软件测试

第一章 绪论

两个基本职责：

- 验证：检验软件是否已正确实现说明书所定义的系统功能和特性
- 有效性确认：确认软件是否满足用户真正的需求

Beizer 关于软件测试目的的分级定义：

- 第 0 级：测试就是调试程序 ❌
- 第 1 级：测试的目的就是显示程序的正确性 ❌
- 第 2 级：测试的目的就是找出软件失败
- 第 3 级：测试的目的不是要证明一些具体的成功或失败，而是减少使用软件的风险
- 第 4 级：测试是一种精神上的修行，这种修炼可以帮助所有 IT 专业人员开发出质量更好的软件

软件测试的原则：

- 测试用例要精确定义程序预期输出
- 避免测试自己编写的程序（包括个人和组织）
- 应彻底检查测试执行结果，避免遗漏
- 有效和无效输入都要测试
- “做了应做的”和“没做不应做的”都要检查
- 避免测试用例用后即弃
- 计划测试工作时不应默许假定不会发现错误
- 发现错误越多，残存错误越多。大部分错误存于少数模块，应重点测试
- 软件测试富有创造性，极具智力挑战。

第二章 软件测试的基本概念、过程和分类

软件缺陷的含义：

- 软件未实现产品说明书要求的功能
- 软件出现了产品说明书指明不应该出现的错误
- 软件实现了产品说明书未提到的功能
- 软件未实现产品说明书虽未明确提及但应该实现的目标
- 软件难以理解、不宜使用、运行缓慢或者最终用户认为不好

V 模型：

- 特点
    - 定义了基本的开发过程和测试行为
    - 标明了测试过程中存在不同类型、不同级别的测试
    - 描述了不同测试阶段和开发过程间各阶段的对应关系

- 局限
    - 仅仅把测试过程作为在需求分析、系统设计及编码之后的一个阶段，忽视了对需求分析，系统设计的验证，需求的满足情况一直到后期的验收测试才被验证

W 模型：

- 特点
    - 增加了软件各开发阶段中应同步进行的验证和有效性确认活动
    - 基于“尽早地和不断地进行软件测试”的原则
- 局限
    - 把软件的开发视为需求、设计、编码等一系列的串行活动，无法解决需求变更等变更调整
    - 开发和测试保持线性的前后关系，上一阶段完成才能开始下一阶段，无法有效、快速支持产品迭代
    - 顺序模型中没有很好体现测试流程的完整性

黑盒测试：将被测程序当作看不见内部的黑盒子，主要根据规格说明书设计测试用例，不涉及程序内部构造和内部特性，只依靠被测程序输入和输出之间的关系或程序的功能设计测试用例。它是一种从用户观点出发的测试，一般被用来确认软件功能的正确性和可操作性。

白盒测试：将被测程序看作一个打开的盒子，测试者能够看到被测源程序，可以分析被测程序的内部结构，一般根据程序内部结构设计测试用例。它是一种从程序员观点出发的测试一般要求对某些程序的结构特性做到一定程度的覆盖。单元测试经常使用白盒测试。

冒烟测试：在对一个新版本进行系统大规模的测试之前，先验证一下软件的基本功能是否实现，是否具备可测试性。

随机测试：测试中所有的输入数据都是随机生成的，其目的是模拟用户的真实操作，并发现一些边缘性的错误。

X 模型存在一个探索性测试

对比维度	黑盒测试	白盒测试
测试视角	外部视角（用户视角）	内部逻辑视角（开发者视角）
测试依据	需求规格说明	代码逻辑、代码结构
核心目标	验证功能是否符合需求，发现功能缺陷	验证代码逻辑、代码结构正确性，发现内部缺陷
适用阶段	需求确定后（贯穿全周期，侧重系统测试、验收测试）	编码后（单元测试、集成测试为主）
方法举例	等价类划分；边界值分析；因果图	语句覆盖；判定覆盖；条件覆盖；路径覆盖；循环覆盖

