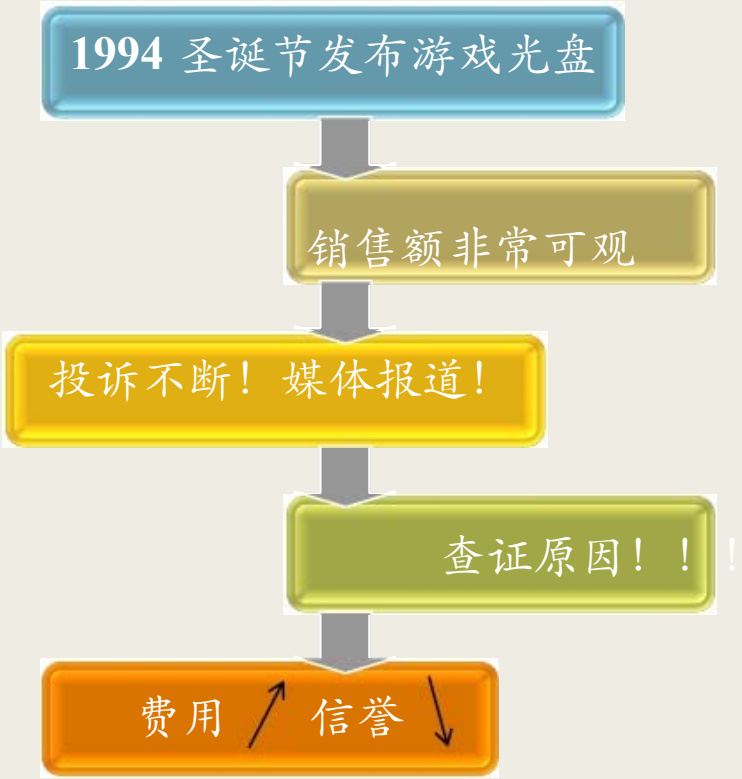
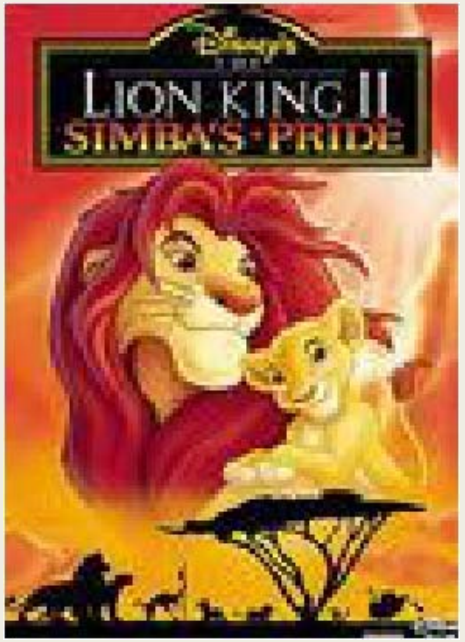
软件测试

张程
Email：bootan@cqu.edu.cn
QQ:80463125

软件测试的背景

迪斯尼的狮子王游戏软件缺陷（1994-95）





爱国者导弹防御系统缺陷（1991）

- 闻名于海湾战争，多次成功防御伊拉克飞毛腿导弹
- 但在对抗导弹仍多次失利，甚至在沙特多哈击毙28名美国士兵
- 原因
  - *系统时钟的一个很小的计时错误*
  - *错误累积到14小时后，跟踪系统不再准确*
  - *多哈的袭击中，系统已经运行了100多小时*

千年虫问题（1974–2000）

- 在某些使用了计算机程序的智能系统（包括计算机系统、自动控制芯片等）中，由于其中的年份只使用两位十进制数来表示，因此当系统进行（或涉及到）跨世纪的日期处理运算时（如多个日期之间的计算或比较等），就会出现错误的结果，进而引发各种各样的系统功能紊乱甚至崩溃

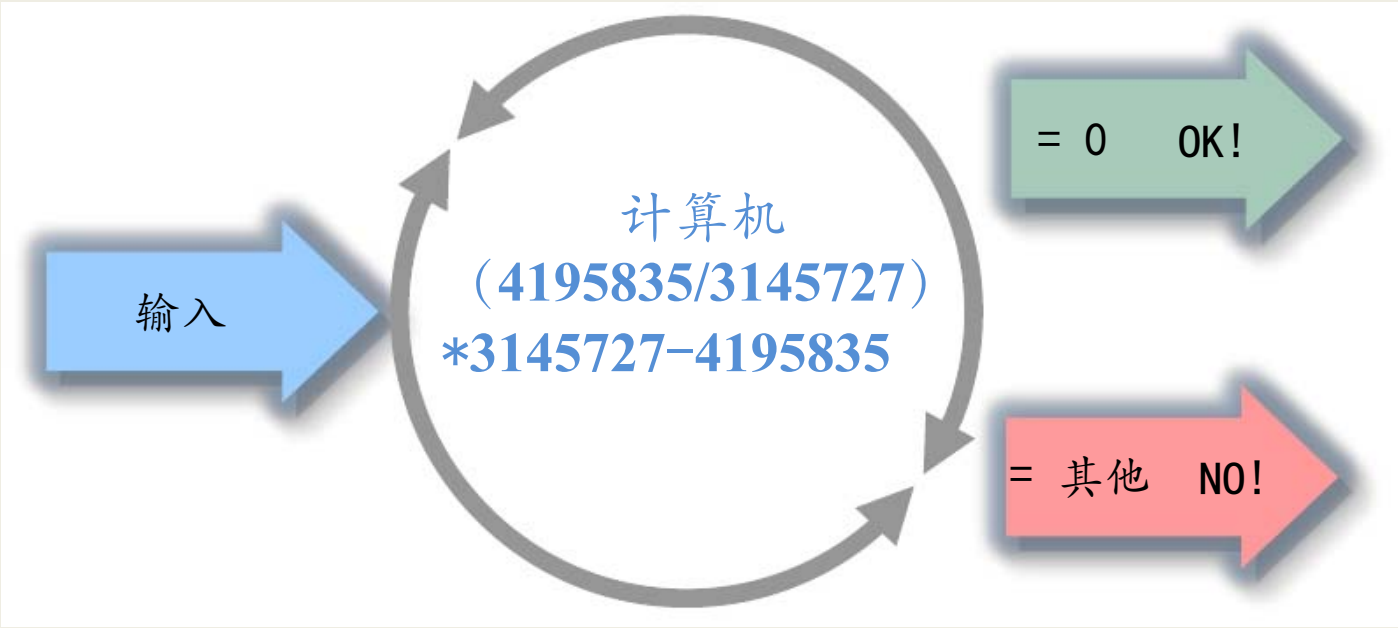
- 根源始于60年代。当时计算机存储器的成本很高，如果用四位数字表示年份，就要多占用存储器空间，就会使成本增加，因此为了节省存储空间，计算机系统的编程人员采用两位数字表示年份
- 影响巨大
  - 冈比亚政府已宣布当天（周一）为非工作日，以暂时减轻出事机关所要承受的压力
  - 著名的7-Eleven便利连锁店的计算机把2001年当成1901年，使许多使用信用卡的用户感到不便
  - 由于电脑千年虫作怪，瑞典多达10万网上银行客户在进入2000年之后无法进入网上账户

美国航天局火星登陆探测器缺陷（1999）

- 1999年12月3日，美国航天局的火星极地登陆者号探测器试图在火星表面着陆时坠毁
- 故障原因：一个数据位被意外置位，小组间协作有漏洞
  - *多个组对探测器进行测试*
  - *一个小组测试飞船的脚折叠过程。该组不去注意着地数据位是否置位（因为这不是他们负责的范围）*
  - *另一个组测试此后的着陆过程。该组总是在开始测试之前复位计算机，清楚数据位。*
- 警醒：内部测试!!!

英特尔奔腾浮点除法缺陷(1994)

- Intel的Pentium老式芯片在浮点数除法存在软件缺陷
- 软件测试工程师在芯片发布前的内部测试中**已发现**该问题，Intel管理层认为没严重到一定要修正甚至公开的程度
- 软件缺陷被发现后，Intel通过公开声明试图弱化该问题的已知严重性
- Intel最终道歉且花费4亿美元支付更换问题芯片



软件质量

基本概念

- ISO8492：质量是产品或服务所满足明示或暗示需求能力的特性和特征的集合
- IEEE：质量是被普遍接受的概念。是系统、部件或过程满足明确需求
- 软件质量主要涉及软件开发所采用的技术、软件开发人员的能力、对软件开发过程的质量控制和软件开发所用的时间与成本等因素
- GB/T 16260-2006《软件工程 产品质量》标准认为：软件产品质量的需求一般要包括对于内部质量、外部质量和使用质量的评估准则，以满足开发者、维护者、需方以及最终用户的需要。

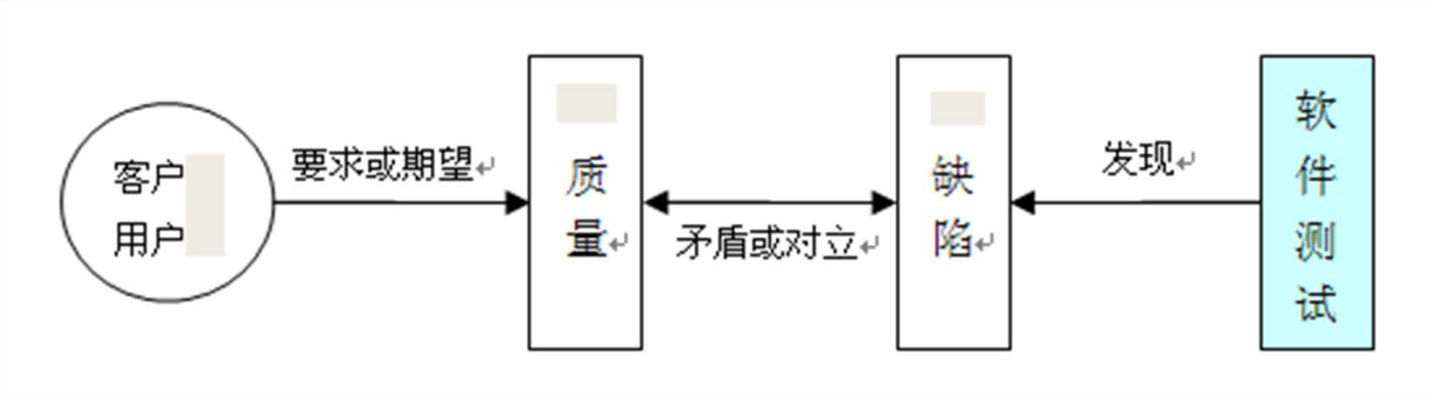
软件质量包含的特性

- 功能性
  - *适合性、准确性、互操作性、安全保密性、依从性*
- 可靠性
  - *成熟性、容错性、易恢复性、吸引性、依从性*
- 易用性
  - *易理解性、易学性、易操作性、吸引性、依从性*
- 效率
  - *时间特性、资源利用性、依从性*
- 维护性
  - *易分析性、易改变性、稳定性、易测试性、依从性*
- 可移植性
  - *适应性、易安装性、共存性、易替换性、依从性*

软件缺陷

缺陷是质量的对立面

- 缺陷是相对质量而存在的，违背了质量、违背了客户的意愿，不能满足客户的要求，就会引起缺陷或产生缺陷



与缺陷相近的词

- 缺点
- 偏差
- 故障
- 失败
- 问题
- 不一致
- 错误
- 缺陷
- 异常。。。。。。

什么是软件缺陷

- 当满足以下几种情况之一，则可以称为发生了软件缺陷
  - *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功能*
  - *软件难以理解、不易使用、运行缓慢或者（从测试员的角度看）最终用户会认为不好*

- [illegible]，
  [illegible]。
  规定号键，结果没有任何反应。

思考：ATM机吞卡现象是不是一个bug?

