第 1 章 测试计划

本章介绍电子商务管理系统的测试计划书的编写。

本章重点：

- “5W1H”规则，明确内容与过程
- 电子商务管理系统的测试计划设计

撰写软件测试计划是软件测试流程中的第一个环节。软件测试的成功与失败是相对的，即要参照测试计划来判断。如果达到测试中预定的管理目标，则可以说软件测试是成功的，否则是失败的。软件测试计划一般是由测试经理来写的，但也不绝对，在小公司里面没有专门的测试部，一个项目组里面只有一个测试人员，就得由测试人员来写。

工作任务 1.1　知识储备

1.1.1　关于软件测试

一、软件测试概述

1. 关于软件测试

软件测试是什么？软件测试就是对项目开发过程的产品（编码、文档等）进行差错审查，保证其质量的一种过程。

软件业的迅猛发展也就是近几十年的过程，时间虽短，但许多误解似乎已根深蒂固，对测试的偏见也是如此。“软件的重点在于需求、在于分析、在于设计、在于开发，而测试容易，没什么技术含量，找一些用户，对照需求尽力去测即可；有时间多测，没时间就少测。”这种看法在许多项目经理、软件负责人的心中固守着，难以改变。

这种观念的结果有目共睹，是什么？很简单，是大量软件 Bug、缺陷的“流失”，从测试人员手中悄然而过，流失到用户手中，流失到项目维护阶段。随之而来的，便是用户无休止的抱怨、维护人员无休止的“救火”、维护成本无休止的增加。这是软件人员的梦魇！噩梦总有醒来时，经过无数教训的重击，在不堪回首而不得不回首的经历中，软件业的管理者发现：是他们错了，软件测试是不可忽视的。

“所有这些问题，假如在项目中测试到，便不会有造成不可收拾的结果了。”——人们终于意识到测试简单而纯真的真谛。

软件测试从直观上来讲是对测试对象进行检查、验证，似乎很简单，但实际不然，它是由许多处理环节构成的。根据测试目标、质量控制的要求，它被划分为以下各类环节，并被设置了不同的准入、准出标准。

（1）软件测试原则

① 尽早和持续不断的测试；

② 彻底完全的测试是不可能的；

③ 软件测试是有风险的行为；

④ 并非所有的软件错误都能修复；

⑤ 反向思维逻辑；

⑥ 由小到大的测试范围；

⑦ 避免测试自己的项目；

⑧ 从用户需求入手。

（2）为什么不能完全测试

① 测试数据输入量太大；

② 输出结果太多；

③ 软件的操作步骤太多；

④ 软件说明书并非“盲人手册”。

（3）并非所有的错误都能修复，Bug 不能被关闭的原因

① 不算真正的软件错误；

② 没有足够的时间；

③ 修复的风险太大；

④ 不值得修复。

（4）错误集中发生现象

① 软件开发人员的疲劳，造成大量代码坏块；

② 程序人员往往会犯同样的错误，因为大部分代码都是复制、粘贴而来的；

③ 软件的基础构架问题，有些软件的底层支撑系统因为“年久失修”变得越来越力不从心了；

④ 发现缺陷的时间越早，Bug 所造成的损失会越小。

（5）避免检查自己的代码的原因

① 程序员从来都不会承认自己写的程序有错误；

② 程序员的测试思路有明显的局限性；

③ 多数程序员没有经过严格正规的职业训练；

④ 程序员无良好的 Bug 跟踪和回归测试经验。

2. 测试过程

正常的测试案例使用方式如图 1-1 所示，测试设计阶段，相关测试设计人员会对测试对象进行了解、分析，为保证测试顺利进行，保证测试覆盖尽量多的测试对象，会设计测试案例、测试方案，在测试期间进行使用；测试发现错误时，软件技术人员会根据测试的缺陷反馈结果及技术人员的软件修改信息对测试程序进行修改，完毕后再进行回归测试。

但是由于软件测试这个行业本身就兴起较晚，现在仍然处于比较不规范且存在很多问题尚未解决的阶段。传统的测试过程，测试管理不严密，测试人员未建立完整的测试库，未将测试案例、测试程序、测试方案进行有效保存，等到回归测试时，相关测试程序等往往已不知去向，无处可寻了；即使能找到这些程序、案例，可往往因为回归测试过于频繁、项目期

限日益迫近，已经没有时间来修改、完善这些程序及案例，只能凭借经验、记忆及技术人员的口述对程序修改过的地方草草重测一遍而已，缺乏正规化的测试过程，造成测试的虎头蛇尾。

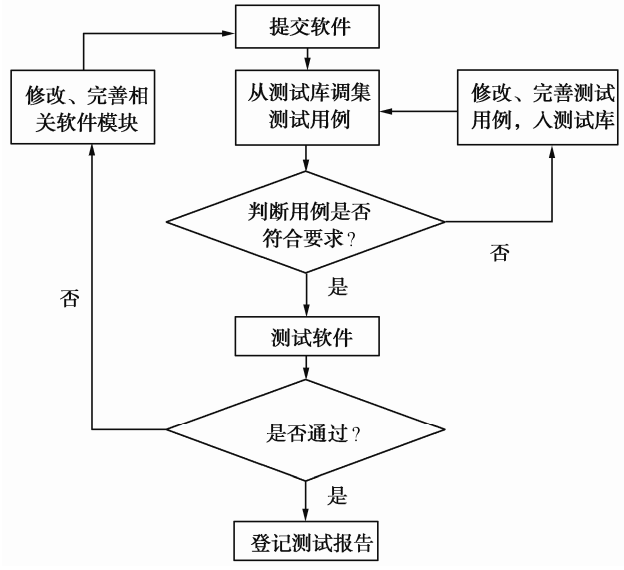


图 1-1　测试流程

二、软件测试概念

通常对软件测试的定义有两种描述：

定义 1：软件测试是为了发现错误而执行程序的过程。

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的一批测试用例，并利用这些测试用例运行程序以及发现错误的过程，即执行测试步骤。

