# 软件测试的艺术

## 第三章 代码检查、走查与评审

1 代码检查与走查是两种主要的人工测试方法。有很多共同的相似点，一旦发现错误，通常能在代码中对其进行精确定位，这就降低了调试的成本。另外，这个过程通常发现成批的错误，这样错误就可以同时得到修正，而基于计算机的测试通常只能暴露出错误的某个表症。代码检查/走查与基于计算机的测试是互补的。缺少其中一种，错误检查的效率都会降低。

2代码检查时主要进行两项活动，1是由编码人员逐条语句讲述程序的逻辑结构。2对着历来常见的编译错误列表分析程序。除了发现错误这个主要作用之外，代码检查还有几个有益的附带作用。其他参与者可以通过接触其他程序员的错误和编程风格同样受益匪浅，另外代码检查还是早期发现程序中最易出错部分的方法之一，有助于在计算机的测试过程中将更多的注意力集中在这些地方。

3 用于代码检查的错误列表：数据引用错误（初始值，数组界限） 数据声明错误  运算错误 比较错误 控制流程错误 接口错误 输入/输出错误

4代码走查与代码检查很相似，都是以小组为单位进行代码阅读，是一系列规程和错误检查技术的集合。每个测试用例都在人们脑中进行推演，把测试数据沿程序的逻辑结构走一遍，程序的状态记录在纸张或白板上以供监视。

5桌面检查可视为单人进行的代码检查或代码走查，由一个人阅读程序，对照错误列表检查程序，对程序推演测试数据。桌面检查胜过没有检查，但远远逊色于代码见检查和代码走查。

6 代码检查小组：一般由4人组成，一个协调人（不是代码作者）、代码作者、测试专家、其他成员一般是程序的设计人员。

7 同行评审是一种基于程序整体质量、可维护性、可扩展性、应用性和清晰性对匿名程序进行评价的技术。

## 第四章 测试用例的设计

### 黑盒测试：

等价类划分、边界值分析、因果图分析、错误猜测

### 白盒测试：

**逻辑覆盖（语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、多重条件覆盖）**

白盒测试关注的是测试用例执行的程度或覆盖程序逻辑结构的程度。

1 语句覆盖有很大的不足，他通常没什么用处；

2 判定覆盖或分支覆盖式较强一些的逻辑覆盖准则；判定覆盖要求每个判定必须有“是”和“否”的选择，并且每条语句都必须被执行一次；

3 条件覆盖比判定覆盖更强一些，条件覆盖要编写足够的测试用例以保证将一个判定汇中的每个条件的所有可能的结果至少执行一次。

4 判定/条件覆盖将一个判断中的每个条件的所有可能的结果至少执行一次，将每个判断的所有可能的结果至少执行一次，将每个入口点都至少调用一遍。

5 多重条件覆盖，要求编写足够多的测试用例，将每个判定中的所有可能的条件结果的组合，以及所有的入口点都至少执行一次。

### 黑盒测试

**1 等价划分**

确定等价类、生成测试用例

**2 边界值分析**

**3 因果图**

边界值分析和等价划分的一个弱点是未对输入条件的组合进行分析

**4 错误猜测**

## 第五章 模块（单元）测试

### 测试用例设计

使用一种或多种白盒测试方法分析模块的逻辑结构，然后使用黑盒测试方法对照模块的规格说明以补充测试用例。

### 增量测试

非增量式：首先，对每个模块进行单独的模块测试，将每一个模块视为独立实体，这些模块可以同时或者按次序进行测试，最后将这些模块组装或集成为完整的程序。

增量测试：首先将下一个要测试的模块组装到前面已经测试过的模块集合中去。

**注意：（前4条优点，后两点不足）**

非增量测试所需的工作量要多一些

如果使用了增量测试，可以较早地发现模块中与不匹配接口。不正确假设相关的编程错误，

如果使用增量式，调试会进行的容易一些。

增量测试会将测试进行的更彻底

非增量测试所占用的机器时间显得少一些，但是非增量测试需要更多的驱动模块和桩模块

模块测试阶段开始时，如果使用非增量测试，就会有更多的机会进行并行操作

### 自顶向下测试和自底向上测试

**自顶向下的测试**

要成为下一个被测试的模块，调用他的模块必须经过了测试

tips：如果程序中存在关键模块，要将关键模块尽可能早地添加进去；I/O模块应尽可能早地添加进去

此方法缺点：P87 输入模块和被测模块之间有别的模块，有的测试用例不能生成，往输入模块输入什么数据很费脑力，通过输出模块判断被测模块的实际输出也很困难。而且，我们在设计程序结构的较低层次时，可能会对较高层次进行合理的变更或改进，如果程序的较高层次已经完成了编码和测试，那么这些理想的改进就只能被摒弃。

