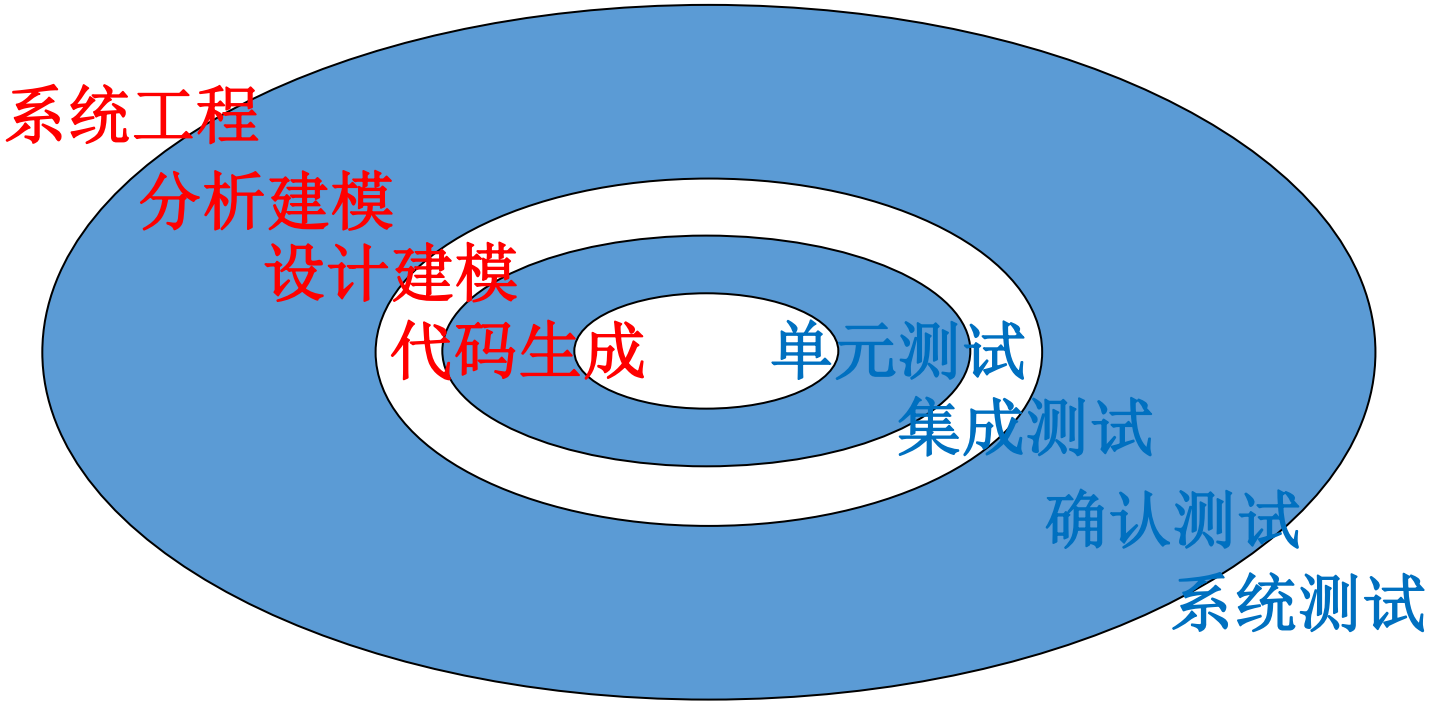
测试策略

- 为了保证测试效率，应当在测试之前进行技术评审，这样可以在测试之前消除很多错误
- 测试从组件级开始，然后逐渐向外直到整个软件系统都完成了集成
- 不同的测试技术适用于不同的软件工程方法以及不同的时间
- 测试和调试是不同的活动，但是测试方法应当适应调试的需要