三、软件测试人才的需要

1. 软件测试需求

你们那儿缺什么人？随便咨询 IT 企业的 HR（human resource，人力资源管理。），他们必然仰天长叹一声，百分百地回答：软件测试人员！

（1）企业的需求

几乎每个大中型 IT 企业的软件产品在发布前都需要大量的质量控制、测试和文档工作，而这些工作必须依靠拥有娴熟技术的专业软件人才来完成，软件测试工程师担任的就是这样一个企业重头角色。

（2）许多 IT 企业没有专职的测试机制

软件产品的质量控制与管理越来越受重视，并逐渐成为企业生存与发展的核心。在许多 IT 企业中，软件测试并非只担当“挑错”的角色，没有专职的测试机制。越来越多的 IT 企业已逐渐意识到测试环节在软件产品研发中的重要性。此类软件质量控制工作均需要拥有娴熟技术的专业软件测试人才来协作完成，软件测试工程师作为一个重头角色正成为 IT 企业招聘的热点，其中软件测试工程师成为 IT 就业市场的最新风向标。

（3）软件测试工程师的需求量

由于我国企业对于软件测试自动化技术在整个软件行业中的重要作用认识较晚，因此，软件测试工程师的数量不足，开发和测试人员的比例不合理。据调查，较好的企业中测试人

员和开发人员的比例是 1∶8，有的是 1∶20，甚至没有专职的测试工程师。软件测试专业技术人员在供需之间存在着巨大的缺口。据有关数据显示，我国目前软件从业人才缺口高达 40 万人。即使按照软件开发工程师与测试工程师 1∶5 的岗位比例计算，我国对于软件测试工程师的需求便有数十万之多。业内专家预计，在未来 5～10 年中，我国社会对软件测试人才的需求量还将继续增大。在国展举办的一次招聘会上，多家企业纷纷打出各类高薪招聘软件测试人员的海报，出人意料的是收到的简历尚不足招聘岗位数的 50%，而合格的竟不足 30%。有行业专家表示，软件测试人才供远小于求的现实问题正影响着我国软件业的健康发展。软件测试是一项需具备较强专业技术的工作。在具体工作过程中，测试工程师要利用测试工具按照测试方案和流程对产品进行性能测试。目前已经陷入“有活没人干”的尴尬局面。

（4）网络测试的需求量

包括微软在内的公司对基于网络测试也没有一套完整的体系，仍处于探索中。网络测试是一个全新的、富有挑战性的工作，软件测试工程师的职业之路充满希望。

2. 软件测试工程师未来的发展空间

软件测试工程师未来的职业发展方向如下：

（1）走技术路线，成长为高级软件测试工程师，再向上可以成为软件测试架构设计师。

（2）向管理方向发展，做项目管理。

（3）做开发人员，很容易转去做产品编程。主要软件测试人员有如下四大魅力元素：

① 就业竞争小；

② 高薪没商量；

③ 多元化发展；

④ 无性别歧视。

3. 职位描述

（1）按照测试流程和计划，构建测试环境，设计测试脚本和用例，执行测试脚本和测试用例，寻找 Bug；

（2）分析问题所在并进行准确定位和验证，按照标准格式填写并提交 Bug 报告；

（3）跟踪并验证 Bug，并确认问题得以解决；

（4）按照标准格式填写并提交测试报告，编写其他相关文档；

（5）完成软件开发的集成测试工作。

4. 一则招聘代码测试工程师的招聘广告

职位要求：

（1）熟练操作计算机，计算机基础知识扎实；

（2）熟悉常用的软件测试方法、软件工程知识，熟悉面向对象设计的测试工作；

（3）熟悉常用的软件开发环境，编程工具；

（4）有良好的英语阅读能力，能够阅读英文测试资料；

（5）责任心强，具备良好的沟通能力。

1.1.2 软件测试阶段和软件测试种类

一、软件测试阶段

软件的测试阶段就是测试将按照什么样的思路和方式进行。通常，软件测试要经过单元

测试、集成测试、确认测试、系统测试以及验收测试。这些阶段和开发过程是相对应的。关于测试阶段的叫法有多种，比如测试类型，测试策略等。

1. 单元测试

（1）什么是单元测试

单元测试是针对软件设计的最小单位——程序模块甚至代码段进行正确性检验的测试工作。

（2）什么时候进行单元测试

在程序员编码之后，代码已通过编译后进行单元测试，而且在前期就应该做一些准备工作．如单元测试计划，单元测试用例等。

（3）由谁来进行单元测试

白盒测试工程师或开发人员。

（4）单元测试的依据是什么

源程序本身、《详细设计》文档。

（5）单元测试通过的标准是什么

程序通过所有单元测试的用例；

语句的覆盖率达到 100%；

分支的覆盖率达到 85%。

（6）如何进行单元测试

单元测试主要用白盒测试方法，一般先静态检查代码是否符合规范，然后动态地运行代码，检查其运行结果。一个单元测试的小例子如下：

该程序实现的功能如下，在主函数 main()里面定义了一个含有 5 个整型元素的数组，用一个循环来实现数组元素的输入，每次循环都调用 1 次 iszero 函数，如果输入的数组元素不等于 0，则打印输出本身，如果为 0，则输出 1。

单元测试的一般步骤为编译运行程序（查看能否正确运行）—静态测试（检查代码是否符合规范）—动态测试（深入检查代码的正确性，容错性和边界值等）。

① 编译运行程序。

首先编译程序，没有语法上的错误，编译通过。然后运行程序，输入 12340，按回车键。输出 12341，符合预期结果。

② 静态测试。

检查程序中不符合编码规范的地方。

```
#include<stdio.h>
void  iszero(int m)
{
if(m!=0)
printf("% d",m);
 else
 printf("% d",1);
}
void main(void)
{
int a[5];
int i=0;
printf("请输入 5 个整数\n");
```

```
for(i=0;i<=4;i++)
{
scanf("% d",&a[i]);
iszero(a[i]);
}
}
```

