# 软件测试

软件测试，描述一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程。换句话说，软件测试是一种实际输出与预期输出之间的审核或者比较的过程。软件测试的经典定义是：在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程。

目录

[软件测试 1](#_Toc2785647)

[1 测试原则 1](#_Toc2785648)

[2 测试目标 1](#_Toc2785649)

[3 测试对象 1](#_Toc2785650)

[4 测试过程 1](#_Toc2785651)

[5 测试内容 2](#_Toc2785652)

[6 测试方法 2](#_Toc2785653)

[等价类 2](#_Toc2785654)

[边界值 4](#_Toc2785655)

[详细分类 5](#_Toc2785656)

[7 测试流程 7](#_Toc2785657)

[8 测试阶段 7](#_Toc2785658)

[单元测试 7](#_Toc2785659)

[集成测试 7](#_Toc2785660)

[系统测试 8](#_Toc2785661)

[回归测试 8](#_Toc2785662)

## 测试原则

1. 测试应该尽早的进行，最好在需求阶段就开始介入，因为最严重的错误不外乎就是系统不能满足用户的需求
2. 程序员应该避免检查自己的程序，软件测试应该用第三方来负责
3. 设计测试用例时应该考虑到合法的输入和不合法的输入以及各种边界条件，特殊情况下还要制造极端状态和意外状态，如网络异常中断、电源断电等
4. 应该充分注意测试中的群集现象
5. 对错误结果要进行一个确认过程。一般由A测试出来的错误，一定要由B来确认。严重的错误可以召开评审会议进行讨论和分析，对测试结果要进行严格的确认，是否真的存在这个问题以及严重程度等
6. 制定严格的测试计划。一定要制定测试计划，并且要用例有指导性。测试时间安排尽量宽松，不要希望在极短的时间内完成一个高水平的测试
7. 妥善保存测试计划、测试用例、出错统计和最终分析报告，为维护提供方便

## 测试目标

1. 发现一些可以通过测试避免的开发风险
2. 实施测试来降低所发现的风险
3. 确定测试何时可以结束
4. 在开发项目的过程中将测试看作是一个标准项目

## 测试对象

程序。

数据。

文档。

## 测试过程

第一步：对要执行测试的产品/项目进行分析，确定测试策略，制定测试计划。该计划被审核批准后转向第二步。测试工作启动前一定要确定正确的测试策略和指导方针，这些是后期开展工作的基础。只有将本次的测试目标和要求分析清除，才能决定测试资源的投入。

第二步：设计测试用例。设计测试用例要根据测试需求和测试策略来进行，进度压力不大时，应该设计得详细，如果进度、成本压力较大，则应该保证测试用例覆盖到关键性的测试需求。该用例被批准后转向第三步。

第三步：如果满足“启动准则（EntryCriteria）”，那么执行测试。执行测试主要是搭建测试环境，执行测试用例。执行测试时要进行进度控制、项目协调等工作。

第四步：提交缺陷。这里要进行缺陷审核和验证等工作。

第五步：消除软件缺陷。通常情况下，开发经理需要审核缺陷，并进行缺陷分配。程序员要修改自己负责的缺陷。在程序员修改完成后进入到回归测试阶段。如果满足“完成准则（ExitCriteria）”，那么正常结束测试。

第六步：撰写测试报告。对测试进行分析，总结本次的经验教训，在下一次的共走中改进。

软件测试过程管理，主要包括软件测试是什么样的过程，如何评价一个软件测试过程，如何进行配置管理和测试风险分析以及成本的管理。

## 测试内容

软件测试的主要工作内容是验证（verification）和确认（validation），下面分别给出其概念：

**验证**是保证软件正确正确地实现了一些特定功能的一系列活动，即保证软件以正确的方式来做了这个事情（do it right）

1. 确定软件生存周期中的一个给定阶段的产品是否达到前阶段确立的需求的过程。
2. 程序正确性的形式证明，即采用形式理论证明程序符合设计规约规定的过程
3. 评审、审查、测试、检查、审计等各类活动，或对某些项处理、服务或文件等是否和规定的需求相一致进行判断和提出报告。

**确认**是一系列的活动和过程，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。即保证软件做了你所期望的事情（do the right thing）

1. 静态确认，不在计算机上实际执行程序，通过人工或程序分析来证明软件的正确性。
2. 动态确认，通过执行程序做分析，测试程序的动态行为，以证实软件是否存在问题。

软件测试的对象不仅仅是程序测试，软件测试应该包括整个软件开发期间各个阶段所产生的文档，如：需求规格说明书、概要设计文档、详细设计文档，当然软件测试的主要对象还是源程序。

## 测试方法

### 等价类

#### 定义

是把所有可能的输入数据，即程序的输入域划分成若干部分（子集），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。该方法是一种重要的，常用的黑盒测试用例设计方法。

#### 划分等价类

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，并合理地假定：测试某等价类的代表值就等于对这一类其他值的测试，因此，可以把全部输入数据合理划分为若干等价类，在每一个等价类中取一个数据作为测试的输入条件就可以用少量代表性的测试数据取得较好的测试结果。等价类划分可以有两种不同的情况：有效等价类和无效等价类。