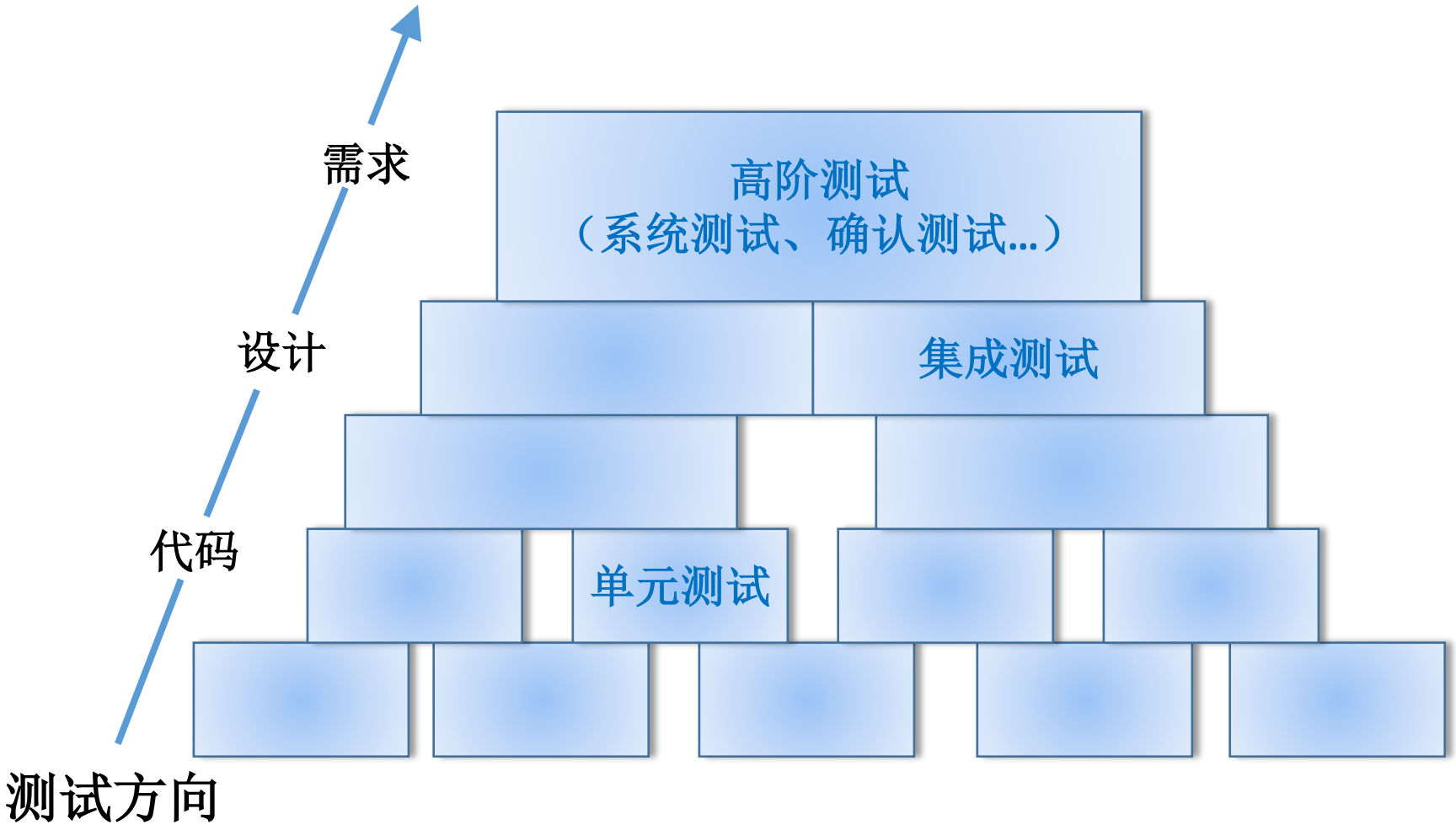
测试策略：从小到大

- 首先从“较小的测试”开始，然后逐步进行为“较大的测试”
- 对于结构化程序
  - ✓ 模块（组件）是最初的关注点
  - ✓ 然后是模块间的集成直至完整的系统
- 对于面向对象程序
  - ✓ 最初的关注点是封装了属性和操作的类
  - ✓ 然后是类的集成、模块间集成直至完整的系统