软件缺陷的特征

- “看不到”
  - *软件的特殊性决定了缺陷不易看到*
- “看到但抓不到”
  - *发现了缺陷，但不易找到问题发生的原因所在*

软件缺陷产生的原因

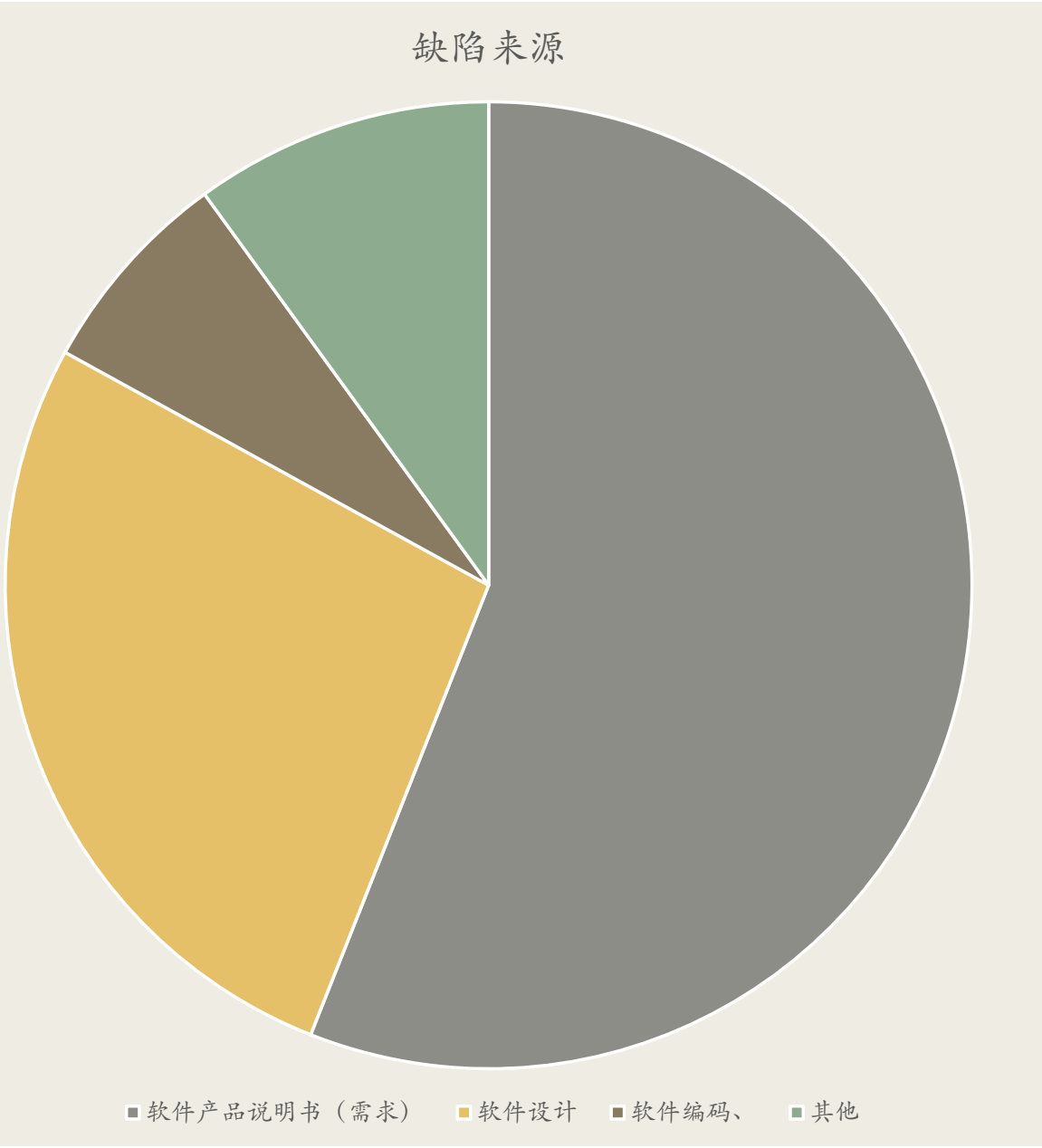
- 技术问题
  - *算法错误。*
  - *语法错误。*
  - *计算和精度问题。*
  - *系统结构不合理，造成系统性能问题。*
  - *接口参数不匹配出现问题*
- 团队工作
  - *系统分析时对客户的需求不是十分清楚，或者和用户的沟通存在一些困难。*
  - *不同阶段的开发人员相互理解不一致，软件设计对需求分析结果的理解偏差，编程人员对系统设计规格说明书中某些内容重视不够，或存在着误解。*
  - *设计或编程上的一些假定或依赖性，没有得到充分的沟通*

软件缺陷产生的原因

- 软件本身
  - *文档错误、内容不正确或拼写错误。*
  - *数据考虑不周全引起强度或负载问题。*
  - *对边界考虑不够周全，漏掉某几个边界条件造成的错误。*
  - *对一些实时应用系统，保证精确的时间同步，否则容易引起时间上不协调、不一致性带来的问题。*
  - *没有考虑系统崩溃后在系统安全性、可靠性的隐患。*
  - *硬件或系统软件上存在的错误。*
  - *软件开发标准或过程上的错误。*

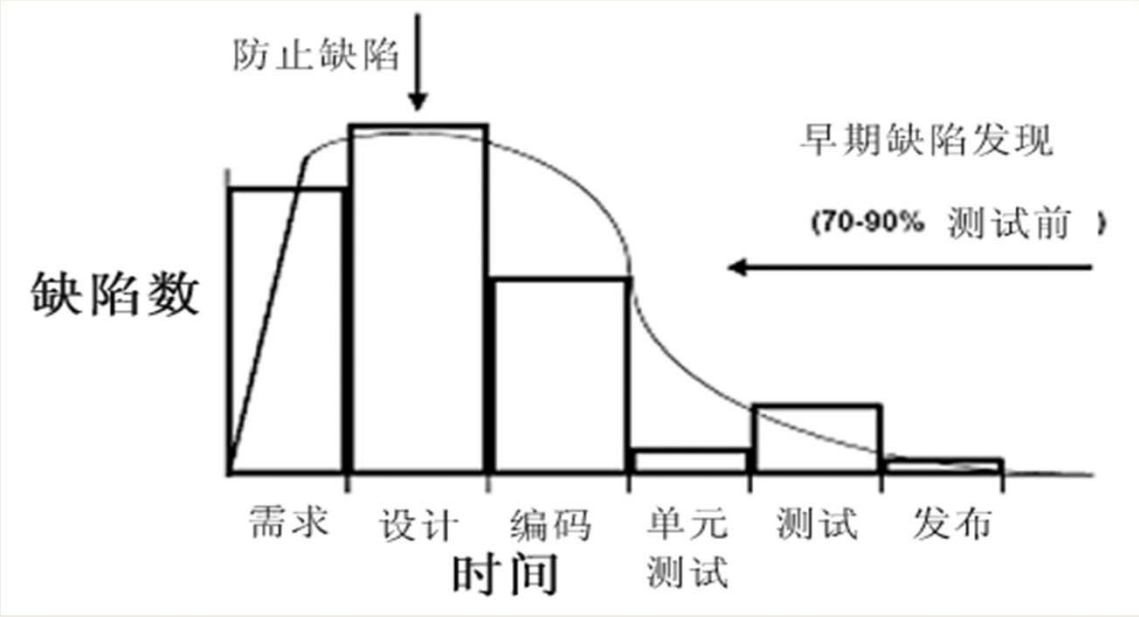
软件缺陷来源

- **为什么需求阶段最多**
  - *用户一般是非计算机专业人员，软件开发人员和用户的沟通存在较大困难，对要开发的产品功能理解不一致。*
  - *由于软件产品还没有设计、开发、完全靠想象去描述系统的实现结果，所以有些特性还不够清晰。*
  - *需求变化的不一致性。用户的需求总是在不断变化的，这些变化如果没有在产品规格说明书中得到正确的描述，容易引起前后文，上下文的矛盾。*
  - *对规格说明书不够重视，在规格说明书的设计和写作上投入的人力，时间不足。*
  - *没有在整个开发队伍中进行充分沟通，有时只有设计师或项目经理得到比较多的信息。*



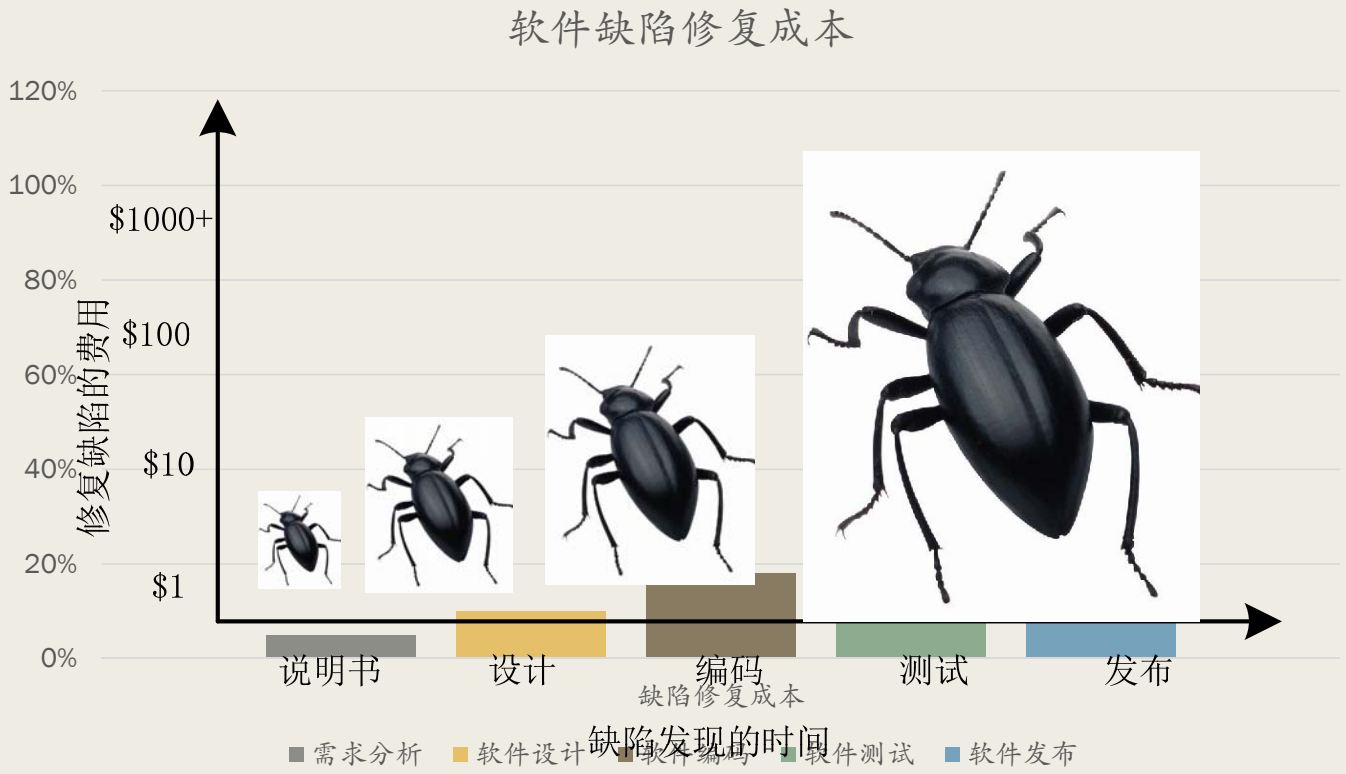
软件缺陷在不同阶段的发布

- 在真正的程序测试之前，通过审查、评审会可以发现更多的缺陷
- 规格说明书的缺陷会在需求分析审查、设计、编码、测试等过程中会逐步发现，而不能在需求分析一个阶段发现



软件缺陷修复成本

- 从需求、设计、编码、测试一直到交付用户公开使用后的过程中，都有可能产生和发现缺陷。
- 随着整个开发过程的时间推移，更正缺陷或修复问题的费用呈几何级数增长



不同阶段的缺陷

- 需求阶段的Bug：这个阶段的BUG是最难发现、最难修复的，而且值得注意的是需求阶段的BUG如果没有及时发现等到实现阶段发现时，那么修复它的费用要比当初修复它要高15~75倍。
  - *主要的原因如下：*
    - 模糊、不清晰的需求；
    - 被忽略的需求；
    - 相互冲突的需求
- 分析设计阶段的BUG：设计中的BUG比需求阶段产生的BUG特征明显易于捕获，但是其维修代价很高，原因是设计BUG已经作为一个整体影响着整个系统的实现。
  - *主要的原因如下：*
    - 忽略设计；
    - 混乱的设计；
    - 模糊的设计

不同阶段的缺陷（续）

- 实现阶段的BUG：是软件系统中最普通、最一般的“常规BUG”。可以将实现阶段出现的BUG分为下面几类：
  - *消息错误*
  - *用户界面错误*
  - *遗漏的功能*
  - *内存溢出或者程序崩溃*
  - *其他实现错误*
- 配置阶段的BUG：比较典型的原因是旧的代码覆盖了新的代码，或者测试服务器上的代码和实现人员本机最新代码版本不一致，也可能是实现人员操作配置管理工具不正确引起的；还可能体现了测试人员或者最终用户操作不正确
- 短视将来的BUG：“千年虫”问题就是当初的设计人员为了节省一点硬件成本给全球造成了难以估量的损失。
- 静态文档的BUG：档BUG的定义很简单，即说明模糊、描述不完整和过期的都属于文档BUG。

缺陷严重性划分原则

- 表示软件缺陷所造成的危害的恶劣程度
- 严重级
  - *严重：系统崩溃、数据丢失、数据损坏*
  - *较严重：操作性错误、错误结果、遗漏功能*
  - *一般：小问题、错别字、UI布局、罕见故障*
  - *建议：不影响使用的瑕疵或更好的实现*

缺陷优先级划分原则

- 优先级表示修复缺陷的重要程度与次序
- 优先级
  - *最高优先级：立即修复，停止进一步测试*
  - *次高优先级：在产品发布之前必须修复*
  - *中等优先级：如果时间允许应该修复*
  - *最低等优先级：可能会修复，不修复也能发布*

缺陷等级划分

等级	描述	说明
5-紧急	发现可重复出现的致命问题	——导致系统崩溃； ——导致程序模块丢失； ——主业务流程出现断点； ——内存泄漏； ——导致死机
4-非常高	发现可重复出现的严重问题	——被测功能不能正确实现； ——软件错误导致数据丢失； ——被测数据处理错误； ——用户需求未实现。
3-高	一般性的错误或功能实现有不完美处	——操作界面错误； ——打印内容、格式错误； ——简单的输入限制未放在前台进行控制； ——删除操作未给出提示。
2-中	细小的错误	——界面不规范； ——辅助说明描述不清楚； ——输入输出不规范； ——长操作未给用户提示； ——提示窗口文字未采用行业术语。
1-低	建议类错误	需求说明书、用户手册中未说明，但影响用户对软件使用的方便性等

