软件工程

第19章 测试面向对象的应用系统

徐本柱 软件学院

2018-01

主要内容

- 扩展测试的视野
- 测试OOA和OOD模型
- 面向对象测试策略
- 面向对象测试方法
- 类级可应用的测试方法
- 类间测试用例设计

要点浏览

- OO软件的体系结构是包含协作类的一系列分层的子系统
  - 每个元素（子系统和类）完成有助于满足系统需求的功能。
  - 有必要在不同的层次上测试面向对象系统
  - 发现类间协作及子系统跨体系结构层通信时可能发生的错误。
- OOA与OOD模型在结构和内容上与所得的OO程序类似
  - “测试”可以起始于对这些模型的评审。一旦代码已经生成
  - 面向对象测试开始进行“小规模”的类测试
  - 检查类操作和类协作是否存在错误。
  - 当类集成起来构成子系统时，应用
    - 基于线程的测试、基于使用的测试、簇测试和基于故障的测试彻查协作类
    - 最后，利用用例发现软件确认级的错误。

引言

- 为了充分测试面向对象的系统，必须做3件事情：
  - (1)对测试的定义进行扩展，使其包括应用于面向对象分析和设计模型的错误发现技术；
  - (2)单元测试和集成测试策略必须彻底改变；
  - (3)测试用例设计必须考虑面向对象软件的独特性质。

19.1 扩展测试的视野

- 面向对象软件的构造开始于需求分析和设计模型的创建
- 面向对象软件工程模式的进化特性
  - 模型开始于系统需求的不太正式的表示
  - 进化到更详细的类模型、类关系、系统设计和分配以及对象设计
  - 每个阶段都要对模型进行“测试”，避免错误传播到下一轮迭代
- 面向对象分析和设计模型的评审非常有用
  - 因为相同的语义结构出现在分析、设计和代码层次。
  - 在分析期间所发现的类属性的定义问题会防止副作用的发生
- 在开发后期，OOA和OOD模型
  - 提供了有关系统结构和行为的实质性信息
  - 编写代码之前，需要对这些模型进行严格的评审
- 应该在模型的
  - 语法、语义和语用方面对所有的面向对象模型
  - 进行正确性、完整性和一致性测试。

19.2 测试OOA和OOD模型

- 不能在传统意义上对分析和设计模型进行测试
  - 因为这些模型是不能运行的。
  - 可以使用技术评审检查模型的正确性和一致性。

19.2.1 OOA和OOD模型的正确性

- 符号和语法的正确性
  - 用于表示分析和设计模型的符号和语法是
  - 与为项目所选择的特定分析和设计方法连接在一起的
  - 语法的正确性是基于符号表示的正确使用来判断的
- OOA/D模型反映真实世界
  - 根据模型是否符合真实世界的问题来评估模型的语义正确性
  - 如果模型准确地反映了真实世界，则在语义上是正确的
  - 实际上，为了确定模型是否反映了真实世界的需求，
  - 由问题领域专家检查类定义和层次中是否有遗漏和不清楚的地方。
  - 对类关系进行评估，确定是否准确地反映了真实世界的对象连接。

19.2.2 面向对象模型的一致性

- 为了评估一致性，应该检查每个类及与其他类的连接
- 可以使用CRC模型或对象-关系图来辅助此活动。
- 推荐使用下面的步骤对类模型进行评估：

1.检查CRC模型和对象-关系模型。
2.检查每一张CRC卡的描述以确定委托责任是定义协作者的一部分
3.反转连接，确保每个提供服务的协作者都从合理的地方收到请求
4.使用步骤3中反转后的连接，确定是否真正需要其他类，或者责任在类之间的组织是否合适。
5.确定是否可以将广泛请求的多个责任组合为一个责任。

用于评审的CRC索引卡片实例

类的名称：CreditSale	
类的类型：交易事件	
类的特性	nontangible, atomic, sequential, permanent, guarded
责任：	协作者：
读信用卡	信用卡
取得授权	信用权利
显示购物金额	产品票
	销售总账
	审计文件
生成账单	账单

图19-1 用于评审的CRC索引卡片实例

OO模型的一致性检查

- 系统设计描述
  - 总体的产品体系结构、组成产品的子系统、将子系统分配给处理器的方式、将类分配给子系统的方式以及用户界面的设计。
- 系统设计评审：
  - 检查面向对象分析期间所开发的对象-行为模型，
  - 将所需要的系统行为映射到为完成此行为而设计的子系统上。
  - 在系统行为的范畴内也要对并发和任务分配进行评审。
  - 对系统的行为状态进行评估以确定并发行为。
  - 使用用例进行用户界面设计。
- 对象模型描述
  - 每个类的细节以及实现类之间的协作所必需的消息传送活动。
- 对照对象-关系网检查对象模型，
  - 确保所有的设计对象包括必要的属性和操作，
  - 以实现为每个CRC卡所定义的协作。
  - 要对操作细节的详细规格说明进行评审。