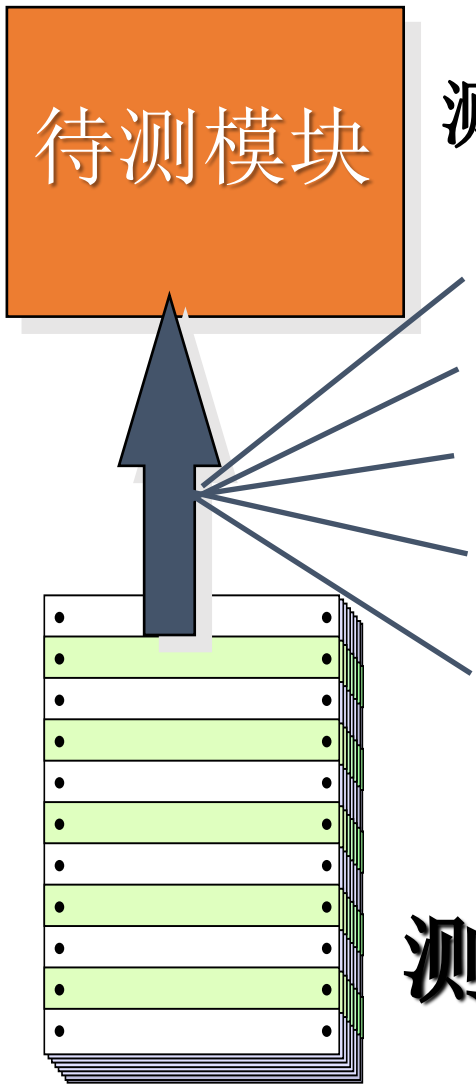
软件测试策略



软件测试步骤



单元测试

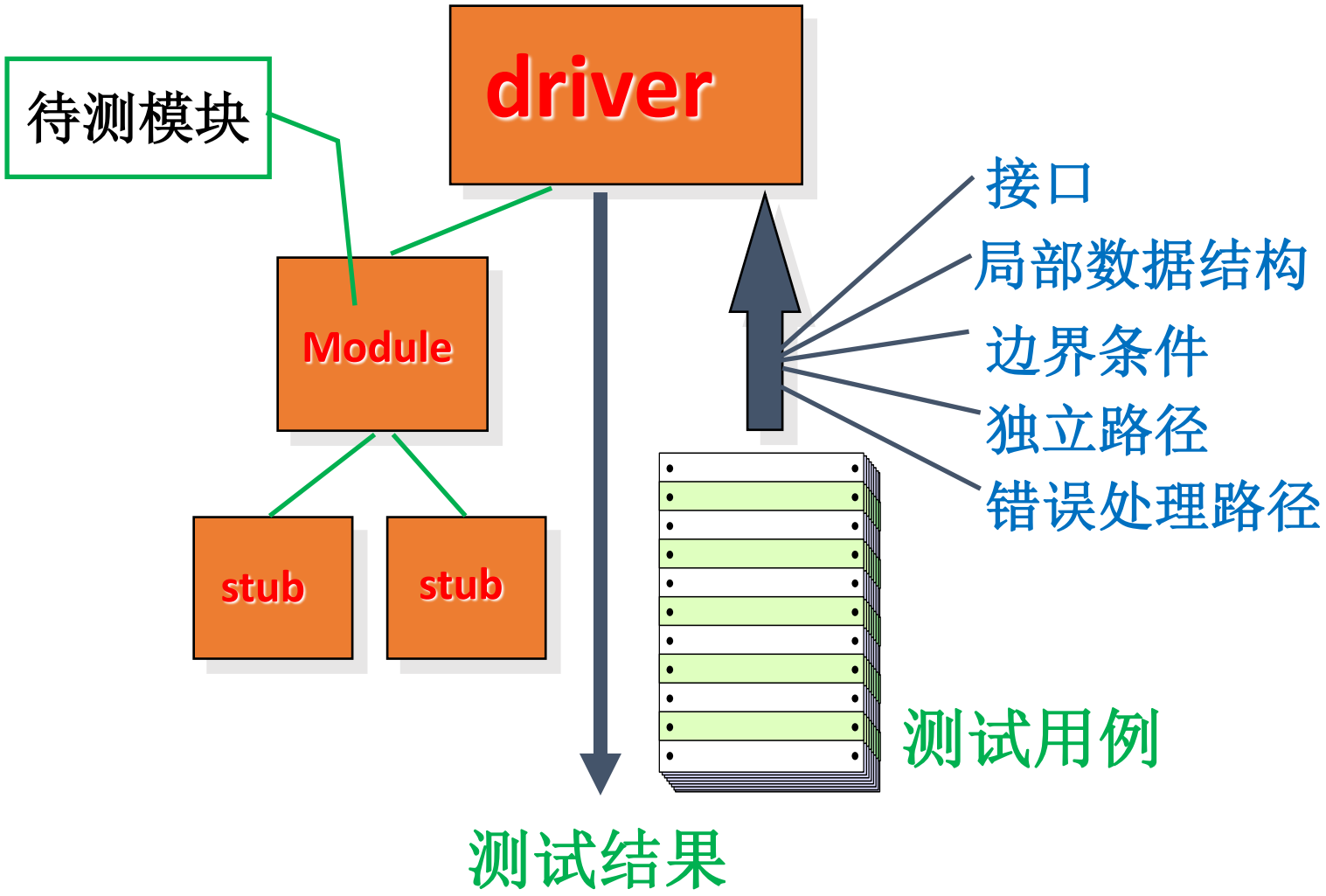


测试什么？

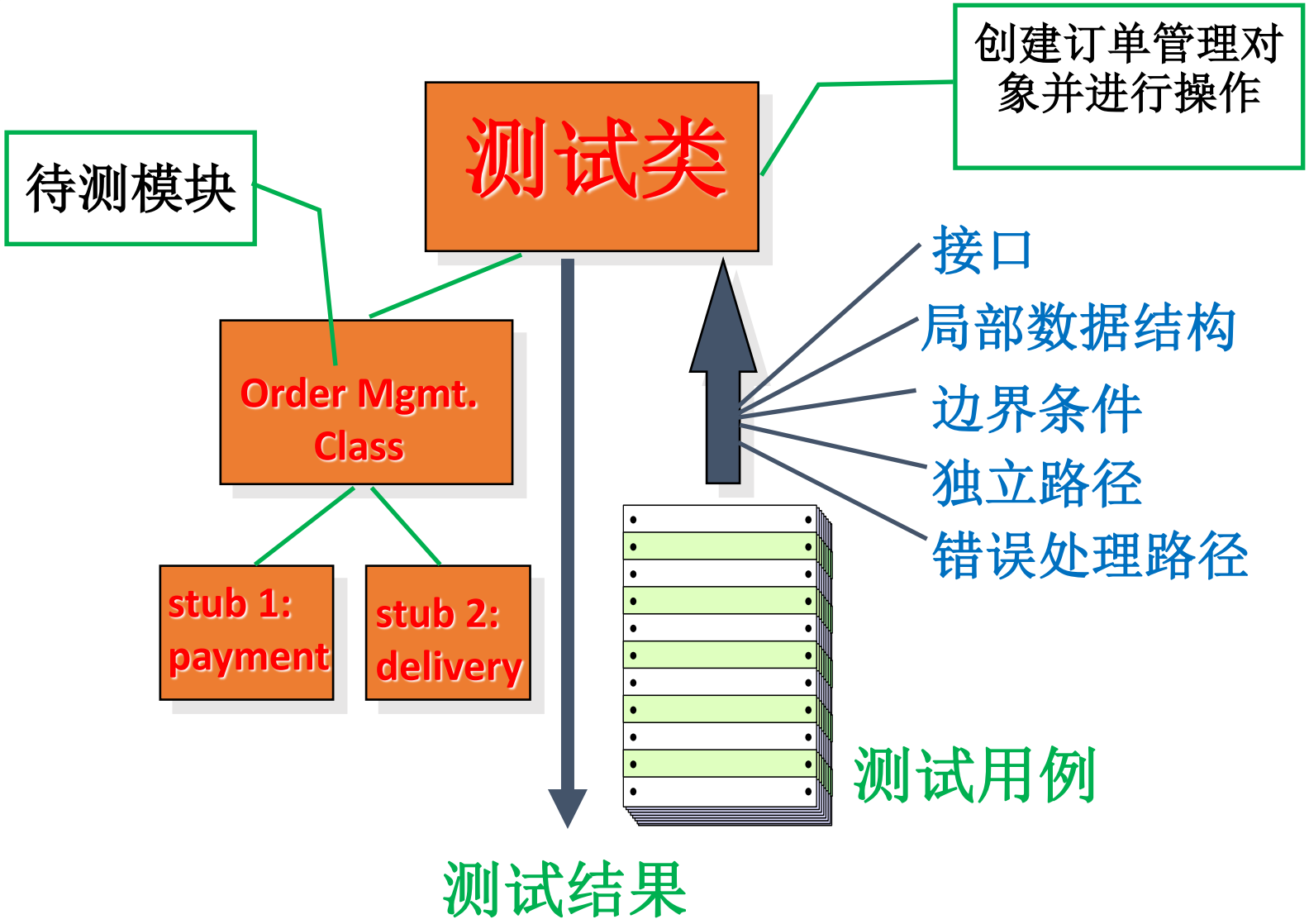
- 接口
- 局部数据结构
- 边界条件
- 独立路径
- 错误处理路径

