软件测试
--概述

引言 --- 为什么开设这门课程

- 软件规模的日益庞大和复杂
- 软件支撑各行各业的核心业务
- 软件质量要求日益提高
- 软件测试作为保证软件质量的最直接最重要的一种手段，在软件开发过程中日益重要
- 软件测试工程师的需求量日益增加

引言

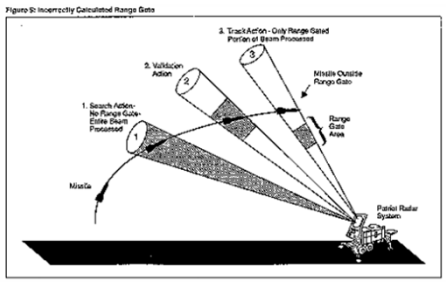
- 最大方的自动柜员机
  - 2001年1月，欧元正式流通后几天，德国的某些自动柜员机竟能在提款后不记入帐目！

引言

- 1996年6月，欧洲航天局（ESA）阿利亚娜5型火箭在发射37秒以后爆炸（8年多，80亿美元）
  - 原因：运算溢出错误

引言

- 爱国者导弹反导失败（海湾战争，1991年2月）
  - 原因：积累的时钟误差



主要内容

- 为什么需要测试
- 测试用例
- 标识测试用例
- 错误与缺陷分类
- 测试级别

主要内容

- 为什么需要测试
- 测试用例
- 标识测试用例
- 错误与缺陷分类
- 测试级别

什么是软件测试

- **软件测试是为了发现错误而执行程序的过程**
- **广义上讲，在软件开发过程中的所有评审、确认、检验等活动都是软件测试**
- **软件测试与软件使用是不一样的**

为什么要进行测试

- 对质量或可接受性做出判断
- 发现问题

- 测试不能表明软件中不存在错误！！

软件测试的重要性

- 软件开发成本分析

软件类型	开发成本按阶段分布（%）		
	需求与设计	实现	测试
控制软件	46	20	34
航空航天软件	34	20	46
操作系统	33	17	50
科技计算软件	44	26	30
商业应用软件	44	28	28

术语解释

- 错误（error）-- 同义词是过错（mistake），人们在编写代码时会出现过错，这种过错叫做bug
- 缺陷（fault）-- 缺陷是错误的结果。分为过错缺陷和遗漏缺陷
- 失效（failure）-- 当缺陷执行时会发生失效
- 事故（incident）-- 当出现失效时，可能会也可能不会呈现给用户（或客户或测试人员）

术语解释

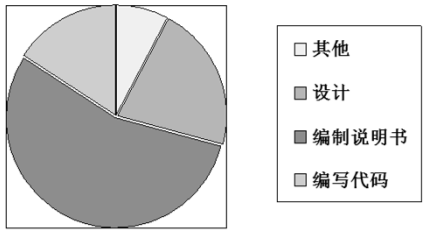
- 测试（test）-- 测试显然要处理错误、缺陷、失效和事故。测试是采用测试用例执行软件的活动
- 测试目标：找出失效，或演示正确的执行
- 测试用例（test case）-- 测试用例有一个标识，并与程序行为有关。测试用例还有一组输入和一个预期输出表
- **测试过程（test process）** - 测试计划、测试用例开发、运行测试用例以及评估测试结果