通过静态测试检查可以发现的问题有：函数之间没有空行；低层次语句与高层次语句之间没有缩进；没有注释。

③ 动态测试

边界值问题没有提示输入 1234567，运行结果 12345，结果正确，但输入超过 5 个元素，程序没有提示非法数据的容错性。

（7）附：编码规范

① 一行代码只做一件事情，如只定义一个变量，或只写一条语句，容易阅读和注释；

② 代码行的最大长度值控制在 70～80 个字，否则不便于阅读和打印；

③ 函数与函数之间，定义语句和执行语句之间最好加空行，空行不会浪费内存；

④ 在程序的开头加注释，说明程序的基本信息；在重要的函数模块处加注释，说明各函数的功能；

⑤ 低层次的语句比高层次的语句缩进一个 TAB 键（4 个空格），使程序结构更清晰；

⑥ 不要漏掉函数的参数和返回值，如果没有，则用 void 表示。

2. 集成测试

集成测试是将模块按照设计要求组装起来进行测试，主要目的是发现与接口有关的问题。由于在产品提交到测试部门前，产品开发小组都要进行联合调试，因此在大部分企业中集成测试是由开发人员来完成的。

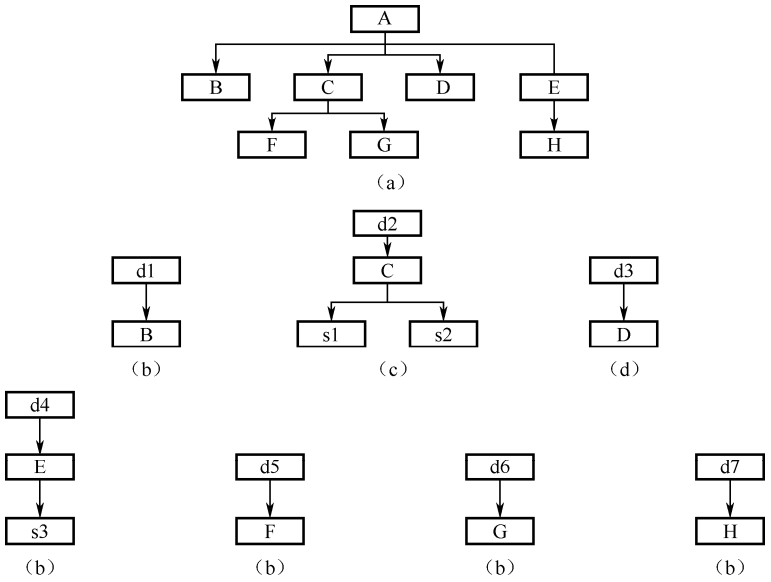
时常有这样的情况发生，每个模块都能单独工作，但这些模块集成在一起之后却不能正常工作。主要原因是，模块相互调用时接口会引入许多新问题。例如，数据经过接口可能丢失；一个模块对另一模块可能造成不应有的影响；几个子功能组合起来不能实现主功能；误差不断积累达到不可接受的程度；全局数据结构出现错误，等等。综合测试是组装软件的系统测试技术，按设计要求把通过单元测试的各个模块组装在一起之后，进行综合测试以便发现与接口有关的各种错误。

某设计人员习惯于把所有模块按设计要求一次全部组装起来，然后进行整体测试，这称为非增量式集成。这种方法容易出现混乱。因为测试时可能发现一大堆错误，为每个错误定位和纠正非常困难，并且在改正一个错误的同时又可能引入新的错误，新旧错误混杂，更难断定出错的原因和位置。与之相反的是增量式集成方法，程序一段一段地扩展，测试的范围一步一步地增大，错误易于定位和纠正，界面的测试亦可做到完全彻底。下面讨论两种增量式集成方法。

（1）集成测试的非增量方式

非增量式测试采用一步到位的方法构造测试。在对所有模块进行测试后，按照程序结构图将各模块连接起来，把连接后的模块当成一个整体进行测试。

实例：对如下程序结构（a），如何进行非增量方式测试，测试过程如图（b）～（h）所示。



（2）增量式测试

增量式集成测试可按照不同次序实施，由此产生了两种不同的方法，即自顶向下结合的方法和自底向上结合的方法。

① 自顶向下结合方法

自顶向下集成是构造程序结构的一种增量式方式，它从主控模块开始，按照软件的控制层次结构，以深度优先或广度优先的策略，逐步把各个模块集成在一起。深度优先策略首先是把主控制路径上的模块集成在一起，至于选择哪一条路径作为主控制路径，这多少带有随意性，一般根据问题的特性确定。以图 1-2 为例，若选择了最左一条路径，首先将模块 M1，M2，M5 和 M8 集成在一起，再将 M6 集成起来，然后考虑中间和右边的路径。广度优先策略则不然，它沿控制层次结构水平地向下移动。仍然以图 1-2 为例，首先把 M2、M3 和 M4 与主控模块集成在一起，再将 M5 和 M6 与其他模块集成起来。

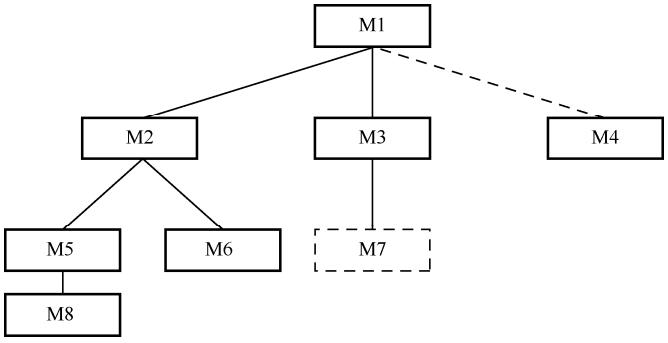


图 1-2　自顶向下集成

由于下文用到桩模块的概念，在此先向读者介绍一下。在集成测试前要为被测模块编制一些模拟其下级模块功能的“替身”模块，以代替被测模块的接口，接受或传递被测模块的数据，这些专供测试用的“假”模块称为被测模块的桩模块。

