**第一章**

**第一讲 软件测试背景**

1. **软件 = 程序 + 文档 + 数据**

**第二讲 软件测试基础知识**

1. 测试的含义

首先是一项活动，在这项活动中某个系统或组成的部分将在特定的条件下运行，结果将被观察和记录，并对系统或组成部分进行评价。

1. 软件测试

使用人工或自动化手段，来运行或测试某个系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别（IEEE）

1. 软件测试的根本目的

发现\修改缺陷 满足需求，提高用户满意程度 优化软件品质

一个好的测试用例在于发现了还未曾发现的错误；一次成功的测试则是发现了错误的测试。

1. 软件测试对象
2. 软件测试不等于程序测试
3. 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、用户文档都是软件测试的对象

在软件生命周期中，每个阶段都有不同的测试对象，形成了不同开发阶段的不同类型的测试。

1. **软件测试分类**
   1. **测试组织：开发方+用户方+第三方**
   2. **测试用例设计方法：黑盒+白盒+灰盒**
   3. **测试策略与过程：单元—>集成—>系统—>验收**
   4. **基本要求和适用要求：功能、性能**
   5. **回归测试、冒烟测试、随机测试**

按测试组织：开发方测试、用户测试、第三方测试

按测试技术：黑盒测试（不去看代码）、白盒测试、

灰盒测试是介于[白盒测试](http://baike.baidu.com/view/51297.htm)与[黑盒测试](http://baike.baidu.com/view/51274.htm)之间的，可以这样理解，灰盒测试关注输出对于输入的正确性，同时也关注内部表现，但这种关注不象白盒那样详细、完整，只是通过一些表征性的现象、事件、标志来判断内部的运行状态，有时候输出是正确的，但内部其实已经错误了，这种情况非常多，如果每次都通过白盒测试来操作，效率会很低，因此需要采取这样的一种灰盒的方法。

按测试过程：单元测试、集成测试、系统测试、验收测试.

按测试类型：功能、性能、界面、易用性测试、兼容性测试、安全性测试、安装测试

（单元测试：在编码过程中，对每个小程序单元测试）

（集成测试：将单元集成在一起后，可称为组件）

回归测试、冒烟测试、随机测试

（冒烟测试：是指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，是否具备可测性。专门针对某一项功能的测试---主干功能）

验收测试（α测试，不能由[程序员](http://baike.baidu.com/view/39175.htm)或测试员完成。β测试，测试版，公众）

* 我们使用QQ2009 Beta版进行聊天（用户测试、黑盒测试、验收测试）
* 程序员A对刚开发出来的add函数进行检验 （开发方测试、白盒测试、单元测试）
* SCIS系统开发完成了，几名测试老师对该系统进行检验（开发方测试、黑盒测试、系统测试
* 测试老师首先验证SCIS的登录功能和其他基本主功能（开发方测试、黑盒测试、系统测试、冒烟测试）
* 针对SCIS测试老师提出系统缺陷，开发者修改后，测试老师再次检验（开发方测试、黑盒测试、系统测试、回归测试）
* SCIS系统通过测试老师的检验，从各班中抽取几名同学代表进行系统初体验（用户测试、黑盒测试、验收测试（ alpha 测试或系统初验））

1. **测试环境（软件 + 硬件 + 网络）**

指测试软件的时候，软件运行的平台

软件开发环境：VB、VC、VS等

软件生产运行环境：最终用户环境

1. **测试环境搭建原则（记住）**

* 真实:（项目软件、产品软件）：尽量模拟用户的真实使用环境
* 干净：测试环境中尽量不要安装其他与被测软件无关的软件
* 无毒：测试环境没有病毒
* 独立：测试环境和开发环境要独立开来

项目软件：有明确的客户群体，周期较短

产品软件：面向大众去发送，周期较长

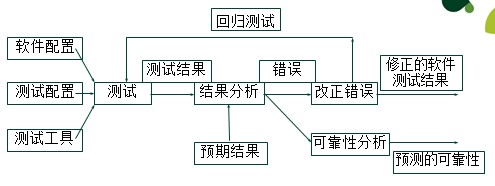
1. **软件测试时间**

* 软件生命周期

计划、需求🡪设计、编码、测试🡪运行维护🡪淘汰停用

* 软件在**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。
* **软件测试尽早开展。**
* **软件测试覆盖整个软件生命周期。**

1. **测试信息流程（简单看）**



测试过程中需要三类输入：软件配置、测试配置和测试工具。

测试最终的结果或输出：得到软件的功能正确或者找出缺陷。

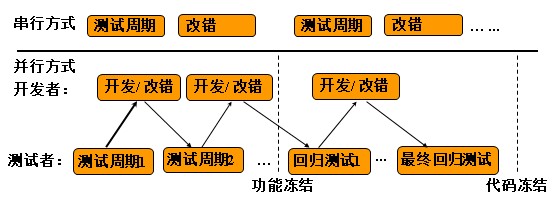
软件配置：包括软件需求规格说明、软件设计规格说明、源代码等；

测试配置：包括测试计划、测试用例、测试驱动程序等。

测试工具：为提高测试效率，采用测试工具支持测试工作，包括测试数据自动生成程序、驱动测试的测试数据库等。

1. 软件测试周期

软件测试的周期性是指测试、改错、再测试、再改错。这样一个循环过程，如下图所示。



**12.软件测试停止依据（理解，第五种推荐）**

* 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
* 第三类标准：使用特定的测试用例设计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和严重程度决定是否停止测试。**

**13.软件测试原则（理解）**

* 尽早地和及时地测试

软件缺陷分布图 **软件产品说明书（需求阶段）56%、**设计阶段27%、编写代码7%、其他10%。**需求阶段产生缺陷最多**

* 测试前应当准备好**测试数据**和与之对应的**预期结果**这两部分

**测试数据 +　预期结果　＝　测试用例**

* 测试输入数据应包括合理的输入条件和不合理输入条件
* 程序提交测试后，应当由专门的测试人员进行测试
* 严格执行测试计划，排除测试的随意性（随意测试，丢失功能点）

回归测试更要注重测试计划

* 测试用例的所有相关预期结果做全面的检查
* 充分注意测试当中的群体现象
* 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
* （补充）缺陷具有免疫性 （每修复3-4个缺陷，一般就会产生一个新的缺陷）

14.测试工作误解

* + 误区一：软件测试技术要求不高，至少比编程容易多了
  + 误区二：若发布的软件有质量问题，那是软件测试人员的错
  + 误区三：软件测试是测试人员的事，与开发人员无关
  + 误区四：根据软件开发瀑布模型，软件测试是开发后期的一个阶段
  + 误区五：有时间就多测试一些，来不及就少测试一些
  + 误区六：软件测试是非建设性的工作，甚至是破坏性的，测试中发现错误是对责任人工作的一种否定

**第三讲 软件测试模型**

1. 软件产品需要各种开发投入

产品说明书、设计文档、进度计划、产品审查、上一版本信息反馈、商业竞争对手的同类软件产品情况、客户调查、易用性数据、观察与感受说明书等

1. 产品开发过程

* 需求调研
  + 对客户调查所收集的详细信息
  + 以前软件的使用情况及存在问题
  + 竞争对手的软件产品信息等
  + **目的：确定将要开发的软件产品应该具有哪些功能**
* 需求规格说明书
  + 对客户需求信息进行综合描述
  + 包括用户未提出、但必须要实现的要求
  + **目的：针对产品进行定义并确定其功能**
* 设计说明书

