第3章 软件测试计划与策略

学习目的与要求

本章介绍软件测试计划与软件测试策略的基本概念。通过本章的学习，能够掌握软件测试计划制订的原则、内容和标准及其软件测试策略等。要求学会如何制订软件测试计划，需要重点掌握软件测试策略，以及软件测试和软件开发过程的关系。

本章主要内容

- 软件测试计划。
- 软件测试策略。
- 软件测试过程。
- 软件测试与软件开发过程。
- 软件自动化测试概述。

3.1 软件测试计划

软件测试是有计划、有组织和有系统的软件质量保证活动，而不是随意的、松散的、杂乱的实施过程。为了规范软件测试内容、方法和过程，在对软件进行测试之前，必须创建测试计划，编写软件测试计划文档，以保证软件测试的有效、顺利进行。测试计划描述了如何进行测试。有效的测试计划会驱动测试工作的完成，使测试执行、测试分析以及测试报告的工作开展更加顺利。

软件测试计划是整个开发计划的重要组成部分，同时又依赖于软件产品开发过程、项目的总体计划、质量保证体系。在测试计划活动中，首先要确认测试目标、范围和需求，然后制定测试策略，并对测试任务、时间、资源、成本和风险等进行估算或者评估。测试计划是为了解决项目的测试目标、任务、方法、资源、进度和风险等问题，当这些问题解决或找到相应

的解决对策和措施后，就是编制好测试计划文档。测试计划是一个过程，不仅仅是编制一个测试文档，还必须随项目情况的变化而不断进行调整，以便优化资源和提高测试效率。

在《ANSI/IEEE 软件测试文档标准 829—1983》中，测试计划被定义为："一个叙述了预定的测试活动的范围、途径、资源及进度安排的文档。它确认了测试项、被测特征、测试任务、人员安排，以及任何偶发事件的风险。"

软件测试计划是指导测试过程的文件，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、测试交流和风险分析等内容。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。

测试计划描述了所有要完成的测试工作，包括被测试项目的背景、目标、范围、方式、资源、进度安排、测试组织以及与测试有关的风险等。

编写测试计划具有以下好处。

（1）使软件测试工作进行更顺利。一般讲"预则立"，计划便是软件测试工作的预先安排，为整个测试工作指明方向。

（2）促进项目参加人员彼此沟通。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。这种形式使测试工作与开发工作紧密地联系起来。

（3）使软件测试工作更易于管理。领导能够根据测试计划做宏观调控，进行相应资源配置等；其他人员了解测试人员的工作内容，进行有关配合工作。按照这种方式，资源变更成为可控制的风险。

3.1.1 制订测试计划的原则

通常，在测试需求分析前编制总体测试计划书，在测试需求分析后编制详细测试计划书。测试计划的编写是一项系统工作，编写者必须对项目了解，对测试工作所接触到的方方面面都要有系统的把握，因此一般情况下由具有丰富经验的项目测试负责人进行编写。在编制测试计划时，应该遵循以下原则。

（1）制订测试计划应尽早开始。尽早进行测试计划，就可以从最根本的地方去了解所要测试的对象及内容，对完善测试计划很有好处。

（2）保持测试计划的灵活性。测试计划不是固定的，在测试进行过程中会有一定的变动，测试计划的灵活性为持续测试提供了很好的支持。

（3）保持测试计划简洁和易读。测试计划制订出来后应该能够让测试人员明白自己的任务和计划。避免测试计划的"大而全"，即测试计划文档应包含详细的测试技术指标、测试步骤和测试用例，避免篇幅冗长且重点不突出。最好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。