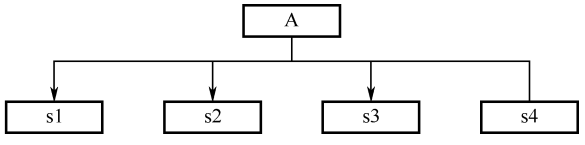
自顶向下综合测试的具体步骤为：

- 以主控模块作为测试驱动模块，把对主控模块进行单元测试时引入的所有桩模块用实际模块替代；
- 依据所选的集成策略（深度优先或广度优先），每次只替代一个桩模块；
- 每集成一个模块立即测试一遍；
- 只有每组测试完成后，才着手替换下一个桩模块；
- 为避免引入新错误，须不断地进行回归测试（即全部或部分地重复已做过的测试）。

从第二步开始，循环执行上述步骤，直至整个程序结构构造完毕。图 1-2 中，实线表示已部分完成的结构，若采用深度优先策略，下一步将用模块 M7 替换桩模块 S7，当然 M7 本身可能又带有桩模块，随后将被对应的实际模块替代。最后直至桩模块 S4 被替代完毕为止。

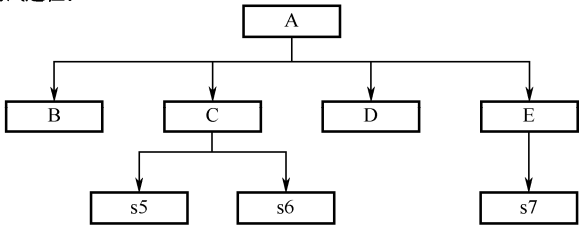
自顶向下集成的优点在于能尽早地对程序的主要控制和决策机制进行检验，因此较早地发现错误。缺点是在测试较高层模块时，低层处理采用桩模块替代，不能反映真实情况，重要数据不能及时回送到上层模块，因此测试并不充分。解决这个问题有几种办法，第一种是把某些测试推迟到用真实模块替代桩模块之后进行，第二种是开发能模拟真实模块的桩模块；第三种是自底向上集成模块。第一种方法又回退为非增量式的集成方法，使错误难于定位和纠正，并且失去了在组装模块时进行一些特定测试的可能性；第二种方法无疑要大大增加开销；第三种方法比较切实可行，下面专门讨论。

例：有如下程序结构图（a），按照自顶向下的方法完成测试活动的过程如下。

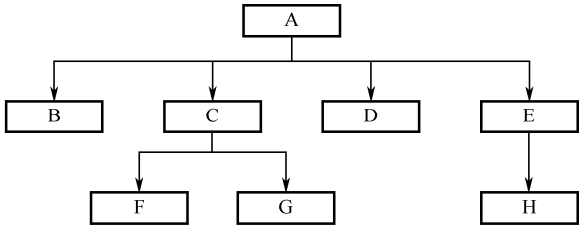


（a）

测试过程：



（b）



（c）

② 自底向上集成

自底向上测试是从“原子”模块（即软件结构最低层的模块）开始组装测试，因测试到较高层模块时，所需的下层模块功能均已具备，所以不再需要桩模块。

自底向上综合测试的步骤分为：

- 把底层模块组织成实现某个子功能的模块群（cluster）；
- 开发一个测试驱动模块，控制测试数据的输入和测试结果的输出；
- 对每个模块群进行测试；
- 删除测试使用的驱动模块，用较高层模块把模块群组织成为完成更大功能的新模块群。

从第一步开始循环执行上述各步骤，直至整个程序构造完毕。

图 1-3 说明了上述过程。首先“原子”模块被分为三个模块群，每个模块群引入一个驱动模块进行测试。因模块群 1、模块群 2 中的模块均隶属于模块 Ma，因此在驱动模块 D1、D2 去掉后，模块群 1 与模块群 2 直接与 Ma 接口，这时可将 D3 去掉，将 Mb 与模块群 3 直接接口，对 Mb 进行集成测试。这样继续下去，直至最后将驱动模块 D4、D5 也去掉，最后 Ma、Mb 和 Mc 全部集成在一起进行测试。

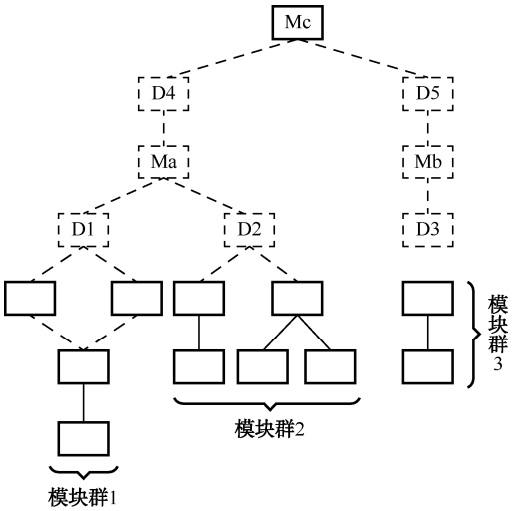
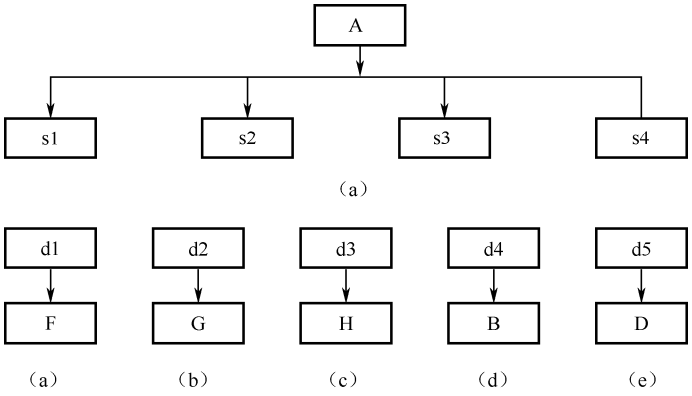
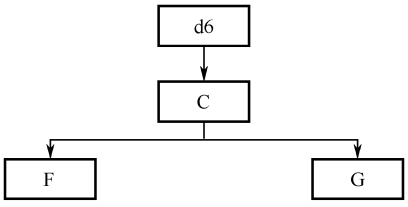