架构、数据流示意图、状态变化示意图、流程图、注释代码

（建筑一栋大楼，在施工前必须先进规划、设计、绘制各类工程图纸。）

* 开发进度表
* 软件产品组成部分

程序代码、帮助文件、用户手册、样本和示例、产品支持信息、图表和标志、标签、产品和宣传材料、软件的安装、软件说明文件、测试错误提示信息

* 测试文档

一般测试文档所包含的内容

**测试计划：描述用于验证软件是否符合产品说明书和客户需求的整体方案。**

**测试用例：依据测试的项目，并描述验证软件的详细步骤。**

**软件测试报告：描述依据测试用例找出的问题，通常提交测试报告。**

**归纳、统计和总结：采用图表、表格和报告等形式来描述整个测试过程。**

1. 开发模型
2. 大棒开发法
   * 优点：思路简单， 通常可能是开发者的“突发奇想”
   * 缺点：开发过程是非工程化的，随意性大，结果不可预知
   * 测试：开发任务完成后，修复较困难
3. 边写边改发
   * 优点：简单考虑到了软件的需求，产品周期短
   * 缺点：没有计划和文档的编制
   * 测试工作： 由于新的版本不断产生，测试工作长期循环

采用边写边改法的软件开发通常只是有了比较粗略的想法就开始进行简单的设计、然后进行较长的反复编写、测试与修复这样一个循环的过程。在认为无法更精细的描述软件产品要求时，就发布产品。

1. 瀑布发

瀑布模式是将软件生命周期的各项活动，规定为按照固定顺序相连的若干个阶段性工作，形如瀑布流水，最终得到软件产品。

* 优点：易于理解；调研开发的阶段性（明确的阶段性划分）；强调早期计划及需求调查；确定合适能够交付产品及何时进行评审与测试
* 缺点：需求分析只进行一次，不能适应需求变化；顺序的开发流程，使得开发中的经验教训不能反馈到该项目的开发中去；不能反映出软件开发过程的反复与迭代性；没有包含任何类型的风险评估；开发中出现的问题直到开发后期才能够显露，因此失去及早纠正的机会。

不适应需求变化；各步骤分立、没有交叉；反复、迭代不可显示、无法后退；

缺少风险评估；后期暴露问题

* 测试工作：测试工作被定义为在开发后才开始进行

1. 快速原型法

根据客户需求在较短的时间内解决用户最迫切解决的问题，完成可演示的产品。这个产品只实现最重要功能，在得到用户的更加明确的需求之后，原型将丢弃。增加用户反馈阶段。

1. 螺旋模型法

瀑布模型 +　边写边改模型　＋　风险评估　＝　螺旋模型

主要思想是在开始时不必详细定义所有细节，而是从小开始，定义重要功能，尽量实现，接受客户反馈，进入下一阶段，并重复上述过程，直到获得最终产品。

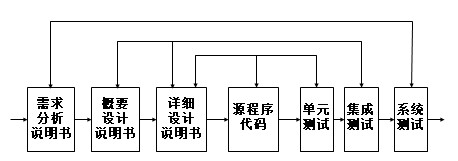
（一个螺旋包括6个步骤：1.确定目标，可选方案有限制条件；2.指出并解决风险；3.评估方案；4.本阶段开发和测试；5.计划下一阶段；6.确定进入下一阶段的方法。[测试](http://www.spasvo.com/)一直在进行，直到最后宣布成功！）

每一螺旋（开发阶段）包括5个步骤：确定目标，选择方案-🡪评估方案，解决风险🡪本阶段的开发和测试🡪计划下一阶段🡪确定下阶段方法

* 优点：严格的全过程风险管理；强调各开发阶段的质量；提供机会评估项目是否有价值继续下去。(发现问题早)

1. **开发与测试的关系（了解）**

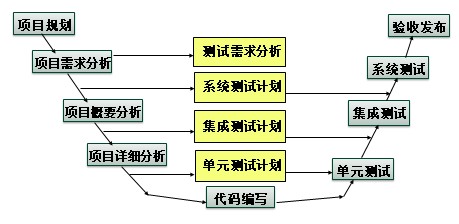
* 测试与开发各阶段的关系



* 测试在开发阶段的作用
* 项目规划阶段：负责从单元测试到系统测试的整个测试阶段的监控。
* 需求分析阶段：确定测试需求分析、系统测试计划的制定，评审后成为管理项目。
* 详细设计和概要设计阶段：确保集成测试计划和单元测试计划完成。
* 编码阶段：由开发人员进行自己负责部分的测试代码。在项目较大时，由专人

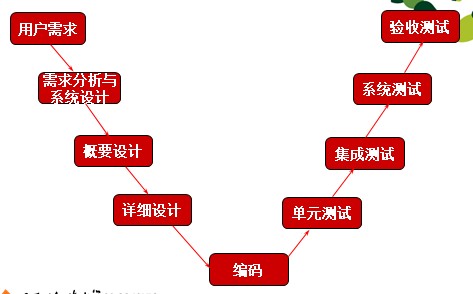
进行编码阶段的测试任务。

* 测试阶段（单元、集成、系统测试）：依据测试代码进行测试，并提交相应的测试状态报告和测试结束报告。
* 完整的软件开发流程



1. **软件测试模型介绍（重点 模型特点、模型图）**

* **软件测试过程模型-------V模型**



* V模型是**最具有代表意义**的测试模型 。
* V模型是软件开发**瀑布模型**的变种，它反映了测试活动与需求分析和设计的关系 。
* 从左到右，描述了**基本的开发过程和测试行为**，非常明确地标明了测试过程中存在的不同级别，并且清楚地描述了这些测试阶段和开发过程期间各阶段的**对应关系** 。
* **箭头**代表了时间方向，左边下降的是开发过程各阶段，与此相对应的是右边上升的部分，即各测试过程的各个阶段。
* **V模型存在一定的局限性**

测试过程编码之后的一个阶段

需求分析阶段的隐藏的问题一直到后期的验收测试才被发现

* V模型的改进

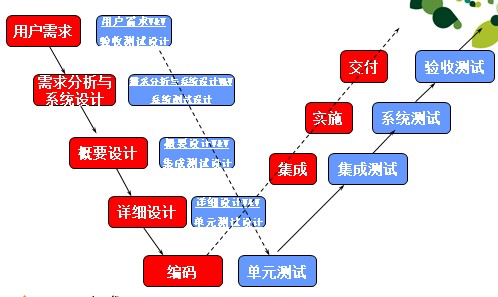
V模型 + 各阶段同步测试 = W模型

* W模型体现了“尽早的和不断进行软件测试”的原则
* **软件测试过程模型------W模型**

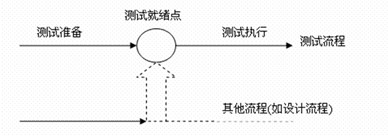
W模型强调测试伴随着整个软件开发周期，而且测试的对象不仅仅是程序，需求、设计等同样需要测试，也就是说测试和开发是同步进行的。

**关于W模型的缺陷：**在W模 型中，需求、设计、编码等活动被视为串行的，同时，测试和开发活动也保持着一种线性的前后关系，只有上一阶段完成后，才可以开始下一阶段的活动，不能支持 迭代和适应开发过程中的变更调整。

**W模型的优点：**W模型可以说是V模型自然而然的发展。它强调：**测试伴随着整个软件开发周期，而且测试的对象不仅仅是程序，需求、功能和设计同样要测试。**



* **软件测试过程模型------H模型**



H模型揭示出：

* 软件测试应尽早准备尽早执行
* 软件测试是一个独立的流程，贯穿于整个产品周期，与开发并行
* 软件测试不仅仅指测试的执行，还包括很多其他的活动
* 软件测试是根据被测物的不同而分层次进行的，不同层次的测试活动可以是按照某个次序先后进行的，但也可以是反复的

**第四讲 软件测试与软件质量**

1. 软件质量的概念

* 质量是系统、部件或过程满足客户或者用户需要的程度。
* 软件质量是软件符合明确叙述的功能和性能**需求**、文档中明确描述的**开发标准**、以及所有专业开发的软件都应**具有的隐含特征**的程度。