**自底向上的测试**

要成为下一个被测试的模块，他的从属模块必须经过了测试

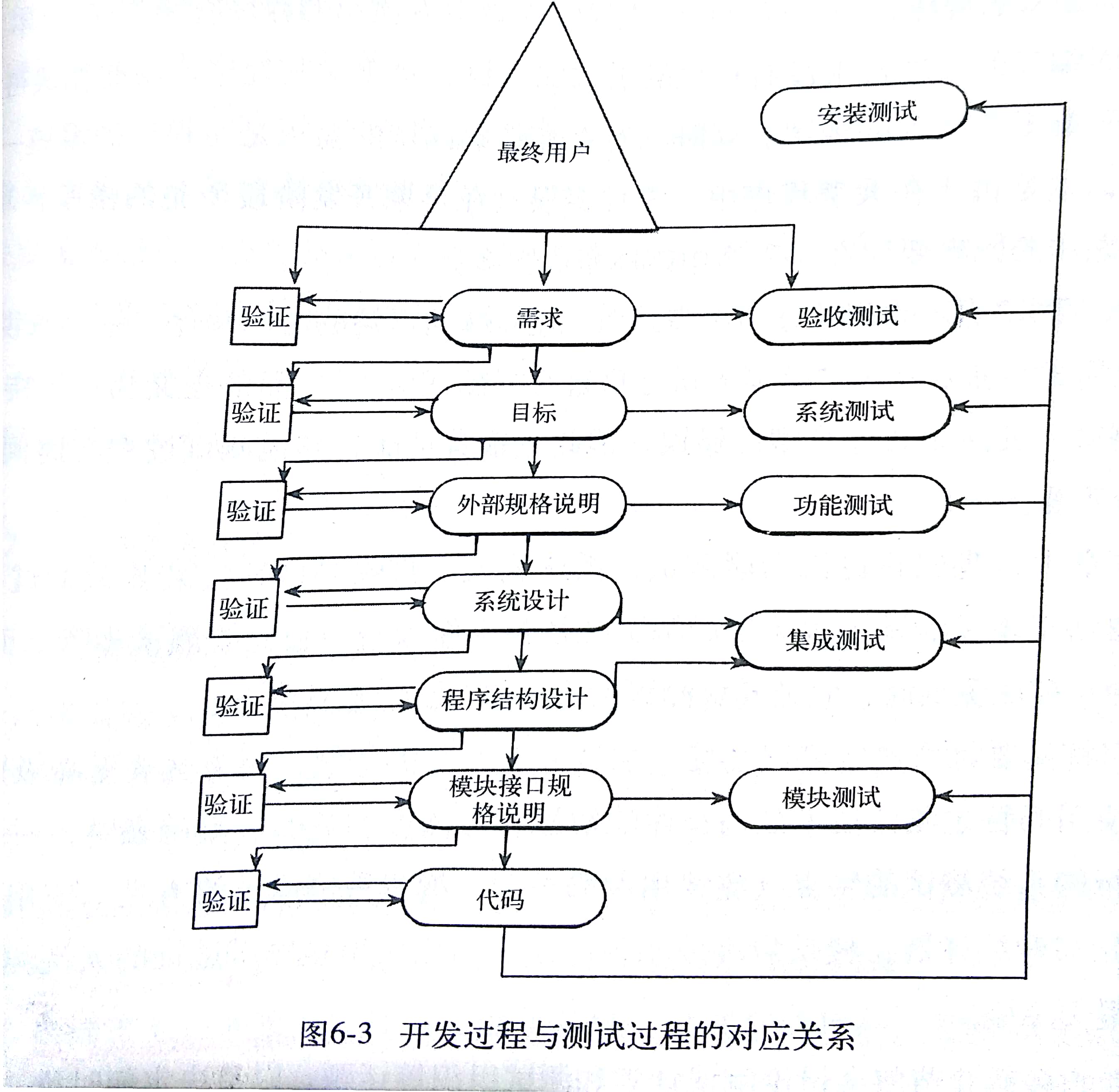
自顶向下测试的优点是自底向上测试的缺点，自顶向下测试的缺点是自底向上测试的优点。