图 1-3　自底向上集成

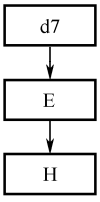
自底向上集成方法不用桩模块，测试用例的设计亦相对简单，但缺点是程序最后一个模块加入时才具有整体形象。它与自顶向下综合测试方法的优缺点正好相反。因此，在测试软件系统时，应根据软件的特点和工程的进度，选用适当的测试策略，有时混和使用两种策略更为有效，上层模块用自顶向下的方法，下层模块用自底向上的方法。

例：有如下程序结构图（a），用自底向上的测试方法进行测试的过程如下：

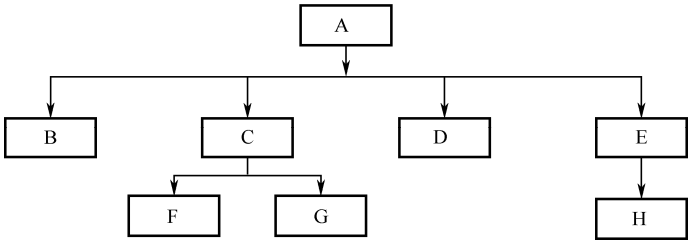




（f）



（g）



（h）

此外，在综合测试中尤其要注意关键模块，所谓关键模块一般都具有下述一个或多个特征：

- 对应几条需求；
- 具有高层控制功能；
- 复杂、易出错；
- 有特殊的性能要求。关键模块应尽早测试，并反复进行回归测试。

3. 系统测试

系统测试是在集成测试通过后进行的，目的是充分运行系统，验证各子系统是否都能正常工作并完成设计的要求。它主要由测试部门进行，是测试部门最大最重要的一个测试，对产品的质量有重大的影响。

计算机软件是基于计算机系统的一个重要组成部分，软件开发完毕后应与系统中其他成分集成在一起，此时需要进行一系列系统集成和确认测试。对这些测试的详细讨论已超出软件工程的范围，这些测试也不可能仅由软件开发人员完成。在系统测试之前，软件工程师应完成下列工作：

- 为测试软件系统的输入信息设计出错处理通路；
- 设计测试用例，模拟错误数据和软件界面可能发生的错误，记录测试结果，为系统测试提供经验和帮助；
- 参与系统测试的规划和设计，保证软件测试的合理性。

系统测试应该由若干个不同测试组成，目的是充分运行系统，验证系统各部件是否都能正常工作并完成所赋予的任务。下面简单讨论几类系统测试。

（1）恢复测试

恢复测试主要检查系统的容错能力。当系统出错时，能否在指定时间间隔内修正错误并

重新启动系统。恢复测试首先要采用各种办法强迫系统失败，然后验证系统是否能尽快恢复。对于自动恢复需验证重新初始化（reinitialization）、检查点（checkpointing mechanisms）、数据恢复（data recovery）和重新启动（restart）等机制的正确性；对于人工干预的恢复系统，还需估测平均修复时间，确定其是否在可接受的范围内。

（2）安全测试

安全测试检查系统对非法侵入的防范能力。安全测试期间，测试人员假扮非法入侵者，采用各种办法试图突破防线。例如，① 想方设法截取或破译口令；② 专门定做软件破坏系统的保护机制；③ 故意导致系统失败，企图趁恢复之机非法进入；④ 试图通过浏览非保密数据，推导所需信息，等等。理论上讲，只要有足够的时间和资源，没有不可进入的系统。因此系统安全设计的准则是，使非法侵入的代价超过被保护信息的价值。此时非法侵入者已无利可图。

（3）强度测试

强度测试检查程序对异常情况的抵抗能力。强度测试总是迫使系统在异常的资源配置下运行。例如，① 当中断的正常频率为每秒一至两个时，运行每秒产生十个中断的测试用例；② 定量地增长数据输入率，检查输入子功能的反应能力；③ 运行需要最大存储空间（或其他资源）的测试用例；④ 运行可能导致操作系统崩溃或磁盘数据剧烈抖动的测试用例，等等。

（4）性能测试

对于那些实时和嵌入式系统，软件部分即使满足功能要求，也未必能够满足性能要求，虽然从单元测试起，每一测试步骤都包含性能测试，但只有当系统真正集成之后，在真实环境中才能全面、可靠地测试运行操作系统，性能测试就是为了完成这一任务。性能测试有时与强度测试相结合，经常需要其他软硬件的配套支持。

4. 验收测试

验收测试（acceptance testing）以需求阶段的《需求规格说明书》为验收标准，测试时要求模拟实际用户的运行环境。对于实际项目可以和客户共同进行，对于产品来说就是最后一次的系统测试。测试内容为对功能模块的全面测试，尤其要进行文档测试。

验收测试是系统开发生命周期方法论的一个阶段，这时相关的用户和/或独立测试人员根据测试计划和结果对系统进行测试和接收。它让系统用户决定是否接收系统。它是一项确定产品是否能够满足合同或用户所规定需求的测试。这是管理性和防御性控制。

验收测试是部署软件之前的最后一个测试操作。验收测试的目的是确保软件准备就绪，并且可以让最终用户将其用于执行软件的既定功能和任务。

验收测试是向未来的用户表明系统能够像预定要求那样工作。经集成测试后，已经按照设计把所有的模块组装成一个完整的软件系统，接口错误也已经基本排除了，接着就应该进一步验证软件的有效性，这就是验收测试的任务，即软件的功能和性能如同用户所期待的那样。

通过综合测试之后，软件已完全组装起来，接口方面的错误也已排除，软件测试的最后一步——确认测试即可开始。确认测试应检查软件能否按合同要求进行工作，即是否满足软件需求说明书中的确认标准。

实现软件确认要通过一系列黑盒测试。确认测试同样需要制订测试计划和过程，测试计划应规定测试的种类和测试进度，测试过程则定义一些特殊的测试用例，旨在说明软件与需求是否一致。无论是计划还是过程，都应该着重考虑软件是否满足合同规定的所有功能和性