软件缺陷的种类

- 文档缺陷：文档在静态检查过程中发现的缺陷。通过测试需求文档、文档审查对被分析或被审查的文档发现的缺陷
- 代码缺陷：对代码进行同行评审、审计或代码走查过程中发现的缺陷
- 测试缺陷：由测试执行活动发现的被测对象（可执行的代码、系统，不包括静态测试发现的问题）的缺陷。
  - *测试活动：内部测试、连接测试、系统集成测试、用户验收测试、测试活动中发现的缺陷为测试缺陷*
- 过程缺陷：通过过程审计、过程分析、管理评审、质量评估、质量审核等活动发现的关于过程的缺陷和问题。
  - *过程缺陷的发现者一般是质量经理、测试经理、管理人员*

软件缺陷数目估计

- 撒播模型
  - *乒乓球法（参入黑球）*
  - *撒播模型是利用概率论的方法来估算程序中的错误数目，其基本原理类似于估计一个大箱子中存放的乒乓球的数目。*
  - *假设一个大箱子里有许多白色的乒乓球N，向箱子里置入M个黑色的乒乓球，并将箱中的球搅拌均匀，然后，从箱子中随机的取出足够多的球，假设白色的乒乓球有n个，黑色的乒乓球m个，则可以根据下列公式估计N*

$$\frac{N}{N+M}=\frac{n}{n+m}$$

$$N=\frac{n}{m}*M$$

软件缺陷数目估计

- 撒播模型
  - *Mills缺陷预测（植入缺陷）*
  - *用人工随机的向待估算的软件置入错误；然后进行测试，待测试到足够长的时间后，对所测试到的错误进行分类，看看哪个是人工置入的错误，哪个是程序中固有的错误；最后，根据上述公式即可估算出程序中所有的错误。*
  - *缺点*
    - 程序中固有的缺陷是未知的，每个错误被检测的难易程度也同样是未知的。
    - 人工置入的缺陷是否和程序中存在缺陷检测的难易程度一致也是未知的。

软件缺陷数目估计

- 撒播模型
  - *Hyman缺陷预测（相同缺陷）*
  - *假设软件总的排错时间是X个月，即经过X个月的排错时间，假设程序中将不再存在错误。*
  - *让两个人共同对程序进行排错，假设经过足够长(X的一半或更少)的排错时间后，第一个人发现了n个错误，第二个人发现了m个错误，其中属于两个人共同发现的错误有m_1个，则公式为：*

$$\frac{N}{n} = \frac{m}{m_1}$$

$$N = \frac{m}{m_1} * n$$

软件缺陷数目估计

- 静态模型
  - *Akiyama模型*
  - *N=4.86+0.018*L*

    其中：N是缺陷数；L是可执行的源语句数目。
  - *基本的估计是1KLOC有23个缺陷。这是早期的研究成果。该模型可能对某个人或某类专门的程序是有效的，模型的提出明显是一种实践上的统计结果。该模型太简单了，实际价值不大。*

软件缺陷数目估计

- 静态模型
  - *谓词模型*
  - *N=C+J*

    其中：C是谓词数目，J是子程序数目。
  - *程序的许多错误来自于程序中包含的谓词。但将每个谓词和子程序都假定一个错误也显然是不合适的，但也没有其他更好的办法来准确地描述它们之间的关系。总之，该模型也有一定的参考价值。*

软件缺陷数目估计

- 静态模型
  - *Halstead模型*
  - *N=V/3000*

    *其中：V=x*lny，x=x1+x2，y=y1+y2*

    x1：程序中使用操作符的总次数；

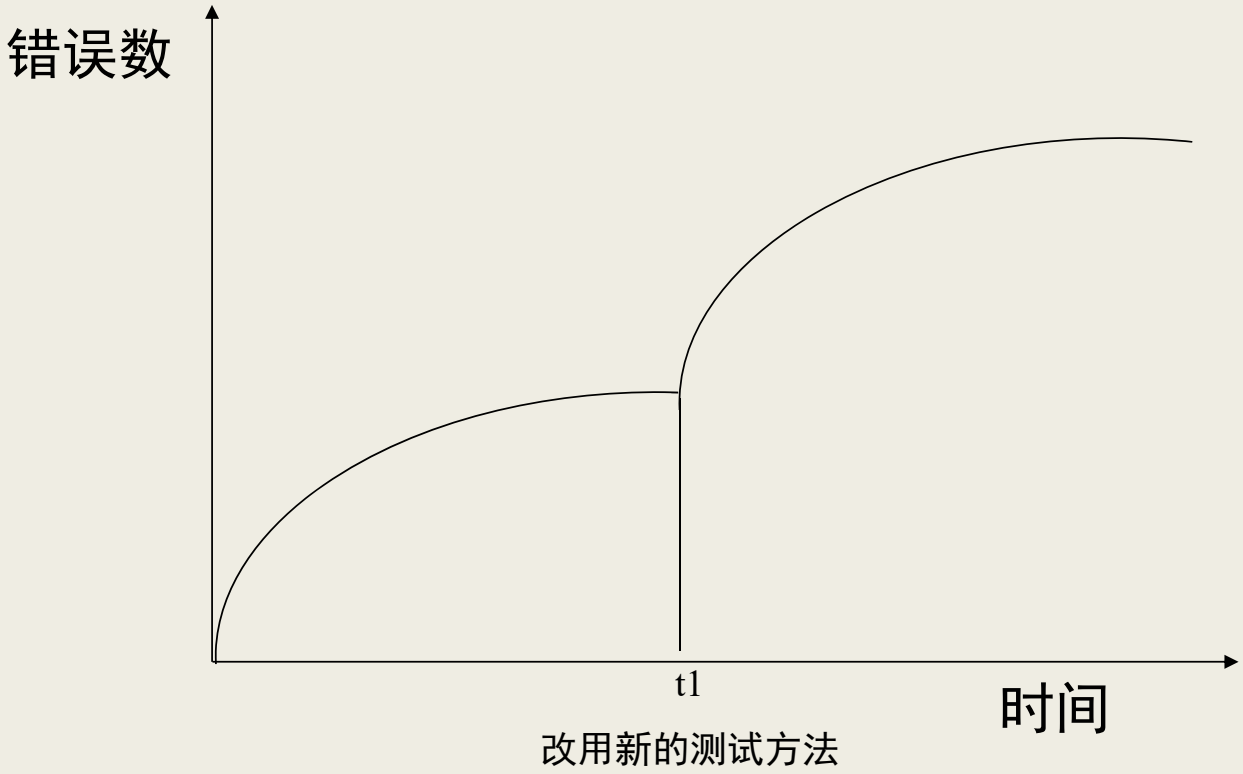
    x2：程序中使用操作数的总次数；

    y1：程序中使用操作符的种类；

    y2：程序中使用操作数的种类；
  - *根据Halstead的理论，V是程序的体积，即程序占内存的比特数目，该模型认为，平均3000bit就有一个错误。该模型和Akiyama模型有些类似，也完全是大量程序的统计结果，但难以说清楚哪一个更好。*

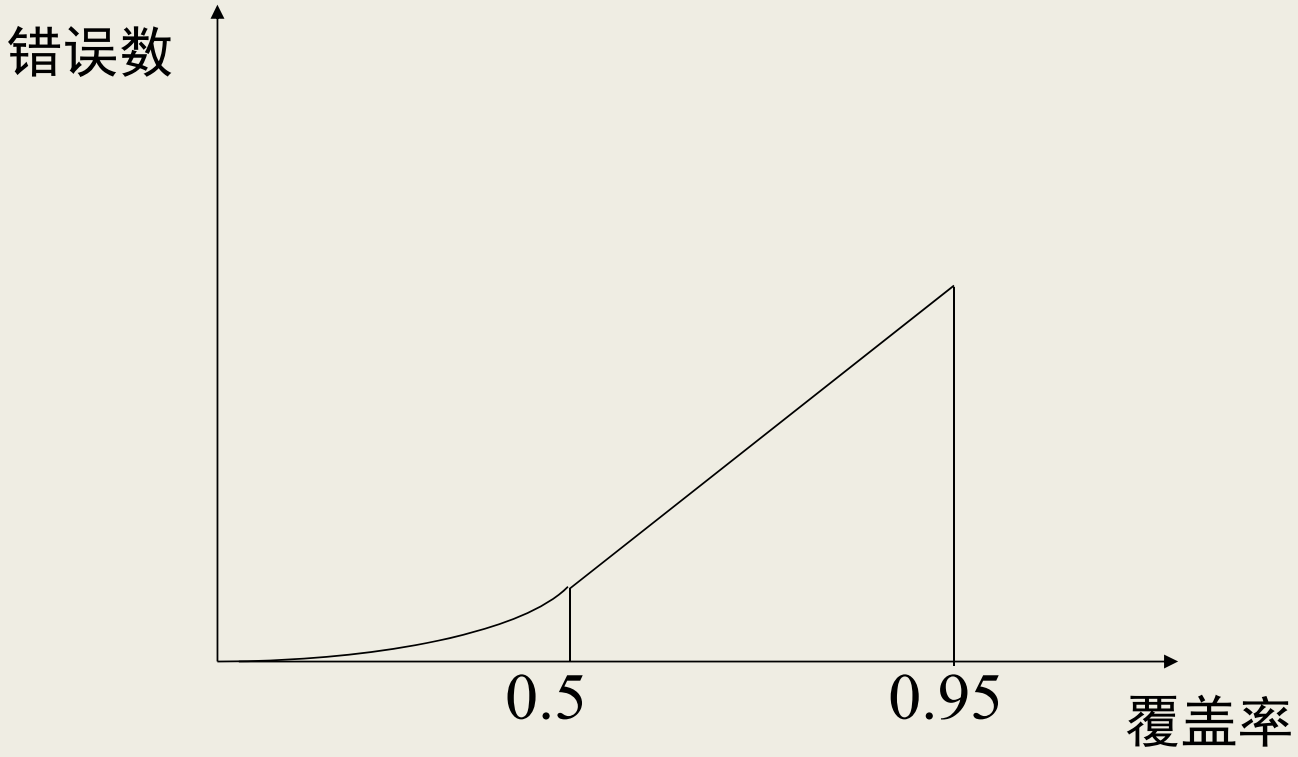
软件缺陷数目估计

- 覆盖率预测模型
  - 错误与时间曲线



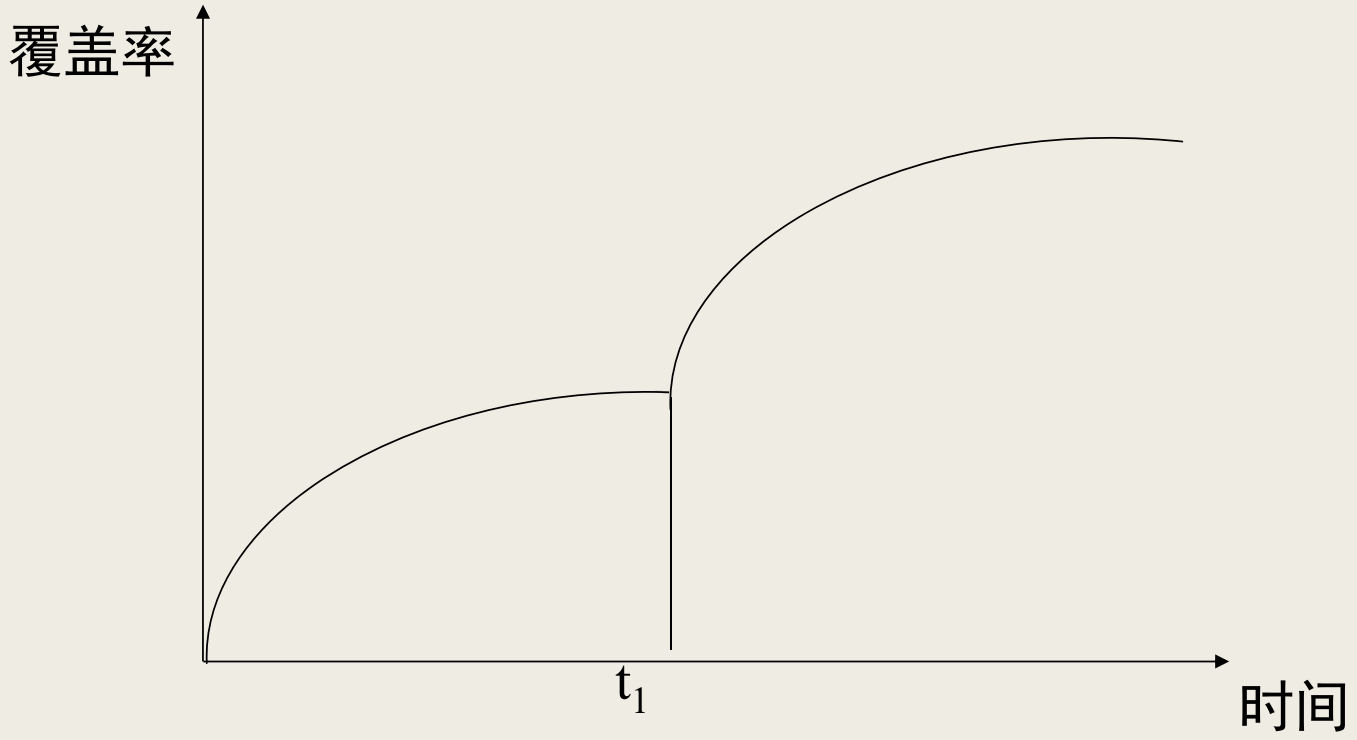
软件缺陷数目估计

- 覆盖率预测模型
  - 错误与覆盖率曲线



软件缺陷数目估计

- 覆盖率预测模型
  - *覆盖率与时间曲线*



软件缺陷管理

- *缺陷管理的目标*
  - *确保每个被发现的缺陷都能够被解决*
    - 这里解决的意思不一定是被修正，也可能是其他处理方式(例如，在下一个版本中修正或不修正)。
  - *收集缺陷数据并根据缺陷趋势曲线识别测试过程的阶段*
    - 决定测试过程是否结束有很多种方式，通过缺陷趋势曲线来确定测试过程是否结束是常用并且较为有效的一种方式。
  - *收集缺陷数据并在其上进行数据分析，作为组织的过程财富*
    - 在对软件缺陷进行管理时，必须先对软件缺陷数据进行收集，然后才能了解这些缺陷，并且找出预防和修复它们的方法，以及预防引入新的缺陷。

