## 6.2 软件测试基础

### 1、什么是软件测试?

简单地说，软件测试就是为了发现错误而执行程序的过程。

目前没有统一的定义，三种代表性描述。



- **定义1(1983年IEEE给出的软件测试)**:

“使用**人工或自动手段**来**运行或测定**某个系统的过程，其**目的**在于检验它**是否满足规定的需求**或是弄清**预期结果与实际结果之间的差别**”。

该定义强调测试的目的是软件**是否满足规定的需求**。



- **定义2**:

软件测试是根据软件开发各阶段的**规格说明和程序的内部结构**而精心设计一批**测试用例**，并利用这些测试用例去执行程序，以**发现软件故障的过程**。

该定义强调寻找故障是测试的目的，侧重于技术角度。



- **定义3**:

**软件测试**是一种**软件质量保证活动**，其动机是通过一些经济有效的方法，发现软件中存在的缺陷，从而**保证软件质量**。

强调软件测试是一种**软件质量保证手段**。侧重于管理的角度。



### 2、软件测试的目的

- 基于不同的立场，存在着两种不同的测试目的。
  - ✓用户的角度（测试人员）：普遍希望通过软件测试暴露软件中隐藏的错误和缺陷，以考虑是否可接受该产品。
  - ✓软件开发者的角度：希望测试成为表明软件产品中不存在错误的过程，验证该软件已正确地实现了用户的要求。



### Myers软件测试目的

- Myers在其名著“The Art of Software Testing”中强调：
  - a. 测试是程序的执行过程，目的在于发现错误；
  - b. 一个好的测试用例在于能发现至今未发现的错误；
  - c. 一个成功的测试是发现了至今未发现的错误的测试。

- 正确认识测试的目标是十分重要的：
  - ✓测试目标决定了测试方案的设计。
  - ✓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(“Dijstra”)。
- 以最少的人力、物力和时间找出软件中潜在的各种错误和缺陷，通过修正各种错误和缺陷，提高软件质量。
  - ✓发现软件的错误
  - ✓检验软件是否满足需求
  - ✓提高软件的质量



## 3、软件测试的原则

- ✓所有的测试都应当追溯到用户要求
- ✓把“尽早地和不断地进行软件测试”作为座右铭
- ✓应该由独立的第三方从事测试工作
- ✓Pareto原则
- ✓从“小规模”测试开始，并逐步进行“大规模”测试
- ✓测试用例应由输入数据和预期输出结果组成，应包括合理的输入条件和不合理的输入条件
- ✓穷举测试是不可能的



- **Seven Principles of Software Testing**

  **Bertrand Meyer,** ETH Zürich and Eiffel Software
  IEEE Computer, August 2008, pp99-101.

- **软件测试的原则**,中国计算机学会通讯，2009年1月，第5卷 第1期 pp54-57.



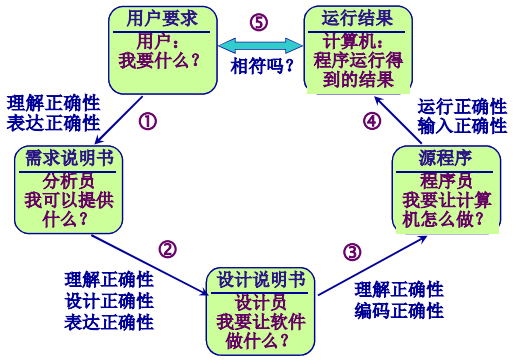
## 4、软件测试的对象

- 软件测试并不等于程序测试。软件测试应贯穿于软件定义与开发的整个期间。
- 需求分析、概要设计、详细设计以及程序编码等各阶段所得到的文档，包括需求规格说明、概要设计规格说明、详细设计规格说明以及源程序，都应成为软件测试的对象。
- 测试的两个方面：
  - ❑缺陷测试
  - ❑V & V（验证和确认）



- 验证(Verification)，检查软件生存期各个阶段过程活动的结果是否满足规格说明的描述，证实各阶段和阶段之间的逻辑协调性、完备性和正确性。
- 确认(Validation)，是比验证更广泛的过程活动。目的是想证实在一个给定的外部环境中软件的逻辑正确性，即是否满足用户的要求。
- Boehm给出两者的区分：
  - ❑验证：我们是否在正确地建造一个产品
  - ❑确认：我们是否在建成一个正确的产品