1. 软件质量标准（评价一个软件质量好坏的准则）

软件质量标准主要分为两个方向：

产品质量:人们实践产物的属性和行为。模型：McCall、ISO 9126 、Boehm

过程质量：人们生产产品的过程的好坏。模型：CMM、ISO 9000

McCall模型：（**3方面11个质量特性 记住，名词含义理解**）

* **产品修订**：
* 可维护性：为了满足用户新的要求或当环境发生了变化，或运行中发现了新的错误时，对一个已投入运行的软件进行相应诊断和修改所需工作量的大小。
* 可测试性：测试软件以确保其能够执行预定功能所需工作量的大小。
* 灵活性：修改或**改进**一个已投入运行的软件所需工作量的大小。
* **产品变迁**：
* 可移植性：讲一个软件系统从一个计算机系统或环境移植到另一个计算机系统或环境所需工作量的大小
* 可复用性：一个软件能再次用于其它应用程序的程度
* 互联性：又称互操作性，连接一个软件和其他系统所需工作量的大小。
* **产品运行**：
* 正确性：在预订环境下，软件满足设计规格说明及用户预期目标的程度。他还要求软件本身没有错误。
* 可靠性：软件按照设计要求，在规定时间和条件下不出故障，持续运行的程度。
* 可使用性：对于一个软件系统，用户学习、使用软件及为程序准备数据和解释输出所需工作量的大小
* 完整性：为某一目的而保护数据，避免它受到偶然的或有意的破坏、改动或遗失的能力。
* 效率：为了完成预定功能，软件系统所需的计算机资源的多少。

ISO9126模型：（**6个性质特性 记住**、21个子特性）

* **功能性**（适合性、准确性、互用性、依从性、安全性）
* **可靠性**（成熟性、容错性、可恢复性）
* **易使用性**（易理解性、易学性、易操作性）
* **效率**（时间特性、资源特性）
* **可维护性**（易分析性、易改变性、稳定性、易测试性）
* **可移植性**（适应性、易安装性、一致性、易替换性）

CMM过程质量

* 能力成熟度模型：是对于软件组织在定义、实施、度量、控制和改善其软件过程的实践中各个发展阶段的描述。
* CMM的意义：迄今为止学术界和工业界**公认**的有关软件工程和管理实践的**最好的软件过程**。为**评估**软件组织的生产能力**提供了标准**。为**提高**软件组织的生产过程**指明了方向**。
* **CMM五级模型（达到3级基本以成熟 记住）**

**第一级：初始级** 特点是软件过程**无秩序**，有时甚至是**混乱**的。软件过程定义几乎**没有章法和步骤可循**，软件产品所取得的成功往往依赖极**个别人**的努力和机遇（需求、设计、工作计划没有，做完之后就算完了，重新做一个系统的时候就直接再开始做）

**第二级：可重复级** 已建立了基本的项目管理过程，可用于对成本，进度和功能特性进行跟踪。对类似的应用项目，有章可循，并能重复以往所取得的成功。（1.Milestone可见，按计划开发2.软件开发的首要问题不是技术问题而是管理问题。因此，可重复级的焦点集中在软件管理过程上。3.一个可管理的过程则是一个可重复级的过程，一个可重级的过程则能逐渐进化和成熟。4.该级管理过程包括了需求管理、项目管理、质量管理、配置管理和子合同管理五个方面。5.项目管理分为计划过程和跟踪监控过程两个过程。6.通过实施这些过程，从管理角度可以看到一个按计划执行的且阶段可控的软件开发过程。）

**第三级：定义级** 用于管理的、工程的软件过程均已实现文档化、标准化，并形成了整个软件组织的标准软件过程。全部项目均已采用与实际情况相吻合的、适当修改的标准软件过程来进行。

* 每个阶段的内部活动可见
* 较普遍的看法是，**只有当达到了第3级能力成熟度时，才表明这个软件组织的软件能力“成熟”了。**

**第四级：定量管理级** 软件过程和产品质量**有详细的度量标准**，软件过程和产品质量得到了定量的认证和控制。

**第五级: (不断)优化级** 通过对来自过程、新概念和新技术等方面各种有用信息的定量分析，能够不断地、持续性的对过程进行改进。

ISO 9000标准

ISO9000是关于**质量管理**和质量评价的一系列国际标准。定义了一套基本达标的做法，有助于公司一致的交付符合客户质量要求的产品或者服务。

ISO 9000与 CMM区别

ISO9000 国际标准，广泛领域 CMM 行业标准，软件行业

1. **软件质量相关人员种类（区别理解）**

**QC**(quality control)着眼于软件、保证质量符合规定、发现问题、受控

**QA**(assurance)着眼于过程整体、建立体系，确保执行、预防问题、监督

QA工作内容：监控软件开发过程、参与评审，确保文.档有效、保证产品和过程符合相应标准和规程、保证问题得以处理

**第五讲 软件测试职业与素质**

1. 第三方测试：独立于软件公司自身测试的测试。既不是开发方，也不是客户方，而是指软件公司和软件用户之间的一方

**第六讲 测试流程 + 测试文档（很重要）**

1. 软件测试流程

前期：参加需求评审会议

拟定软件测试计划—>设计和生成测试用例—>搭建测试环境—>实施测试—>测试评估—>测试总结

测试阶段：指定测试计划—>测试设计与开发—>执行测试—>评估总结

1. **制定测试计划**

* 测试计划阶段主要处于测试的先期**准备**工作阶段，在该阶段中主要是对将要进行的**测试工作**做整体计划安排。
* 本阶段的主要工作内容

对需求规格说明书的仔细研究

将要测试的产品分解成可独立测试的单元

为每个测试单元确定采用的测试技术

为测试的下一个阶段及其活动制定计划

概要测试计划 +　详细测试计划　＝　制定计划

1. **测试设计与开发**

* 测试设计与开发阶段包含两部分，一部分是**设计**，主要是参照各种相关文档对测试进行设计的工作，包括**测试需求的分析**和**测试用例**的设计；另一部分工作是**开发**，主要是按照设计的测试需求分析与测试用例设计的方案要求进行实施的过程，该过程包括：**测试用例数据的准备**，测试工具的配置、**测试脚本**的开发录制与维护等工作，此阶段的工作可一直持续到软件测试结束。测试用例文档是软件测试的依据，包括测试输入、测试步骤、预期结果等内容。
* **测试用例文档**的本质

从测试的角度对被测对象的功能和各种特性的细化和展开。

测试用例，设计测试用例 –>测试用例文档

* 测试用例文档的好处

保证测试功能不被遗漏，也不被重复测试。合理安排测试人员。使得软件测试不依赖于个人

1. **实施测试**

* 实施软件测试阶段主要是将设计和开发阶段的测试用例和数据应用于软件的过程，执行的过程，执行过程中发现缺陷，可提交缺陷报告。
* **缺陷报告**：是软件测试过程中最重要的文档，它的内容包括：1)记录问题发生的环境，如:各种资源的配置情况。2)记录问题的在现步骤3）记录问题性质的说明4)记录问题的处理进程--问题处理进程从一定角度上反映测试的进程和被测软件的质量状况以及改善过程。
* 初测期——测试主要功能和关键的执行路径，排除主要障碍。
* 细测期——依据测试计划和测试用例，逐一测试大大小小的功能、方方面面的特性、性能、用户界面、兼容性、可用性等等；预期可发现大量不同性质、不同严重程度的错误和问题。
* 回归测试期——系统已达到稳定，在一轮测试中发现的错误已十分有限；复查已知错误的纠正情况，确认未引发任何新的错误时，终结回归测试。

1. 评估总结

评审阶段是在测试结束后对整个测试过程与产品进行评估的过程。

1. 测试文档类型



**第七讲 测试计划**

1. 测试计划：就是在软件测试工作正式实施之前明确测试的对象，并且通过对资源、时间、风险、测试范围和预算等方面的综合分析和规划，保证有效的实施软件测试。
2. 为什么制定测试计划

1）把知识和经验直接转化为执行任务的具体方法2）为组织、安排和管理测试项目提供一个整体框架3）促进团队间关于测试任务和过程的交流4）对项目执行过程中的风险进行分析，并制定相关的应对策略