测试用例在测试中占据中心地位

课程的重点在于如何标识有用的测试用例集合

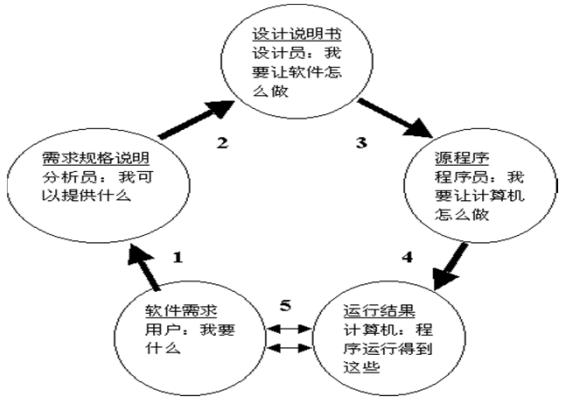
为什么会出现软件缺陷？

- 从小程序到大项目的无数研究得出：导致软件缺陷最大的原因是产品说明书（需求）
- 其次的原因是设计方案的问题

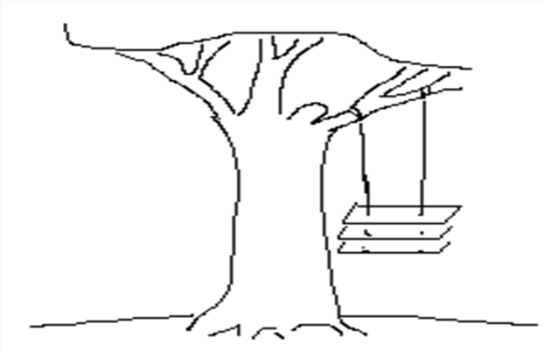


13

软件开发主要环节

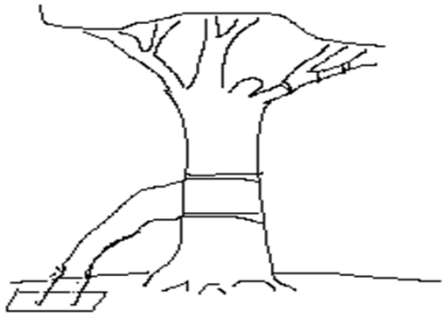


分析员的设想

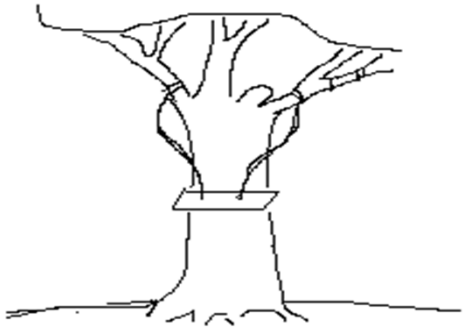


分析员的描述

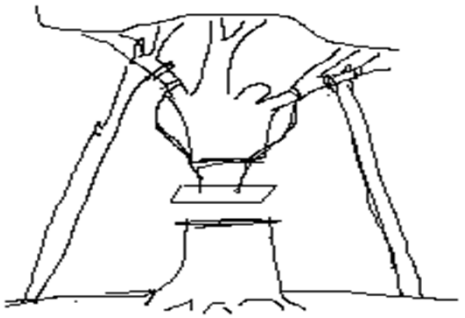
完成的设计



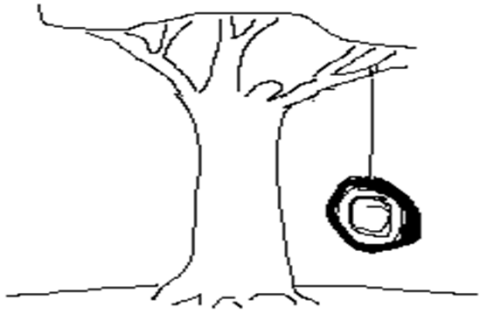
程序员做出的产品



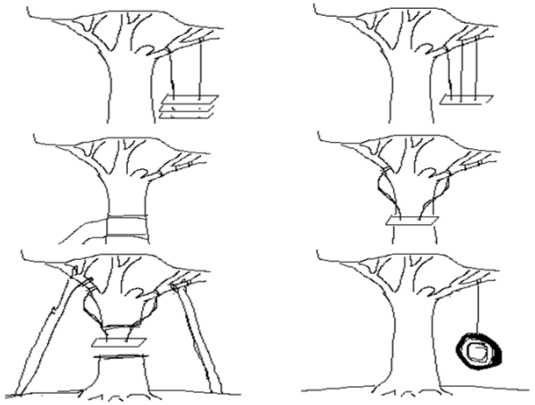
用户安装后的结果



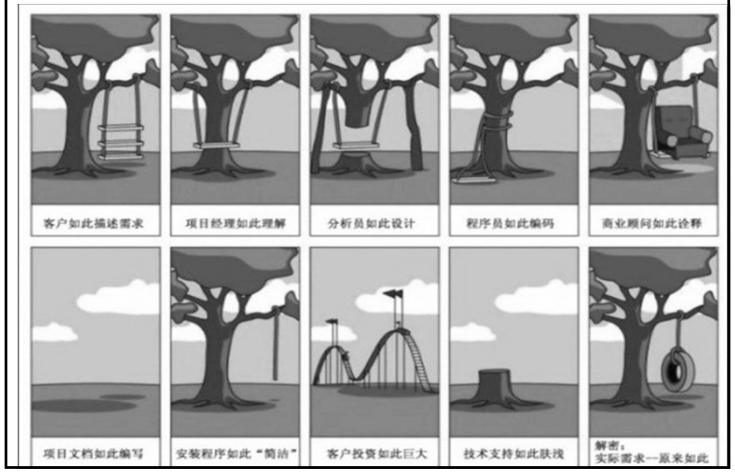
用户的原始需求