19.3 面向对象测试策略

- 面向对象环境的单元测试
  - 封装的类常是单元测试的重点
  - 面向对象软件的类测试等同于传统软件的单元测试
- 面向对象环境的集成测试
  - 基于线程的测试，集成响应系统的一个输入或事件所需的一组类
  - 基于使用的测试，
    - 通过测试很少使用服务类的独立类开始构造系统，独立类测试完后，
    - 利用独立类测试下一层次的类(依赖类)。继续依赖类的测试直到完成整个系统
  - 簇测试
- 面向对象环境的确认测试
  - 传统的黑盒测试方法可用于驱动确认测试
  - 可从对象-行为模型导出测试用例，
  - 也可从创建的事件流图导出测试用例

19.4 面向对象测试方法

- 面向对象体系结构产生了封装协作类的分层子系统
  - 每个系统成分（子系统和类）完成有助于满足系统需求的功能。
  - 有必要在不同的层次上测试面向对象系统，以发现错误。
  - 在类相互协作以及子系统穿越体系结构层通信时可能出现错误。
- 面向对象测试与传统的测试用例设计是不同的，
  - 传统测试用例根据输入-处理-输出视图或单个模块的算法细节设计
  - 面向对象测试侧重于设计适当的操作序列以检查类的状态

面向对象测试方法

1. 每个测试用例都应该被唯一地标识，并明确地与被测试的类相关联。
2. 应该叙述测试的目的。
3. 应该为每一个测试开发测试步骤，并包括以下内容 [BER94]:
   a. 将要测试的类的指定状态列表；
   b. 作为测试结果要进行检查的消息和操作列表；
   c. 对类进行测试时可能发生的异常列表；
   d. 外部条件列表（即软件外部环境的变更，为了正确地进行测试，这种环境必须存在）；
   e. 有助于理解或实现测试的补充信息。

面向对象测试方法

- 1 面向对象概念的测试用例设计含义
  - 封装是OO的本质特征之一，但可能成为测试的一个小障碍
  - 继承也为测试用例设计者提出了额外的挑战
  - 必须设计新的测试用例集
- 2 传统测试用例设计方法的可应用性
  - 白盒测试方法可以应用于类中定义的操作
  - 用例可为黑盒测试和基于状态的测试设计提供有用的输入
- 3 基于故障的测试
  - 目标是设计测试，该测试最有可能发现似然故障
  - 若分析和设计模型可以洞察有可能出错的事物，
  - 则基于故障的测试可以花费相当少的工作量而发现大量的错误

面向对象测试方法

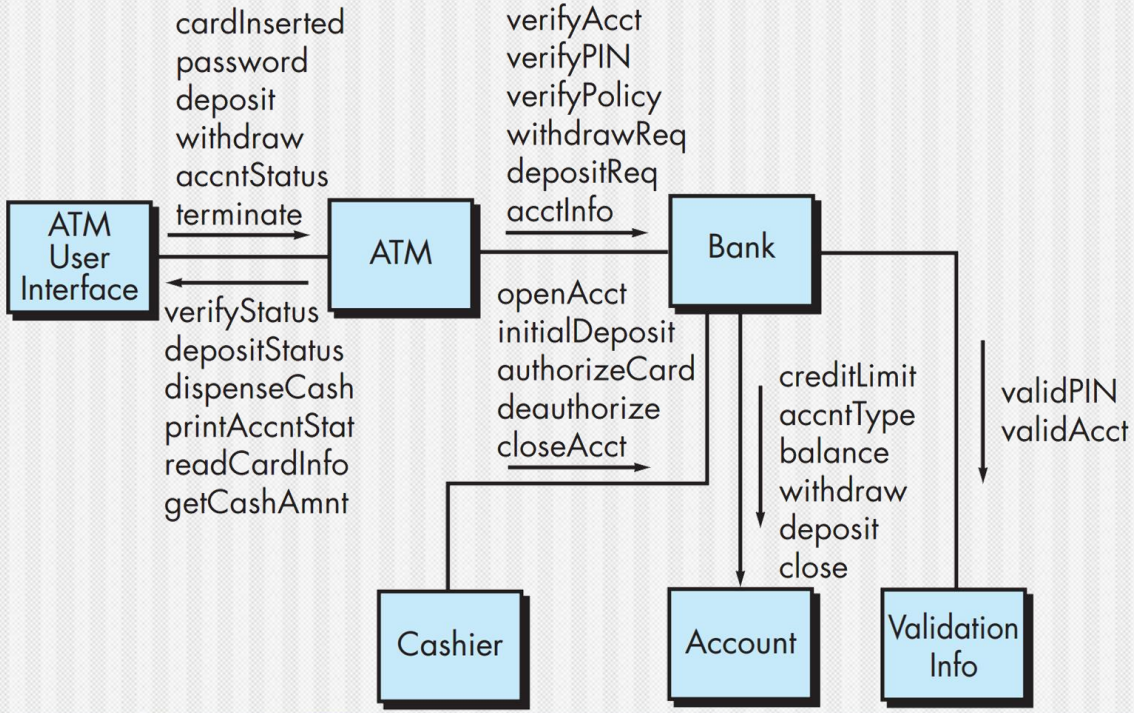
- 4 基于场景的测试
  - 基于故障测试忽略了两种主要错误类型：
    - （1）不正确的规格说明；
    - （2）子系统间的交互
  - 基于场景的测试关心用户做什么，而不是产品做什么
  - 基于场景的测试倾向于用单一测试检查多个子系统