不足：没有早起程序框架的概念，直到最后一个模块被添加进来，才形成了可工作的程序。

### 执行测试

程序员不应测试自己编写的模块，而应交换模块进行测试；编写调用模块的程序员始终是测试被调用模块的最佳候选人

## 第六章 更高级别的测试



模块测试的目的是发现程序模块与其接口规格说明之间的不一致

功能测试的目的是证明程序未能符合外部规格说明

系统测试的目的是为了证明软件产品与其初始目标不一致

集成测试往往并不作为一个独立的测试步骤，而且在进行增量模块测试的时候，他是模块测试的隐含部分。

### 功能测试

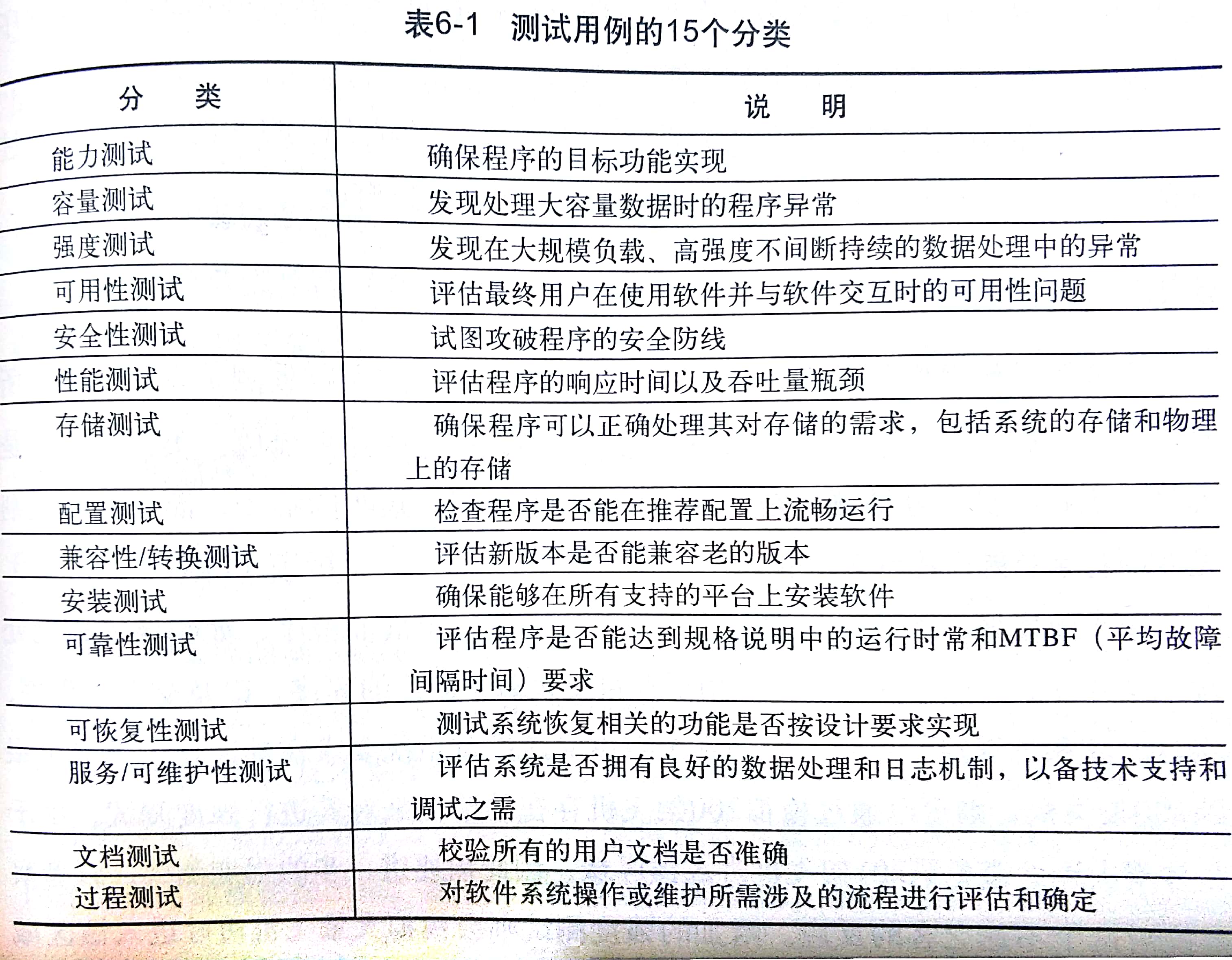
功能测试的目的是证明程序未能符合外部规格说明

功能测试通常是一项**黑盒操作，**要依赖早期的模块测试的过程来实现理想的白盒逻辑覆盖准则。

### 系统测试

系统测试并不是测试整个系统或程序功能的测试，因为有了功能测试，这样就会显得多余，

系统测试的目的是为了证明软件产品与其初始目标不一致



强度测试是在短时间内达到数据或者操作的数量峰值，基于web的应用程序是最常接受强度测试的软件之一。

基于web的应用程序常常比绝大多数程序所需的安全测试级别更高。电子商务网站尤其如此。

性能测试一般描述为特定负载和配置环境下程序的响应时间和吞吐率。

## 第七章 可用性测试

### 需要多少用户进行测试

E=100\*（1-（1-L）N）

E找到错误的比例

N测试人数

L单个测试人员发现的可用性问题比例（如31%）

### 数据采集方法

“发声思考”“眼球追踪”

## 第八章 调试

执行了一个成功的测试用例，发现了一个错误之后。第一步、确定程序中可疑错误的准确性质和位置；第二步、修改错误。

### 暴力法调试

利用内存信息、打印语句、自动化调试工具

### 归纳法调试

确定相关数据、组织数据、做出假设、证明假设、解决问题。

