**概述**

**软件测试的目的：**

1. 测试是程序的执行过程，目的在于发现错误。
2. 测试是为了证明程序有错，而不是证明程序无错
3. 一个好的测试用例在于发现至今未发现的错误
4. 一个成功的测试是发现了至今未发现的错误的测试

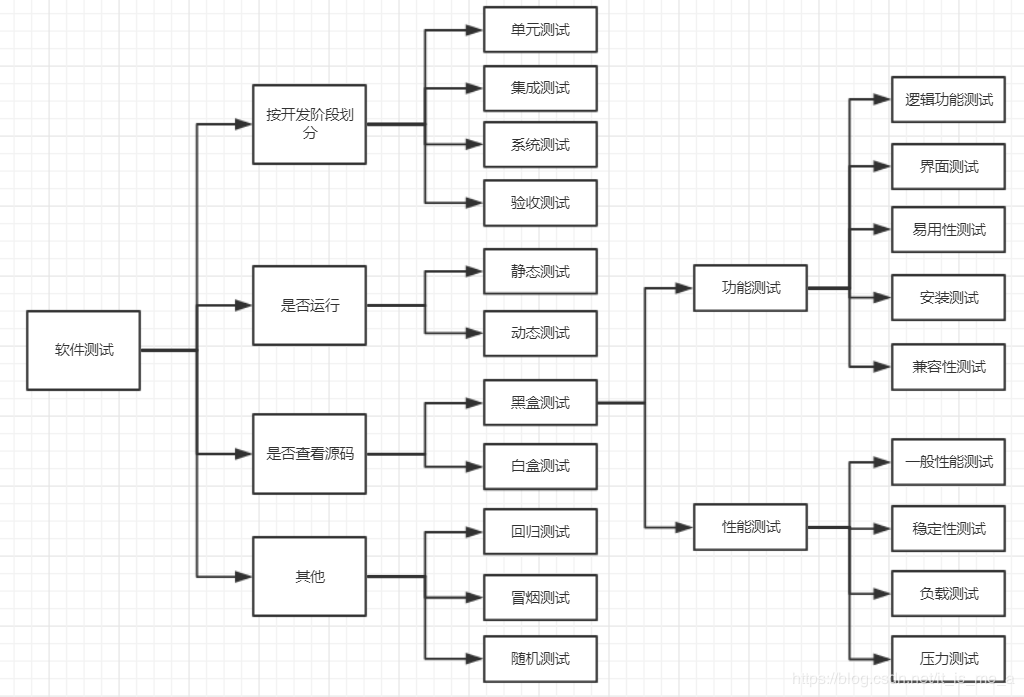
**软件测试的特征：**

1. 可以从需求开始，为不仅仅是代码
2. 即使静态活动也是动态活动
3. 用来预防失效
4. 有助于在软件生命周期内尽早的发现问题，以降低修复缺陷所需成本
5. 过程中应创建可重用的测试件

**软件测试的关键性原则：**

△软件测试是证伪而非证真  
△尽早地和不断地进行软件测试  
△重视无效数据和非预期使用习惯的测试  
△程序员应该避免检查自包的程序  
△充分注意测试中的群集现象  
△用例要定期评申  
△应当对每一个测试结果做全面检查  
△测试现场保护和资料归档  
△软件测试的经济型原则

**分类**

**软件测试的分类：**  


**单元测试（unit testing）：**  
单元测试又称模块测试，针对秋件设计中的量小单位一程序模块，进行正确性检查的测试工作。单元测试需要从程序的内部结构出发设计测试用例。多个模块可以平行地独立进行单元测试。

*单元*：C中指一个函数，Java中指一个类等。

* 1：什么时候进行单元测试？  
  编码后，编译通过后进行
* 2： 由谁来完成单元测试？  
  白盒测试工程师或者开发工程师，最好不要白己做白己代码的测试
* 3： 单元测试的依据？  
  源程序(代码+注释) +《详细设计文档》
* 4： 单元测试通过的标准？  
  程序通过所有单元测试用例  
  语句的覆盖率达到100%  
  分支的履盖率达到85%
* 5： 如何进行单元测试？  
  单元测试主要用白盒测试，先静态地检查代码是否符合规范，然后动态运行代码，检查其实际运行结果，检查程序的运行结果是否正确是一个最基本的要求，还要关注容错处理，程序的边界值处理等。