软件缺陷管理

- 作为一个缺陷跟踪管理系统，需要正确的记录错误信息和错误处理信息的全部内容

Bug记录信息

- ✓ 测试软件名称
- ✓ 测试版本号
- ✓ 测试人名称
- ✓ 测试用例
- ✓ 标题
- ✓ 测试软件和硬件配置环境
- ✓ 发现软件错误的类型
- ✓ 错误严重等级
- ✓ 详细步骤
- ✓ 必要的附图
- ✓ 发生错误的模块…

Bug处理信息

- ✓处理者姓名
- ✓处理时间
- ✓处理步骤
- ✓缺陷记录的当前状态

软件缺陷状态

- 软件缺陷的主要状态包括以下的内容
  - *新建(New)：　测试中新报告的软件缺陷；*
  - *打开(Open)：　被确认并分配给相关开发人员处理；*
  - *修正(Fixed)：　开发人员已完成修正，等待测试人员验证；*
  - *拒绝(Declined)：　拒绝修改缺陷；*
  - *延期(Deferred)：　不在当前版本修复的错误，下一版修复*
  - *关闭(Closed)：　错误已被修复。*
- 测试人员提交新发现的缺陷入库，缺陷状态为“New”
- 高级测试人员验证错误
  - *如果确认是错误，则分配给相应的开发人员，设置状态为“Open”*
  - *如果不是错误，则拒绝，设置为“Declined”状态*
- 开发人员查询状态为“Open”的缺陷，对其进行处理
  - *如果不是错误，则状态置为“Declined”*
  - *如果是错误，则修复并置状态为“Fix”*
  - *如果不能解决，要留下文字说明并保持缺陷状态仍为“Open”*
  - *对于不能解决或者延期解决的缺陷，不能由开发人员自己决定，一般要通过某种会议（评审会）才能认可*
- 测试人员查询状态为“Fix”的缺陷，验证缺陷是否已解决，做如下处理
  - *如果问题解决了，置缺陷的状态为“Closed”*
  - *如果问题没有结果，则置状态为“Reopen”*

缺陷报告

- 一个完整的缺陷报告需要包含以下的信息

ID	唯一的识别缺陷的序号
标题	对缺陷的概括性描述，方便快速浏览、管
前提	在进行实际执行的操作之前所具备的条件
环境	缺陷发现时所处的测试环境，包括操作系
操作步骤	导致缺陷产生的操作顺序的描述
期望结果	按照客户需求或设计目标事先定义的操作设计规格说明书等保持一致
实际结果	按照操作步骤而实际发生的结果。实际结异
频率	同样的操作步骤导致实际结果发生的概率
严重程度	指因缺陷引起的故障对软件产品使用或某户的角度出发，由测试人员决定。一般小
优先级	缺陷被修复的紧急程度或先后次序，主要影响，但要考虑开发过程的需求(对测试组(产品经理、测试/开发组长)决定。一正常排队、低优先级
类型	属于哪方面的缺陷，如功能、用户界面、
缺陷提交人	缺陷提交人的名字(会和邮件地址联系起

缺陷指定解决人	估计修复这个缺陷的开发人员，在缺陷状态下由开发组长指定相关的开发人员；自动和该开发人员的邮件地址联系起来。当缺陷被报出来时，系统会自动发出邮件
来源	缺陷产生的地方，如产品需求定义书、设计规格说明书、代码的具体组件或模块、数据库、在线帮助、用户手册等
产生原因	产生缺陷的根本原因，包括过程、方法、工具、算法错误、沟通问题等，以寻求流程改进、完善编程规范和加强培训等，有助于缺陷预防
构建包跟踪	用于每日构建软件包跟踪，是新发现的缺陷，还是回归缺陷，基准(baseline)是上一个软件包
版本跟踪	用于产品版本质量特性的跟踪，是新发现的缺陷，还是回归缺陷，基准是上一个版本
提交时间	缺陷报告提交的时间
修正时间	开发人员修正缺陷的时间
验证时间	测试人员验证缺陷并关闭这个缺陷的时间
所属项目/模块	缺陷所属哪个具体的项目或模块，要求精确定位至模块、组件级
产品信息	属于哪个产品、哪个版本等
状态	当前缺陷所处的状态，见2.2.3

缺陷书写规范

- 标题：应保持简短、准确，提供缺陷的本质信息
  - *尽量按缺陷发生的原因与结果的方式书写；*
  - *避免使用模糊不清的词语，例如：“功能中断，功能不正确，行为不起作用”等。应该使用具体文字说明缺陷的症状；*
  - *为了便于他人理解，避免使用术语、俚语或过分具体的测试细节。*
- 复现步骤：应包含如何使别人能够很容易的复现该缺陷的完整步骤。为了达到这个要求，复现步骤的信息必须是完整的、准确的、简明的、可复现的。

  常见问题：
  - *包含了过多的多余步骤，且句子结构混乱，可读性差，难以理解；*
  - *包含的信息过少，丢失了操作的必要步骤；*
  - *没有对软件缺陷发生的条件和影响区域进行隔离。*

缺陷书写规范（续）

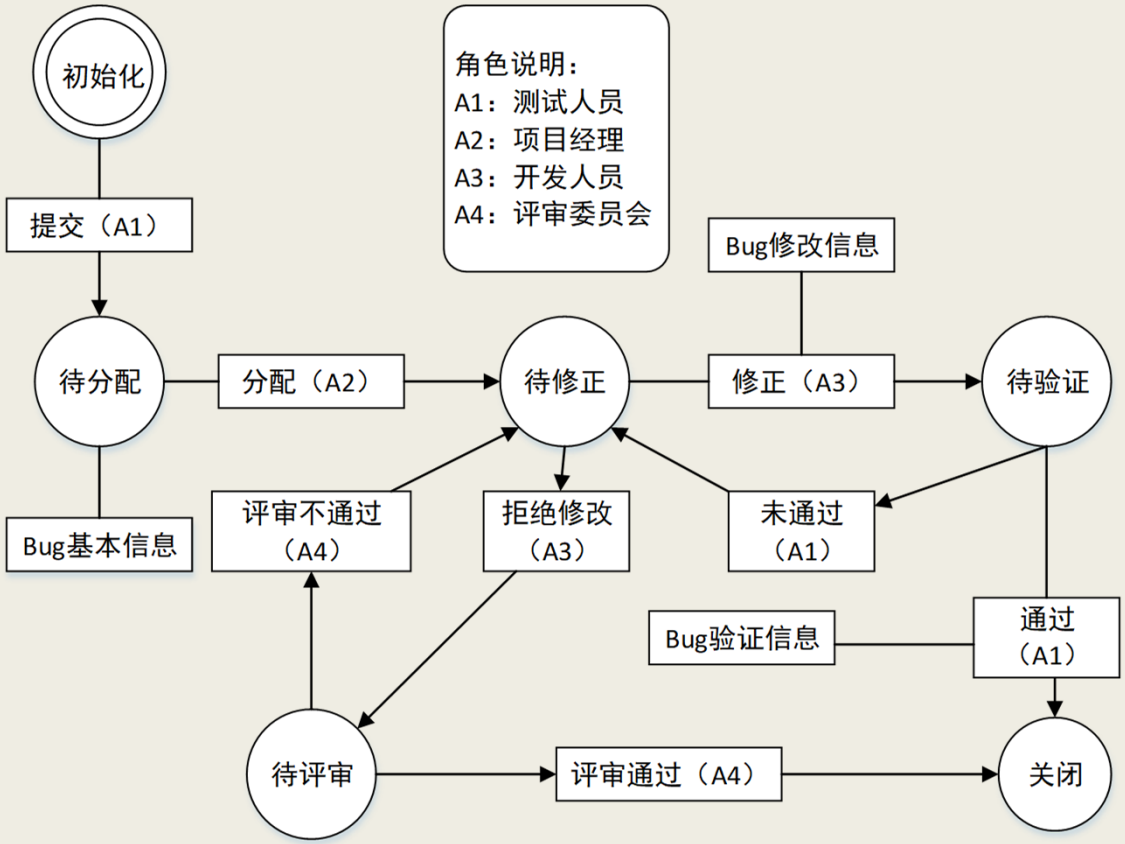
- 复现步骤的正确书写方式：
  - *提供测试的环境信息；*
  - *简单地一步步引导复现该缺陷，一个步骤包含的操作不要多；*
  - *每个步骤前使用数字对步骤编号；*
  - *尽量使用短语或短句，避免复杂句型句式；*
  - *复现的步骤要完整、准确、简短；*
  - *将常见步骤合并为较少步骤；*
  - *按实际需要决定是否包含步骤执行后的结果。*
- 实际结果：是执行复现步骤后软件的现象和产生的行为。
  - *实际结果的描述应向标题信息那样，要列出具体的缺陷症状，而不是简单地指出“不正确”或“不起作用”。*
- *期望结果：描述应与实际结果的描述方式相同。通常需要列出期望的结果是什么。*

缺陷书写规范（续II）

- 附件：对缺陷描述的补充说明，可以是以下一些类型：
  - *缺陷症状的截图；*
  - *测试使用的数据文件；*
  - *缺陷交流的记录，例如相关邮件等；*
  - *解决缺陷的补丁程序*
- 其它：
  - *选择合适的缺陷严重性属性；*
  - *按相应的规定，填写相应的字段信息*
- 避免常见的错误：
  - *避免使用我、你等人称代词，可以直接使用动词或必要时使用“用户”代替*
  - *避免使用情绪化的语言和强调符号；*
  - *避免使用诸如“似乎”、“看上去可能”等含义模糊的词汇，而需要报告确定的缺陷结果；*
  - *避免使用自认为比较幽默的语句，只需客观地描述缺陷的信息；*
  - *避免提交不确定的测试问题，自己至少需要重现一次再提交。*

软件缺陷管理

- 软件缺陷管理流程



缺陷管理工具

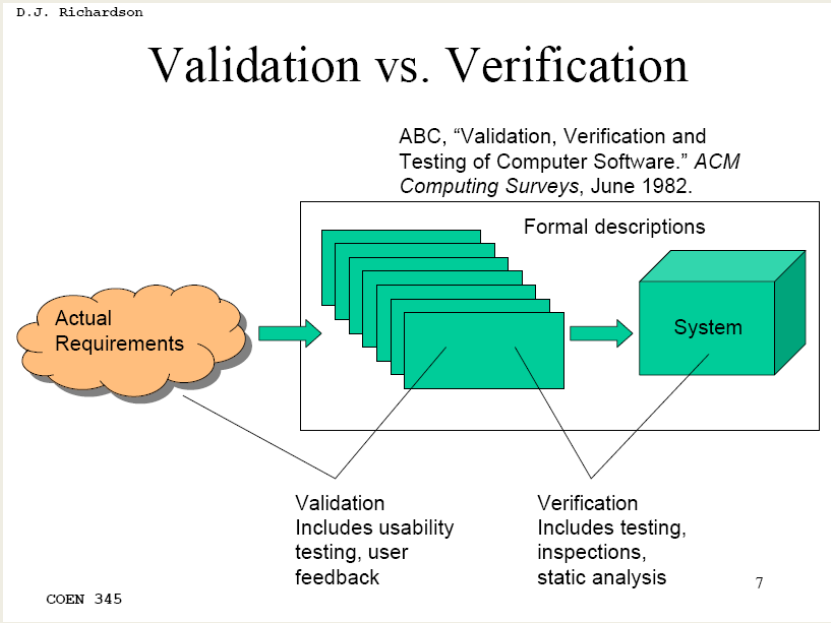
- TrackRecord(商用) http://www.compuware.com/products/trackrecord.htm
  - *一个高级的需求变更和缺陷管理工具，可以帮助组织建立一个系统方法来协调软件开发、调试、测试和实现。TrackRecord支持并加速各种开发过程，并具有针对开发、测试和管理需求而设计的灵活、开放的体系结构。TrackRecord可以与康博软件其它的开发、测试和支持产品以及第三方产品集成，从而进行自动化缺陷跟踪、项目管理和整个企业应用的可靠性保证*
- Rational ClearQuest https://www.ibm.com/products/rational-clearquest
  - Rational ClearQuest 是基于团队的缺陷和变更跟踪解决方案，它包含在Rational Suite中。Rational Suite 是针对分析人员、开发人员和测试人员进行了优化的一套软件开发全面解决方案。作为它主要组件之一的Rational ClearQuest 是一套高度灵活的缺陷和变更跟踪系统，适用于在任何平台上，任何类型的项目中，捕获各种类型的变更