1. 何时制定测试计划

软件测试计划应当尽早的制定，需求说明书确定之后进行

软件测试计划在测试活动中处于中心位置

1. 使用和维护测试计划

使用过程中要对测试计划进行必要的监测

* 1. 测试计划要经过评审
  2. 测试项目是否按照计划执行
  3. 测试计划是否需要调整或修改

1. 测试计划基本结构

测试计划的简介、测试项目说明**、测试范围、测试手段和策略**、项目**通过**或**失败**的标准、暂停和重新启动测试的**标准\原则**、测试的可交付性**、测试任务分配、环境的需求、职责**、人员和培训需求**、进度表、风险及偶然事故的预测**

1. 正确认识测试计划

* 谁是测试计划的最终用户
* 测试计划的**最终**用户一般是**研发团队**
* 测试计划作为**产品**提交给用户（特殊需求、军方、外包测试用户）
* 关于测试计划的格式和内容

**用户是研发团队：**

* 测试计划的价值取决于它能在多大的程度上帮助你管理你的测试项目和帮助你发现错误。
* 千万不要为了写测试计划而写测试计划，测试计划务必能指导测试工作，切实具有可用性。
* 简单的套用模版，没有意义。

**用户是特殊用户：**按**用户要求**填写

1. 怎样进行软件风险分析

软件风险分析目的：确定被测模块的优先级、深度。

**第二章**

**第1讲 项目测试环境搭建**

1. 测试环境 = 软件环境 + 硬件环境 + 网络环境

对软件系统进行各级测试所基于的软/硬件设备和支持(包括被测软件的运行平台、用于各级测试的工具和与该测试有关的软硬件环境，越接近用户环境越好)

* 硬件环境：指测试必需的服务器、客户端、网络连接设备，以及打印机/扫描仪等辅助硬件设备所构成的环境
* 软件环境：指被测软件运行时的操作系统、数据库及其他应用软件构成的环境
* 网络环境：指被测软件运行时的网络系统、网络结构及其他网络设备构成的环境
* 对测试环境的要求：a.尽可能真实的环境，b.营造纯净、独立的测试环境c.无毒的环境\*符合软件运行的最低要求d.选用比较普及的操作系统和软件平台

1. 测试环境的备份

* 测试过程中会遇到多种不可预测的事情发生，一但造成系统崩溃，则会造成测试数据丢失、测试过程中断或者测试环境的重新搭建
* 经常对测试环境进行多次必要的备份是一个必备的预防措施和一个比较好的习惯
* 对测试环境的备份可以挽回不必要的损失、节省测试的时间、保持测试的连续性

1. 测试环境的恢复

* 一旦测试环境遭到破坏，可以还原最近备份的系统，实现测试环境的恢复
* 目的：维持测试环境的一致性、恢复测试数据、恢复测试环境的当前状态
* 工具：Ghost（Symantec）、Partimage(Linux)

1. 测试环境的快速恢复

* 将对测试环境备份获得的文件复制到其他硬件配置相同的计算机上
* 利用备份恢复工具进行恢复
* 实现多台机器同样的测试环境的快速搭建，从而节省搭建测试环境的时间

1. 测试数据的获取

* 测试数据也是影响测试环境的一个因素
* 测试数据直接影响着测试的质量
* 测试数据有多种来源：用户、测试人员设计、随机生成
* 功能测试不需要大量的数据，需要数据的覆盖率高，测试数据要求尽量真实
* 性能测试需要大量的数据，测试数据应尽可能的达到符合实际的数据分配
* 用户数据：从最终用户处获得实际数据
* 手动 ：对测试数据进行手动的添加
* 自动：利用自动测试工具进行自动的数据添加
* 编码：利用数据库接口通过编制程序来获得数据
* 随机生成：任意的向软件中输入数据

**第2讲 测试策略与过程概述**

1. 软件测试复杂性

无法对程序进行完全测试

a.测试所需要的**输入量**太大 b.测试的**输出结果**太多 c.软件**实现的途径**太多（修改文章途径太多了）d.软件规格说明没有一个客观标准

1. 软件测试策略

是为软件工程过程定义的一个软件测试模板——也就是把特定的测试用例方法放置进去的一系列步骤。

根据软件测试的条件、需求来指定相应的测试方案称为软件测试策略。

**即设计相应的测试方案、方案应用到软件测试过程中每个阶段中。**

开发人员 + 测试组 = 测试 测试 ≠ 调试

1. 软件测试工作准则

* 软件测试的风险

如果不能做到去测试软件所有的情况，则该软件就是有风险的。在客户手中发现缺陷时，代价是很高的。

* 工作原则：
  1. 将各种可能性减小到可控范围
  2. 针对风险恰当选择
  3. 找到最佳测试量

1. 软件测试的方法

* 静态测试与动态测试
* 黑盒测试与白盒测试
* 传统测试与面向对象测试

1. 静态测试

静态测试不实际运行软件，而是静态的来进行检查，主要是对软件编程格式、结构等方面进行评估，也就是 **代码验证测试**（检查代码是否符合相应编码标准和规范），**界面测试**（软件中的实际界面与需求中的说明是否相符） 以及**文档测试** （主要测试用户手册和需求说明是否真正符合用户的实际需求）

总之并不真正运行被测程序，只进行特性分析的这样的测试。

* 代码检查
* 类型：代码走查，代码审查（都是在开发组内部进行开展的，代码审查是一种正式的评审活动，而代码走查的讨论过程是非正式的。）
* 代码走查：采用讲解、讨论和模拟运行的方式进行的查找错误的活动。
* 代码审查：采用讲解、提问并使用编码模板进行的查找错误的活动。一般有正式的计划、流程和结果报告。
* 优点：在实际使用中，代码检查比动态测试更有效率，能快速找到缺陷，发现30%～70%的逻辑设计和编码缺陷；代码检查看到的是问题本身而非征兆。
* 缺点：非常消耗时间，且代码检查需要知识和经验的积累。
* 静态结构分析
* 静态结构分析主要是以图形的方式表现程序的内部结构。

函数调用关系图以**直观的图形方式描述一个应用程序中各个函数的调用和被调用关系**；

控制流图显示一个函数的逻辑结构，由许多节点组成，一个节点代表一条语句或数条**语句**，连接结点的叫边，边表示**节点间的控制流向**。

* 代码质量量度

进行度量的**目的**是为了减少程序复杂程度，提高软件简单性和可理解性，并使软件开发费用减少，开发周期缩短，软件内部潜藏错误减少

1. 动态测试

* 动态方法的**主要特征**是：计算机必须真正运行被测试的程序，通过输入测试用例，对其运行情况即输入与输出的对应关系进行分析，以达到检测的目的。
* 动态测试包括：
* 功能确认与接口测试：**通过1动态执行程序能发现模块间参数传递是否正确**
* 覆盖率分析：
* .主要对代码的执行路径覆盖范围进行评估，**执行了很多用例**
* 性能分析：通过执行程序或者系统，大家可以发现运行速度是不是能接受,是不是缓慢。**代码或者系统运行缓慢是开发过程中常见的重要问题**。应用程序运行速度较慢，开发者找不到问题所在， 如不能解决应用程序的性能问题，则将降低并影响应用程序的质量，**因此查找和修改性能瓶颈成** 为调整整个代码性能的关键。
* 内存分析：内存泄漏会导致系统运行的崩溃，尤其对于嵌入式系统这种资源比较匮乏、应用非常广泛，而且 往往又处于重要部位的，将可能导致无法预料的损失。通过测量内存使用情况，可以了解程序内存 分配的真实情况，发现对内存的不正常使用，在问题出现前发现征兆，在系统崩溃前发现内存泄露 错误；发现内存分配错误，并精确显示发生错误时的上下文情况，指出发生错误的原由。

1. **黑盒测试**

黑盒测试是在程序接口进行测试，在**完全不考虑程序内部结构和内部特性的情况**下，在程序接口进行测试，它只检查程序功能是否按照规格说明书的规定正常使用，**验证程序是否能适当地接收输入数据而产生正确的输出信息。**也被称为用户测试。主要针对软件界面和软件功能进行测试。