第三章 黑盒测试

因果图

- `E` 约束：`a b` 不能同时为 `1`
- `I` 约束：`a b` 不能同时为 `0`

- O 约束：a b 有且只有一个为 1
- R 约束：$a = 1 \Rightarrow b = 1$
- （结果）M 约束：$a = 1 \Rightarrow b = 0$

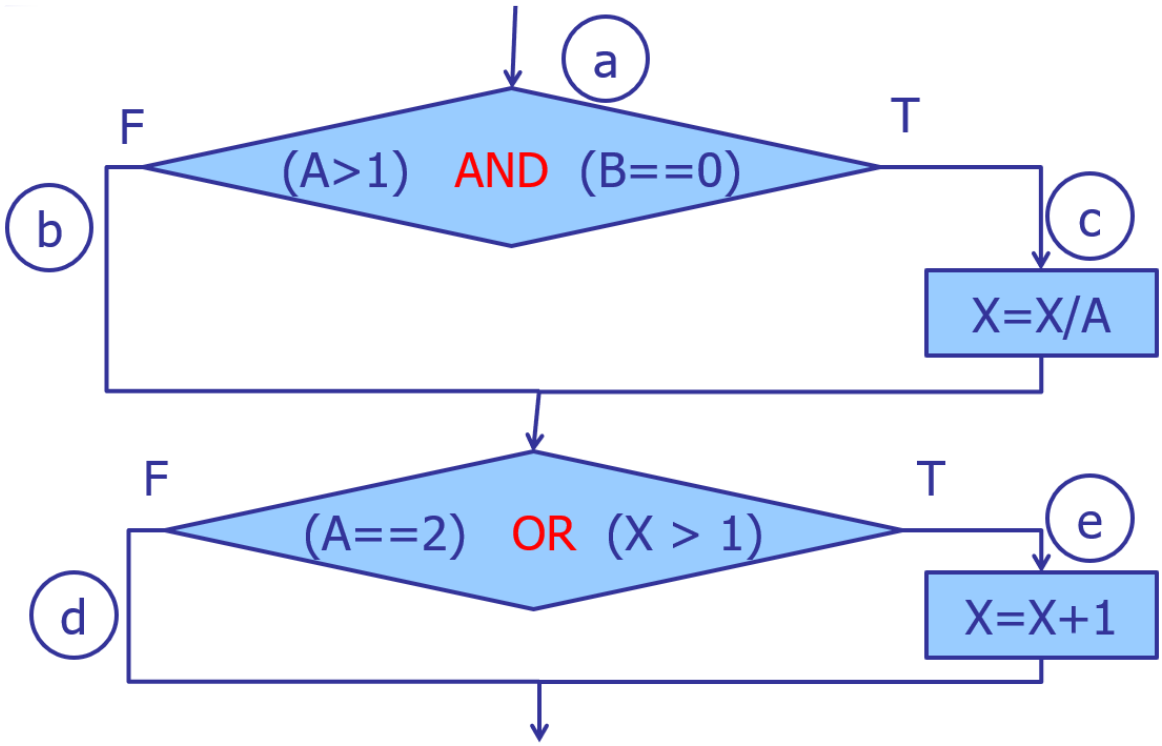
第四章 白盒测试

环形复杂度

基于控制流图

- V(G) = 区域数
- V(G) = E - N + 2（边 - 点 + 2）
- V(G) = P + 1（判定点 + 1）

路径覆盖反例

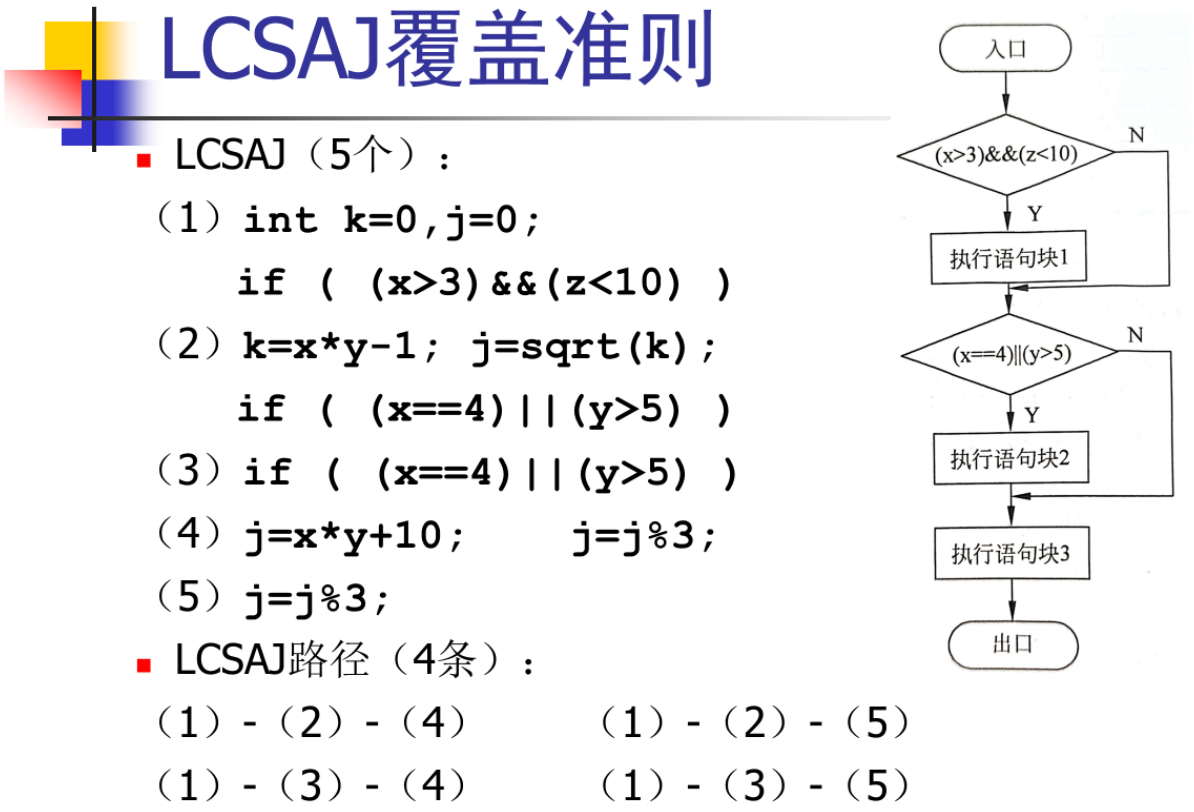


用例	A B X	路径	覆盖组号	覆盖条件
Case 1	2 0 4	ace	1, 5	T1 T2 T3 T4
Case 3	1 0 1	abd	3, 8	F1 T2 F3 F4
Case 5	2 1 1	abe	2, 6	T1 F2 T3 F4
Case 8	3 0 1	acd	1, 8	T1 T2 F3 F4

LCSAJ 覆盖准则

第 1 层：语句覆盖
第 2 层：分支覆盖
第 3 层：LCSAJ 覆盖，即程序中的每一个 LCSAJ 都至少在测试中经历过一次。
第 4 层：两两 LCSAJ 覆盖，即程序中的每两个相连的 LCSAJ 组合起来在测试中都要经历一次。
第 n + 2 层：每 n 个首尾相连的 LCSAJ 组合在测试中都要经历一次。

- 含义：在程序中，一个 LCSAJ 是一组顺序执行的代码以控制跳转为其结束点。
- 起点：根据程序本身决定的。它的起点可以是程序第一行或转移语句的入口点，或是控制流可跳达的点。



循环测试方法

测试简单循环，设其循环最大次数为 n，可采用以下测试集：

- 跳过整个循环
- 只循环一次
- 只循环两次
- 循环 m 次，其中 m < n
- 分别循环 n - 1，n，n + 1 次

测试嵌套循环：

- 测试从最内层循环开始，所有外层循环次数设置为最小值；
- 对最内层循环按照简单循环的测试方法进行；
- 由内向外进行下一个循环的测试，本层循环的所有外层循环仍取最小值，而由本层循环嵌套的循环取某些“典型”值；