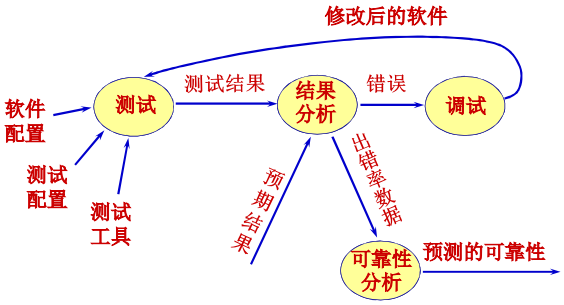




- 在整个生存周期中各个阶段都必须应用V & V技术。两个主要目标是：
  1)发现系统中的缺陷；
  2)判断在给定的操作环境下系统是否可用；
- V & V技术分为两种：
  1)软件检查：对系统的各种表示进行静态分析，以发现问题。这些检查可以借助工具进行文档和代码分析。
  2)软件测试：通过执行测试数据观察系统的操作特征（行为）是否符合要求。



## 5、测试信息流






- 软件配置：软件需求规格说明、软件设计规格说明、源代码等；
- 测试配置：测试计划、测试用例、测试程序等；
- 测试工具：测试数据自动生成程序、静态分析程序、动态分析程序、测试结果分析程序、以及驱动测试的测试数据库等等。
- 测试结果分析：比较实测结果与预期结果，评价错误是否发生。



- 排错（调试）：对已经发现的错误进行错误定位和确定出错性质，并改正这些错误，同时修改相关的文档。
- 修正后的文档再测试：直到通过测试为止。
- 通过收集和分析测试结果数据，对软件建立可靠性模型
- 利用可靠性分析，评价软件质量。
- 如果测试发现不了错误，可以肯定，测试配置考虑得不够细致充分，错误仍然潜伏在软件中。

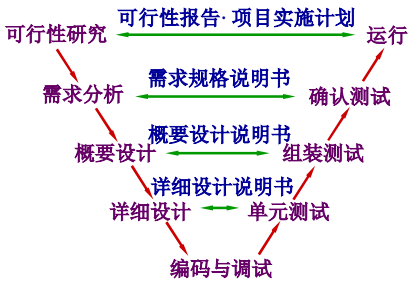


## 6、测试与软件开发各阶段的关系

- 软件开发过程是一个自顶向下，逐步细化的过程
  - ✓软件计划阶段定义软件范围（作用域）
  - ✓软件需求分析阶段建立软件信息域、功能和性能需求、约束等
  - ✓软件设计阶段建立软件体系结构、接口、数据结构和过程设计
  - ✓程序编码阶段把设计用某种程序设计语言转换成程序代码



- 测试过程是依相反顺序安排的自底向上，逐步集成的过程。



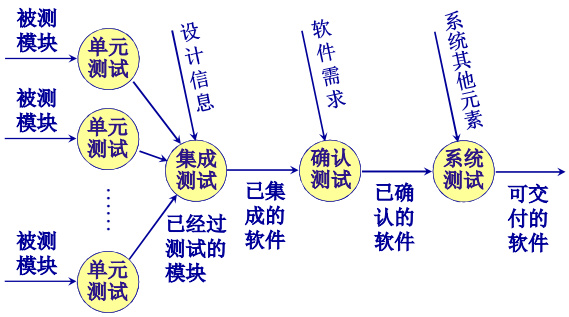
## 6.3 软件测试的阶段与策略

- 测试过程按5个步骤进行:
  - ✓单元测试集中对用源代码实现的每一个程序单元进行测试，检查各个程序模块是否正确地实现了规定的功能。
  - ✓集成测试把已测试过的模块组装起来，主要对与设计相关的软件体系结构的构造进行测试。



- ✓确认测试则是要检查已实现的软件是否满足了需求规格说明中确定了的各种需求，以及软件配置是否完全、正确。
- ✓系统测试把已经经过确认的软件纳入实际运行环境中，与其他系统成份组合在一起进行测试。是否完全、正确。
- ✓验收测试是向用户表明所开发的软件系统能够像用户所预定的那样工作。