**集成/系统/验收测试：黑盒**

* 定义：若测试规划是基于产品的功能，目的是检查程序各个功能是否能够实现，并检查其中的功能错误，则这种测试方法称为黑盒测试方法。
* 黑盒测试又称为**功能测试**（性能测试）、**数据驱动测试**和基**于规格说明**的测试。
* 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和性能的正确性和可操作性。
* 用户观点：用户不关心程序怎样写出来的,只关心如何使用
* 黑盒测试的目的主要是在已知软件产品所应具有的功能的基础上：
* 检查程序功能能否按需求规格说明书的规定正常使用，测试各个功能是否有遗漏，检测性能等特性要求是否满足。
* 检测人机交互是否错误，检测数据结构或外部数据库访问是否错误，程序是否能适当地接收输入数据而产生正确的输出结果，并保持外部信息（如数据库或文件）的完整性。
* 检测程序初始化和终止方面的错误
* 特点：
* 黑盒测试与软件的具体实现过程无关，在软件实现的过程发生变化时，测试用例仍然可以使用。
* 黑盒测试用例的设计可以和软件实现同时进行，这样能够压缩总的开发时间。

1. **白盒测试**

**白盒测试需要完全了解程序结构和处理过程，它按照程序内部逻辑测试程序，检验程序中,每条通路是否按预定要求正确工作。也被称为程序员测试。**

**单元测试：白盒**

定义：若测试规划基于产品的内部结构进行测试，检查内部操作是否按规定执行，软件各个部分功能是否得到充分使用，则这种测试方法称为白盒测试方法。

白盒测试又称为**结构测试**、**逻辑驱动测试** 或 **基于程序的测试**，

一般用来分析程序的内部结构。

1. 关系对比（动静、黑白）

**黑盒：动态（运行程序，只看输入和输出）；**

**静态（不运行程序，只看界面）；**

**白盒：动态（运行程序，并分析代码结构）；**

**静态（不运行程序，只静态查看代码）**

1. 传统测试与面向对象测试

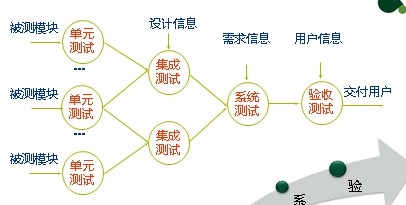
* 前期：参加需求评审会议
* **测试计划拟定阶段**：依据项目开发计划和需求分析结果来制定，是整个测试过程中最重要的阶段，为实现可管理且高质量的测试过程提供基础
* 本阶段的主要工作内容：规划后期测试工作、拟定测试计划书
* 测试用例设计和生成阶段：（本阶段的主要工作内容）
* 对需求说明书、测试计划和原型仔细研究
* 将待测产品分解成可独立测试的单元
* 为每个测试单元确定采用的测试技术
* 编写测试用例，测试脚本
* **测试环境搭建阶段**：软件环境、硬件环境、网络环境
* **实施测试阶段**：（本阶段的主要工作内容）

执行用例脚本🡪 记录测试结果🡪 缺陷提交跟踪及管理 🡪回归 🡪执行用例脚本 🡪 。。。

* 测试评估阶段：（本阶段的主要工作内容）
* 审核系统功能及系统质量
* 汇总bug数量和级别、统计修改率
* 对遗留bug进行审核
* 评估小组一般由项目负责人、营销人员、部门经理等组成，也可能是由客户指定的第三方人员组成
* 测试总结阶段：
* 本阶段的主要工作内容：

选择和保留测试用例、测试结果、测试工具

提交软件测试报告

* 收尾工作的意义和重要性：产品如果升级、功能变更或维护，只要对保留下来的相关测试数只要作相应调整，就能够进行新的测试
* **软件测试过程**
* **单元测试：**针对每个单元的测试， 以确保每个模块能正常工作为目标。
* **集成测试：**对已测试过的模块进行组装，进行集成测试。目的在于检验与软件设计相关的程序结构问题。
* **确认（有效性）测试 （很多公司为：验证程序员自测结果）**是检验所开发的软件能否满足所有功能和性能需求的最后手段。
* **系统测试：**检验软件产品能否与系统的其他部分（比如，硬件、数据库及操作人员）协调工作。
* **验收（用户）测试：**检验软件产品质量的最后一道工序。主要突出用户的作用，同时软件开发人员也应有一定程度的参与。
* 

**第3讲 软件测试技术\_黑盒（1）**

1. 测试用例概述

* **测试用例的定义**：

（1）测试用例是为特定的目的而设计的一组测试输入、执行条件和预期的结果。（2）测试用例是执行的最小实体。

* 测试用例的**特征**：

（1）最有可能抓住错误的；（2）不是重复的、多余的；（3）一组相似测试用例中最有效的；（4）既不是太简单，也不是太复杂。

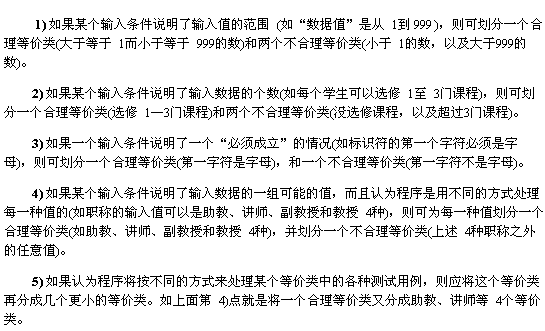
* 测试用例的代表性：能够代表并覆盖各种合理的和不合理的、合法的和非法的、边界的和越界的以及极限的输入数据、操作和环境设置等。
* 测试结果的可判定性：即测试执行结果的正确性是可判定的，每一个测试用例都应有相应的期望结果。
* 测试结果的可再现性：即对同样的测试用例，系统的执行结果应当是相同的。

1. 测试用例文档

测试用例文档是为了完成一个测试用例项的输入、预期结果、测试执行条件等因素的文档。**在测试计划评审通过后开始进行，正式执行测试前完成**

1. 黑盒测试技术概述

完全不考虑程序内部结构和内部特性的情况下进行。

* **黑盒测试常用方法：**等价类划分法、边界值分析法、错误推测法：因果图法、决策表法、正交试验法、场景法
* **等价类划分法：**依据软件需求说明对输入的范围进行细分，然后再分出的每一个区域内选取一个有代表性的测试数据开展测试。它将不能穷举的测试过程进行合理分类，从而保证设计出来的测试用例具有完整性和代表性。
* **等价类是指某个输入域的子集合。在该子集合中，各个输入数据对于揭露程序中的错误都是等效的，它们具有等价特性，即每一类的代表性数据在测试中的作用都等价于这一类中的其它数据。这样，对于表征该类的数据输入将能代表整个子集合的输入。**
* 等价类划分法是把所有可能的输入数据，即程序的输入域划分为若干部分（等价类），然后从每一个子集中选取少数具有代表性的数据作为测试用例。可以保证——测试某等价类的代表值就是等效于对于这一类其它值的测试。
* **有效等价类**：符合需求说明，合理地输入数据集合
* **无效等价类：**不符合需求说明，无意义地输入数据的集合
* **有效等价类 +　无效等价类　＝　等价类**
* **操作步骤**：a.依据常用方法划分等价类　b. 为等价类表中的每一个等价类分别规定一个**唯一的编号** c. 设计一个新用例，使它能够**尽量多**覆盖尚未覆盖的有效等价类。重复该步骤，直到所有**有效等价类**均被用例所覆盖D. 设计一个新用例，使它仅覆盖一个尚未覆盖的**无效等价类**。重复该步骤，直到所有的无效等价类均被用例所覆盖
* 常用举例：
* **边界值分析法**（基础的，最常用的一种）**：**就是对输入或输出的边界值进行测试的一种测试方法。