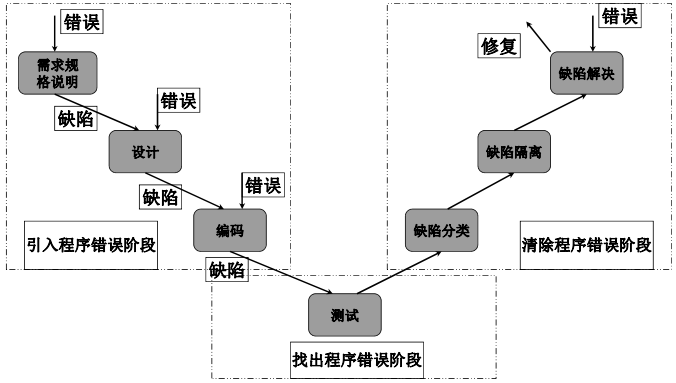
需求的变迁



需求的变迁

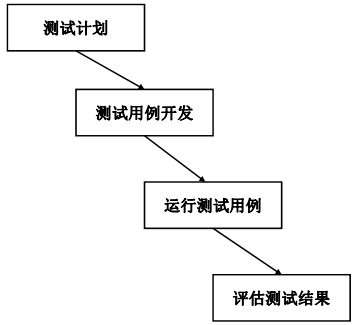


测试生命周期模型



测试子过程

- 测试过程细分步骤：



主要内容

- 为什么需要测试
- 测试用例
- 标识测试用例
- 错误与缺陷分类
- 测试级别

测试用例

- 软件测试的本质是针对要测试的内容确定一组测试用例
- 测试用例的输出部分常常被忽视，因为输出部分的确定往往很困难
- 工业界的一种回答叫做“参考测试”(Reference Testing)，
- 即系统要在专家用户的指导下进行测试，这些专家要判断被执行的一组测试用例的输出是否为可接受的
- 测试活动要建立必要的前提条件，提供测试用例输入，观察输出，然后将这些输出与预期输出进行比较，以确定该测试是否通过

测试用例的组成

- 标准：
  - 输入： 前提、由某种测试方法所标识的实际输入
  - 预期输出：后果和实际输出
  - 测试结果：实际执行结果是否与预期结果一致
- 附加：
  - 操作过程：一个测试用例中的操作步骤
  - 验证过程: 验证测试用例是否通过的操作步骤

