# 第8章 软件测试

在软件开发过程中以及软件开发完成后，如何验证软件满足了要求，不存在缺陷呢？由于软件系统复杂性和对其它软件、硬件的依赖性，没有办法通过数学证明或者别的技术手段来回答这个问题。在实际软件开发项目中，软件测试是一个不可缺少的环节，它通过将实际输出与预期输出进行审核或者比较，来揭示软件中存在的缺陷，以便开发者进行改进。由于不可能执行所有的情况，因此我们是通过设计一些测试用例希望它们能够揭露尽可能多的软件中存在的缺陷。同时，由于测试的时间和费用有限，我们也需要认真规划测试过程，使测试达到的效果最好。

另一方面，广义的测试活动不是软件开发后续过程中的一个阶段，测试的对象也不仅是程序本身。测试活动应贯穿于软件开发的整个过程，只有这样，才能更有效率地的开发出有质量保障的优质软件系统。尽管测试工作贯穿了软件开发的全过程，但是我们所称呼的“测试阶段”一般都是发生在软件开发生命周期的末期。

## 8.1 软件测试的主要内容

为了使软件测试能够真正发挥作用，并与整个开发过程相配合，软件测试工作必须要通过制定测试计划、设计测试、测试准备、执行测试、评估测试几个阶段来完成。

### 8.1.1 测试计划的制定

测试计划是对测试的对象、测试中需要的资源、测试的时间安排等进行规划的过程。通过确定任务、识别和分析风险、安排资源和确定进度，并以文档的方式记录下来。当然，测试计划也不是静态的，它可能在早期作为整个开发计划的一部分，进行较为粗略的制订，而后随着项目的进行，不断完善，并在执行过程中进行动态的调整。一般来说，测试计划应该包含以下几个方面。

（1）测试范围，也就是测试对象的界定

（2）风险的确定

（3）资源的规划

（4）时间进度的制定

### 8.1.2 测试用例和测试流程的设计

测试开始前需要设计测试用例，以及具体的测试流程。测试的设计需要依据测试需求进行。

1．测试需求分析

对测试需求进行分析时，需要对需求进行分解，察看各种文档，和用户进行交流。测试需求分析中考虑的问题如下：

（1）确定软件的主要用例

（2）对每一个用例，确定其输入、输出、限制以及要达到的性能指标，形成测试需求

（3）对每一个测试需求，判断其属于的测试类型（功能、性能、安全测试等）。可以构造测试跟踪需求矩阵，针对每一个用例，列出其多项测试需求以及相应的测试类型

（4）对于整个软件，考虑是否需要进行以下的测试：

* 整个系统的性能测试
* 整个系统的安全性测试
* 整个系统的兼容性测试
* 整个系统的容量测试
* 系统的界面测试
* 系统的安装测试

2．测试用例的设计

测试用例指对一项特定的测试任务的描述，体现了测试方案、技术、策略和具体的方式。值得提出的是，测试用例都是从数量极大的可用测试方案中精心挑选出的，能够最大限度发现软件缺陷的方案。由于测试用例的执行需要耗费时间，因此，过多的测试用例可能会需要过长的时间，然而，过少的测试用例可能会带来软件中潜在的、未发现的缺陷数量的过大，因此，如何平衡这两方面的要求，进行合理的测试用例的选择，一直是软件测试工作的重点和难点。

评价测试用例的好坏有以下两个标准：① 是否可以发现尚未发现的软件缺陷？② 是否可以覆盖全部的测试需求？

### 8.1.3 测试的准备

测试的准备是指准备测试环境、准备测试数据、准备测试脚本和测试辅助代码等过程。

1．准备测试环境

（1）测试规程准备

（2）技术准备

（3）配置软件、硬件环境，包括测试工具

（4）人员

2．测试数据准备

测试数据中包括测试输入以及测试输出，它具有两种情形：

（1）正常事务的测试数据

（2）引发异常的测试数据

3．测试脚本的准备和测试辅助代码的编写

在测试过程中，有些时候也需要进行代码的编写工作。

在集成测试中，需要根据测试策略编写驱动模块和桩模块。驱动模块用以模拟被测模块的上级模块，驱动模块接受测试数据，把相关的数据传送给被测模块，启动被测模块，并接收和记录相应的返回数据。而桩模块模拟被测模块工作过程中所调用的模块，一般只进行很少的数据处理。