通常边界值分析法是作为对等价类划分法的补充。

其测试用例来自等价类的边界，应对边界值进行特别关注。

* **与等价类分析法区别：**
* 边界值分析法不是从一个等价类中任选一例子作代表，而是选一个或几个例子，使得该等价类的边界值情况成为测试的主要目标。
* 边界值分析不仅注意输入条件，它还根据输出的情况（即按输出等价类）设计测试用例。
* **在等价类的边界上以及两侧的情况设计测试用例**
* 首先确定边界情况。通常输入或输出等价类的边界就是应该着重测试的边界情况。
* 选取正好等于、刚刚大于或刚刚小于边界的值作为测试数据，而不是选取等价类中的典型值或任意值。
* 如果输入条件规定了值的范围，则应取刚达到这个范围的边界值以及刚刚超过这个范围边界的值作为测试输入数据。
* 如果输入条件规定了值的个数，则用最大个数、最小个数和比最大个数多1个、比最小个数少1个的数作为测试数据。
* 如果程序的规格说明给出的输入域或输出域是有序集合 （如有序表、顺序文件等），则应选取集合中的第一个和 最后一个元素作为测试用例。
* 分析程序规格说明，找出其它可能的边界条件。
* **小练习：**
* 如果程序的规格说明中规定：“重量在10.00公斤至50.00公斤范围内的邮件，其邮费计算公式为……"。

应取10 50 ， 10.01 50.01 ，9.99 49.99 等。

* 一个输入文件应包括1~255个记录。

可取1和255，还应取0及256等。

* 某程序的规格说明要求计算出“每月保险金扣除额为0至1165.25元”。

可取0.00及1165.25、以及-0.01及1165.26等。

* 情报检索系统，要求每次“最少显示1条、最多显示4条情报摘要”。

可取1和4，还应包括0和5等

**第4讲 软件测试技术\_黑盒（2）**

1. **错误推测法：**基于**经验和直觉**推测程序中所有可能存在的各种错误， 从而有**针对性**的设计测试用例的方法。

错误推测法本身不是一种测试技术，而是一种可以应用到所有测试技术中产生更加有效的测试的一种技能。

**经验分享**：

* 时间性测试：提交操作时限、未到达的日期是否可选择、前后时间限制问题、系统时间的调整
* 密码输入框缺陷：明文显示（超级用户）、复制密码，明文显示、一致性
* 配置安全性
* 同时操作问题：a.在不同机器上登录同一用户b.对一条记录在不同机器上进行不同操作（修改、删除）解决方式一：锁定记录。解决方式二：给出提示信息c.一人审核，一人退回d.两人修改同一张工单

1. 常用控件介绍（看课件）

* 文本框测试

数据的内容、长度、类型（注：大小写）、格式（行、日期）、唯一性、空、空格、复制/粘贴+手动、特殊字符、功能键等

* 按钮测试

按钮功能是否实现（关联）

对于不符合业务背景的输入数据是否有相应的处理

提示信息是否正确（正确、友好、无法恢复时）

* 单选按钮测试
  + 各单选按钮功能是否能正确完成
  + 单选按钮是否只能选中一个
  + 是否有默认被选中的选项
  + 功能键（边界）
* 复选框测试
  + 多个复选框可以同时选中
  + 多个复选框可以部分选中
  + 多个复选框可以不被选中
  + 逐一执行每个复选框的功能
  + 组合执行复选框的功能

**第5讲 软件测试技术\_白盒（3）**

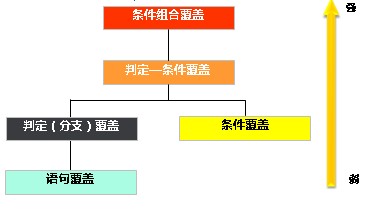
1. 白盒测试技术概述

白盒测试是针对**被测单元内部**是如何进行工作的测试。它根据程序的控制结构设计测试用例，主要用于软件程序验证。

**特点**:主要是检查程序的内部结构、逻辑、循环和路径。

1. **逻辑驱动覆盖测试**：（非常传统的一种测试方法）

根据覆盖目标的不同，逻辑覆盖可分为：**（看课件）**



逻辑驱动覆盖，五种覆盖（大题，小题都要掌握）画图一定要画正确，满足覆盖最少用的测试用例的条数，五种覆盖的强弱关系

* **语句覆盖：**是一个比较弱的测试标准，**含义**：选择足够的测试用例，使得程序中**每个语句**至少都能被执行一次。

更通俗一点来讲，就是让每一个判定，获得每一种可能的情况，真值和假值，有很多种选择

* **判定（分支）覆盖：**是一个比”语句覆盖”稍强的测试标准，含义：选择足够的测试用例，使得程序中**每个分支**至少都能被执行一次。

更通俗一点来讲，就是让每一个判定，获得每一种可能的情况，真值和假值，有很多种选择

* **条件覆盖：**更强的测试标准，含义：选择足够的测试用例，使得判定中**每个条件获得各种可能的取值。**
* **判定-条件覆盖：**综合判定覆盖和条件覆盖，**含义**：选择足够的测试用例，使得判定中每个条件获得各种可能的取值，**使得每个判定取到各种可能的结果。**
* **条件组合覆盖：**选择足够的测试用例，使得**判定中条件的各种组合**都至少出现一次。

**第三章**

**第1讲 软件缺陷概述**

1. 什么是软件缺陷

* 任何程序、系统、以及文档中的问题，同产品设计书的不一致性，不能满足用户的需求

**从产品内部看：**软件缺陷是软件产品开发或维护过程中所存在的错误、毛病等各种问题；

**从产品外部看：**软件缺陷是系统所需要实现的某种功能的失效或违背。

* **至少满足以下5个规则之一，才称为发生一个软件缺陷：（掌握理解）**
* 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，不易使用，运行缓慢或者----最终用户会认为不好

1. 软件缺陷术语

缺点、偏差、谬误、失败、问题、矛盾、毛病、错误、异常

1. **Bug产生原因及分布（能够分析出来）**

* 问题出现在哪里

a.人与人的交流比写程序困难得多。B.没有充分的文档资料。C.项目没有被很好地理解；计划不周，最终导致进度拖延。D.软件可靠性缺少度量的标准，质量无法保证。E.软件难以维护、不易升级。等

* 软件缺陷的产生
* 技术问题：算法错误，语法错误，计算和精度问题，接口参数传递不匹配
* 团队工作：误解、沟通不充分
* 软件本身：a.文档错误、用户使用场合(user scenario)b.时间上不协调或不一致性所带来的问题c.系统的自我恢复或数据的异地备份、灾难性恢复等问题
* **软件缺陷构成**：**软件产品说明书（需求）产生缺陷最多**、设计、编写代码、其他

为什么软件产品规格说明书中缺陷最多？

* 需求：沟通难度
* 未设计、开发在黑暗中摸索前行
* 忽视文档的重要作用
* 需求变动导致信息不一致
* 团队合作不够

1. Bug修复成本

* 软件在从需求、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。

**第2讲 缺陷级别和生命周期**

缺陷严重性含义，能判断严重级别优先级。 严重性跟优先级 缺陷状态和流程（重点掌握）英文的含义，什么状态下使用

三款软件 （简单理解，不需要掌握） JIRA BUGZILL BUGFREE 工具是做什么的

1. **软件缺陷—严重性**：表示软件缺陷所造成的危害的恶劣程度

* **Fatal**：致命的错误，造成系统或应用程序崩溃、死机、系统悬挂，或造成数据丢失、主要功能完全丧失等。
* **Critical**：严重错误，主要指功能或特性没有实现，主要功能部分丧失，次要功能完全丧失，或致命的错误声明。
* **Major**：主要错误，这样的缺陷虽然不影响系统的使用，但没有很好地实现功能，没达到预期效果。如提示信息不太准确，或用户界面差，操作时间长等。
* **Minor**：一些小问题，对功能几乎没有影响，产品及属性仍可使用。
* **Suggestion**：一些友好的建议。

**A类—严重错误:**

1．由于程序所引起的死机,非法退出 2．死循环3．数据库发生死锁 4．因错误操作导致的程序中断 5．功能错误6．与数据库连接错误7．数据通讯错误

**B类—较严重错误:**

1．程序错误2．程序接口错误 3．数据库的表、业务规则、缺省值未加完整性等约束条件

**C类—一般性错误:**

1.操作界面错误（包括数据窗口内列名定义、含义是否一致） 2.打印内容、格式错误3.简单的输入限制未放在前台进行控制 4.删除操作未给出提示5.数据库表中有过多的空字段