典型测试用例信息

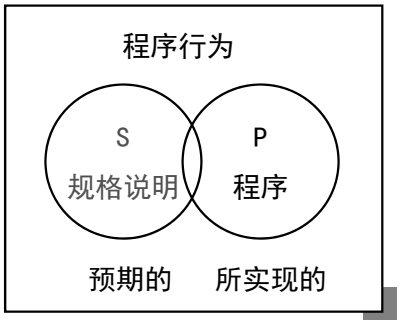
测试用例ID
目的
前提
输入
预期输出
后果
执行历史
日期 结果 版本
执行人

- 清楚地给出这些信息会使测试用例更有价值
- 测试用例需要被开发、评审、使用、管理和保存

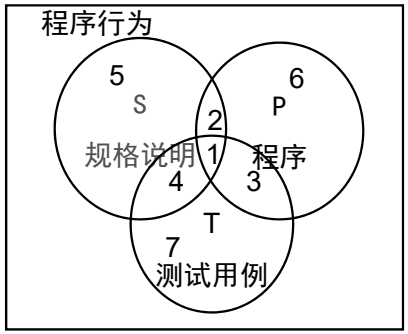
通过维恩图理解测试

- 测试基本上关心的是行为，而行为与软件(和系统)开发人员很常见的结构视图无关
- 结构视图关注的是它是什么，而行为视图关注的是它做什么
- 基本文档通常都是由开发人员编写，强调的是结构信息

- **集合S是所描述的行为，集合P是用程序实现的行为**



通过维恩图理解测试



- **因此测试最主要的是使S、P、T都相交的区域，即区域1尽可能的大。**

主要内容

- 为什么需要测试
- 测试用例
- 标识测试用例
- 错误与缺陷分类
- 测试级别

设计测试用例

- 功能性测试，又叫黑盒测试
- 结构性测试，又叫白盒测试

什么是功能性测试

- 功能性测试的基本观点是，任何程序都可以看作是将从输入定义域取值映射到输出值域的函数
- 采用功能性测试方法设计测试用例的唯一依据是软件的规格说明
- 由于功能性方法基于已描述行为，因此很难想象这些方法能够标识没有被描述的行为

功能性测试的优缺点

- 优点:
  - 功能性测试与软件如何实现无关，所以如果实现发生变化，测试用例仍然有用
  - 测试用例开发可以与实现并行进行，因此可压缩项目开发时间
- 缺点:
  - 测试用例之间可能存在严重的冗余
  - 此外可能还会有未测试的软件漏洞

功能性测试的维恩图表示

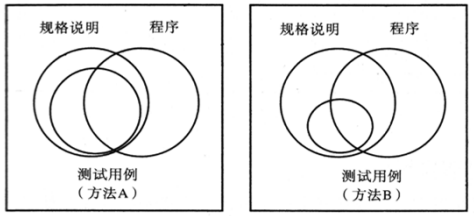


图1-6 功能性测试用例标识方法比较

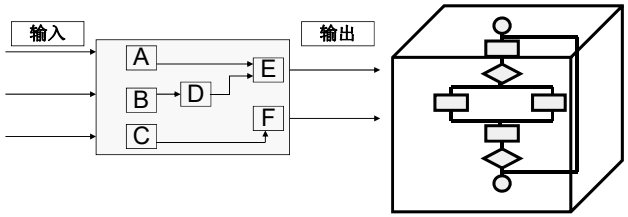
方法A标识了比方法B更大的测试用例集合。请注意，对于这两种方法，测试用例集合完全局限在已描述行为集合内。

功能性测试的主流方法

- 边界值分析
- 健壮性分析
- 最坏情况分析
- 特殊值测试
- 输入等价类
- 输出等价类
- 基于决策树的测试

什么是结构性测试

- 结构性测试又叫做白盒测试，或透明盒测试
- 结构性测试是根据程序实现来设计测试用例



结构性测试

- 把测试对象看做一个透明的盒子，它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
- 通过在不同点检查程序的状态，确定实际的状态是否与预期的状态一致