（4）尽量争取多渠道评审测试计划。通过不同的人来发现测试计划中的不足及缺陷，可以很好地改进测试计划的质量。

（5）计算测试计划的投入。投入测试中的项目经费是有一定限额的，制订测试计划时一定要注意测试计划的费用情况，要量力而行。

3.1.2 制订测试计划

软件测试必须以一个好的测试计划为基础。尽管测试的每一个步骤都是独立的，但是必定要有一个起到框架结构作用的测试计划。测试计划应该作为测试的起始步骤和重要环节。一个测试计划应包括产品基本情况调研、测试需求说明、测试策略和记录、测试资源配置、计划表、问题跟踪报告、测试计划的评审等。

1）产品基本情况调研

这部分应包括产品的一些基本情况介绍，例如产品的运行平台和应用的领域，产品的特点和主要功能模块等。对于大的测试项目，还要包括测试的目的和侧重点。

2）测试需求说明

这部分要列出所有要测试的功能项。凡是没有出现在清单里的功能项都排除在测试的范围外。万一有一天在一个没有测试的部分里发现了一个问题，应该高兴有了这个记录在案的文档，可以证明自己测了什么，没测什么。具体要点包括以下方面。

（1）功能的测试：理论上测试是要覆盖所有的功能项，例如在数据库中添加、编辑、删除记录等。

（2）设计的测试：对于一些用户界面、菜单的结构、窗体的设计是否合理等的测试。

（3）整体考虑：这部分测试需求要考虑数据流从软件中的一个模块到另一个模块的过程中的正确性。

3）测试策略和记录

这是整个测试计划的重点，描述如何公正客观地开展测试，要考虑模块、功能、整体、系统、版本、压力、性能、配置和安装等因素的影响。要尽可能地考虑到细节，越详细越好，并制作测试记录文档的模板，为即将开始的测试做准备。测试记录主要包括以下部分。

（1）公正性声明：要对测试的公正性、遵照的标准做一个说明，证明测试是客观的。整体上，软件功能要满足需求，实现正确，与用户文档的描述保持一致。

（2）测试案例：描述测试案例是什么样的，采用了什么工具，工具的来源是什么，如何执行的，用了什么样的数据。测试记录中要为将来的回归测试留有余地，当然也要考虑同时安装的其他软件对正在测试的软件可能会造成的影响。

（3）特殊考虑：有时针对外界环境的影响，要对软件进行特殊的测试。

（4）经验判断：对在以往的测试中经常出现的问题加以考虑。

（5）设想：采取一些发散性的思维，往往能帮助找到测试的新途径。

4）测试资源配置

制订项目资源配置计划，包含每一个阶段的任务、所需要的资源，当发生类似到了使用期限或者资源共享时，要更新项目资源计划。

5）计划表

测试的计划表可以做成一个或多个项目通用的形式，根据大致的时间来制作，操作流程要以软件测试的常规周期作为参考，也可以根据什么时间测试哪一个模块来制定。

6）问题跟踪报告

在测试的计划阶段，应该明确如何准备去做一个问题跟踪报告，以及如何去界定一个问题的性质。问题跟踪报告要包括问题的发现者和修改者，问题发生的频率，用了什么样的测试案例测出该问题的，以及明确问题产生时的测试环境。

7）测试计划的评审

测试计划的评审又称测试规范的评审。在测试实施开展前，必须认真负责地检查一遍，并获得整个测试部门人员的认同，包括部门的负责人的同意和签字。

3.2 软件测试策略

测试策略是指制定测试整体策略、所使用的测试技术和方法。为了检验开发的软件能否符合规格说明书的要求，测试活动可以采用各种不同的策略。这些策略的区别在于它们表明了不同的出发点、不同的思路，以及采用的不同手段和方法，具体包括要使用的测试技术和工具、测试完成标准、影响资源分配的特殊考虑等。在此，着重讨论要使用的测试技术。