随着自动化测试工具的流行，我们需要编写测试脚本，从而可以使测试能够自动进行。许多工具中提供了通过“录制回放”的方式生成自动化测试脚本的方法，但是它不够灵活。因此，测试人员可以利用脚本语言自行编写测试程序。

### 8.1.4 执行测试

执行测试是按照计划执行所有的测试用例，并观察其测试结果的过程。在执行测试过程中，应该按照模版填写相应的信息。如果采用了自动测试工具，那么需要能够熟练使用这些工具完成测试过程。

### 8.1.5 测试评估

当测试完成后，需要对测试工作进行一个总结。总结主要从两个方面进行，一个是覆盖性，一个是测试质量。

1．测试的覆盖性

覆盖性回答的是“测试的完全程度如何”这一问题。覆盖性可以从两个角度来看待，一个是从需求的角度，一个是从代码的角度。从需求的角度，就是要求每个需求是否满足都被测试到，从代码角度，就是要求按照一定的标准，所有的代码都被测试到。

2．测试的质量

测试质量评估的是测试过程发现缺陷的能力。

## 8.2 测试类型

从不同的角度，测试可以分为不同的类型。我们将按照测试阶段和测试

### 8.2.1按照测试阶段的划分

从测试发生的阶段看，可以分为：需求分析审查、设计审查、代码审查、单元测试、集成测试和系统测试等。

需求分析审查、设计审查、单元测试中的代码评审及各阶段测试用例的评审都是通过对相关文档或代码的走读活动来实现的。虽然这种方法比较简单，但是实践证明，如果安排合理，能够以较低的成本发现大部分的缺陷。通过检查方式而非执行代码来发现缺陷的测试称为静态测试。单元测试、集成测试及系统测试等通过运行软件来检验软件的动态行为和运行结果正确性的测试方法称之为动态测试。

1．需求分析审查

　　需求规格说明书是系统设计、系统测试等的基础，也是整个项目的规划、验收的基础。因此，对其进行检查，确定其中存在的缺陷对保障软件项目的成功具有突出的意义。软件需求规格说明书的评审是由评审专家组进行的，主要看其是否尽可能完整地、无二义地描述了功能和性能需求，以及在实现上是否具有可行性。评审时经常以会议方式进行，通过将缺陷确认、整理、修改到评审专家组认可。

2．设计审查

　　对软件设计的审查是通过评审专家对设计文档进行预审后，在评审会议上与设计人员将问题一一确认来实现。

　　评审专家依据需求规格说明书审查设计是否覆盖到用户提出的每个用例，对子系统的划分、类的确定、类中属性和方法的确定、数据访问方式、安全性保障、性能保障等进行详细的审查。

3．代码审查

　　代码审查是通过代码走读的方式来实现的。代码走读是开发人员在对某个模块的代码（必须编译通过）完成编码后，进行的代码评审活动。代码走读前依据企业制订的统一标准，按照统一的流程进行。代码审查可以在不同的层面进行，单元审查是对一个方法或一个类，查找代码层面的错误；集成审查关注接口和流程，包括传入的参数检查、返回值检查及流程能否顺利地进行和正确集成等；第三个层面可称之为系统审查，关注功能层面和业务逻辑。