结构性测试的优缺点

- 优点:
  - **可构成测试数据对特定程序部分测试，可以检测代码中的每条分支和路径**
  - **揭示隐藏在代码中的错误**
  - **对代码的测试比较彻底**
  - **有一定的充分性度量手段**
- 缺点:
  - **工作量大, 成本高，通常只用于单元测试**
  - **无法检测代码中遗漏的路径和数据敏感性错误**
  - **不能验证规格说明的正确性**
  - **无法对规格说明中未实现的部分进行测试**

结构性测试的维恩图表示

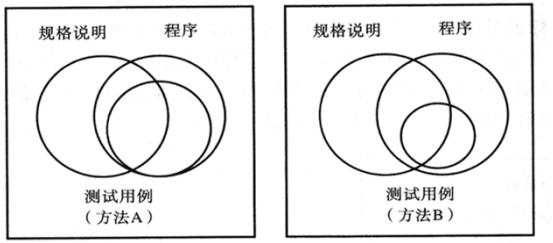


图1-7 结构性测试用例标识方法比较

对于结构性测试，不管采用什么方法，测试用例集合都完全局限于已编程实现的行为集合中。

（基于程序行为）

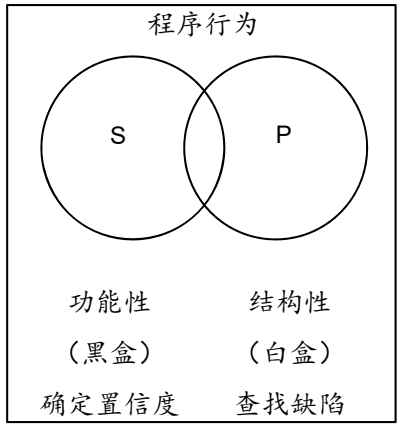
结构性测试的主要方法

- 路径测试
  - DD-路径
  - 基路径测试
- 数据流测试
  - 定义-使用测试
  - 基于程序片的测试

功能性测试和结构性测试的比较

- **功能性测试**只利用**规格说明**来设计测试用例
- **结构性测试**使用**程序源代码**作为设计用例的基础
- 因此：
  - 如果所有已描述行为都没有被实现，则结构性测试永远也不会认识到这一点
  - 如果程序实现了没有被描述的行为，功能性测试永远不会揭示这一点
- 明智的组合会带来功能性测试的置信，以及结构性测试的度量，即功能性测试与结构性测试相结合（灰盒测试）

测试用例的来源



主要内容

- 为什么需要测试
- 测试用例
- 标识测试用例
- 错误与缺陷分类
- 测试级别

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

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， 0.00美元账单
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	扩展到其他系统的系统停机

一些典型缺陷

输入/输出缺陷

类型	举例
输入	不接受正确的输入
	接受不正确的输入
	描述有错或遗漏
	参数有错或遗漏
输出	格式有错
	结果有错
	在错误的时间产生正确的结果(太早、太迟)
	不一致或遗漏结果
	不合逻辑的结果
	拼写/语法错误
	修饰词错误

数据缺陷

不正确的初始化
不正确的存储/访问
错误的标志/索引值
不正确的打包/拆包
使用了错误的变量
错误的数据引用
缩放数据范围或单位错误
不正确的数据维数
正确的下标
不正确的类型
不正确的数据范围
传感器数据超出限制
出现I次断开
不一致的数据

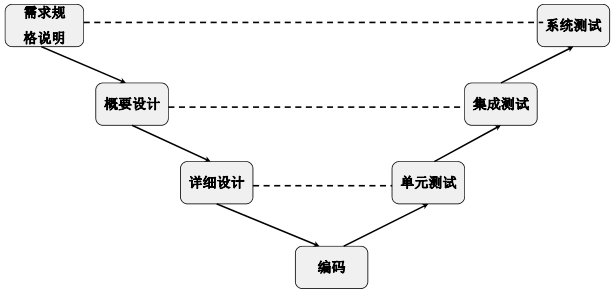
一些典型缺陷

计算缺陷	接口缺陷	逻辑缺陷
不正确的算法	不正确的中断处理	遗漏情况
遗漏计算	I/O时序有错	重复情况
不正确的操作数	调用了错误的过程	极端条件出错
不正确的操作	调用了不存在的过程	解释有错
括号错误	参数不匹配(类型，个数)	遗漏条件
精度不够(四舍五入，截断)	不兼容的类型	外部条件有错
错误的内置函数	过量的包含	错误变量的测试
		不正确的循环迭代
		错误的操作符(例如用 < 取代了<=)