- BugFree （开源） https://www.zentao.net/
  - *BugFree是借鉴微软的研发流程和Bug管理理念，使用PHP+MySQL独立写出的一个Bug管理系统。简单实用、免费并且开放源代码*
  - *已升级为禅道，包括了项目管理和bug管理等。有开源版本*
- Bugzilla（开源） https://www.bugzilla.org/
  - *Bugzilla 是一个“缺陷跟踪系统”或者“bug跟踪系统”，帮助个人或者小组开发者有效的跟踪已经发现的错误. 多达数商业缺陷跟踪软件收取昂贵的授权费用，bugzilla做为一个免费软件，拥有许多商业软件所不具备的特点，因而，现在已经成为全球许多组织喜欢的缺陷管理软件*
- *Mantis https://gitee.com/mirrors/mantisbt*
  - 基于PHP技术的轻量级的开源缺陷跟踪系统，以Web操作的形式提供项目管理及缺陷跟踪服务。在功能上、实用性上足以满足中小型项目的管理及跟踪

软件测试

软件测试的定义

- 软件测试是由“验证Verification”和“有效性确认Validation”活动构成的整体
  - *“验证”是检验软件是否已正确地实现了产品规格书所定义的系统功能和特性*
  - *“有效性确认”是确认所开发的软件是否满足用户真正需求的活动。*

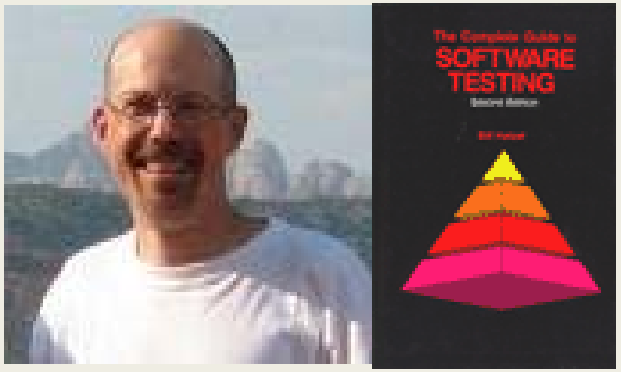


软件测试的对象

- 软件测试不等于程序测试
- *软件测试贯串于软件定义和开发的整个过程。*
- *软件开发过程中所产生的需求规格说明、概要设计规格说明、详细设计规格说明以及源程序都是软件测试的对象。*

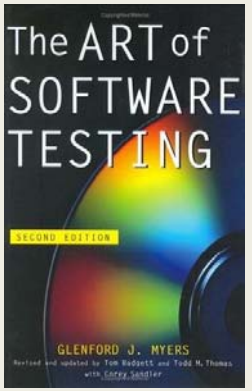
软件测试的正面性

- Bill Hetzel博士（正向思维的代表）：
  - *软件测试就是为程序能够按预期设想那样运行而建立足够的信心。*
  - *“软件测试是一系列活动以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”*
  - *测试是为了验证软件是否符合用户需求，即验证软件产品是否能正常工作*

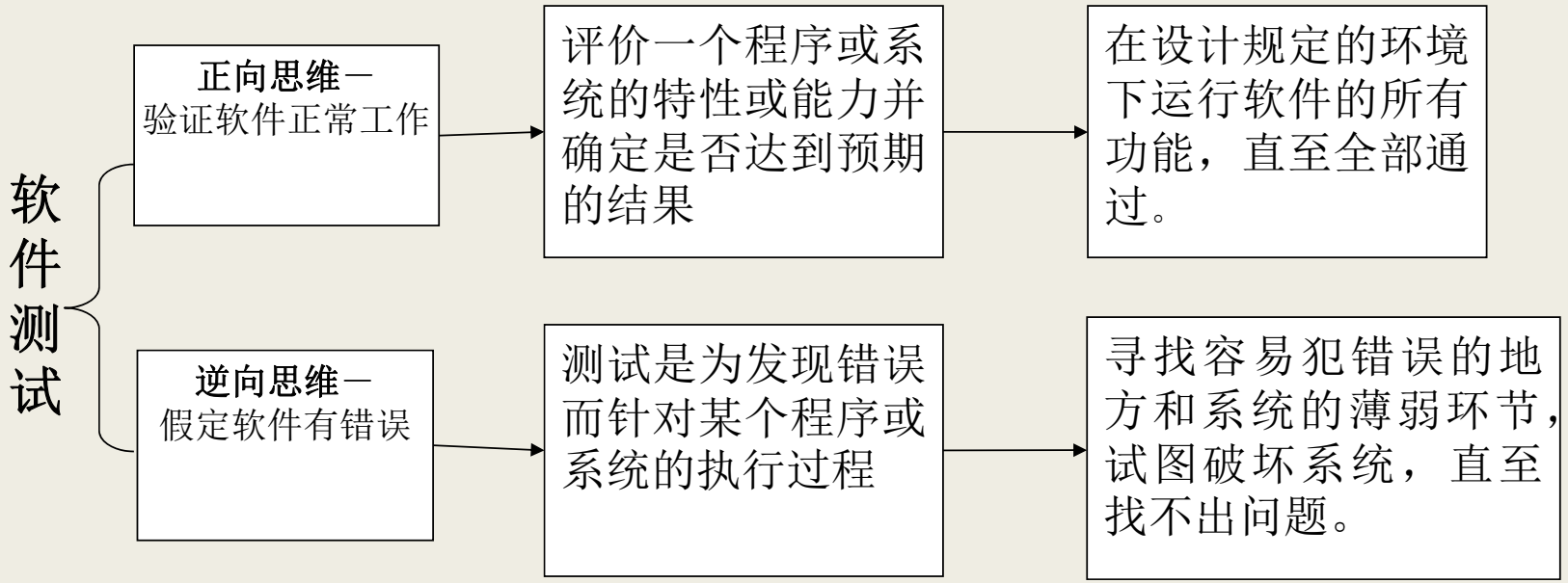


软件测试的反面性

- Glenford J. Myers （反向思维的代表）：
  - *测试是为了证明程序有错，而不是证明程序无错误*
  - *一个好的测试用例是在于它能发现至今未发现的错误*
  - *一个成功的测试是发现了至今未发现的错误的测试*



软件测试的两面性


软件测试
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。
逆向思维－
假定软件有错误
测试是为发现错误而针对某个程序或系统的执行过程
寻找容易犯错误的地方和系统的薄弱环节，试图破坏系统，直至找不出问题。

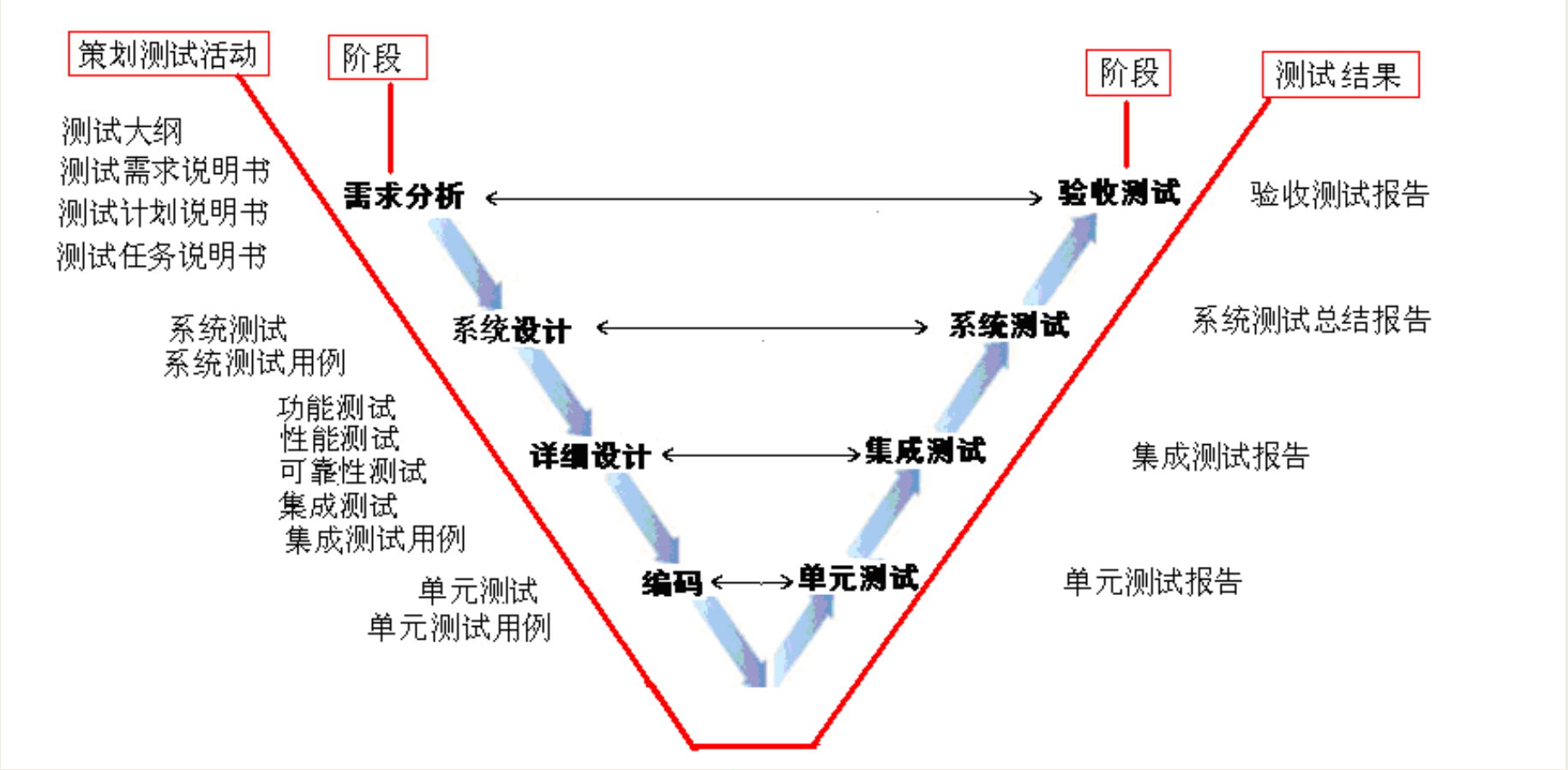
软件测试的目的

- 根据Grenford.J.Myers的观点，软件测试的目的：
  （1）测试程序的执行过程，目的在于发现缺陷；
  （2）一个好的测试用例在于能发现至今尚未发现的缺陷；
  （3）一个成功的测试是发现了至今未发现的多个缺陷的测试；
  测试的目的：不仅仅是为了发现软件缺陷与错误，而且也是对软件质量进行度量与评估，以提高软件质量。
  - *测试是想以最少的时间和人力，系统地找出软件中潜在的各种缺陷，通过修正缺陷提高软件质量，回避软件发布后由于潜在缺陷造成的隐患所带来的商业风险；*
  - *测试的附带收获是，它能够证明软件的功能和性能是否与需求说明书相符合；*
  - *实施测试收集到的测试结果数据为可靠性分析提供了依据；*
  - *测试不能表明软件中不存在错误，它只能说明软件中存在错误。*

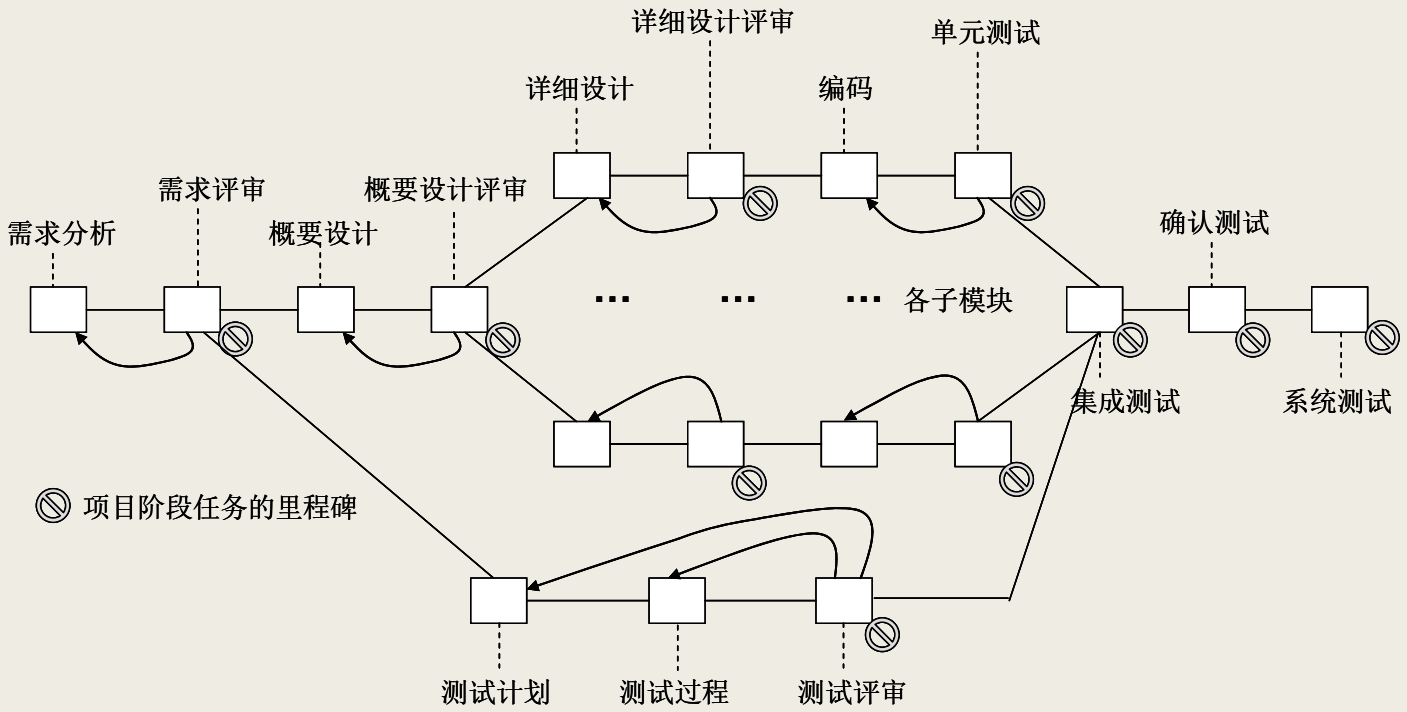
软件测试的原则

- 尽早地和及时地测试；
- 测试用例应当由测试数据和与之对应的预期结果这两部分组成；
- 在程序提交测试后，应当由专门的测试人员进行测试；
- 测试用例应包括合理的输入条件和不合理的输入条件
- 严格执行测试计划，排除测试的随意性；
- 充分注意测试当中的群体现象；
- 应对每一个测试结果做全面的检查；
- 保存测试计划、测试用例、出错统计和最终分析报告，为维护工作提供充分的资料。