**集成测试（integration testing）：**  
集成测试又叫组装测试，通常在单元测试的基础上，将所有程序模块进行有序的、递增的测试。重点测试不同模块的接口部分。

* 1．什么时候进行集成测试？  
  单元测试＆集成测试同步进行，理论上先有单元测试
* 2．由谁来做集成测试？  
  白盒测试工程师成者开发工程师
* 3．集成测试的依据？  
  单元测试的模块＋《概要设计》文档。

**系統测试 （system testing）：**  
指的是将整个系统看为一个整体进行测试，包括对功能、性能、以及软件所运行的软硬件环境进行测试。

目前系统测试主要由黑盒测试工程师在系统集成完毕后进行测试，前期主要测试系统的功能是否满足需求，后期主要测试系统运行的性能是否满足求，以及系统在不同的软硬件环境中的兼容性等。

**验收测试 （acceptance testing）：**  
验收测试指按照项目任务书或合同、供需双方的定的验收依据文档进行的对整个系统的测试与评审，决定是否接收或拒收系统。在系统测试的后期，以用户测试为主或有测试人员等质量保证人员共同参与的测试。

* α测试：指的是由用户，测试人员、开发人员等共同参与的内部测试。
* β测试：指的是内测后的公测，即完全交给最终用户测试
* 正式的验收测试

**静态测试（ static testing）：**  
是指不实际运行被测软件，而只是静态地检查程序代码、界面或文档中可能存在的错误过程。

* 代码测试：主要测试代码是否符合相应的标准和规范
* 界面测试：主要测试软件的实际界面与需求中的说明是否相符
* 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。
* 工具：（Logiscope）Telelogic，可以用作 Java／C++ 等规范

**动态测试（ dynamic testing）：**  
是指实际运行被测程序，输入相应的测试数据检查实际输出结果和预期结果是否相符的过程。

* 动态测试方法为结构和正确性测试
* 动态测试工具 Robot、QTP等

**黑盒测试（ black-box testing）：**  
指的是把被测的软件看做一个黑盒子，我们不关心盒子里面的结构是什么样子的，只关心软件的输入数据和输出数据。

**静态黑盒测试**

测试内容：

* 开发文档：软件需求说明书、数据库说明、概要说明书、详细设计说明节、可行性研究报告
* 用户文档：用户手册、操作手册，维护修改建议

**白盒测试（ white-box testing）：**  
指的是把盒子打开，去研究里面的源代码和程序结构。

在软件公司中，往往用黑盒测试和白盒测试相结合的方式。

* 软件的整体功能和性能进行黑盒测试
* 软件的源代码用白金测试

**功能测试 （function testing）：**  
是黑盒测试的一方面，它检查实际软件的功能是否符合用户的需求。

* 逻辑功能测试（ logic function testing）
* 界面测试（ UI testing）
* 易用性测试（ usability testing）
* 安装测试（ installation testing）
* 兼容性测试（ compatibility testing）

**性能测试（ performance testing）：**  
是软件测试的高端领域，性能测试工程师的待通和白盒测试工程师不相上下，通常我们所说的高级测试工程师一般就是指性能测试或是白盒测试工程师。

* 时间性能（事务响应时间等）
* 空间性能（系统资源消耗）
* 一般性能测试
* 可靠性测试
* 压力测试
* 负载测试

**回归测试 regression testing**  
是指软件被修改后重新进行的测试，如重复执行上一个版本测试时的用例，是为了保证对软件所做的修改没有引入新的错误而重复进行的测试。

**冒烟测试（ smoke testing）**  
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，是否具备可测试性。

**随机测试（ random testing）**  
是指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。

**软件缺陷**

**软件缺陷定义：**

◆软件未达到产品说明书中标明的功能  
◆软件出现了产品说明书中指明的不会出现的功能。  
◆软件功能超出了产品说明书中指明的范围。  
◆软件未达到产品说明书中指明应达到的目标。  
◆软件测试人员认为软件难以理解和使用、运行速度慢，或最终用户认为不好。  
符合以上任意一种情况，极为软件缺陷。