主要内容

- 为什么需要测试
- 测试用例
- 标识测试用例
- 错误与缺陷分类
- 测试级别

测试级别



软件测试的四个阶段

- 单元测试
  - 模块测试，最小单位的测试
- 集成测试
  - 综合测试，它揭示接口规约中不够完整或错误的地方
- 系统测试
  - 发现软件和需求不符合或与之矛盾的地方
- 验收测试
  - 受客户控制，以用户为主的测试，由用户参加设计测试用例，一般使用生产中的实际数据进行测试

总结

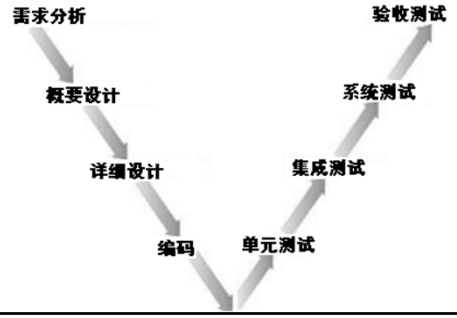
- 错误、缺少、失效、事故的定义是什么？
- 为什么需要软件测试？
- 测试用例是什么？
- 功能性测试、结构性测试分别是什么？他们的异同是什么？
- 软件测试包含哪些级别？

补充---软件测试模型

- **V模型**
- **W模型**
- **X模型**
- **H模型**

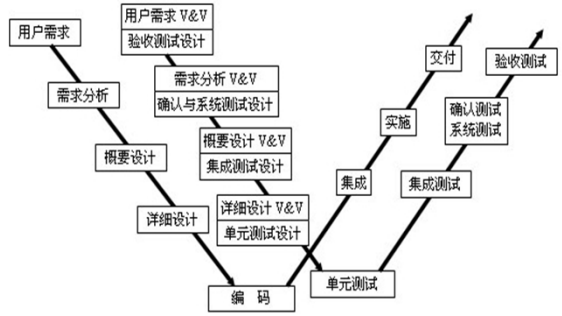
V模型

- 反映了测试活动与分析设计的关系
- 局限性：把测试作为编码之后的最后一个活动，在需求分析、系统设计阶段等前期产生的错误直到后期的验收测试才能发现



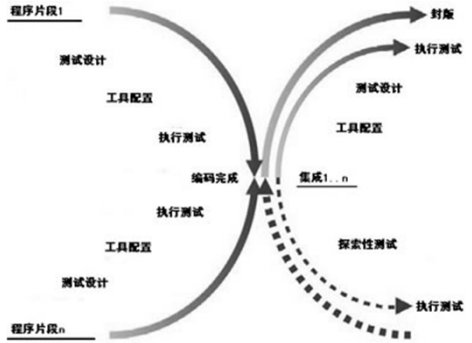
W模型

- 基于“尽早地和不断地进行软件测试”原则。测试与开发同步进行，有利于尽早的发现问题。
- 局限性：仍把开发活动看成是从需求开始到编码结束的串行活动，不能支持迭代，自发性以及变更调整



X模型

- X模型提出针对单独的程序片段进行相互分离的编码和测试,此后通过频繁的交接,通过集成最终合成为可执行的程序



H模型

- H模型揭示了一个原理:软件测试是一个独立的流程,贯穿产品整个生命周期,与其他流程并发地进行
- H模型指出软件测试要尽早准备, 尽早执行。