能，文档资料是否完整、准确，人机界面和其他方面（例如，可移植性、兼容性、错误恢复能力和可维护性等）是否令用户满意。

确认测试的结果有两种可能，一种是功能和性能指标满足软件需求说明的要求，用户可以接受；另一种是软件不满足软件需求说明的要求，用户无法接受。项目进行到这个阶段才发现严重错误和偏差一般很难在预定的工期内改正，因此必须与用户协商，寻求一个妥善解决问题的方法。

确认测试的另一个重要环节是配置复审。复审的目的在于保证软件配置齐全、分类有序，并且包括软件维护所必须的细节。

事实上，软件开发人员不可能完全预见用户实际使用程序的情况。例如，用户可能错误地理解命令，或提供一些奇怪的数据组合，亦可能对设计者自认明了的输出信息迷惑不解，等等。因此，软件是否真正满足最终用户的要求，应由用户进行一系列“验收测试”。验收测试既可以是非正式的测试，也可以是有计划、有系统的测试。有时，验收测试长达数周甚至数月，不断暴露错误，导致开发延期。一个软件产品，可能拥有众多用户，不可能由每个用户验收，此时多采用称为α、β测试的过程，以便发现那些似乎只有最终用户才能发现的问题。

α测试是指软件开发公司组织内部人员模拟各类用户对即将面市的软件产品（称为α版本）进行测试，试图发现错误并修正。α测试的关键在于尽可能逼真地模拟实际运行环境和用户对软件产品的操作并尽最大努力涵盖所有可能的用户操作方式。经过测试调整的软件产品称为β版本。紧随其后的测试是指软件开发公司组织各方面的典型用户在日常工作中实际使用β版本，并要求用户报告异常情况、提出批评意见。然后软件开发公司再对β版本进行改错和完善。

（1）验收测试过程

过程如下：

① 软件需求分析：了解软件功能和性能要求、软硬件环境要求等，并特别要了解软件的质量要求和验收要求。

② 编制《验收测试计划》和《项目验收准则》：根据软件需求和验收要求编制测试计划，制定需测试的测试项，制定测试策略及验收通过准则，并经过客户参与的计划评审。

③ 测试设计和测试用例设计：根据《验收测试计划》和《项目验收准则》编制测试用例，并经过评审。

④ 测试环境搭建：建立测试的硬件环境、软件环境等（可在委托客户提供的环境中进行测试）。

⑤ 测试实施：测试并记录测试结果。

⑥ 测试结果分析：根据验收通过准则分析测试结果，做出验收是否通过及测试评价。

⑦ 测试报告：根据测试结果编制缺陷报告和验收测试报告，并提交给客户。

（2）验收测试的总体思路

用户验收测试是软件开发结束后，用户对软件产品投入实际应用以前进行的最后一次质量检验活动。它要回答开发的软件产品是否符合预期的各项要求，以及用户能否接受的问题。由于它不只是检验软件某个方面的质量，而是要进行全面的质量检验，并且要决定软件是否合格，因此验收测试是一项严格的正式测试活动。需要根据事先制定的计划，进行软件配置评审、功能测试、性能测试等多方面检测。

用户验收测试可以分为两个大的部分：软件配置审核和可执行程序测试，其大致顺序可分为：文档审核、源代码审核、配置脚本审核、测试程序或脚本审核、可执行程序测试。

要注意的是，在开发方将软件提交用户方进行验收测试之前，必须保证开发方本身已经对软件的各方面进行了足够的正式测试（当然，这里的“足够”，本身是很难准确定量的）。

用户在按照合同接收并清点开发方的提交物时（包括以前已经提交的），要查看开发方提供的各种审核报告和测试报告内容是否齐全，再加上平时对开发方工作情况的了解，基本可以初步判断开发方是否已经进行了足够的正式测试。

用户验收测试的每一个相对独立的部分，都应该有目标（本步骤的目的）、启动标准（着手本步骤必须满足的条件）、活动（构成本步骤的具体活动）、完成标准（完成本步骤要满足的条件）和度量（应该收集的产品与过程数据）。在实际验收测试过程中，收集度量数据，不是一件容易的事情。

① 软件配置审核。对于一个外包的软件项目而言，软件承包方通常要提供如下相关的软件配置内容：

- 可执行程序、源程序、配置脚本、测试程序或脚本。
- 主要的开发类文档：《需求分析说明书》、《概要设计说明书》、《详细设计说明书》、《数据库设计说明书》、《测试计划》、《测试报告》、《程序维护手册》、《程序员开发手册》、《用户操作手册》、《项目总结报告》。
- 主要的管理类文档：《项目计划书》、《质量控制计划》、《配置管理计划》、《用户培训计划》、《质量总结报告》、《评审报告》、《会议记录》、《开发进度月报》。

在开发类文档中，容易被忽视的文档有《程序维护手册》和《程序员开发手册》。

《程序维护手册》的主要内容包括：系统说明（包括程序说明）、操作环境、维护过程、源代码清单等，编写目的是为将来的维护、修改和再次开发工作提供有用的技术信息。

《程序员开发手册》的主要内容包括：系统目标、开发环境使用说明、测试环境使用说明、编码规范及相应的流程等，实际上就是程序员的培训手册。

不同大小的项目，都必须具备上述的文档内容，只是可以根据实际情况进行重新组织。

对上述的提交物，最好在合同中规定阶段提交的时间，以免发生纠纷。

通常，正式的审核过程分为 5 个步骤：计划、预备会议（可选）、准备阶段、审核会议和问题追踪。预备会议是对审核内容进行介绍并讨论。准备阶段就是各责任人事先审核并记录发现的问题。审核会议是最终确定工作产品中包含的错误和缺陷。

审核要达到的基本目标是：根据共同制定的审核表，尽可能地发现被审核内容中存在的问题，并最终得到解决。在根据相应的审核表进行文档审核和源代码审核时，还要注意文档与源代码的一致性。

在实际的验收测试执行过程中，常常会发现文档审核是最难的工作，一方面是由于市场需求等方面的压力使这项工作常常被弱化或推迟，造成持续时间变长，加大文档审核的难度；另一方面，文档审核中不易把握的地方非常多，每个项目都有一些特别的地方，而且也很难找到可用的参考资料。