19.5 类级可应用的测试方法

- “小范围”测试侧重于单个类及该类封装的方法。
- 随机测试和分割是可以用于检查类的测试方法。
- 1 面向对象的随机测试
  - 执行操作序列和其他随机顺序测试，检查不同类实例的生命历史
- 2 类级的划分测试
  - 与传统等价划分相似，划分测试减小测试特定类所需测试用例数量
  - 基于属性划分就是根据它们所使用的属性进行划分类操作
  - 基于类别划分就是根据每个操作所完成的一般功能进行划分类操作

19.6 类间测试用例设计

- 当开始集成面向对象系统时，测试用例的设计变得更为复杂
- 在这个阶段必须开始类间协作的测试。
- 与单个类的测试相类似，类协作测试可以通过运用
  - 随机和划分方法、基于场景测试及行为测试来完成



19.6.1 多类测试

- 生成多类随机测试用例的方法步骤：

  1、对每个用户类，使用类操作列表来生成一系列的随机测试序列。这些操作将向其他服务类发送消息；

  2、对生成的每个消息，确定协作类和服务对象中的相应操作；

  3、对服务对象中的每个操作（已被用户对象发送的消息调用）确定它所发送的消息；

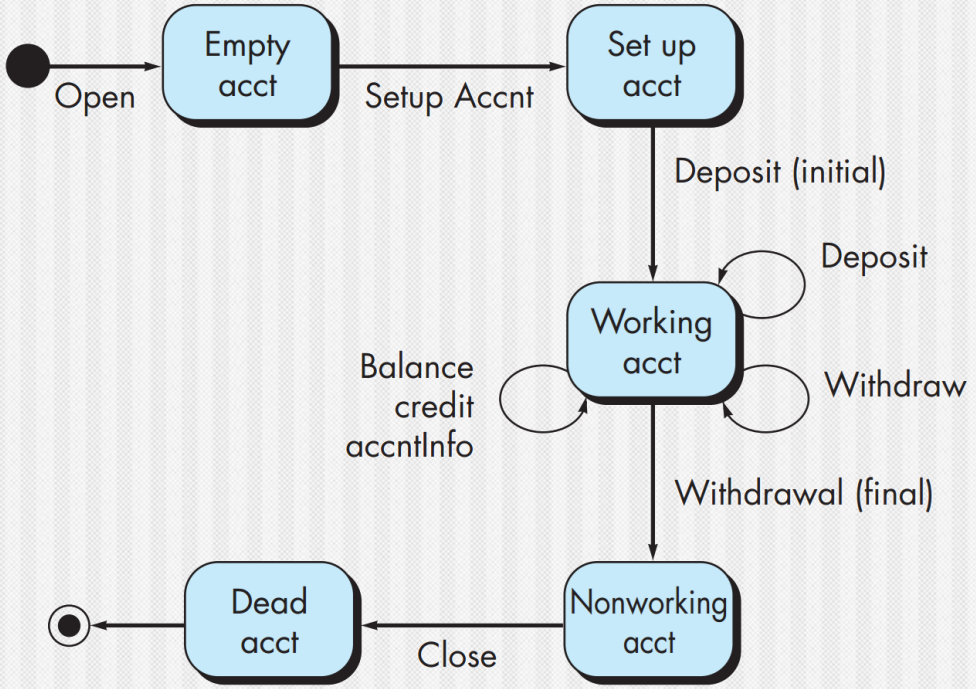
  4、对每个消息，确定下一层被调用的操作并将其引入到测试序列中。

- 多个类的划分测试方法与单个类的划分测试方法类似
  - 另一种划分测试方法是基于特殊类的接口的
  - 基于状态划分可用于进一步细化上述划分

19.6.2 从行为模型中导出的测试

- 类的状态图可辅助生成检查类的动态行为的测试序列。

- 可以通过“广度优先”的方式来遍历状态模型
  - 一个测试用例检查单个变换，
  - 且测试新的变换时，仅使用已测试过的变换



谢谢！