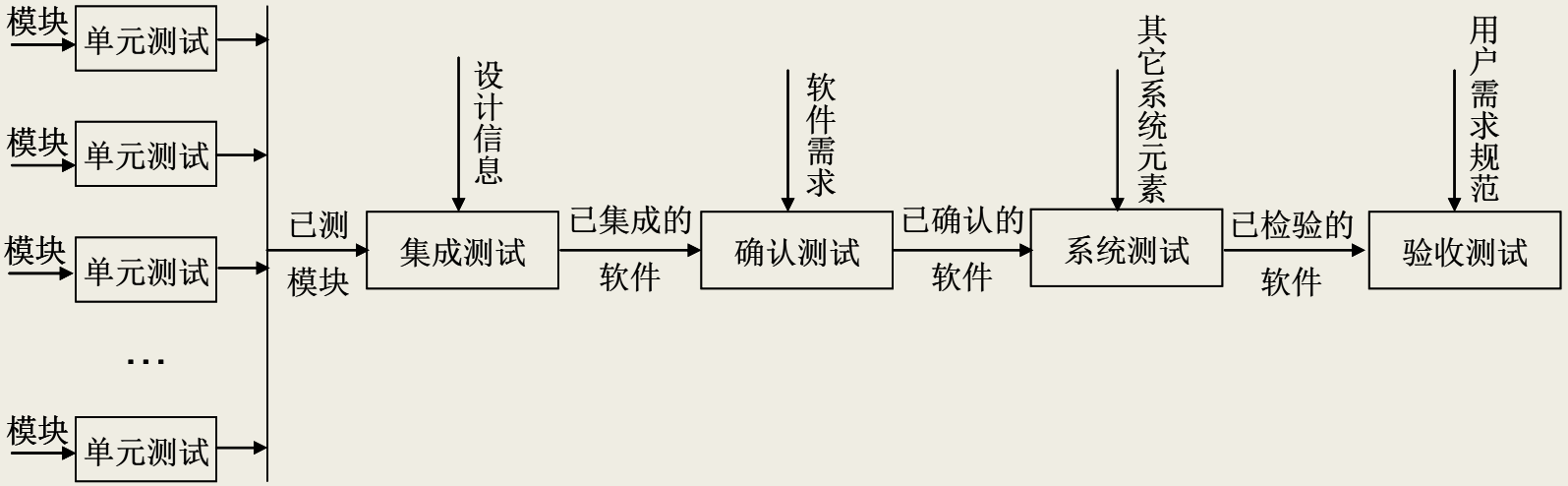
软件测试和软件开发的关系



软件测试和软件开发的关系

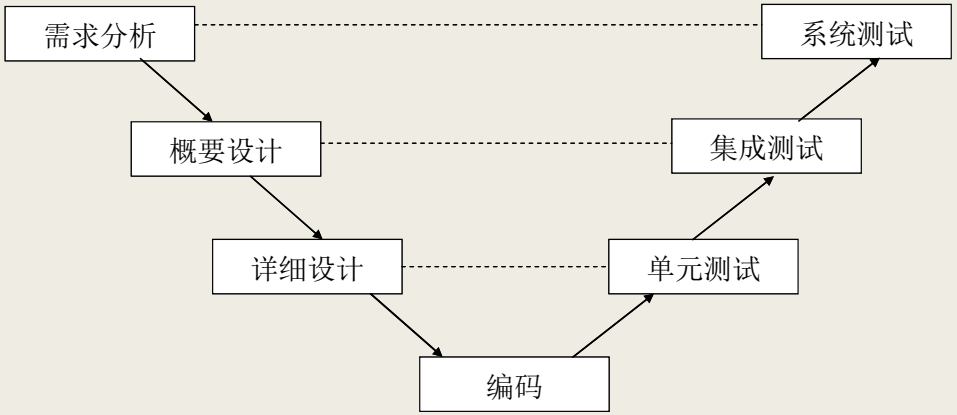


软件测试过程


模块
单元测试
模块
单元测试
模块
单元测试
…
模块
单元测试
已测
模块
集成测试
设计信息
已集成的
软件
确认测试
软件需求
已确认的
软件
系统测试
其它系统元素
已检验的
软件
验收测试
用户需求规范

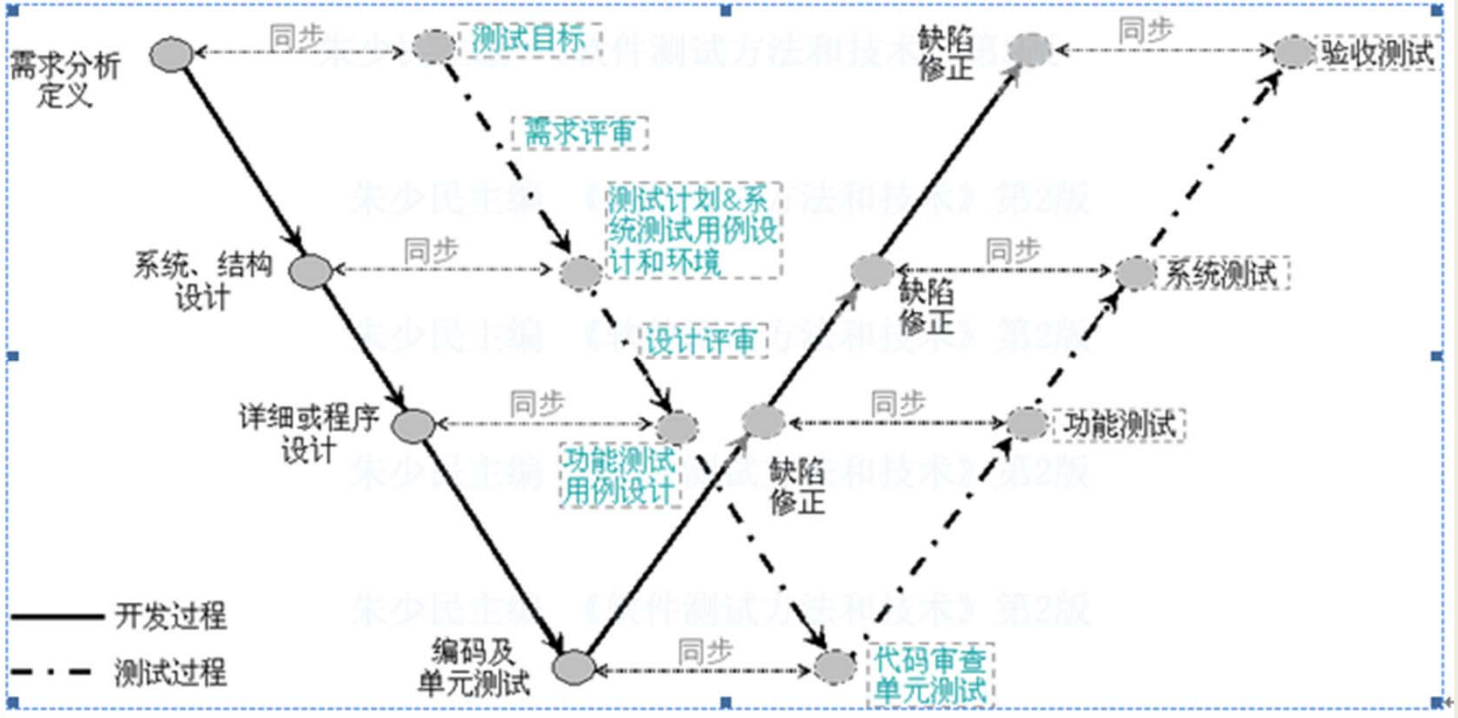
软件测试的过程模型

- V模型
  - *特点：明确地表明了测试的不同级别，清晰地展示了软件测试与开发之间的关系*



软件测试的过程模型

- W模型
  - *强调：测试伴随着整个开发周期，且测试的对象不仅仅是程序，需求、功能和设计同样要测试*

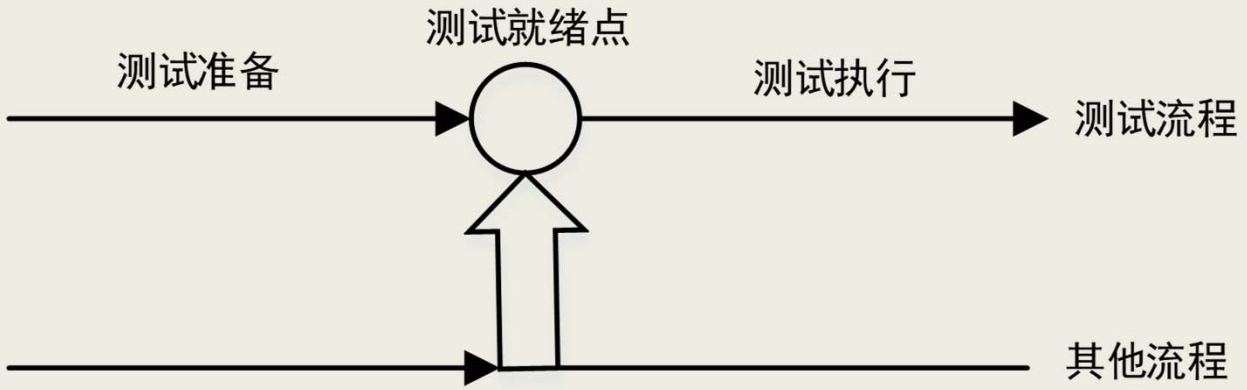


V模型和W模型的问题

- 它们都把软件的开发视为需求、设计、编码等一系列的串行活动。事实上，虽然这些活动之间存在互相牵制的关系，但在大多数时间里，它们互相独立，可并发进行。相应的测试也不存在严格的先后次序，只要条件满足，就可以进行测试
- V模型和W模型都没能很好地表示测试流程的完整性
- 测试流程大致分为两类
  - *测试准备活动：包括测试需求分析、测试计划、测试分析、测试编码、测试验证等*
  - *测试执行活动：包括测试运行、测试报告、测试分析等*