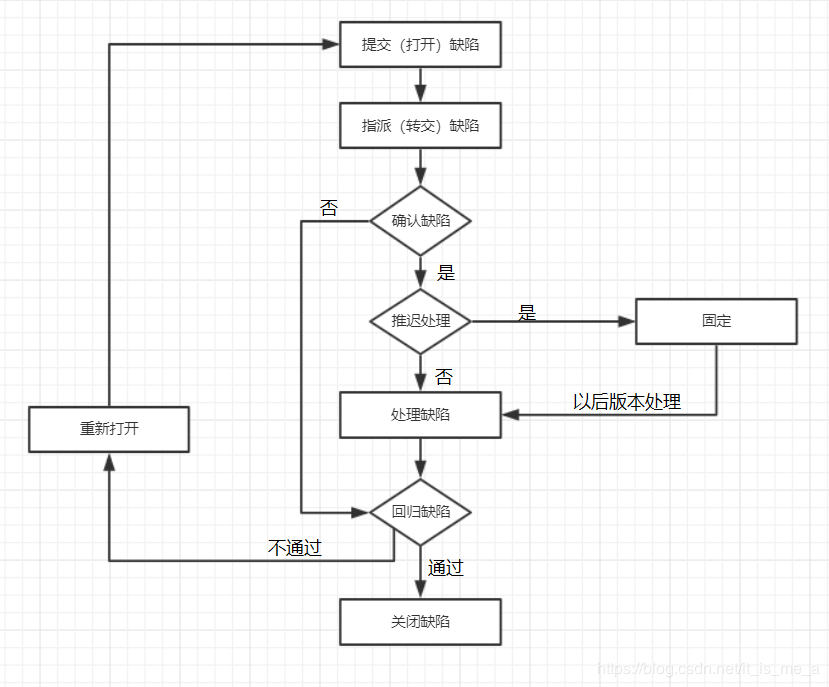
**软件缺陷管理的目标：**  
确保每个被发现的缺陷都能有效地解决。（注意：这里的解决不一定是被修复）

**软件缺陷属性**  
发现缺陷后，需要提交缺陷单，通常情况下，缺陷单需要包含以下的内容：  


**缺陷状态**

| **序号** | **优先级** | **描述** |
| --- | --- | --- |
| 1 | 提交（Submitted） | 已提交缺陷 |
| 2 | 打开（Open） | 确认待处理的缺陷 |
| 3 | 已拒绝（Rejected） | 被拒绝处理的缺陷 |
| 4 | 已解（Resolved） | 以修复的缺陷 |
| 5 | 已关闭（Closed） | 确认解决的缺陷 |
| 6 | 重新打（Reopen） | 修复验证不通过，被重新打开的缺陷 |

**软件缺陷生命周期**

  
**常见缺陷管理工具**

（1） Bugzilla  
（2） BugFree  
（3） HP Quality Center  
（4） JIRA  
（5） Mantis

**软件质量**

**软件质量特性**

软件质量特性分为静态质量特性和动态质量特性。

静态质量特性包括结构化的，可维护的，可测试的代码以及正确而又完整的文档。

动态质量特性包括 正确性，可靠性， 完整性，一致性， 易用性， 性能等。

* 正确性：如果软件针对其输入域中的每个元素都能得到预期的结果，则称该软件是正确的。
* 可靠性：是指软件在给定时间间隔和给定条件下无故障运行的概率。
* 易用性：是指软件使用的难易程度。其本身是一个研究领域，有大量技术可用于易用性测试，心理学在易用性测试设计中也扮演着重要的角色。
* 完整性：指得到软件需求规格说明书或者用户手册中所有功能的可能性
* 一致性：指软件对常规惯例和假设的遵循程度。例如，用户界面中所有按钮遵从统一的颜色编码规定。
* 性能：可以简单理解为软件完成规定任务所花费的时间。