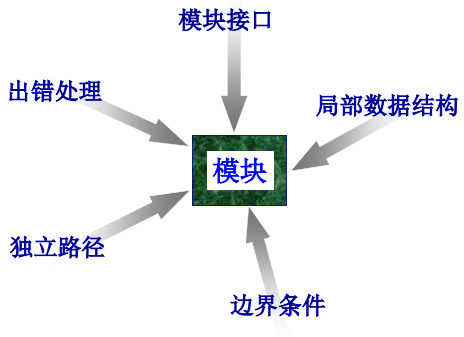

## 6.3.1 单元测试

- 单元测试又称模块测试，是针对软件设计的最小单位—程序模块，进行正确性检验的测试工作。其目的在于发现模块内部可能存在的各种差错。
- 单元测试和编码通常属于软件过程的同一个阶段。
- 单元测试主要使用白盒测试技术（即从程序的内部结构出发设计测试用例），而且对多个模块的测试可以并行地进行。



- 单元测试的内容：
  - ✓依据详细设计说明书和源程序清单，了解该模块的I / O 条件和模块的逻辑结构；
  - ✓主要采用白盒测试的测试用例，辅之以黑盒测试的测试用例，
  - ✓对任何合理的输入和不合理的输入，都能鉴别和响应。





## (1) 模块接口测试

- 在单元测试的开始，应对通过被测模块的数据流进行测试。测试项目包括：
  - ✓ 调用本模块的输入参数(数目、次序、属性等)是否正确
  - ✓ 本模块调用子模块时传递给子模块的参数是否正确
  - ✓ 全局量的定义和用法在各模块中是否一致



- 在做内外存交换时要考虑：
  - ✓ 文件属性是否正确
  - ✓ OPEN与CLOSE语句是否正确
  - ✓ 缓冲区容量与记录长度是否匹配
  - ✓ 在进行读写操作之前是否打开了文件
  - ✓ 在结束文件处理时是否关闭了文件
  - ✓ 正文书写 / 输入错误
  - ✓ I / O错误是否检查并做了处理



## (2) 局部数据结构测试

- ✓ 不正确或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
- ✓ 错误的初始值或错误的缺省值
- ✓ 变量名拼写错或书写错
- ✓ 不一致的数据类型
- ✓ 全局数据对模块的影响



## (3) 路径测试

- 选择适当的测试用例，对模块中重要的执行路径进行测试。
- 应当设计测试用例查找由于错误的计算、不正确的比较或不正常的控制流而导致的错误。
- 对基本执行路径和循环进行测试可以发现大量的路径错误。



## (4) 错误处理测试

- ✓ 出错的描述是否难以理解
- ✓ 出错的描述是否能够对错误定位
- ✓ 显示的错误与实际的错误是否相符
- ✓ 对错误条件的处理正确与否
- ✓ 在对错误进行处理之前，错误条件是否已经引起系统的干预等



## (5) 边界测试

- ✓ 注意数据流、控制流中刚好等于、大于或小于确定的比较值时出错的可能性。对这些地方要仔细地选择测试用例，认真加以测试。
- ✓ 如果对模块运行时间有要求的话，还要专门进行关键路径测试，以确定最坏情况下和平均意义下影响模块运行时间的因素。

## 2. 单元测试的实施

- 单元测试可以采用人工测试或计算机测试实现。
- 人工测试，可以由编写者本人非正式地进行(充当测试者的角色)，也可以由审查小组正式进行，后者称为代码审查。请参见软件项目管理章节的走查和审查。
- 代码审查一种非常有效的程序验证技术。对于典型的程序来说，可以查出30%～70%的逻辑设计错误和编码错误。