### 演绎法调试

列出所有的可能和假设、利用数据排除可能的原因、提炼剩下的假设、证明剩下的假设、修复问题。

### 回溯法调试

### 测试法调试

使用测试用例进行调试

## 第九章 敏捷开发模式下的测试

敏捷开发提倡迭代式和增量式的开发模型，并强调测试在其中的重要性。

敏捷开发三个共同点：依赖客户的参与、测试驱动以及紧凑的迭代开发周期。

### 敏捷测试

是协同测试的一种形式，要求每一个人都参与到测试计划的设计、实现以及执行中去。

一旦开发者的代码库稳定，客户需要开始验收测试并向开发团队提供反馈。说明测试不是独立的一个阶段而是和开发紧密联系并驱动开发。

先写测试用例和测试配件，然后开始编写可以通过单元测试验证的代码。

### 极限编程（XP）与测试

首先创建单元测试和性能测试，然后才能创建代码库，这种形式的测试成为极限测试。

XP计划：通常需求收集与应用设计结合起来。

XP测试：关键是进行连续的测试，单元测试占主要部分。

**单元测试：先写单元测试用例代码，再写原代码**

**验收测试：验收测试由客户而不是开发人员或者编程搭档执行。**

## 附 牛客错题集

1 负载测试是验证要检验的系统的能力最高能达到什么程度。请判断这句话的正确与否。

错 系统的最高能力是压力测试，而负载测试是在超荷情况下的性能测试

2 设计系统测试计划需要参考的项目文挡有哪些？

【软件需求】是软件开发之前做好的，软件开发是根据这个做的，那么软件测试自然也需要参考该文件 【迭代计划】是软件的某个周期的计划，自然也需要参考 【可行性】可行性研究报告，在软件开发前做好了，在开发前项目经理已召开进行评估，通过后才开始开发此软件，所以在测试过程中，不再需要参考次报告

等值分析测试=等价类划分+边界值分析测试

边界值法既可以用于黑盒测试用例，也可以用于白盒测试用例。

基本路径法用于白盒测试。

正交试验设计法用于黑盒测试。

逻辑覆盖法用于白盒测试用例设计