测试用例

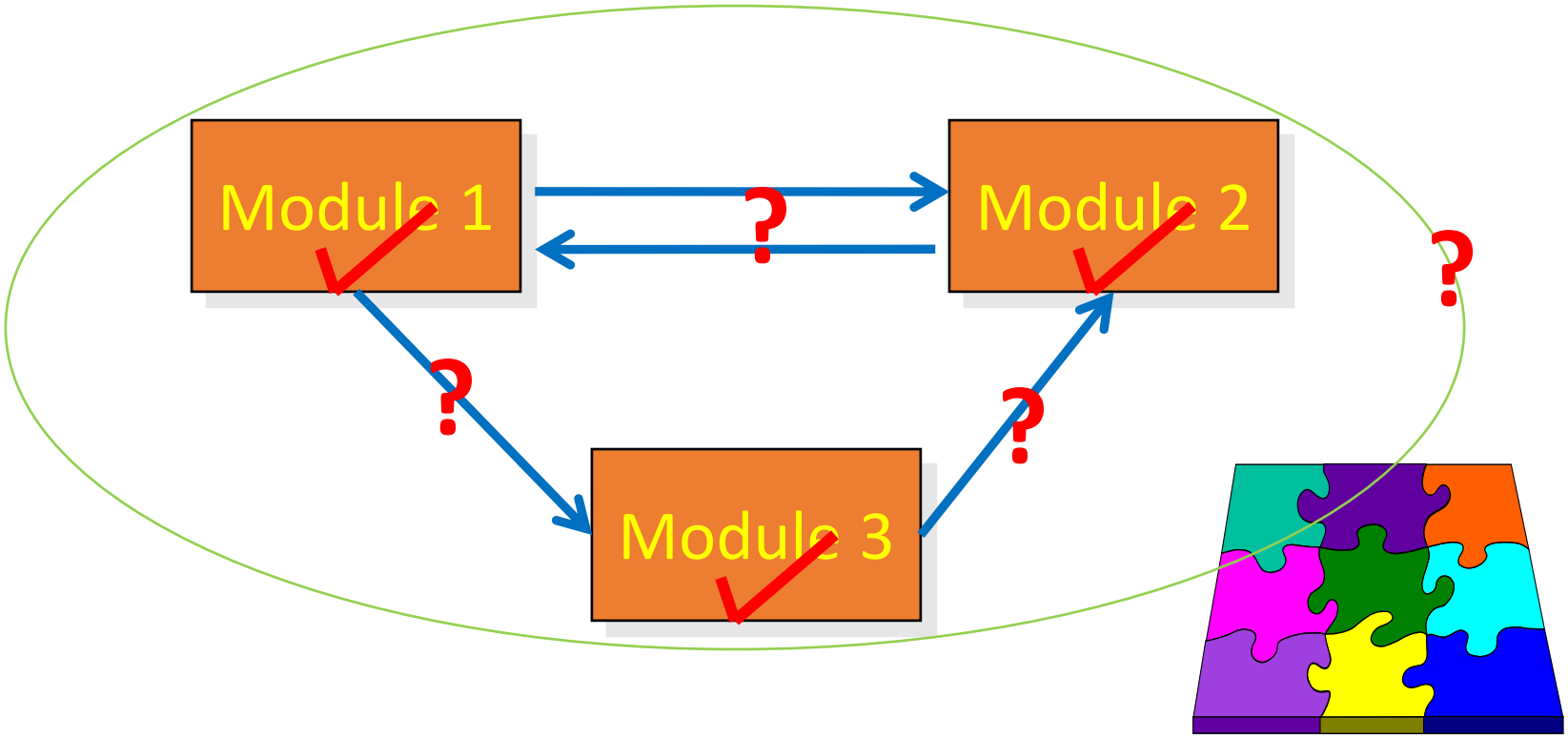
单元测试环境


待测模块
driver
Module
stub
stub
接口
局部数据结构
边界条件
独立路径
错误处理路径
测试用例
测试结果

单元测试环境：示例

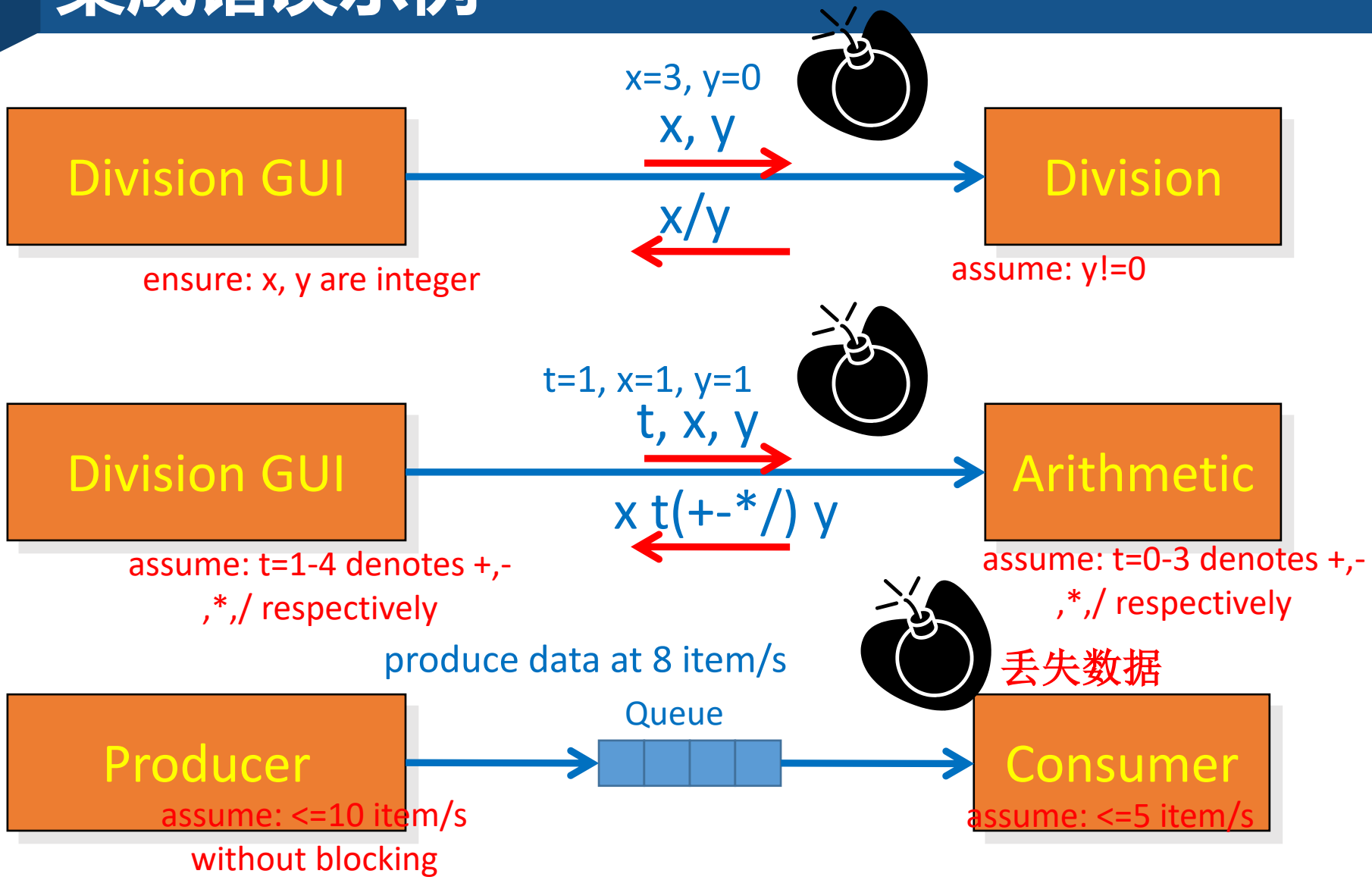

创建订单管理对象并进行操作
测试类
待测模块
Order Mgmt. Class
stub 1: payment
stub 2: delivery
接口
局部数据结构
边界条件
独立路径
错误处理路径
测试用例
测试结果

集成测试



- 集成测试：根据软件体系结构设计构造完整系统并在此过程中进行测试，以发现与接口和交互相关的问题
- 目标：在通过单元测试的软件组件基础上根据软件体系结构设计构造完整的软件系统