4．单元测试

　　单元测试是对软件中的基本组成单位进行测试，检验其函数的正确性（包括功能正常，输出正确）。

　　一般来说，单元测试用例的编写最早可以在设计评审完成后就启动。单元测试用例编写的目的是覆盖性，覆盖的方法有：语句覆盖、分支覆盖、条件覆盖、条件组合覆盖和路径覆盖等。为了以最少的资源做最多的测试检查，首选路径覆盖的方法。路径覆盖是设计足够的测试用例，运行所测程序并覆盖程序中所有可能的路径。

5．集成测试

　　集成测试是软件系统在集成过程中所进行的测试。其主要目的是检查软件单位之间的接口是否正确。

　　集成测试用例的编写要紧扣与程序相关的各个接口，使每类接口的数据流或控制流均通过接口，从而实现接口测试的完全性。注意：对同一数据流要分别进行正确数据流与错误数据流的用例设计，对边界值的输入最好有单独的用例。集成测试还应关注接口的性能问题，根据系统的性能需求还要设计相关的接口性能测试用例。集成测试的执行主要是借助测试工具——桩程序来实现。桩程序的编写最好由他人来完成，以防止一个人对接口定义理解有偏差而使意外发生。

6．系统测试

　　系统测试是对已经集成好的软件系统进行彻底的测试，以验证软件系统的正确性以及性能等是否满足各系统的需要。除了验证系统的功能外，还需要验证系统的容错能力、错误恢复能力、安全性、性能等。

7. 验收测试

验收测试旨在向软件的购买者展示该软件系统满足其用户的需求。它的测试数据通常是系统测试的测试数据的子集。所不同的是，验收测试常常有软件系统的购买者代表在现场，甚至是在软件安装使用的现场。

除了上述按照不同阶段进行划分的测试类型外，还有几种测试也会安排，它包括：

1. 回归测试

回归测试是在软件维护阶段，对软件进行修改之后进行的测试。其目的是检验对软件进行的修改是否正确，以及是否引发了其他的错误。为了保证软件修改后没有引发新的错误，原先的测试用例都需要再次执行。

1. Alpha 测试

这是在系统开发接近完成时对应用系统的测试，它是由最终用户或其他人员在开发人员准备好的环境中进行的测试。

1. Beta 测试

当开发和测试完成、准备发布前，将软件部署到用户的环境时中所做的测试，。这种测试一般由最终用户或其他人员完成

### 8.2.2按测试手段的划分

1. 白盒测试

白盒测试是指基于代码的内部结构信息，采用某种覆盖准则，即使覆盖全部代码、分支、路径或者条件的测试。白盒测试可以检验程序中的每条通路是否都能按预定要求正确工作。白盒测试可用于单元测试、集成测试中。

2 黑盒测试

黑盒测试是指不基于内部设计和代码的任何知识，而基于需求和功能性进行的测试。黑盒测试在测试时，把程序看作一个不能打开的黑盆子，在完全不考虑程序内部结构和内部特性的情况下，测试者在程序接口进行测试，它检查程序功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数锯而产生正确的输出信息，并且保持外部信息（如数据库或文件）的完整性。黑盒测试方法主要有等价类划分、边值分析、因—果图、错误推测等。黑盒测试可以用于系统测试、验收测试中。

## 8.3 软件测试工具

为了提高软件测试的速度和质量，目前已经开发出了不少有效的软件测试工具，为软件测试提供了帮助。这些工具包括：

1. 测试管理工具

这些工具可以帮助完成测试计划、跟踪测试运行结果，还包括了有助于需求、设计、编码测试及缺陷跟踪的工具。这些工具如Quality Center, TestManager，QACenter, TestLodge等。