- 重复上一步的过程，直到测试完所有循环

测试串接循环：

- 若串接的各个循环相互独立，则可分别采用简单循环的测试方法；
- 否则采用嵌套循环的测试方法。

Z 路径覆盖是路径覆盖的一种变体，它是将程序中的循环结构简化为选择结构的一种路径覆盖。即：循环要么执行、要么跳过。

计算在 Z 路径覆盖下达到路径覆盖的最少测试用例数（N-S 图）

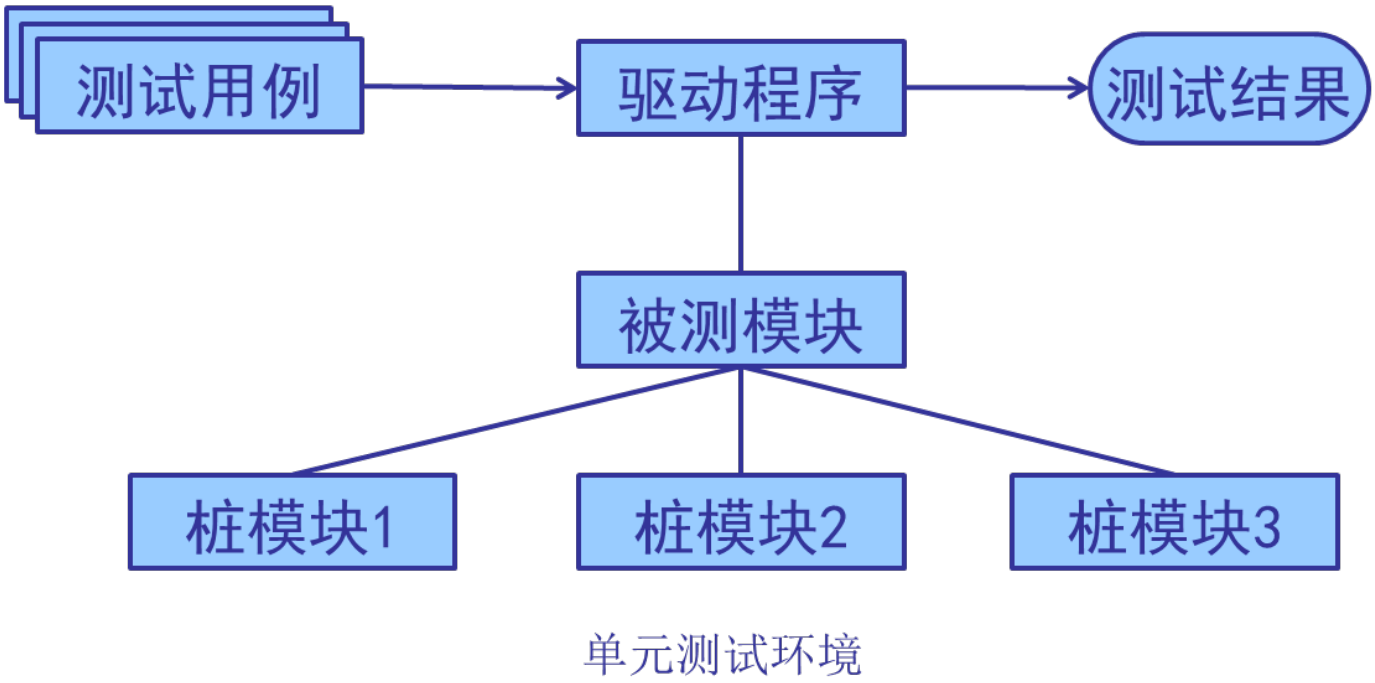
第五章 单元测试

驱动模块：相当于被测基本单元的主程序，它接收测试数据，并把数据传送给被测单元，调用被测模块，最后输出实测结果。

桩模块（存根模块）：用来代替被测基本单元调用的其他基本单元。

驱动模块和桩模块是测试时使用的软件，而不是软件产品的组成部分，但它需要一定的开发费用。桩模块和驱动模块的设计都需要一定的研发成本。

会画下面的图：



单元测试环境

五个基本特性：

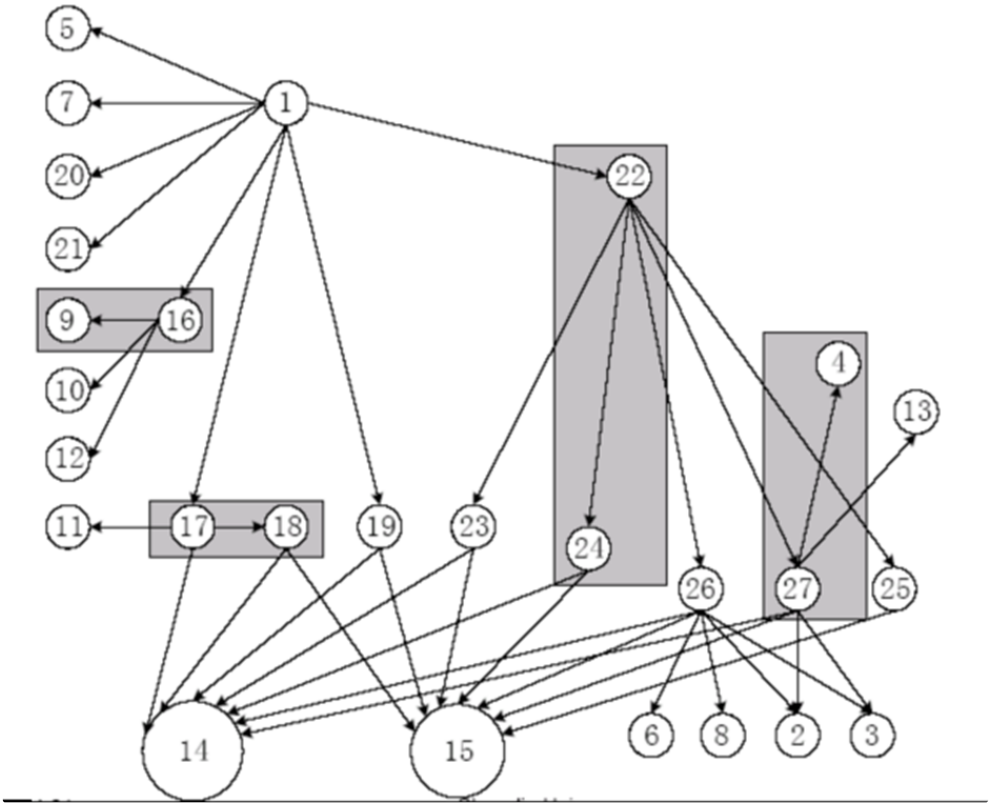
- 模块接口
- 局部数据结构
- 边界条件
- 重要执行路径
- 出错处理

第六章 集成测试

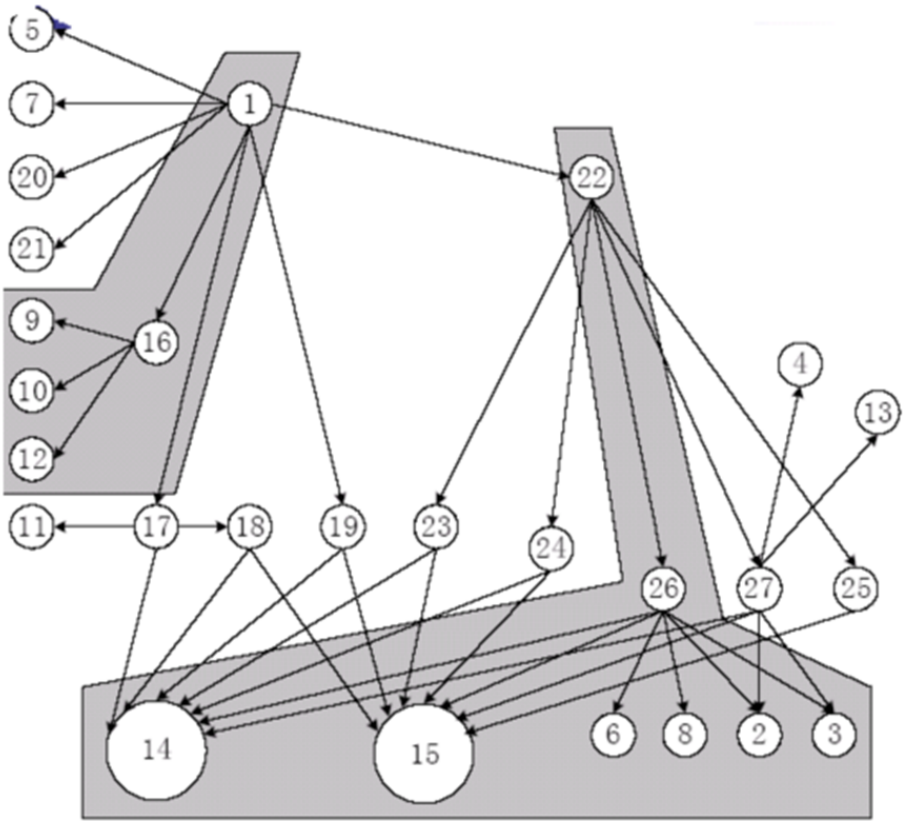