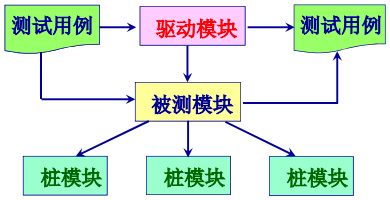


- 实施代码审查时，审查小组最好由下述4人组成：
  - ✓ 组长
  - ✓ 程序的设计者
  - ✓ 程序的编写者
  - ✓ 程序的测试者
- 代码审查的过程如下：
  - ✓ 审查之前，小组成员应该先研究设计说明书；
  - ✓ 审查会上，由程序的编写者解释怎样用程序代码实现这个设计的，通常是逐个语句地讲述程序的逻辑；
  - ✓ 小组其他成员力图发现其中的错误。通常依据程序设计常见错误清单，分析审查这个程序。



- 在采用计算机测试时，由于模块并不是一个独立的程序，因此为了测试某个单元，同时要考虑它和外界的联系，用一些辅助模块去模拟与被测模块相联系的其他模块。
- 驱动模块接收测试数据，把这些数据传送给被测试的模块，并且印出有关的结果。
- 桩模块代替被测模块所调用的模块,使用被它代替的模块的接口，可能做最少量的数据操作，印出对入口的检验或操作结果，并且把控制归还给调用它的模块。






如何编写驱动模块与桩模块？



- 如果一个模块要完成多种功能，可以将这个模块看成由几个小程序组成。必须对其中的每个小程序先进行单元测试，对关键模块还要做性能测试。
- 驱动程序和桩程序是采用计算机测试方法的额外代价。
- 人工审查与计算机测试各有所长，互为补充。
- 模块的内聚程度高可以简化单元测试过程。



## 6.3.2 集成测试

- 集成测试又称组装测试、综合测试。在单元测试的基础上，将所有模块按照设计要求组装成为系统。应考虑如下问题：
  - ➢ 在把各个模块连接起来的时侯，穿越模块接口的数据是否会丢失；
  - ➢ 一个模块的功能是否会对另一个模块的功能产生不利的影响；
  - ➢ 全局数据结构是否有问题；
  - ➢ 各个子功能组合起来，能否达到预期要求的父功能；
  - ➢ 单个模块的误差累积起来，是否会放大，从而达到不能接受的程度。

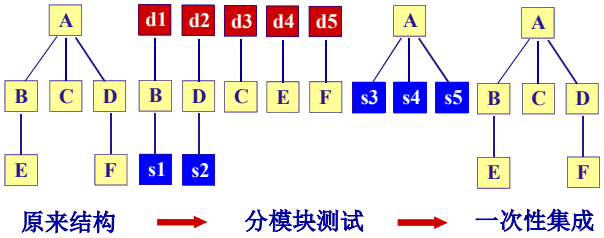
- 在单元测试的同时可进行集成测试，发现并排除在模块连接中可能出现的问题，最终构成要求的软件系统。
- 子系统的集成测试特别称为部件测试，它所做的工作是要找出集成后的子系统与系统需求规格说明之间的不一致。
- 通常，把模块组装成为系统的方式有两种：
  - ✓ 一次性集成方式
  - ✓ 增量式集成方式



## 1. 一次性集成方式

- 它是一种非增量式集成方式。也叫做整体拼装。
- 使用这种方式，首先对每个模块分别进行模块测试，然后再把所有模块集成在一起进行测试，最终得到要求的软件系统。








## 2. 增量式集成方式

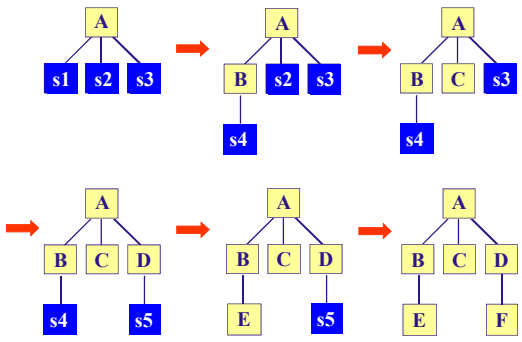
- 又称渐增式组装。首先对一个个模块进行模块测试，然后将这些模块逐步组装成较大的系统；在组装的过程中边连接边测试，以发现连接过程中产生的问题。
- 进行集成测试时普遍采用渐增式测试方法：
  - ✓ 非渐增式集成测试可能遇到若干的错误，改正错误更是极端困难。
  - ✓ 渐增式集成测试容易定位和改正错误；对接口可以进行更彻底的测试；可以使用系统化的测试方法。