**软件测试关键过程：**

* 单元测试
* 集成测试
* 系统测试
* 验收测试
* 文档测试/评审
* 缺陷测试

**单元测试**是对程序的最小模块，最小单元进行测试，他可以是一个方法/类等，但这个单元有可能不能单独运行，得要比别的程序调用他或者它需要调用别的函数/类进行，所以测试的时候我们需要一些辅助的方法：驱动程序和桩程序。

单元测试包括：

1. 模块接口测试
2. 局部数据测试
3. 路径测试
4. 错误处理测试
5. 边界测试

模块接口测试

* 调用所测模块时的输入参数与模块的形式参数在个数、属性、顺序上是否匹配
* 所测模块调用子模块时，它输入子模块的参数与子模块的形式参数在个数、属性、顺序上是否匹配
* 输出给标准函数的参数在个数、属性、顺序上是否匹配
* 全局变量的定义在各模块是否一致

局部数据结构测试

* 检查不正确或者不一致的数据类型说明
* 使用尚未赋值或尚未初始化的变量
* 错误的初始值或者默认值；
* 变量名拼写错误或书写错误
* 不一致的数据类型。

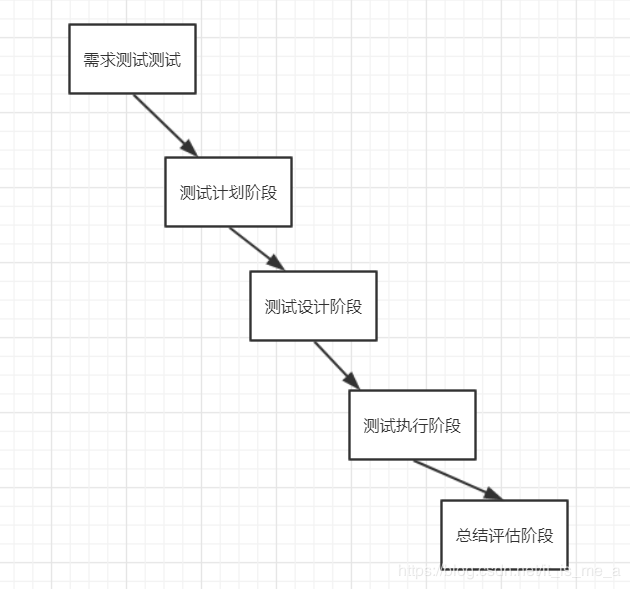
**集成测试**包括：

1. 自顶向下增式测试
2. 自底向上增式测试

一个集成测试策略必须回答三个问题：

* 哪些是集成测试的重点？
* 模块接口应以什么样的顺序进行检测？
* 应该使用哪种测试设计技术检测每个接口？

**测试项目周期：**



**例子一**

测试三角形程序：  
输入3个整数a、b和c，作为三角形的3条边，通过程序判断由这3条边构成的三角形类型是等边三角形、等腰三角形还是一般三角形，并打印出相应的信息。

需要测试用例  
（1）3数相等  
（2）3数中有2个数相等，比如AB相等  
（3）3数中有2个数相等，比如BC相等  
（4）3数中有2个数相等，比如AC相等  
（5）3数均不相等  
（6）2数之和不大于第3数。比如最大数是A  
（7）2数之和不大于第3数。比如最大数是B  
（8）2数之和不大于第3数，比如最大数是C  
（9）含有零数据  
（10）含有负整数  
（11）少于3个整数  
（12）含有非整数  
（13）含有非数字符  
（14）2数之和等于第3数  
（15）输入3个零  
（16）输入3个负数