② 可执行程序的测试。

在文档审核、源代码审核、配置脚本审核、测试程序或脚本审核都顺利完成后，就可以进行验收测试的最后一个步骤——可执行程序的测试，它包括功能、性能等方面的测试，每

种测试也都包括目标、启动标准、活动、完成标准和度量等五部分。

要注意的是不能直接使用开发方提供的可执行程序用于测试，而要按照开发方提供的编译步骤，从源代码重新生成可执行程序。

在真正进行用户验收测试之前一般应该已经完成了以下工作（也可以根据实际情况有选择地采用或增加）：

- 软件开发已经完成，并全部解决了已知的软件缺陷。
- 验收测试计划已经过评审并批准，并且置于文档控制之下。
- 对软件需求说明书的审查已经完成。
- 对概要设计、详细设计的审查已经完成。
- 对所有关键模块的代码审查已经完成。
- 对单元、集成、系统测试计划和报告的审查已经完成。
- 所有的测试脚本已完成，并至少执行过一次，且通过评审。
- 使用配置管理工具且代码置于配置控制之下。
- 软件问题处理流程已经就绪。
- 已经制定、评审并批准验收测试完成标准。

具体的测试内容通常可以包括：安装（升级）、启动与关机、功能测试（正例、重要算法、边界、时序、反例、错误处理）、性能测试（正常的负载、容量变化）、压力测试（临界的负载、容量变化）、配置测试、平台测试、安全性测试、恢复测试（在出现掉电、硬件故障或切换、网络故障等情况时，系统是否能够正常运行）、可靠性测试等。

性能测试和压力测试一般情况下是在一起进行，通常还需要辅助工具的支持。在进行性能测试和压力测试时，测试范围必须限定在那些使用频度高的和时间要求苛刻的软件功能子集中。由于开发方已经事先进行过性能测试和压力测试，因此可以直接使用开发方的辅助工具。也可以通过购买或自己开发来获得辅助工具。具体的测试方法可以参考相关的软件工程书籍。

如果执行了所有的测试案例、测试程序或脚本，用户验收测试中发现的所有软件问题都已解决，而且所有的软件配置均已更新和审核，可以反映出软件在用户验收测试中所发生的变化，用户验收测试就完成了。

尽管测试阶段的划分十分明确，但是在具体的项目和产品的测试中，尤其在执行测试时，会根据实际需要来开展。关于软件测试阶段的理论可以参看朱少民编写的《全程软件测试》一书中软件测试过程和开发过程的 V 模型关系。

测试的各个阶段所用时间比例是不同的，根据该阶段的性质，确定时间长短。当然，具体情况要具体掌握，根据不同项目不同情况，各测试阶段所用时间长短可随时调节。图 1-4 直观地描述出了测试各阶段所用的时间对比。

二、软件测试的种类

对于测试种类的说法很多，最多的能达到几十种测试种类。但是实际工作中很多测试是互相包含的。按照企业中实际工作需要，通常主要进行下面几种类型的测试：功能测试、健壮性测试、接口测试、强度测试、压力测试、性能测试、用户界面测试、可靠性测试、安装/反安装测试、帮助文档测试。

下面介绍几种重要的测试种类及其测试的内容：

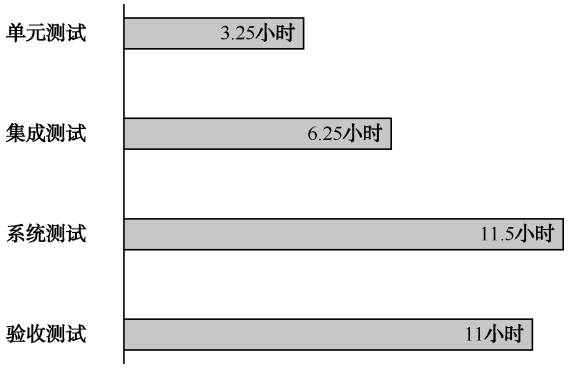


图 1-4　测试各阶段所用的时间

1. 功能测试

功能测试主要针对产品需求说明书的测试，是验证功能是否适合需求，包括原定功能的检验、是否有冗余功能、遗漏功能。这类测试应由测试员做，这并不意味着程序员在发布前不必检查他们的代码能否工作，他们也需要进行基本功能的测试。

2. 接口测试

程序员对各个模块进行系统联调的测试，包含程序内接口和程序外接口测试。这个测试，在单元测试阶段进行了一部分工作，而大部分都是在集成测试阶段完成的。由开发人员进行。

3. 性能测试

在交替进行负荷和强迫测试时常用的术语。性能测试关注的是系统的整体。它和通常所说的强度、压力/负载测试有密切关系。所以压力和强度测试应该与性能测试一同进行。

4. 用户界面测试

对系统的界面进行测试，测试用户界面是否友好、是否方便易用、设计是否合理、位置是否正确等一系列界面问题。

5. 安装/反安装测试

安装测试主要检验软件是否可以正确安装，安装文件的各项设置是否有效，安装后能否影响原系统；反安装是逆过程，测试是否删除干净，是否影响原系统等。

6. 文档测试

主要测试开发过程中针对用户的文档，以需求、用户手册、安装手册等为主，检验文档是否和实际应用存在差别。文档测试不需要编写测试用例。

测试种类的划分不要拘泥于上面的形式，总体来说应该服从于测试策略，可以根据具体工作的特点进行安排，为了工作更容易开展，完全可以把一些测试合并在一起进行。在后面的性能测试用例的编写上，充分体现了这一思想。

综合上面的分析，测试种类、测试阶段以及执行人员具体的关系如表 1-1 所示。

表 1-1　测试的种类、阶段和执行人员的关系

测试阶段	测试类型	执　行　者
单元测试	模块功能测试，包含部分接口测试、路径测试	开发工程师
集成测试	接口测试、路径测试，含部分功能测试	开发工程师（如果测试人员水平较高，可以由测试人员执行）