1. 静态分析工具

这种工具直接分析代码来检测某些缺陷，它比用其它方法更有效，开销也更小。这类工具包括Purify，Sprint，Checkstyle，Jtest等。

1. 覆盖率工具

这种工具评估通过一系列测试后，软件被执行的程度，如PureCoverage、TrueCoverage、Logiscope等。

1. 动态分析工具

这些工具评估正在运行的系统的性能，例如，检查系统运行过程中的内存使用情况，是否有内存越界、内存泄露等等，这类工具有Purify、BoundChecker等。

1. 测试执行工具

这类工具可使测试能够自动化进行，并且支持各个层次（单元测试、集成测试、系统测试）上的自动测试。例如系统测试阶段有功能测试自动化工具，如Robot、Winrunner、SilkTest等；性能测试工具如Loadrunner、SilKPerformer等。

1. 白盒测试工具

静态分析工具、覆盖率工具和动态分析工具能够支持白盒测试。

7. 黑盒测试工具

主要有：

（1）客户端功能测试：MI公司的winrunner，compuware的qarun, Rational的robot；

（2）服务器端压力性能测试：MI公司的winload,compuware的qaload, Rational的SQAload等等；

（3）Web测试工具：MI公司的Astra系列，rsw公司的e-testsuite；

（4）测试管理工具：Rational的testmanager, compuware的qadirector等

（5）缺陷跟踪工具：如trackrecord，Testtrack

8. 单元测试框架

单元测试框架允许定义单元测试代码，控制测试的执行，还提供了应用程序来运行测试，并在成功完成测试套件中的每个测试后给出报告。

针对不同的编程语言，目前出现了不同的单元测试工具，例如JUnit是Java语言的单元测试框架，NUnit是.net单元测试框架，cppunit是C++的单元测试框架。

## 8.3 测试阶段交付物

### 8.3.1 软件测试计划

软件测试计划是描述测试目的、范围、方法和[软件测试](http://baike.baidu.com/view/16563.htm)的重点等的[文档](http://baike.baidu.com/view/55621.htm)。软件测试计划作为[软件项目计划](http://baike.baidu.com/view/2235776.htm)的子计划，在项目启动初期就必须规划。

[软件测试](http://baike.baidu.com/view/16563.htm)计划是指导测试过程的纲领性文件，包含了产品概述、测试策略、测试方法、测试区域、测试配置、测试周期、测试资源、测试交流、风险分析等内容。借助软件测试计划，参与测试的项目成员，尤其是测试管理人员，可以明确测试任务和测试方法，保持测试实施过程的顺畅沟通，跟踪和控制测试进度，应对[测试过程](http://baike.baidu.com/view/651488.htm)中的各种变更。

1. **测试策略**
   1. 整体测试策略

对整个测试的流程安排，参加的人员，预期达到的结果进行总体概要说明。

1.2进入准则

说明测试活动启动需要满足的进入准则，即开始执行本测计划之前必须完成的各项工作，包括集成/系统测试开始前需要进行的产品构建等。

1.3 暂停/退出准则

暂停准则说明测试异常中止的触发条件，一般为发现严重的妨碍测试继续进行的错误。

退出准则作为测试活动完成与否的判据，应当明确的予以说明。

1. **测试范围和测试方法**

本节中将围绕软件的功能需求、性能需求、接口需求等各种需求确定测试需求。

* 1. **测试的子系统对象**

列出需要测试的子系统，以及不需要测试的子系统。

* 1. **测试需求**

**1）功能需求**

参考软件需求规格说明，针对每一个用例确定测试需求。

（1）用例1

* 列出对应的测试需求，每一条测试需求需要采用一定的格式进行编号，例如，采用“*TR-对应的需求编号－序号*”进行编号。一般一个用例对应多条测试需求；
* 针对该用例的测试需求，使用的测试方法（黑盒、白盒、自动化、手工），如果采用了自动化测试，说明采用的工具。

1. 用例2

……

**2）其他需求**

针对软件需求规格说明中提出的需求，逐条确定是否需要进行测试：

* 对每一条需要测试的需求，简要列出需求项，并针对性地列出测试需求；
* 针对测试需求，列出使用的测试方法（黑盒、白盒、自动化、手工），如果是自动化测试，说明采用的工具。