**D类—较小错误:**

1.界面不规范 2.辅助说明描述不清楚 3．输入输出不规范4.长操作未给用户提示 5.提示窗口文字未采用行业术语6.可输入区域和只读区域没有明显的区分标志

**E类—测试建议**

1. **软件缺陷—优先级:** 表示修复缺陷的先后次序的指标

一般优先级的划分用ABCD或数字1—4表示，A或1表示最高级别，D或4表示最低级别。

**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

**最低优先级：**可能会修复，但是也能不修复

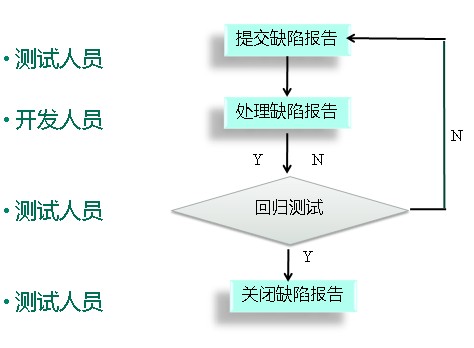
* **一般情况：严重程度高的缺陷优先级高**
* **特殊情况：不成正比**

1. **软件缺陷—状态**

缺陷通过一个跟踪修复过程的进展情况，也就是在缺陷生命周期中的状态基本定义



1. 软件缺陷—生命周期



1. JIRA介绍

JIRA是集项目计划、任务分配、需求管理、错误跟踪于一体的商业软件。

JIRA 是目前比较流行的基于**Java架构**的管理系统，其在开源领域，认知度比其他的产品要高得多，而且易用性也好一些。同时，**开源**则是其另一特色，在用户购买其软件的同时，也就将源代码也购置进来，方便做二次开发。

1. Bugzilla
2. BugFree

**第3讲 缺陷报告编写**

缺陷报告的编写，编写原则，思路（理解）一个缺陷报告只包含一个缺陷

1. 如何提交Bug

* 空中接龙模式：口头描述，该方法国内一些小企业还在使用
* 缺点：容易忘记；沟通理解容易出现差异；打乱开发人员思路
* 流水记帐模式：直接记录，内容可以记录成txt,word,excel等格式
* 缺点：不能及时反映bug情况，延误重要问题修改时间；不方便管理
* 系统管理模式：使用专业的缺陷管理工具帮助提交缺陷，常用的这些工具有Qc,TD,bugfree,bugzllia,JIRA等，在软件企业中广泛使用
* 优点：可以标识其中的任意一个缺陷，并可以追踪这些缺陷，能及时有效的进行bug修复
* 测试人员：直接将缺陷描述提交到缺陷管理工具中
* 开发人员：定时到缺陷管理工具中去查找自己需要修改的缺陷

1. 缺陷报告用途：记录缺陷、缺陷分类（为解决缺陷分配资源）、缺陷跟踪
2. 怎样编写缺陷报告

* 保证重现缺陷
* 判断一个缺陷报告撰写好坏的简单方法：让**非缺陷报告撰写者**（技术人员）依据缺陷报告重现缺陷，如果能**简单**、**迅速**的**重现**缺陷，表明缺陷报告较好。
* 分析故障——使用最少步骤重现缺陷
* 减少开发人员重现缺陷的时间
* 使开发人员更准确的定位缺陷
* 包含所有重现缺陷的**必要步骤**

测试人员假定常用的操作步骤开发人员不一定熟悉，省略了必要的步骤常常造成开发人员无法重现缺陷

* 方便阅读
* 一个缺陷一个报告
* 注意自己的语气
* 值得注意的经验:报告小缺陷、报告随机缺陷、不要夸大缺陷

（第四讲，第五讲）理解

**第4讲 BugFree搭建**

**第5讲 BugFree企业级应用**

1. **BugFree系统简介**

BugFree基于PHP和MySQL开发，是免费且开发源代码的缺陷管理系统。服务器端在Linux和Windows平台上都可以运行；客户端无需安装任何软件，通过IE、FireFox等浏览器就可以自由使用。

**第四章**

单元测试的主要内容，不需要特别记忆，但要理解

性能测试四种分类（记住，理解，对应例子能对上号）

回归测试任何阶段都有可能发生，有新版本就会产生

验收测试 具体实施过程中项目软件跟产品软件相应的验收过程，分清项目软件跟产品软件 a阿尔法测试，b贝塔测试

**第1讲 单元测试**

1. 单元与单元测试的概念

**单元测试**：是对软件基本组成单元进行的测试。是检验程序最小单位，即检查模块有无错误，它是在编码完成后必须进行的测试工作。

* 目的：执行单元测试，是为了证明某段代码的行为确定和开发者所期望的一致。
* 时间：时间：编码—编译—单元测试（静态分析+动态运行）
* 注意：前期完成单元测试计划、设计好用例
* 依据：详细设计说明
* 执行者：程序开发者或白盒测试人员
* 测试方法：以白盒测试法为主（覆盖），先静态检查代码是否符合规范，再动态运行代码，检查结果。
* 好处：可靠性、可读性维护性更强、移植性

1. 单元测试的主要任务

单元测试针对每个程序的模块，主要测试如下5方面：模块接口、局部数据结构、边界条件、独立的路径和错误处理。

* 模块接口：对模块接口进行测试，检查进出程序单元的数据流是否正确。须在其它测试之前进行。

注：主要关注单元中的输入和输出。

* 局部数据结构：测试模块内部的数据能否保持完整性，包括内部数据的内容、形式及相互关系不发生错误。
* 常见错误：不正确的或不一致的类型说明。错误的初始化或默认值。错误的变量名，如拼写错误或书写错误。下溢、上溢或者地址错误。

注：主要关注与被测单元内部的相关数据的类型。

* 路径测试：针对程序路径进行测试，尤为重要。重点关注由于计算错误、不正确的判定或不正常的控制流而产生的错误。

注：主要关注程序的逻辑分支问题。

* 边界条件：采用边界值分析方法设计测试用例，重点关注程序边界处。

注:主要针对于单元测试中的边界问题。

* 出错处理：重点关注模块在工作中发生错误时，出错处理设施是否有

注：主要针对程序中的错误提示。

1. 单元测试的执行过程

单元测试时，如果模块不是独立的程序，需要设置一些辅助测试模块。

* 驱动模块：用来模拟被测试模块的上一级模块，相当于被测模块的主程序。它接收数据，将相关数据传送给被测模块，启动被测模块，并打印出相应的结果。
* 桩模块：用来模拟被测模块工作过程中所调用的模块。它们一般只进行很少的数据处理。

1. 单元测试实例

* **JUnit简介**

JUnit 是 Java 社区中知名度最高的**单元测试工具**，成为Java开发中单元测试框架的事实标准。

* **NUnit简介**

NUnit是一个**单元测试框架**, 它适合于所有.NET语言

**第2讲 集成测试**

1. **集成测试概念：**是每个模块完成单元测试后，按照设计时确定的结构图，将它们连接起来进行测试。

集成测试也称综合测试、组装测试、联合测试。

1. 集成测试基础理论

* **目标：**各单元组合后能按既定意图协作运行，并确保软件的行为正确
* **测试内容：**单元间的接口及集成后的功能。
* **时间：1**单元测试-集成测试（理论上）2同步进行（真实工作中）
* **注意：**前期完成集成测试计划、设计好用例

概要设计说明 + 详细设计说明 🡪 测试依据

开发人员 ＋　测试人员　＝　测试执行者

非增量式　＋增量式 = 测试形式

黑盒 +　白盒　＝　测试技术

* **重点关注：ａ.**数据穿越接口是否丢失。**B.**一模块是否会破坏另一模块功**c.**子功能组装是否达到所要求的主功能。**D.**全局数据结构是否出问题。**e.**误差累积问题。