集成错误示例



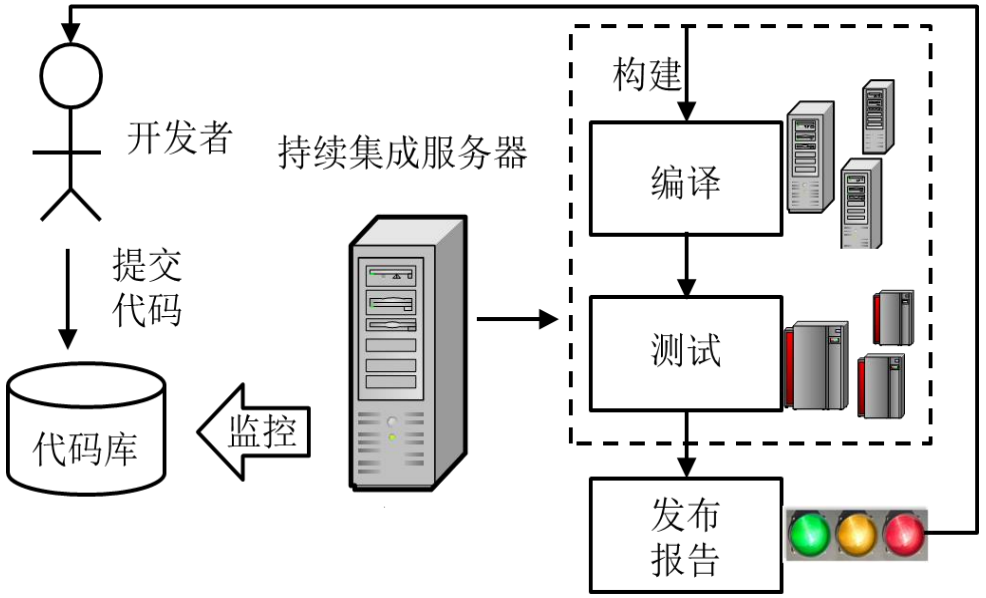
不一致的接口假设

集成测试策略

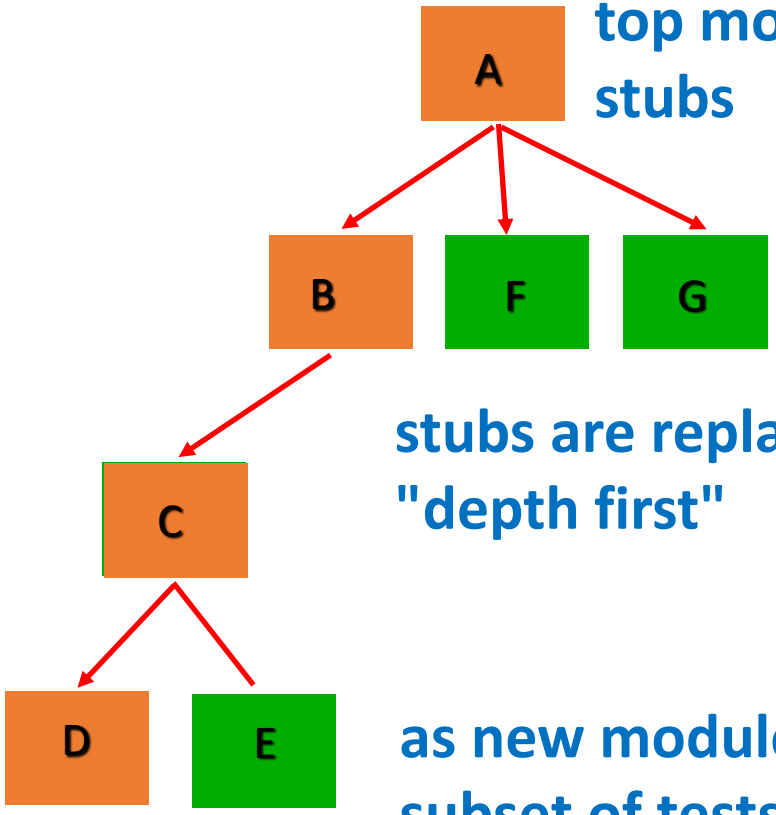
- “大爆炸”（Big Bang）集成
  - ✓ 一种在产品发行阶段一次性进行集成的方式
  - ✓ 缺点
    - 工作量大、耗时长
    - 时间弹性弱、反馈困难且时间长
    - 涉及模块过多，难以进行错误诊断和定位
- 增量集成
  - ✓ 软件的集成通过多个增量，按照一定顺序（例如自底向上）逐步集成为最终的系统
    - 例如，在一个较大规模的嵌入式系统中，开发者可以首先集成操作系统、硬件平台和硬件适配层，然后增加通信层、应用层的各个模块，最后集成各个应用所对应的操作界面等

持续集成

- 一种软件开发实践，其中团队成员经常集成他们的工作
- 每次集成都通过自动构建和自动化测试来验证，以尽快发现集成错误
- 要求每次只引入细小变化、缓慢的但是稳健的保持系统增长
- 目标是始终保持一个可以工作的系统



自顶向下的增量集成：深度优先

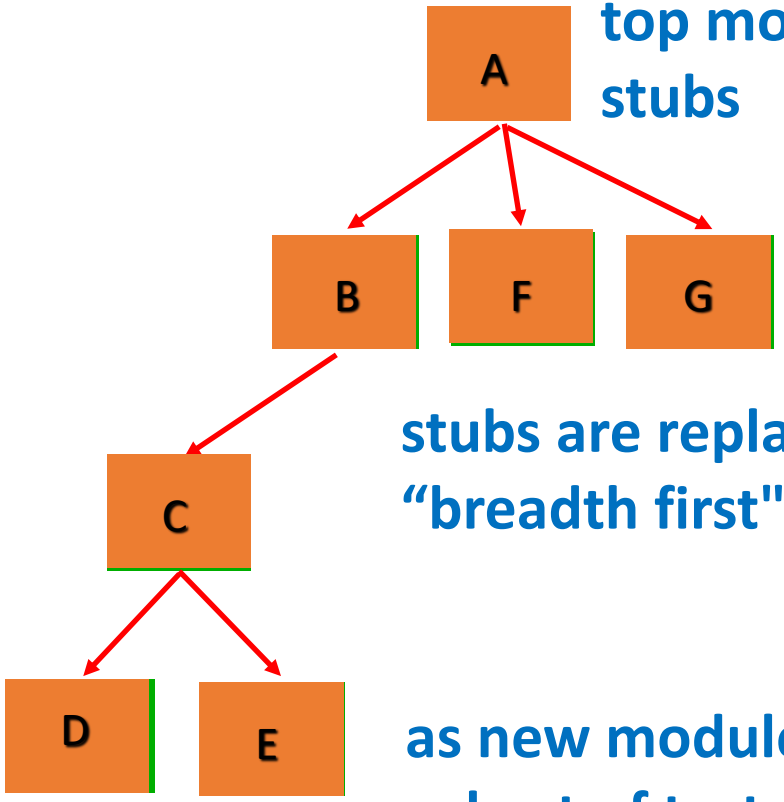


top module is tested with stubs

stubs are replaced one at a time "depth first"