1. **测试用例**

本节针对测试需求设计测试用例。每一个测试需求，可能对应一个或者多个测试用例。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 需求项 | 测试需求编号 | 测试用例编号 | 测试用例 | |
| RQ-01  <需求项名称> | TR-01-01 | TU-01-01-01 | 名称 |  |
| 测试对象 |  |
| 优先级 |  |
| 输入 |  |
| 输出 |  |
| 步骤 |  |
| 说明 |  |
| TU-01-01-02 | 名称 |  |
| 测试对象 |  |
| 优先级 |  |
| 输入 |  |
| 输出 |  |
| 步骤 |  |
| 说明 |  |
| TR-01-02 |  |  |  |
| RQ-02<需求项名称> |  |  |  |  |

其中：

* 需求项：需求项可以是一个用例，也可以是其他需求。需求项的编号为“

*RQ－序号<需求项名称>*”，如果这一需求是用例，那么需求项名称为用例名，如果是其他需求，简要概括该需求作为需求项名称。

* 测试用例编号：可以采用“*TU-需求编号－测试需求编号－用例编号*”的方式。
* 测试对象：说明测试作用的是整个软件系统、某一软件模块、某个组件还是某个类。
* 优先级：分高，较高，中，较低，低各个等级
* 输入：明确表示输入数据，或者输入文件
* 输出：定义期望获得的正确的输出
* 步骤：分步骤描述测试是如何进行的
* 说明：对使用的工具，或者需要注意的地方加以说明。

1. **测试环境**

**4.1 硬件环境**

描述测试所需求的硬件环境

**4.2 软件环境**

描述测试所需求的软件环境

**4.3 通信环境要求**

描述网络通信等方面的要求

**4.4 安全性环境要求**

描述系统安全方面的要求

**4.5 特定测试环境要求**

如对测试资源有特殊要求，请在此处说明。

1. **测试计划安排**

**5.1 工作量估计**

根据工作内容和项目任务对包括测试设计的工作量、测试执行和测试总结的工作量，以人月或人日计， 并确定测试计划、测试设计、测试准备、测试执行和测试评估工作所占的比重。软件测试工作量应为开发工作量的30%-40%为宜。

|  |  |  |
| --- | --- | --- |
| **工作阶段** | **所需工作日** | **占项目的比例** |
| 测试计划阶段 |  |  |
| 测试设计阶段 |  |  |
| 测试准备阶段 |  |  |
| 测试执行阶段 |  |  |
| 测试评估阶段 |  |  |

**5.2人员需求及安排**

下表列出了在此测试活动的人员安排：

|  |  |  |
| --- | --- | --- |
| **角色** | **人员** | **具体职责/备注** |
| 测试经理 |  |  |
| 测试设计 |  |  |
| 测试人员 |  |  |
|  |  |  |

**5.3 进度安排**

下表列出了测试的时间安排：

|  |  |  |  |
| --- | --- | --- | --- |
| **项目里程碑** | **开始时间** | **结束时间** | **输出要求/备注** |
| 测试计划阶段 |  |  |  |
| 测试设计阶段 |  |  |  |
| 测试准备阶段 |  |  |  |
| 测试执行阶段 |  |  |  |
| 测试评估阶段 |  |  |  |

注：所有结束时间为输出要求通过评审的结束时间。

**5.4 其他资源需求及安排**

描述所需的其他资源的安排。

1. **可交付工件**

本节列出了将要创建的各种文档、工具和报告，及其创建人员、交付对象和交付时间。

1. **风险管理**

本节详细描述本次测试所面临的风险（如人力资源风险、测试技术风险、测试资源风险、质量保证风险等）及相应的建议解决办法。

下表列出了一些风险的分析表。分析表必须按影响的程度来排序。实践中，随着时间的推移和环境的变化，必须随时更新风险的分析。

|  |  |  |
| --- | --- | --- |
| **风险** | **发生的可能性** | **负面影响** |
|  |  |  |
|  |  |  |
|  |  |  |