**软件测试终止准则**

软件消亡之前，如果没有测试j结束标准那么软件测试就永无休止。软件测试终止条件需要依据项目具体情况来制定,一般而言，其遵循以下终止准则：

（1）基于测试阶段的原则  
每个软件都经过单元测试、集成测试、系统测试这几个试阶段，我们可以对单元测试、集成测试、系统测试制定各自具体的测试结束标准，当每个阶段的测试结束标准都符合，我们认为该软件达到测试停止标准。

（2）基于测试用例的原则  
测试设计人员设计测试用例，并请项目组成员参与用例评审，一旦评审通过，就可以作为后面测试结束的一个参考标准。比如测试过程中如果发现测试用例通过率太低，可以拒绝继续测试，待开发人员修复后再继续。在比如可以制定功能测试用例通过率100％，非功能性测试用例通过率达到95％以上，即可允许正常测试结東。该准则的关键在于测试用例质量的把握。

（3）基于缺陷收敛趙势及缺陷修复率原则  
可以通过软件缺陷的趋势图的走向，来定测试是否可以结束；缺陷修复率也是常用的一个指标，如严  
重级别错误和主要级别错误要100%修复，较小缺陷修复率达85％以上。

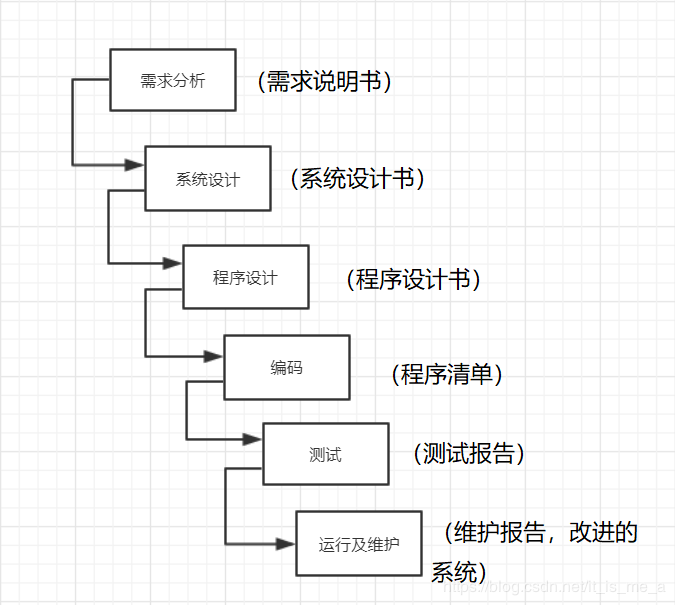
（4）基于覆盖率的原则  
如需求覆盖率达100％，测试用例执行覆盖率达100％，单元测试中语句覆盖率不低于85％等这些准则在软件测试活动中都是比较常用的。

（5）基于验收测试的原则  
即项目通过验收测试，并得到验收测试通过结论，即可结束该项目的测试话动。

（6）软件项目暫停或终止，则测试活动也相应暫停或终止  
如在开发生命周期内出现重大估算、进度偏姜，需要暫停调整或者终止项目，那么测试话动也随之暂停或终止，并备份相应测试数据。