## (1) 自顶向下的增量方式

- 这种集成方式将模块按系统程序结构，沿控制层次自顶向下进行组装。
- 自顶向下的增量方式在测试过程中较早地验证了主要的控制和判断点。
  - ✓ 深度优先的策略：可以首先实现和验证一个完整的软件功能；选择主控制通路有很大任意性。
  - ✓ 宽度优先的策略：沿软件结构水平地移动，把处于同一个控制层次上的所有模块组装起来。

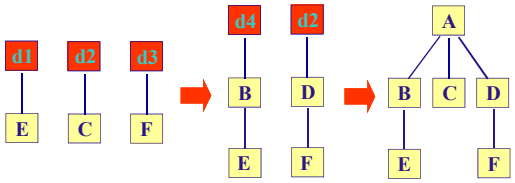




## (2) 自底向上的增量方式

- 这种集成的方式是从程序模块结构的最底层的模块开始集成和测试。
- 因为模块是自底向上进行组装，对于一个给定层次的模块，它的子模块（包括子模块的所有下属模块）已经集成并测试完成，所以不再需要桩模块。在模块的测试过程中需要从子模块得到的信息可以直接运行子模块得到。








## 两种集成测试策略的比较

- 自顶向下测试方法的主要优点是不需要测试驱动程序，能够在测试阶段的早期实现并验证系统的主要功能，而且能在早期发现上层模块的接口错误。
- 自顶向下测试方法的主要缺点是需要桩程序，可能遇到与此相联系的测试困难，低层关键模块中的错误发现较晚，而且用这种方法在早期不能充分展开人力。
- 自底向上测试方法的优缺点与上述自顶向下测试方法的优缺点刚好相反。



## (3) 混合增量式测试

- 改进的自顶向下的增量测试

基本上使用自顶向下的测试方法，但在早期使用自底向上的方法测试软件中的少数关键模块。

✓ 首先对输入 / 输出模块和引入新算法模块进行测试;

✓ 再自底向上组装成为功能相当完整且相对独立的子系统;

✓ 然后由主模块开始自顶向下进行增量测试。



- 自底向上–自顶向下的增量测试

对软件结构中较上层使用的自顶向下方法与对软件结构中较下层使用的自底向上方法相结合。

✓首先对含读操作的子系统自底向上直至根结点模块进行集成和测试;

✓然后对含写操作的子系统做自顶向下的集成与测试。



## 回归测试

- 每当改正软件错误的时候，软件配置的某些成分（程序、文档或数据）也被修改了。如何保证由于调试或其他原因引起的变化，不会导致非预期的软件行为或额外错误？
- 在集成测试过程中每当一个新模块结合进来时，程序就发生了变化，如建立了新的数据流路径、激活了新的控制逻辑等。回归测试是指重新执行已经做过的测试的某个子集，以保证上述这些变化没有带来非预期的副作用。

## 关键模块问题

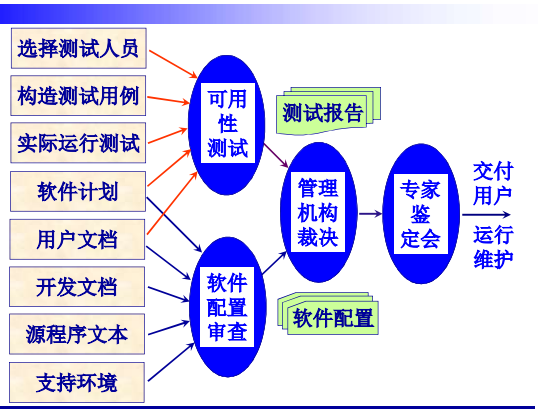
- 在集成测试时，应当确定关键模块，对这些关键模块及早进行测试。
- 关键模块的特征：
  - ✓满足某些软件需求；
  - ✓在程序的模块结构中位于较高的层次（高层控制模块）；
  - ✓较复杂、较易发生错误；
  - ✓有明确定义的性能要求。