as new modules are integrated, some subset of tests is re-run

自顶向下的增量集成：广度优先

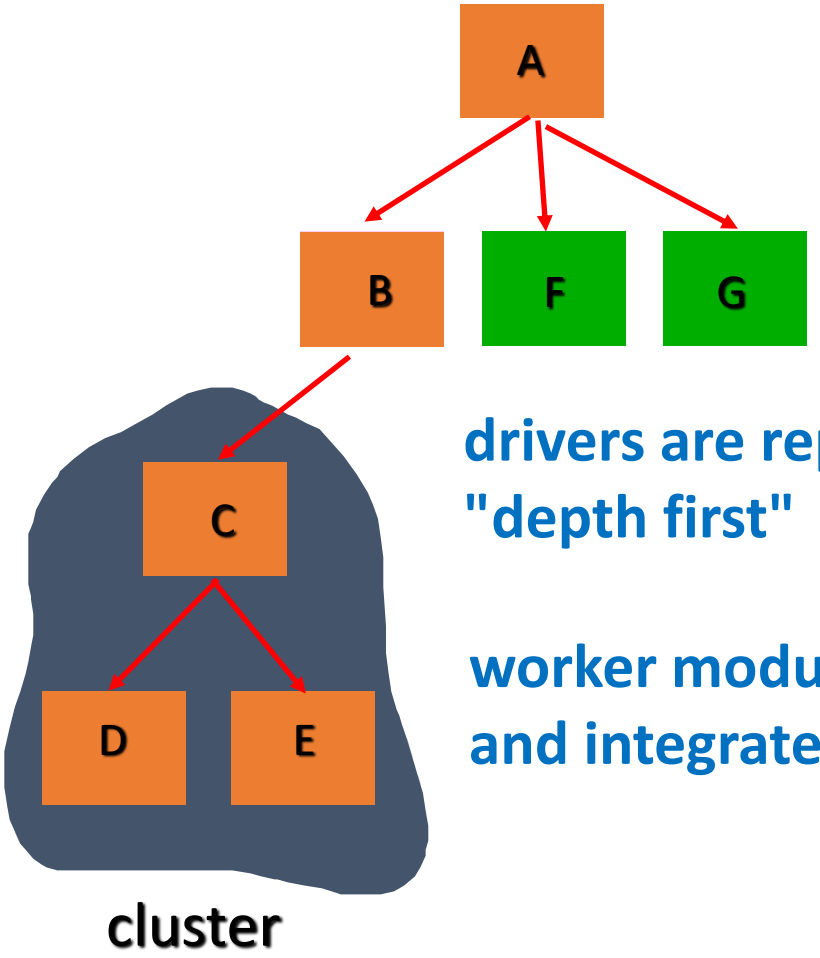


top module is tested with stubs

stubs are replaced one at a time
“breadth first"

as new modules are integrated, some subset of tests is re-run

自底向上的增量集成



drivers are replaced one at a time
"depth first"

worker modules are grouped into builds
and integrated

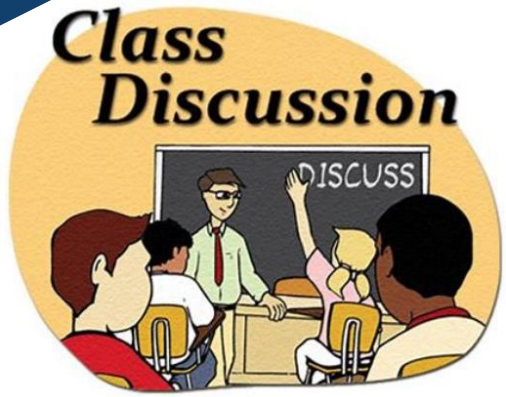
回归测试（Regression Testing）

- 重新执行已经进行过的一些测试，以确保所做的修改没有导致非预期的副作用
- 执行频率高，因此需要尽可能自动化，甚至纳入持续集成
- 回归测试用例需要持续维护

高阶测试

- 确认测试
  - ✓ 验收测试：客户确认所有需求都得到了满足，通常由最终用户进行测试
  - ✓ Alpha/Beta测试
- 系统测试
  - ✓ 恢复测试（Recovery Testing）
  - ✓ 安全测试（Security Testing）
  - ✓ 压力测试（Stress Testing）
  - ✓ 性能测试（Performance Testing）
  - ✓ 冒烟测试（Smoke Testing）

课堂讨论：不同的测试技术区别



课堂讨论：验证与验收测试区别？性能测试与压力测试区别？

- *验证（Verification）* 是指确保软件正确实现特定功能的任务集合。
- *确认（Validation）* 是指不同的任务集，可以确保已经构建的软件可以追溯到客户需求。Boehm [Boe81] 以另一种方式给出解释:
  - Verification: “我们是否正确地构造了产品?"
  - Validation: “我们是否构造了正确的产品?"

配置测试

- 服务器端
  - WebApp与服务器操作系统完全兼容吗?
  - 是否已经利用所选择的分布式服务器配置对WebApp进行了测试？
  - 此WebApp是否与数据库软件进行了适当的集成？是否对数据库的不同版本敏感?
  - 服务器端的WebApp脚本运行正常吗?
  - 等等

安全性测试

- 某些方面所存在的弱点，比如客户端环境，客户端到服务器端之间网络通信以及服务器端环境。
- 在客户端，弱点通常可以追溯到早已存在于浏览器、电子邮件程序、或通信软件中的缺陷。
- 在服务器端，薄弱环节包括拒绝服务攻击和恶意脚本，这些恶意脚本可以被传到客户端，或者用来使服务器操作丧失能力。

性能测试

- 服务器响应时间是否降到了不可接受的程度?
- 在什么情况下（就用户、事务或数据负载来说），性能变得不可接受?
- 哪些系统构件应对性能下降负责?
- 多种负载条件下，对用户平均响应时间是？
- 性能下降是否影响系统的安全性?
- WebApp的可靠性和准确性是否会受影响?
- 负载大于服务器容量最大值时，会怎么样?

负载测试

- 负载测试的目的是确定WebApp和其服务器环境如何响应不同的负载条件。
  - N, 并发用户的数量;
  - T, 每单位时间的在线事务数量;
  - D, 每次事务服务器处理的数据负载。
- 以下面的方式计算总的吞吐量P:
  - P = N x T x D

压力测试

- 系统“逐渐”降级吗？或者，当容量超出时，服务器停机吗？
- 服务器软件会给出“服务器不可用”的提示信息吗？更一般地说，用户知道他们不能访问服务器吗？
- 服务器队列请求增加资源吗？一旦容量要求减少，会释放队列所占用的资源吗？
- 当容量超过时，事务会丢失吗？
- 当容量超过时，数据完整性会受到影响吗？
- N、T和D的哪些值迫使服务器环境失效？如何来证明失效了？自动通知会被发送到位于服务器站点的技术支持人员那里吗？
- 如果系统失效，需要多长时间才能回到在线状态？
- 当容量达到80%或90%时，某些WebApp功能（例如，计算密集的功能、数据流动能力）会被停止吗？

冒烟测试（Smoke Testing）

- 对产品软件进行“每日构造”的一种常见方法。
- 冒烟测试步骤:
  - 已经被翻译为代码的软件构件被集成到构造（“build”）

    一个构造包括所有的数据文件、库、可重用的模块、以及工程化的构件，其中一个构件需要实现一项或多项产品功能。
  - 设计一系列测试来揭示错误，使得此构造不能正确地执行其功能。