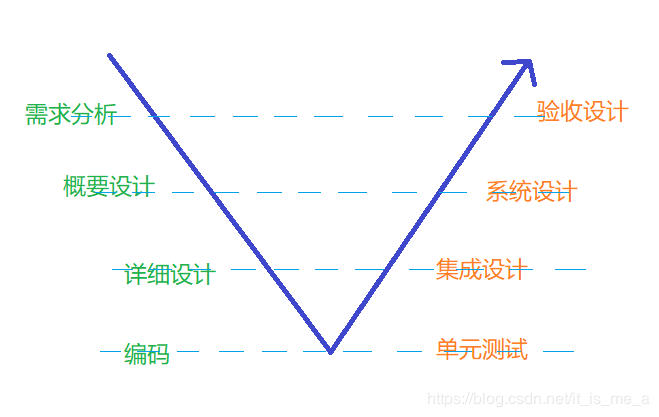
**常见软件模型**

**瀑布模型**

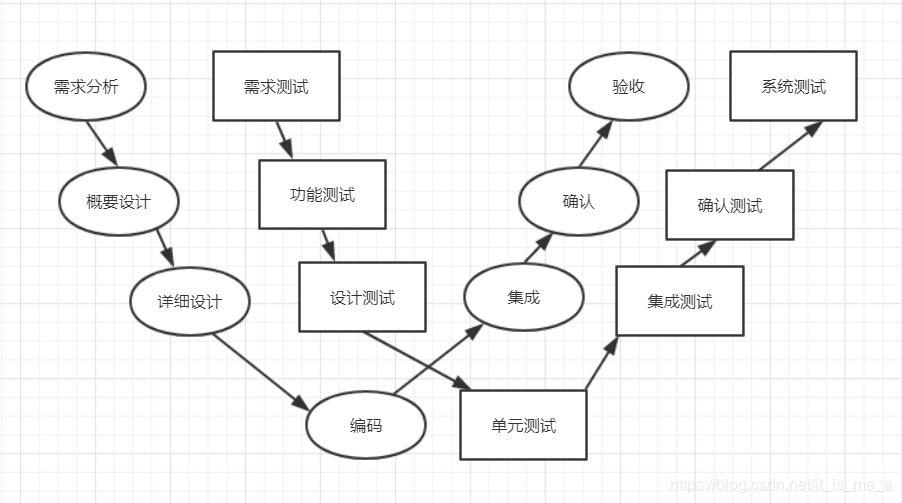
  
使用场景：

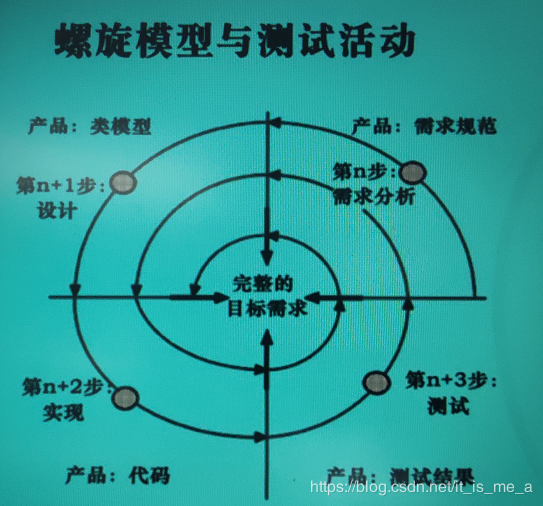
* 需求分析做得比较好的系统
* 二次开发的系统

**V模型与测试活动**

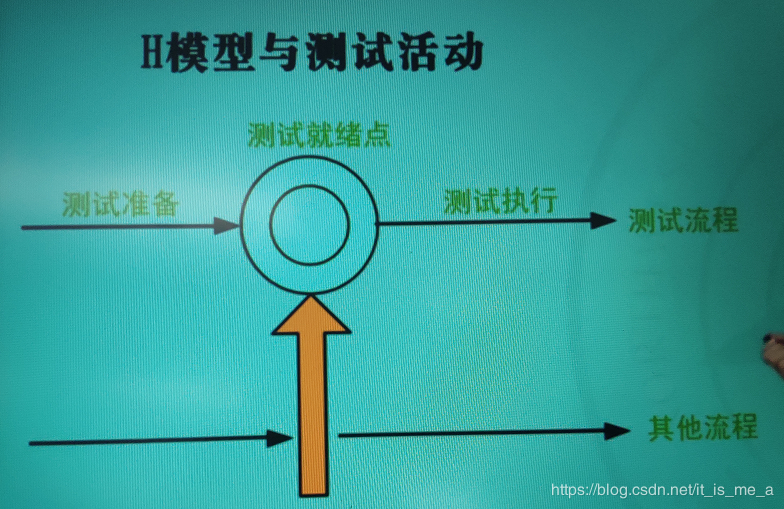


**W模型与测试活动**



**螺旋模型与测试活动**  


**H模型与测试活动**

  
**快速原型模型**

使用场景：

* 开发者在不了解的应用领域开发
* 客户不清楚其开发软件项目的最终目标