## 6.3.3 确认测试

- 确认测试目标是验证软件的有效性。主要使用V&V技术。
- 确认测试的依据是软件需求规格说明书，它明确规定了用户对软件的功能和性能的合理期望。
- 如果软件的功能和性能及其他特性是否与用户的要求一致，则软件是有效的。








## 1. 可用性测试

- 运用黑盒测试的方法，验证被测软件是否满足需求规格说明书列出的需求。
- 首先制定测试计划，规定要做测试的种类。还需要制定一组测试步骤，描述具体的测试用例。
- 确认测试是以用户为主的测试。软件开发人员和质量保证）人员也应参加。由用户参加设计测试用例，使用生产中的实际数据进行测试。



- 通过实施预定的测试计划和测试步骤，确定
  - ✓软件的特性是否与需求相符；
  - ✓所有的文档都是正确且便于使用；
  - ✓对其他软件需求，例如可移植性、兼容性、出错自动恢复、可维护性等，也都要进行测试
- 在全部软件测试的测试用例运行完后，对于测试结果与预期的结果不符合的功能或性能特征，提交一份问题报告。



## 2. 软件配置复查

- 软件配置复查的目的是保证所有按合同或标准必须交付的：
  - ❑ 软件配置的所有成分都齐全；
  - ❑ 各方面的质量都符合要求；
  - ❑ 具有维护阶段所必需的细节；
  - ❑ 而且已经编排好分类的目录。
- 应当严格遵守用户手册和操作手册中规定的使用步骤，以便检查这些文档资料的完整性和正确性。

- 确认测试应交付的文档有：
  - ✓ 确认测试分析报告
  - ✓ 最终的用户手册和操作手册
  - ✓ 项目开发总结报告



## α测试

- α测试是由一个用户在开发环境下进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的测试。
- α测试的目的是评价软件产品的FURPS（即功能、可使用性、可靠性、性能和支持）。尤其注重产品的界面和特色。
- α测试可以从软件产品编码结束之时开始，或在模块测试完成之后开始，也可以在确认测试过程中产品达到一定的稳定和可靠程度之后再开始。



## β测试

- β测试是由软件的多个用户在实际使用环境下进行的测试。测试时，开发者通常不在测试现场，测试是在开发者无法控制的环境下进行的。
- 在β测试中，用户记下遇到的所有问题，并定期向开发者报告。
- β测试主要衡量产品的FURPS。着重于产品的支持性，包括文档、客户培训和支持产品生产能力。
- 只有当α测试达到一定的可靠程度时，才能开始β测试。它处在整个测试的最后阶段。产品的所有手册也应该在此阶段完全定稿。



## 6.3.4 系统测试

- 系统测试：将通过确认测试的软件，作为整个基于计算机系统的一个元素，与计算机硬件、外设、某些支持软件、数据和人员等其他系统元素结合在一起，在实际运行环境下，对计算机系统进行一系列的组装测试和确认测试。
- 系统测试的目的在于:通过与系统的需求定义作比较，发现软件与系统的定义不符合或与之矛盾的地方。



## 6.3.5 验收测试

- 验收测试的目的是向用户表明所开发的软件系统能够像用户所预定的那样工作。
- 主要任务:
  - ✓ 明确规定验收测试通过的标准;
  - ✓ 确定验收测试方法;
  - ✓ 确定验收测试的组织和可利用的资源;
  - ✓ 确定测试结果的分析方法;
  - ✓ 制定验收测试计划并进行评审;
  - ✓ 设计验收测试的测试用例;
  - ✓ 审查验收测试的准备工作;
  - ✓ 执行验收测试;
  - ✓ 分析测试结果，决定是否通过验收



## 测试方案与测试用例

- 测试方案包括具体的测试目的，应该输入的测试数据和预期的结果，其基本目标是：确定一组最可能发现某个错误或某类错误的测试数据。
- 穷尽测试都是不可能的，测试阶段的关键技术问题是测试方案设计。
- 把测试数据和预期的输出结果称为测试用例。
- 设计测试用例是软件测试中最困难的、最核心的问题。不同的测试数据发现程序错误的能力差别很大，为了提高测试效率降低测试成本，选用少量“最有效的”测试数据，做到尽可能完备的测试。