续表

测试阶段	测试类型	执行者
系统测试	功能测试、健壮性测试、性能测试、用户界面测试、安全性测试、压力测试、可靠性测试、安装/反安装测试	测试工程师
验收测试	对于实际项目来说基本同上，并包含文档测试；对于软件产品，主要测试相关的技术文档	测试工程师（根据实际需要，可能包含用户）

总之，测试的种类应该尽量少，这样每次都可以执行更多的测试内容。例如在进行功能测试的同时，完全可以进行健壮性的测试（当然如果产品健壮性方面要求较高，就可以把健壮性测试作为独立的测试）。也就是说测试各个阶段中对执行不同的测试种类采用不同的测试技术。关于测试技术下面进行讲解。

1.1.3 关于测试计划

软件测试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，应对测试过程中的各种变更。

测试计划和测试详细规格、测试用例之间是战略和战术的关系，测试计划主要从宏观上规划测试活动的范围、方法和资源配置，而测试详细规格、测试用例是完成测试任务的具体 战术。所以其中最重要的是测试策略和测试方法（最好是能先评审）。

做好测试计划工作的关键是什么？

1. 明确测试的目标，增强测试计划的实用性

编写软件测试计划的重要目的就是使测试过程能够发现更多的软件缺陷，因此软件测试计划的价值取决于它对帮助管理测试项目，并且找出软件潜在的缺陷。因此，软件测试计划中的测试范围必须高度覆盖功能需求，测试方法必须切实可行，测试工具具有较高的实用性，便于使用，生成的测试结果直观、准确。

2. 坚持“5W”规则，明确内容与过程

3. 采用评审和更新机制，保证测试计划满足实际需求

测试计划设计完成后，如果没有经过评审，直接发送给测试团队，测试计划可能内容不准确或遗漏测试内容，或者软件需求变更引起测试范围的增减，而测试计划的内容没有及时更新，误导测试执行人员。

4. 分别创建测试计划与测试详细规格、测试用例

应把详细的测试技术指标包含到独立创建的测试详细规格文档中，把用于指导测试小组执行测试过程的测试用例放到独立创建的测试用例文档或测试用例管理数据库中。

一、关于测试计划

俗话说：凡事预则立，不预则废！软件测试同样，在测试项目之初就要制订相应的测试计划。接下来了解一下如何编写测试计划。

1. 为什么要编写测试计划

（1）领导能够根据测试计划做宏观调控，进行相应资源配置等；

（2）测试人员能够了解整个项目测试情况以及项目测试不同阶段所要进行的工作等；

（3）便于其他人员了解测试人员的工作内容，进行有关配合工作。

2. 什么时间开始编写测试计划

软件测试计划在项目启动初期就应该规划。

3. 由谁来编写测试计划

软件测试计划一般由具有丰富经验的项目测试负责人来编写。

4. 测试计划编写 6 要素（5W1H）

（1）Why——为什么要进行这些测试；

（2）What——测试哪些方面，不同阶段的工作内容；

（3）When——测试不同阶段的起止时间；

（4）Where——相应文档，缺陷的存放位置，测试环境等；

（5）Who——项目有关人员组成，安排哪些测试人员进行测试；

（6）How——如何去做，使用哪些测试工具以及测试方法进行测试。

二、测试计划模板

因为各个公司的测试计划模板是不同的，这是一个比较完整的测试计划模板，写得很详细（见表 1-2、表 1-3），学生可以参考模板完成天天超市管理系统测试计划的撰写。

项目名称（项目编号）

测试计划

（部门名称）

×××软件公司

表 1-2　测试计划说明表

总页数		正文		附录		生效日期：　　年　　月　　日
编制：			审核：			批准：

表 1-3　修订历史记录

日　　期	版　　本	说　　明	作　　者
<日/月/年>	<x.x>	<详细信息>	<姓名>

目　　录

一、简介

1．目的

<项目名称>的“测试计划”文档的目的是：

（1）提供一个对项目软件进行测试的总体安排和进度计划，确定现有项目的信息和应测试软件构件。

（2）标明推荐的测试需求（高层次）。

（3）推荐可采用的测试策略，并对这些策略加以说明。

（4）确定所需的资源，并对测试的工作量进行估计。

（5）列出测试项目的可交付元素。

2．背景

输入测试对象（组件、应用程序、系统等）及其目标的简要说明。需要包括的信息有：主要的功能和特性、测试对象的构架以及项目的简史。本部分应该只包含 3～5 个段落。

3．范围

描述测试的各个阶段，例如，单元测试、集成测试或系统测试，并说明本计划所针对的测试类型（如功能测试或性能测试）。简要地列出测试对象中将接受测试或将不接受测试的那些特性和功能。

如果在编写此文档的过程中做出的某些假设可能会影响测试设计、开发或实施，则列出所有这些假设。

列出可能会影响测试设计、开发或实施的所有风险或意外事件。

列出可能会影响测试设计、开发或实施的所有约束。

4．使用文档

表 1-4 列出了制订测试计划所用的文档，并标明了文档的可用性。

注：可以视情况删除或添加项目。

表 1-4　测试计划使用文档列表

文档（版本/日期）	已创建或可用	已被接受或已经过复审	作者或来源	备　注
需求规约	□ 是　□ 否	□ 是　□ 否		
功能性规约	□ 是　□ 否	□ 是　□ 否		
用例报告	□ 是　□ 否	□ 是　□ 否		
项目计划	□ 是　□ 否	□ 是　□ 否		
设计规约	□ 是　□ 否	□ 是　□ 否		
原型	□ 是　□ 否	□ 是　□ 否		
用户手册	□ 是　□ 否	□ 是　□ 否		
业务模型或业务流程	□ 是　□ 否	□ 是　□ 否		
数据模型或数据流	□ 是　□ 否	□ 是　□ 否		
业务功能和业务规则	□ 是　□ 否	□ 是　□ 否		
项目或业务风险评估	□ 是　□ 否	□ 是　□ 否		

二、测试需求

下面列出了那些已被确定为测试对象的项目（用例、功能性需求和非功能性需求）。