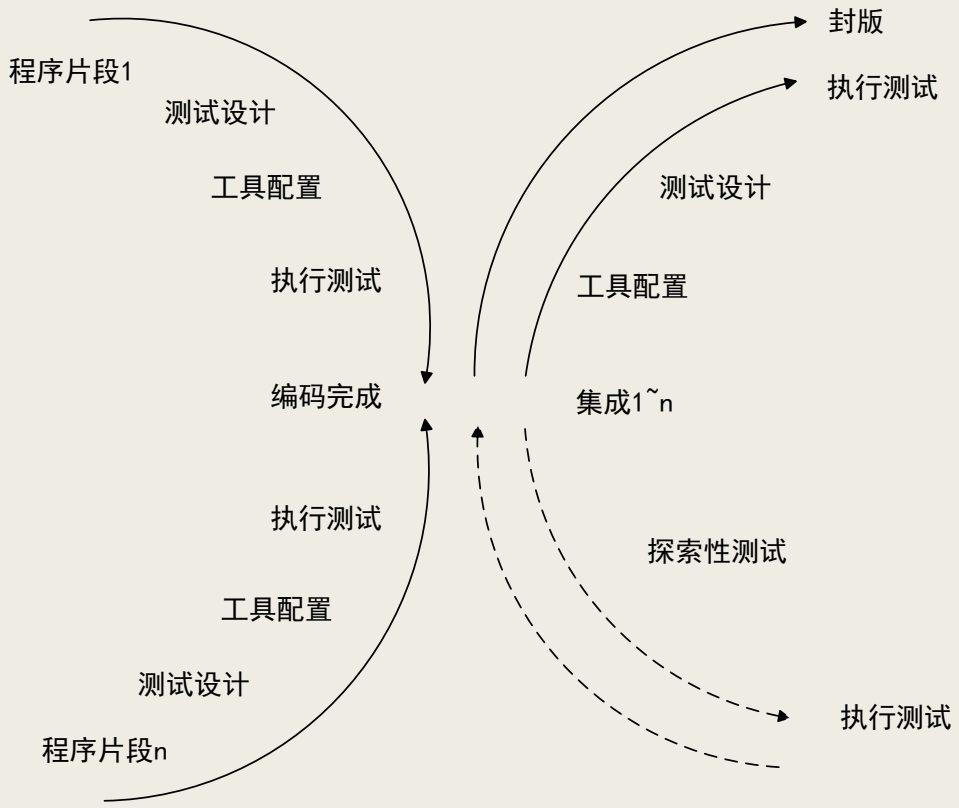
软件测试的过程模型

- H模型

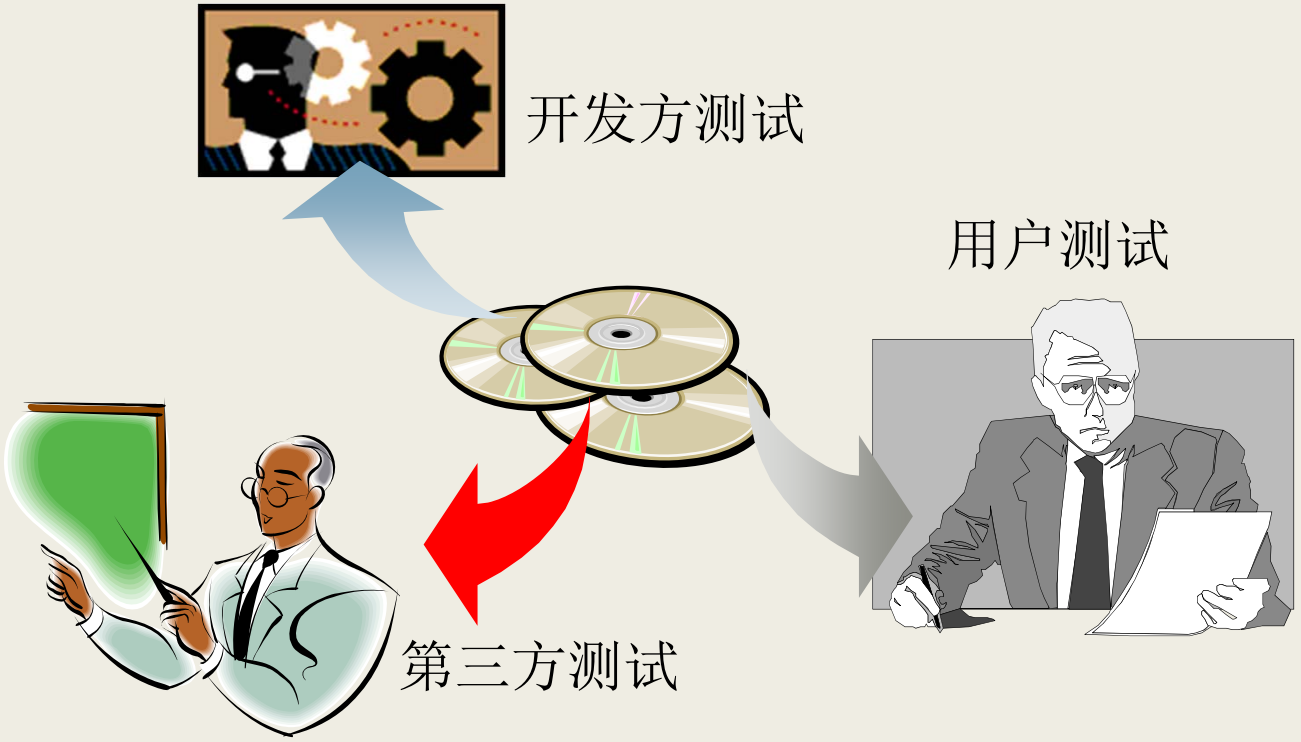


软件测试的过程模型

- X模型



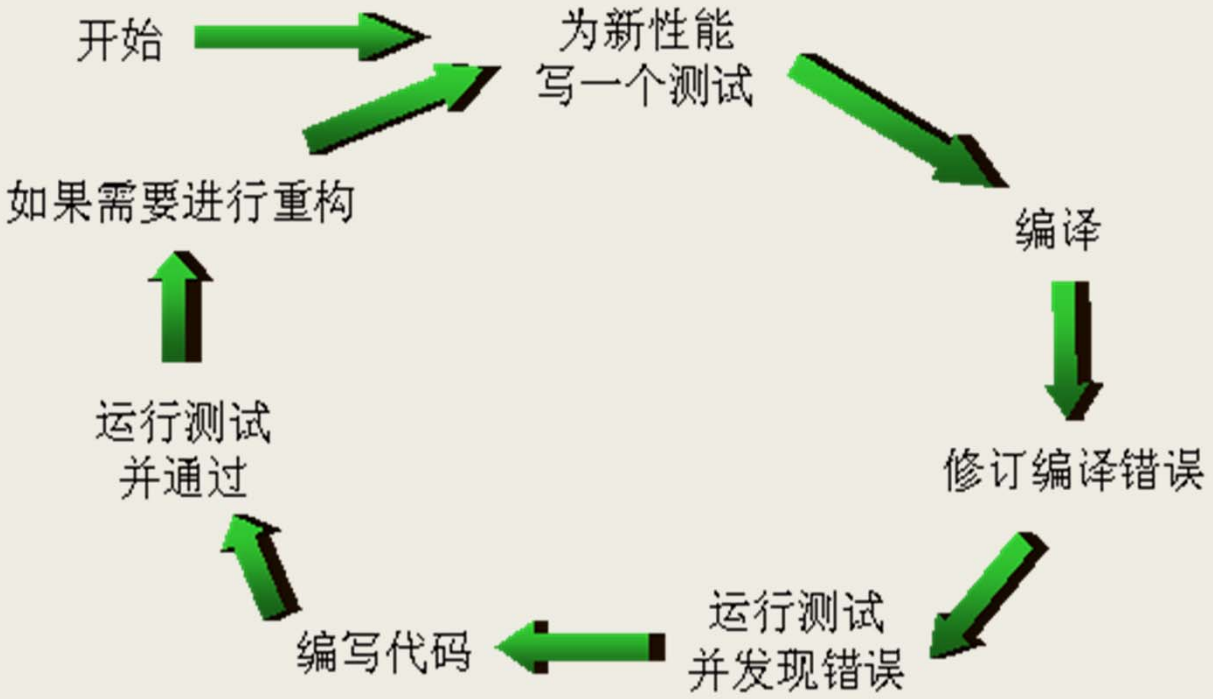
测试的实施组织



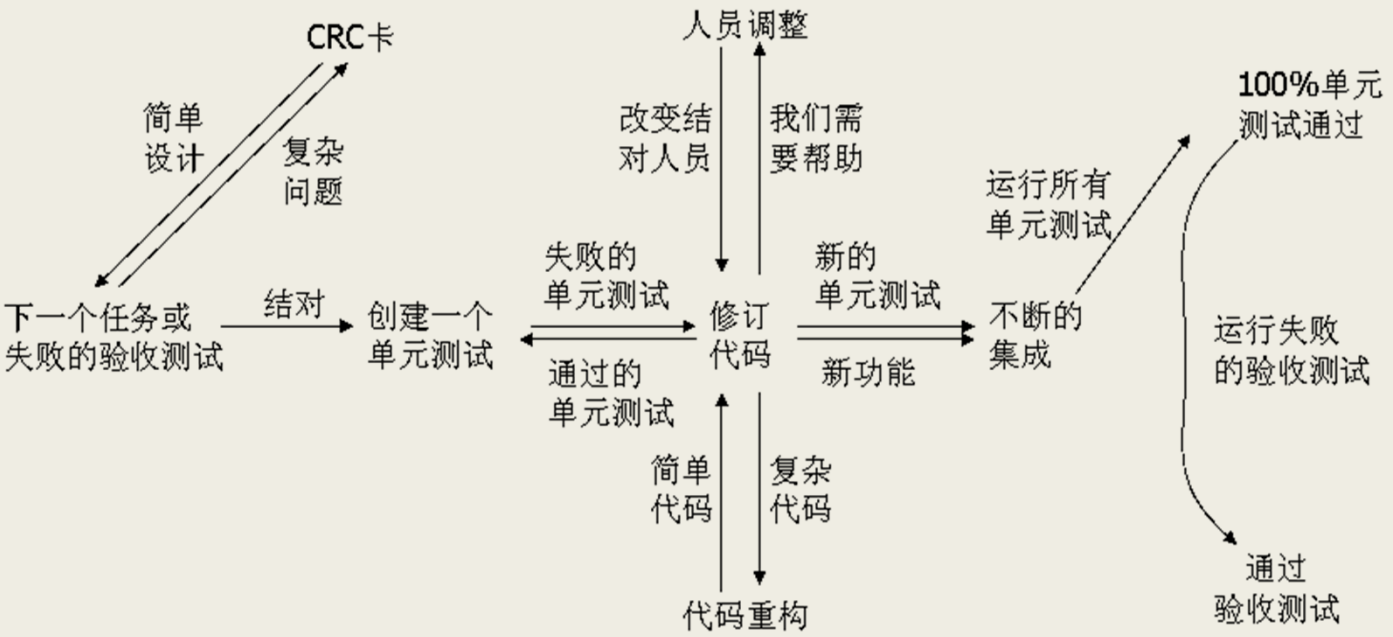
测试停止的依据

- 第一类标准：测试超过了预定时间，则停止测试。
- 第二类标准：执行了所有的测试用例，但并没有发现故障，则停止测试。
- 第三类标准：使用特定的测试方案作为判断测试停止的基础。
- 第四类标准：正面指出停止测试的具体要求，即停止测试的标准可定义为查出某一预订数目的故障。
- 第五类标准：根据单位时间内查出故障的数量决定是否停止测试。

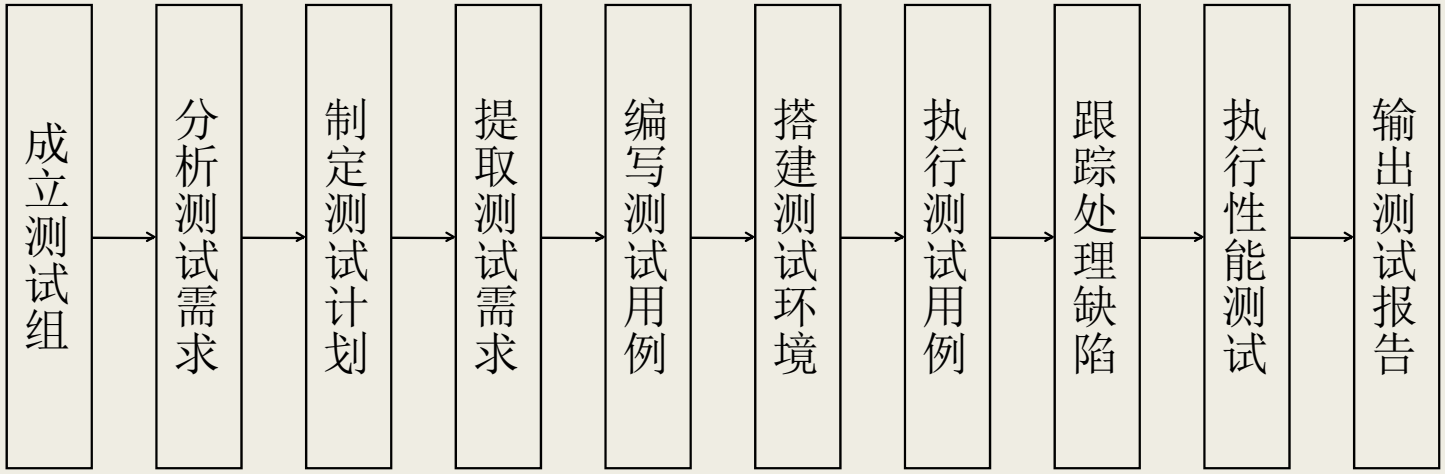
测试驱动开发TDD的思想



测试驱动开发TDD的实践



软件测试流程


成立测试组
分析测试需求
制定测试计划
提取测试需求
编写测试用例
搭建测试环境
执行测试用例
跟踪处理缺陷
执行性能测试
输出测试报告

软件测试流程

- 成立项目组
  - *当需测试的项目分配下来后，该项目的负责人向测试部门提出测试申请，通过测试经理的审批后，由测试经理指派测试组长与测试工程师，成立项目测试组，负责该项目的测试工作*
- 测试组的任务
  - *发现软件程序、系统或产品中所有的问题;*
  - *尽早地发现问题;*
  - *督促开发人员尽快地解决程序中的缺陷;*
  - *帮助项目管理人员制定合理的开发计划;*
  - *并对问题进行分析、分类总结和跟踪*
  - *帮助改善开发流程、提高产品开发效率;*
  - *提高程序编写的规范性、易读性、可维护性等*

软件测试流程

- 分析测试需求
  - *测试经理任命测试组长，测试组长需提前熟悉被测试对象的需求，从总体上掌握项目的进展情况。通过仔细的阅读项目的相关文档（比如项目的进度计划、测试要求等）后，测试组长需安排下一步工作*
- 制定测试计划
  - *测试组长在详细了解项目信息后，根据项目需求、项目进度计划表制定当前项目的测试计划，并以此测试计划来指导测试组开展对应的测试工作。测试计划中需说明每个测试工件输出的时间点、测试资源、测试方法、测试风险规避、测试停测标准等*
- 提取测试需求
  - *测试组长制定好测试计划后，项目组进行评审。评审通过后，测试组即可按照此测试计划开展工作。测试组员根据测试组长的任务分配，进行项目用户需求规格说明书的阅读，开展需求测试工作。需求阅读理解完成后，进行测试需求的提取，也就是列出被测对象需测试的点，这项工作可以利用TestDirector等测试管理工具开展*