1. 非增量式测试：采用一步到位的方法来构造测试。

优点：节省时间

缺点：a.一次集成模块多时，容易出现混乱 b.当发现很多故障时，故障定位和纠正困难 c.修正一个故障同时，可能又引入新故障，新旧故障混杂，难判定出错的具体原因和位置

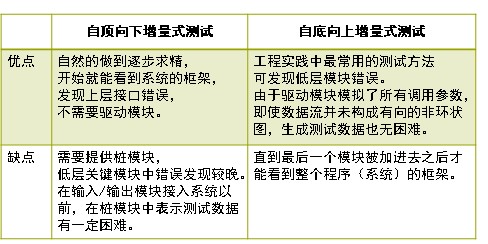
1. 增量式测试：逐次将未曾集成测试的模块和已集成测试的模块（或子系统）结合成较大模块

**按照不同的实施次序**，**增量式集成测试又可以分为三种不同的方法**：自顶向下增量式测试、自底向上增量式测试、混合增量式测试

1. 不同集成测试方法的比较

* 非增量式测试：先分散在集中，若在模块的接口处存在错误，只在最后的集成测试时一下子暴露。修改难度大。
* 增量式：逐步集成和测试，差错分散暴露。一些模块得到较多次的考验

结论：增量式优越于非增量式



**第3讲 系统测试（3课时）**

1. 系统测试的概念和类型

系统测试：将整个软件系统看做一个**整体**进行测试，包括对功能、性能等，以及将计算机硬件、某些支持软件、数据和人员等系统元素**结合**起来，在**实际运行环境**下对软件进行测试 黑盒测试工程师

1. **各类型测试介绍（都属于系统测试）**
2. **功能测试**主要是对产品的各功能点进行验证。根据**需求规格说明书**和**功能测试用例**，逐项测试以检查产品是否达到用户的要求。
3. **界面测试**
4. **易用性测试：**从软件的**使用合理性**和**方便性**等角度对软件系统进行检查，来发现软件不方便用户使用的地方
5. **性能测试：**是通过自动化的测试工具模拟多种**正常**、**峰值**以及**异常负载**条件来对系统的**各项性能指标**进行测试

* 时间性能：软件的一个具体事务的响应时间
* 空间性能：软件运行时所消耗的系统资源

**分类：**一般性能测试、可靠性测试、负载测试、压力测试

* **一般性能测试**：验证软件在正常环境和系统条件下重复使用时是否还能满足性能指标，如运行速度、响应时间、占有系统资源等，不施加任何压力。
* **可靠性测试：**从验证的角度出发，检验系统的可靠性是否达到预期的目标，同时给出当前系统可能的可靠性增长情况。也叫稳定性测试，即连续运行被测系统，检查系统运行时的稳定程度
* **负载测试**：通常让被测系统在其能忍受的压力的极限范围之内连续运行，来测试系统的可靠性。
* **压力测试：**是持续不断地给被测系统增加压力，直到将系统压垮为止，用来测试系统所能承受的最大压力。

1. **兼容性测试：**指测试软件在特定的硬件平台上、不同的应用软件之间、不同的操作系统平台上、不同的网络等环境中是否能很好地运行的测试。

* 兼容性测试的作用
* 兼容性测试能进一步提高产品的质量
* 兼容性测试能是软件与尽可能多的其他软件“和平共处”，尽可能达到平台无关性
* 兼容性测试能尽可能的保证软件存在的价值，它是衡量一个软件质量的重要指标
* 兼容性测试能使软件产品的市场更广阔

**配置测试：**指验证在不同的硬件配置和软件配置下，应用程序能否正常工作。

* 1. **目的：**是保证软件在其相关的硬件上能够正常运行。
  2. **核心：**使用各种硬件来测试软件的运行情况。

**不同版本间的兼容**：指要实现测试平台和应用软件多个版本之间能够正常工作。

**数据兼容性**指要在应用程序之间共享数据，它要求支持并遵守公开的标准，允许用户与其他软件无障碍的传输数据

1. **安全性测试:** 检查系统对非法侵入的防范能力。
   1. **目的：**验证安装在系统内的保护机制能否在实际中保护系统且不受非法入侵，不受非法干扰。（应对入侵、自我防护和应变）
   2. 在安全测试中**，测试者扮演着试图攻击系统的角色：**
      1. 尝试去通过外部的手段来获取系统的密码
      2. 使用可以瓦解任何防守的客户软件来攻击系统
      3. 把系统“瘫痪”，使得其他用户无法访问
      4. 有目的地引发系统错误，期望在恢复过程中侵入系统
      5. 通过浏览非保密的数据，从中找到进入系统的钥匙
   3. 系统的安全测试要**设置一些测试用例试图突破系统的安全保密措施**，检验系统是否有安全保密的漏洞。
2. **国际化/本地化测试**

**软件本地化**是将一个软件产品按照特定国家或语言市场的需要进行全面定制的过程。包括：翻译、重新设计、功能调整、功能测试、当地风俗、文化背景、语言、方言等。

1. **安装测试:** 广义上的安装和卸载
2. **文档测试:** 是关于被测软件的**相关文档的正确性的测试**

**第4讲 回归测试&验收测试**

1. 回归测试的**概念**：对软件的新的版本测试时，对新版本进行重新测试，这时的测试不仅是验证被修复的软件缺陷是否被解决了，且要保证以前所有运行正常的功能依旧保持正常，而不要受到这次修改的影响。
2. **回归测试目的**：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。
3. **回归测试测试阶段**：**任一阶段**
4. **验收测试**是在系统测试之后，以用户测试为主，或有测试人员等质量保障人员共同参与的测试，是检验软件产品质量正式交给用户使用的最后一道工
5. **Alpha测试(内部测试)：**

α测试：软件开发公司组织内部人员模拟各类用户对即将面世的软件产品（称为α版本）进行测试，试图发现错误。由用户、测试人员、开发人员等共同参与的内部测试。

**关键**：尽可能逼真模拟实际运行环境和用户对软件产品的操作，尽最大努力涵盖所有用户操作。

1. **Beta测试 ：** β测试：内测之后的公测，即完全交给最终用户测试。软件开发公司组织各方面的典型用户在日常生活中实际使用β版本，并要求用户报告异常情况、提出批评意见。然后软件开发公司再对β版本进行改错和完善。

**第五章**

用户手册包含哪些内容，编写原则，测试总结报告

**第1讲 用户使用手册写作**

1. **测试文档包括：**测试计划书、测试方案文档、测试计划评审报告、测试用例文档、测试记录、缺陷报告、用户使用手册、用户确认测试报告、测试报告
2. **写作原则：**假设用户什么都不懂、通俗易懂、层次分明（功能OR业务）、详细充实、图文并茂、适当举例
3. 写作方式：

* 功能介绍型：按照菜单、模块划分、按照页面依次介绍
* **特点**1 写法简单，不需业务知识 2 单个模块容易理解 3 大多数说明书采用该方式
* **缺点**1 不方便理解业务 2 各功能间关联性差，即作为不同角色的用户不知道自己到底可以做什么操作
* 业务介绍型：按照角色划分、按照业务流程依次介绍
* **特点：** 1 写法较复杂，需熟悉业务 2 以实际业务处理流程介绍 3 方便熟悉业务 4 这种形式的说明书很少见
* **缺点** ：这样的写法可能不全面

**第2讲 测试总结报告编写**

1. **测试报告文档是**测试阶段最后的文档产出物，把测试的过程和结果写成文档，并对发现的问题和缺陷进行分析，为纠正软件的存在的质量问题提供依据，同时为软件验收和交付打下基础。
2. 测试总结报告内容：

* 文件名称、编号、版本等基本信息
* 引言（编写目的、项目介绍、常用术语、参考文档）
* 测试概要（测试用例设计、测试环境与配置、工具）
* **测试结果与缺陷分析**（测试执行情况与记录；覆盖结果分析；缺陷统计与分析 ）
* **测试结论与建议**（测试结论、测试建议）