集成测试策略

- 基于功能分解
  - 非渐增式集成
  - 渐增式集成
    - 自顶向下集成
      - 深度优先策略
      - 广度优先策略
    - 自底向上集成
    - 三明治集成
    - 修改的三明治集成
  - 基干集成
- 基于调用图
  - 成对集成
  - 相邻集成

成对集成：调用图的每一个边建立并执行一个集成测试会话，最终使得调用图的每条边都有一个集成测试会话，同时又大大降低桩/驱动模块的开发工作。



相邻集成：每个内部结点及其邻居组成集成测试会话并进行测试，如果根节点直接和叶结点相邻，则根节点及其邻居也要组成一个测试会话。



第七章　系统测试

向前兼容：向未来

向后兼容：向过去

易用性测试：主要测试软件易用性。易用性是交互适应性、实用性和有效性的集中体现。

回归测试：对某些已经进行过的测试的某些子集再重新进行一遍，以保证（由于测试或其他原因的）改动不会带来不可预料的行为或另外的错误。

α 测试：是一个用户在开发环境下进行的测试，也可以是开发机构内部的用户在模拟实际操作环境下进行的测试。

β 测试：就是把产品有计划的分发到目标市场，从市场收集反馈信息，把关于反馈信息的评价制成易处理的数据表，再把这些数据分发给所涉及的各个部门。

测试成熟度模型　TMM

- **初始级**：测试过程无序、混乱，测试与调试混为一谈
- **定义级**：测试被定义为软件生命周期的一个阶段，与调试明确被区分开
- **集成级**：测试贯穿整个软件生命周期中，但没有建立起有效的评审制度
- **管理和度量级**：测试是一个管理和质量控制过程，评审作为其中一部分
- **优化级**：不断地进行测试改进

第九章　软件错误与程序排错

错误植入法：测试前由专人在程序中随机植入一些错误。

假设人为植入的错误数为 N_s，经过一段时间的测试后发现 n_s 个植入的错误，此外还发现了 n 个原有的错误。如果认为测试方案发现植入错误和发现原有错误的能力相同，则能够估计出程序中原有错误的总数为：

$$\widehat{N} = \frac{n}{n_s} N_s \tag{1}$$

归纳法：（从特殊到一般，即从具体现象提炼共同特征从而得到缺陷假设）研究与出错相关的信息，找出特征，提出“原因假设”，然后确认或否认该假设。

演绎法：（从一般到特殊，即从通用知识等出发提出可能的缺陷假设，再根据信息验证。）先列举一些可能的原因或假设，然后进行逐个分析，排除不能确立的原因和假设，直到仅剩一个被证实。