    目的是揭示“显示阻塞”错误，这种错误最有可能使软件项目滞后于进度计划。
  - 将此构造与其他构造集成在一起，每天都对整个产品（以其当前的形式）进行冒烟测试。
    - 集成方法可以是自顶向下，也可以是自底向上。

面向对象测试方法

Berard [Ber93] 提出了下面的方法:

- ✓ 每个测试用例都应该标识，并明确地与被测试的类相关联。
- ✓ 应该叙述测试的目的。
- ✓ 应该为每一个测试开发测试步骤，并包括五个步骤:

面向对象测试方法

- 基于故障的测试
  - ✓ 要设计测试用例以检查设计或代码
- 类测试和类层次
  - ✓ 继承并不意味着所有派生类一起测试的需要
- 基于场景的测试设计
  - ✓ 基于场景的测试关心用户做什么，而不是产品做什么

面向对象测试方法: 随机测试

- 随机测试
  - ✓ 标识可应用于类的操作
  - ✓ 定义其使用限制
  - ✓ 标识最小的测试序列
    - 定义类（对象）最小生命期的操作序列
  - ✓ 产生不同的随机（但有效的）测试序列
    - 检查其他（更复杂的）类实例的生命历史

面向对象测试方法: 划分测试

- 划分测试
  - ✓ 划分测试减小测试特定类所需的测试用例数量
  - ✓ 基于状态划分
    - 根据它们改变类状态的能力对类操作进行分类
  - ✓ 基于属性划分
    - 根据它们所使用的属性对类操作进行分类
  - ✓ 基于类别划分
    - 根据每个操作所完成的一般功能对类操作进行分类

面向对象测试方法: 类间测试

- 类间测试
  - ✓ 对每个客户类，使用类操作列表来生成一系列随机测试序列。这些操作将向其他服务类发送消息；
  - ✓ 对生成的每个消息，确定协作类和服务对象中的相应操作；
  - ✓ 对服务对象中的每个操作（已被来自客户对象的消息调用），确定它传送的消息；
  - ✓ 对每个消息，确定下一层被调用的操作，并将其引入到测试序列中。

面向对象测试方法: 行为测试

所设计的测试应该获取所有的状态覆盖[KIR94].

即，操作序列应使得账号（Account）在所有可能的状态之间进行转换。

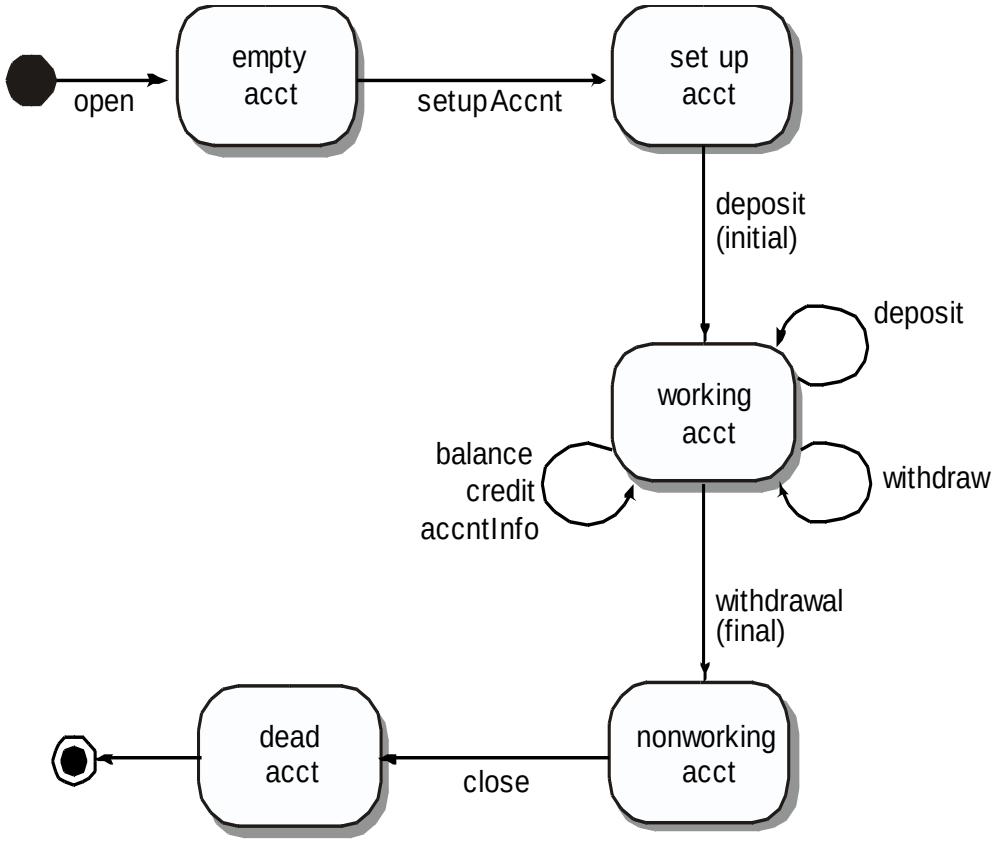


Figure 14.3 State diagram for Account class (adapted from [KIR94])

移动应用测试

- 用户体验测试 – 保证app满足利益相关者可用性和可访问性期望
- 设备兼容性测试 – 在多种设备上进行测试
- 性能测试 – 测试非功能需求
- 连接性测试 – 测试app连接可靠的能力
- 安全性测试 – 确保app满足利益相关者的安全性期望
- 现场测试（Testing-in-the-wild） – 在用户设备和实际的用户环境中测试app
- 认证测试 – app满足分发标准

阅读建议

- 《软件工程》第8章
- 《构建之法》第13章

快速阅读后整理问题
在QQ群中提出并讨论

CS2001
软件工程

End

16. 软件测试
—软件测试概述