### 软件测试总结报告

测试总结报告是对测试情况的汇报，在总结报告中，将概要描述测试方法，详细描述测试过程和测试的结果，并对系统给出评价和建议。

1. **测试概要**

在本节中对测试需求和测试方法进行描述。

* 1. 测试需求与测试用例

可以将测试计划中第2、3节的内容复制过来。

* 1. 测试环境与配置

描述具体的测试环境及其配置情况。

* 1. 测试工具

描述实际采用的测试工具。

1. **测试执行情况**

2.1 测试进度情况

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试活动** | **计划起止日期** | **实际起止日期** | **进度偏差** | **备注** |
| 测试计划 |  |  |  |  |
| 测试设计 |  |  |  |  |
| 测试准备 |  |  |  |  |
| 测试执行 |  |  |  |  |
| 测试评估 |  |  |  |  |

**2.2测试人员**

下表列出了在此测试活动的实际的人员安排：

|  |  |  |
| --- | --- | --- |
| **角色** | **人员** | **具体职责/备注** |
| 测试经理 |  |  |
| 测试设计 |  |  |
| 测试人员 |  |  |
|  |  |  |

1. **测试总结**
   1. **测试用例执行结果**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试需求标识号** | **测试用例**  **标识号** | **用例状态** | **测试结果** | **备注** |
| 测试需求编号 | 测试用例的编号 | 已执行还是未执行 | 测试通过还是未通过 |  |
|  |  |  |  |  |
|  |  |  |  |

* 1. **测试问题解决**

**下表中描述测试中发现的、没有满足需求或其它方面要求的部分。**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试需求标识号** | **测试用例标识号** | **错误或问题描述** | **错误或问题状态** |
|  |  |  | 已解决 |

* 1. **测试结果分析**

**3.3.1 覆盖分析**

1. 测试覆盖分析

测试覆盖率＝41/46 ×100％=89.13％

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **需求编号** | **用例个数** | **执行总数** | **未执行** | **未/漏测分析和原因** |
| 需求编号 | 32 | 32 | 0 | 产生失败数为5，最后均以合理的处理方式解决。 |
| … | 1 | 1 | 0 |  |
| … | 4 | 4 | 0 |  |
| … | 6 | 6 | 0 |  |
| … | 1 | 1 | 0 |  |

2. 需求覆盖分析

对应约定的测试文档，本次测试对系统需求的覆盖情况为：

需求覆盖率＝Y（P）项/需求项总数 ×100％＝83.33％

|  |  |  |
| --- | --- | --- |
| **需求项** | **是否通过**[Y][P][N][N/A] | **备注** |
| 需求项编号 | [N] | 缺少完整的系统安装部署、使用、系统卸载的说明。 |
|  |  |  |

P表示部分通过，N/A表示不可测试或者测试用例不适用。

**3.3.2 缺陷分析**

统计发现的Bug总个数，并按照严重程度列出各项Bug数：

* 严重影响系统运行的错误
* 功能方面一般缺陷，影响系统运行
* 不影响运行但必须修改
* 合理化建议（不是Bug，但是可以改进）

按缺陷在各需求项中的分布情况分：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 严重级别  需求 | A-严重影响系统运行的错误 | B-功能方面一般缺陷，影响系统运行 | C-不影响运行但必须修改 | D-合理化建议 | 总计 |
| 需求项 | 1 | 3 | 4 | 2 | 10 |
| … |  |  |  |  | … |
| 总计 | 15 | 48 | 36 | 14 | 113 |

1. **综合评价**

4.1 软件能力

对软件是否满足交付进行确认，并简要描述系统功能。

4.2 缺陷和限制

指出还存在的不足。

4.3 建议

指出系统未来版本可以进行的修改。