软件测试流程

- 编写测试用例
  - *测试需求提取完毕，经过测试组的评审通过后，测试组员可以进行测试用例的设计，这些工作都是在测试计划中规定的时间内完成。比如测试计划中规定“2008-12-20至2008-12-30完成系统测试用例设计与评审”，那么必须在这个时间段内完成被测对象的测试用例设计。测试用例的设计一般使用Word、Excel等样式，也可使用专用工具进行管理。*
- 搭建测试环境
  - *测试用例设计工作完成后，如果项目开发组告知测试组长可以开展测试的时候，测试组长可从配置管理员处提取测试版本，根据开发组提供的被测对象测试环境搭建单进行测试环境的搭建。测试环境搭建需要测试工程师掌握基本的硬件、软件知识*

软件测试流程

- 执行测试用例
  - *测试环境搭建完成后，测试组员将进行测试用例的执行。根据前期设计并评审通过的测试用例，测试组员进行各个功能模块的测试。在执行测试用例的过程中，如果发现遗漏或者不完善的测试用例，需及时做更新，并用文档记录变更历史。用例执行过程中如果发现bug，则需按照部门或者项目组的bug提交规范，利用一些bug管理工具提交bug*
- 跟踪处理缺陷
  - *大多数公司都有自己的Bug管理流程规范，项目组成员需根据这个流程规范开展日常的Bug处理工作。在缺陷处理阶段，大多要经过四次、甚至更多的迭代过程，多次进行回归测试，直到在规定的时间内达到测试计划中所定义的停测标准为止。*
  - *在这个阶段，主要使用黑盒测试方法开展工作，以被测对象的需求规格说明为依据，重点关注被测对象的界面与功能表现*

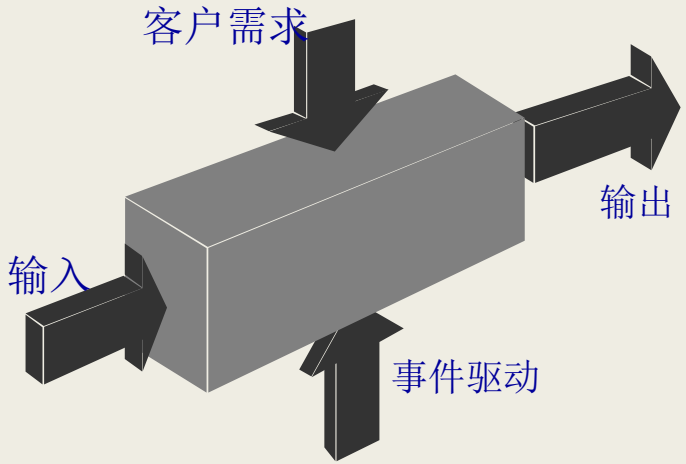
软件测试流程

- 执行性能测试
  - 与功能测试一样，在测试之前，需要进行测试需求的分析、性能指标提取、用例设计、脚本录制、优化、执行、分析等一系列过程。通过使用一些自动化工具进行性能测试是目前性能测试的主要手段，常用的性能测试工具有WAS、QALoad、WebLoad、LoadRunner、Robot等。性能测试阶段主要解决被测对象的性能问题
- 输出测试报告
  - 功能测试、性能测试都完成后，测试组长需要对被测对象做一个全面的总结，以数据为依据，衡量被测对象的质量状况，并提交测试结果报告给项目组从而帮助项目经理、开发组及其他部门了解被测对象的质量情况，以决定下一步的工作计划。
  - 功能测试报告主要包含被测对象的缺陷修复率、Bug状态统计、Bug分布等；性能测试报告主要包含测试指标的达标情况以及测试部的质量评价等。当然，也可以出一份整体的测试报告，包含功能、性能的测试结果

软件测试的分类

- 按测试技术分类
  - *白盒测试技术*
  - *黑盒测试技术*
- 按测试方式分类
  - *静态测试：不执行被测试软件，而对需求规格说明书、软件设计文档、源程序等做结构检查、流程图分析等找出软件错误。可以人工或工具实现*
    - 代码检查
    - 静态结构分析
    - 程序复杂度度量
  - *动态测试：执行被测程序，通过执行结果分析软件可能出现的错误*
    - 功能确认与接口测试
    - 覆盖率分析
    - 性能分析
    - 内存分析

黑盒子和白盒子

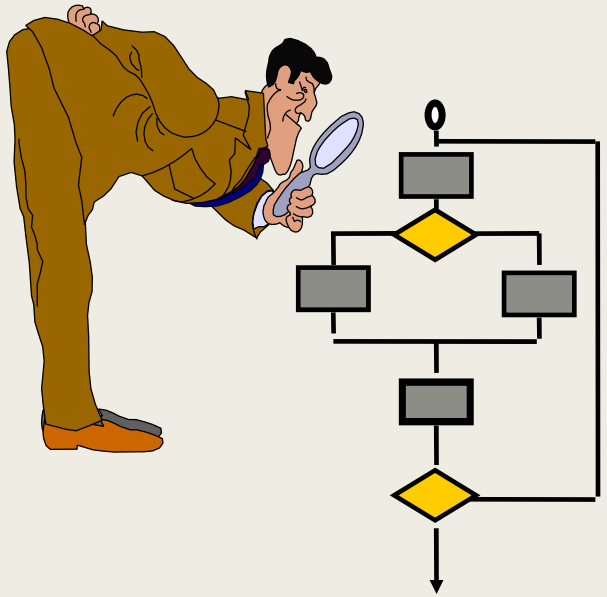


功能测试

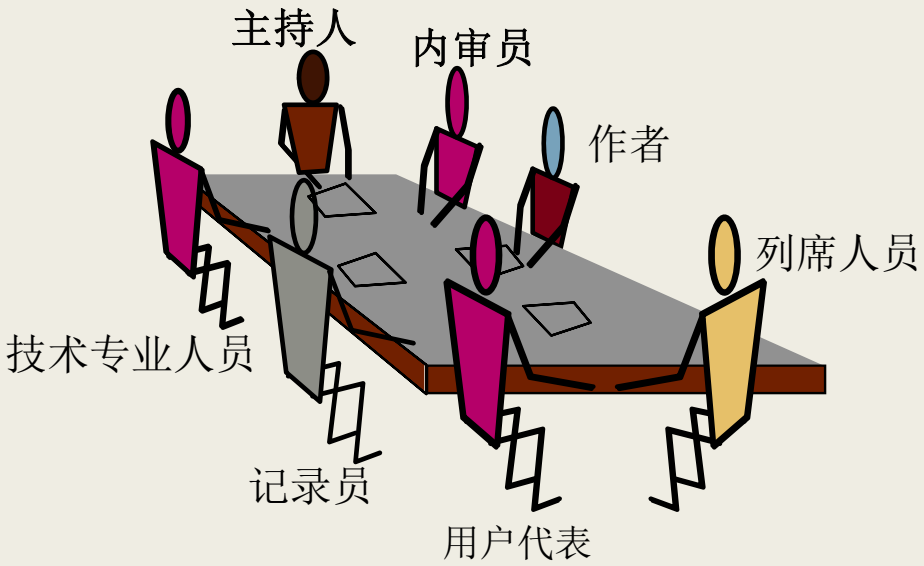
数据驱动测试

结构测试

逻辑驱动测试



静态的和动态的



运行程序

不正式 正式

走读 会议

自动测试和手工测试

手工模拟用户操作

软件测试的分类

- 按测试阶段分类
  - *单元测试*
  - *集成测试*
  - *系统测试*
  - *验收测试*
- 按实施组织分类
  - *开发方测试*
  - *用户方测试*
  - *第三方测试*

软件测试的分类

- 按测试目的分类
  - 功能测试
  - 健壮性测试
  - 接口测试
  - 性能测试
  - 强度测试
  - 压力测试
  - 用户界面测试
  - 安全测试
  - 可靠性测试
  - 安装/反安装测试
  - 文档测试
  - 恢复测试
  - 兼容性测试

测试与调试

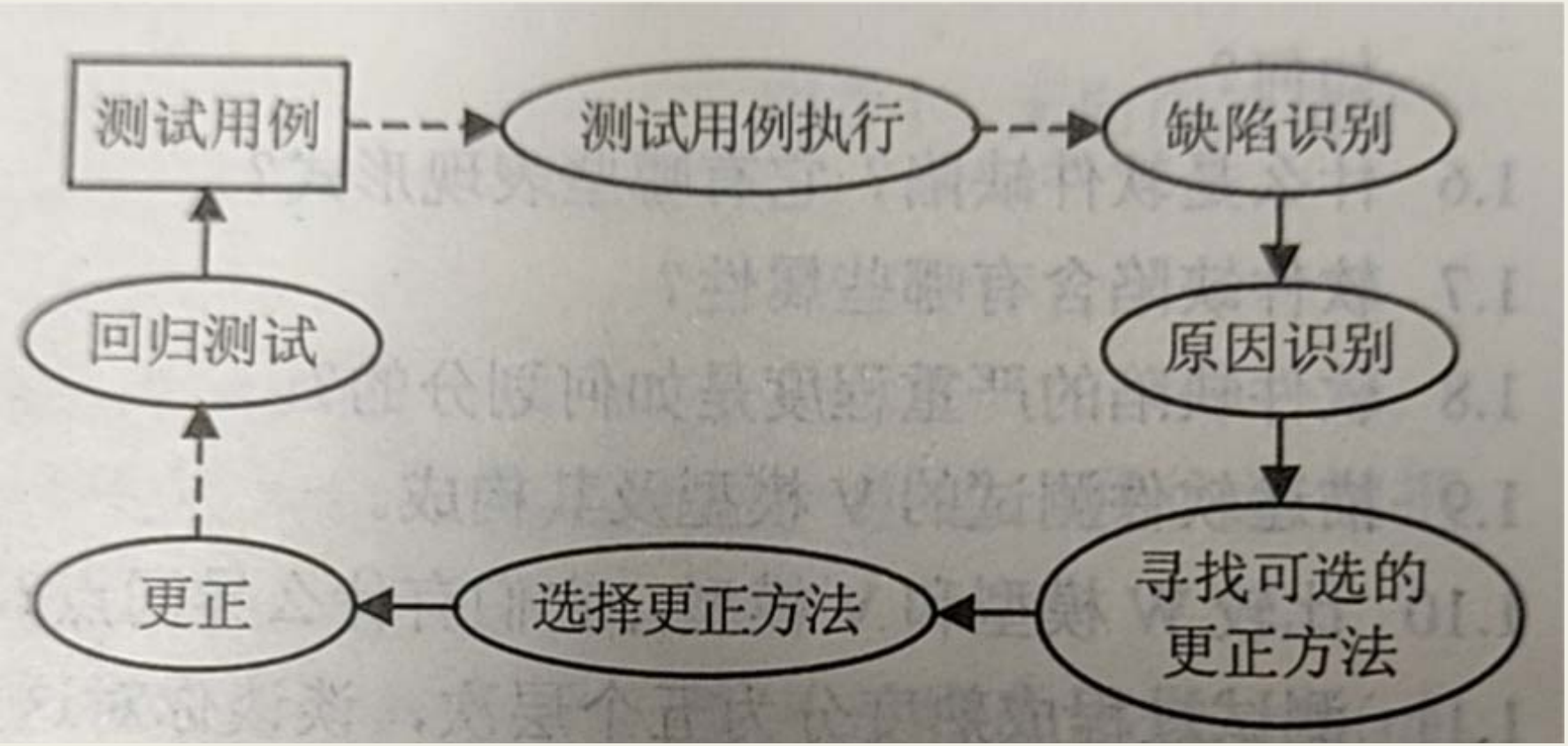
测试的特点

- 测试的目的是增强软件人员对软件正常运行的信心，其过程是从无穷的执行域中选择有穷的测试用例集来对程序行为进行动态验证
- 测试人员的目标是通过设计测试用例，花费最少的时间和最小的代价，系统地找出不同类型的错误
- 测试是通过运行软件以检测将会出现的故障的过程，它提供了一种对质量进行度量的途径。

调试的特点

- 成功地测试后需要调试。调试是在测试发现一个错误后消除错误的过程
- 调试过程从执行一个测试用例开始，直到得到执行结果并发生了预期结果和实际结果不一致的情况。调试过程试图找到症状的原因，从而能够改正错误
- 调试过程有两种结果：
  - *发现问题的原因并将其改正*
  - *未能发现问题的原因*
    - 调试人员应假设一个错误原因，设计测试用例帮助验证此假设，并重复此过程直至最后改正错误

调试的生命周期



软件测试工程师

对软件测试的误解

- 如果发布的软件有质量问题，那是软件测试人员的错。
- 软件测试技术要求不高，至少比编程容易多了。
- 软件测试随便找一个能力差的人就能做。
- 有时间就多测试一些，来不及就少测试一些。
- 软件测试是测试人员的事，与开发人员无关。
- 设计－实现－测试，软件测试是开发后期的一个阶段。

软件测试职业和职位

- 软件测试员
- 软件测试工程师/程序分析员
- 高级软件测试工程师/程序分析员
- 软件测试组负责人
- 软件测试/编程负责人
- 软件测试/质量保证/项目经理

优秀的软件测试工程师的素质

- 他们是群探索者
- 他们是故障排除员
- 他们不放过任何蛛丝马迹
- 他们具有创造性
- 他们是群追求完美者
- 他们判断准确
- 他们注重策略和外交
- 他们善于说服
- 他们有良好的学习能力
- 他们有宽广的知识面