##### 有效等价类

是指对于程序的规格说明来说是合理的、有意义的输入数据构成的集合。利用有效等价类可检验程序是否实现了规格说明书中所规定的功能和性能。

##### 无效等价类

与有效等价类的定义恰巧相反。无效等价类指对程序规格说明书是不合理的或无意义的输入数据所构成的集合。对于具体问题，无效等价类至少应该有一个，也可能有多个。

设计测试用例时，要同时考虑这两种等价类。因为软件不仅要能接收合理的数据，也要能经受意外的考验，这样的测试才能确保软件具有更高的可靠性。

#### 划分等价类的标准

1. 完备测试、避免冗余
2. 划分等价类重要的原则是：集合的划分，划分为互不相交的一组子集，而子集的并集是整个集合
3. 并集是整个集合：完备性
4. 子集互不相交，保证一种形式的无冗余性
5. 同一类中标识（选择）一个测试用例，同一等价类中，往往处理相同，相同处理映射到“相同的执行路劲”

#### 划分等价类的方法

1. 在输入条件规定了取值范围或值的个数的情况下，则可以确立一个有效等价类和两个无效等价类。

如：输入值是学生成绩，范围是0~100

1. 在输入条件规定了输入值的集合或者规定了“必须如何”的条件的情况下，可确立一个有效等价类和一个无效等价类

### 边界值

#### 定义

边界值分析法就是对输入或输出的边界值进行测试的一种黑盒测试方法。通常边界值分析法是作为对等价类划分法的补充，这种情况下，其测试用例来自等价类的边界。

#### 与等价划分的区别

1. 边界值分析不是从某等价类中随便挑选一个作为代表，而是使这个等价类的每个边界都要作为测试条件
2. 边界值分析不仅考虑输入条件，还要考虑输出空间产生的测试情况

#### 边界值分析方法的考虑

长期的测试工作经验告诉我们，大量的错误是发生在输入或输出范围的边界上，而不是发生在输入输出范围的内部，因此针对各种边界情况设计测试用例，可以查出更多的错误。

使用边界值分析方法设计测试用例，首先应确定边界情况。通常输入和输出等价类的边界，就是应着重测试的边界情况。应当选取正好等于，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作为测试数据。

#### 常见的边界值

1. 对16bit的整数而言32767和-32768是边界
2. 屏幕上光标在最左上、最右下位置
3. 报表的第一行和最后一行
4. 数组元素的第一个和最后一个
5. 循环的第0次、第一次和倒数第2次，最后一次

#### 边界值分析

1. 边界值分析使用与等价类划分法相同的划分，只是边界值分析假定错误更多的存在于划分的边界上，因此在等价类的边界上以及两侧的情况设计测试用例

例：测试计算平方根的函数

--输入：实数

--输出：实数

--规格说明：当输入一个0或比0大的数的时候，返回其正平方根；当输入一个小于0的数时，显示错误信息“平方根非法-输入值小于0”并返回0

### 详细分类

#### 角度细分

##### 从是否关心软件内部结构和具体实现的角度划分（按测试分类）

1. 白盒测试
2. 黑盒测试
3. 灰盒测试

##### 从是否执行程序的角度

1. 静态测试
2. 动态测试

#### 阶段细分

##### 从软件开发的过程按阶段划分有

1. 单元测试
2. 集成测试
3. 确认测试
4. 系统测试
5. 验收测试
6. 回归测试
7. Alpha测试
8. Beta测试

\*测试过程按4个步骤进行，即单元测试、集成测试、确认测试和系统测试及发布测试

\*开始是单元测试，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书中确定了的各种需求，以及软件配置配置是否完全、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，与其他系统成分组合在一起进行测试

##### 单元测试（Unit Testing）

单元测试又称模块测试，是针对软件设计的最小单元-程序模块，进行正确性检验的测试工作。其目的在于发现各模块内部 可能存在的各种差错。

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。

###### 单元测试的内容

在单元测试时，测试者需要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，使之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是否正确；

-本模块调用子模块时输入给子模块的参数是否正确；

-全局量的定义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

\*出错的描述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。

###### 单元测试的步骤

## 测试流程

1. 制定测试计划
2. 编辑测试用例
3. 执行测试用例
4. 发现并提交bug
5. 开发组修正bug
6. 对以修正bug进行返测
7. 修正完成的bug将状态置为已关闭，未正确修正的bug重新激活

## 测试阶段

### 单元测试

单元测试时对软件组成单元进行测试，其目的是检验软件基本组成单位的正确性，测试的对象是软件设计的最小单位：模块

### 集成测试

集成测试也称联合测试，将程序模块采用适当的集成策略组装起来，对系统的接口及集成后的功能进行正确性检测的测试工作。其主要目的是检查软件单位之间的接口是否正确，集成测试的对象是已经经过单元测试的模块

### 系统测试

系统测试主要包括功能测试、界面测试、可靠性测试、易用性测试、性能测试。功能测试主要针对包括功能可用性、功能实现程度（功能流程&业务流程、数据处理&业务数据处理）方面测试。

### 回归测试

回归测试指在软件维护阶段，为了检查代码修改而引入的错误所进行的测试活动。回归测试时软件维护阶段的重要工作，有研究表明，回归测试带来的耗费占软件生命周期的1/3总费用以上。

与普通的测试不同，在回归测试过程开始的时候，测试者有一个完整的测试用例集可供使用，因此，如何根据代码的修改情况对已有测试用例集进行有效的复用是回归测试研究的重要方向，此外，回归测试的研究方向还涉及自动化工具，面向对象回归测试，测试用例优先级，回归测试用例补充生成等。