3.2.1 静态测试与动态测试

根据是否运行程序，软件测试技术可分为静态测试与动态测试。

1. 静态测试

静态测试是一种不通过执行程序，只通过检查和审阅而进行测试的技术。可以由人工进行，充分发挥人的逻辑思维优势，也可以借助软件工具自动进行。其关键功能是检查软件的表示和描述是否一致，有没有冲突或者有没有歧义。它瞄准的是纠正软件系统在描述、表示和规格上的错误，是进一步测试的前提。静态测试包括代码检查、静态结构分析、代码质量度量等。经验表明，使用这种方法能够有效地发现 30％～70％的逻辑设计和编码错误。

1）代码检查

代码检查包括代码走查、代码审查等，主要检查代码和设计的一致性，代码对标准的遵循、可读性，代码的逻辑表达的正确性，代码结构的合理性等方面。代码检查可以发现违背程序编写标准的问题，程序中不安全、不明确和模糊的部分，找出程序中不可移植部分、违背程序编程风格的问题，包括变量检查、命名和类型审查、程序逻辑审查、程序语法检查和程序结构检查等内容。

2）编码风格与规范

在程序设计中要使程序结构合理、清晰，形成良好的编程习惯，对程序的要求不仅是可以在机器上执行，给出正确的结果，而且要便于程序的调试和维护，这要求程序员编写的程序不仅自己看得懂，而且也要让别人能看懂。程序如同一篇文章，应该易于被别人看懂，读起来流畅，必要时又容易修改。好的程序设计风格有助于提高程序的正确性、可读性、可维护性、可用性。

好的风格对于好的程序设计具有关键性作用。写好一个程序，当然需要使它符合语法规则、修正其中的错误和使它运行得足够快，但是实际做的远比这多得多。一个写得好的程序比那些写得差的程序更容易读、更容易修改。

3）代码审查

代码走查和代码审查的对比如表 3-1 所示。

代码审查是提高代码质量的良药。代码审查中通常要检查如下几方面的错误。

（1）数据引用错误：是指使用未经正确初始化用法和引用方式的变量、常量、数组、字符串或记录而导致的软件缺陷。例如，变量未初始化，数组和字符串下标越界，对数组的下标操作遗漏[0]，变量与赋值类型不一致，引用的指针未分配内存。

表 3-1　代码走查和代码审查对比

项　目	代 码 走 查	代 码 审 查
准备	通读设计和编码	应准备好需求描述文档、程序设计文档、程序的源代码清单、代码编码标准和代码缺陷检查表
形式	非正式会议	正式会议
参加人员	开发人员为主	项目组成员包括测试人员
主要技术方法	—	缺陷检查表
注意事项	限时、不要现场修改代码	限时、不要现场修改代码
生成文档	会议记录	静态分析错误报告
目标	代码标准规范，无逻辑错误	代码标准规范，无逻辑错误

(2) 数据声明错误：产生的原因是不正确的声明或者使用变量和常量。

(3) 计算错误：计算或运算错误是基本的数学逻辑问题，计算将无法得到预期结果。例如，不同数据类型或数据类型相同但长度不同的变量计算，计算过程中或计算结果溢出，赋值的目的变量上界小于赋值表达式的值，除数/模为 0，变量的值超过有意义的范围(如概率的计算结果不在 0～1 内)。

(4) 比较错误：小于、大于、等于、不等于、真、假，在这些比较和判断时出现的错误很可能是边界条件问题。例如，混淆小于和小于或等于，逻辑表达式中的操作数不是逻辑值，等等。

(5) 控制流程错误：产生该错误的原因是编程语言中循环等控制结构未按预期方式工作，通常由计算或者比较错误直接或间接造成。例如，模块或循环无法终止，存在从未执行的代码，由于变量赋值错误而意外进入循环。

(6) 子程序参数错误：产生该错误的原因是软件子程序不正确地传递了数据。例如，实际传送的参数类型或次序与定义不一致，更改了仅作为输入值的参数。

(7) 输出错误：包括文件读取、接收键盘或鼠标输入以及向打印机或屏幕等输出设备写入错误。例如，软件没有严格遵守外部设备读写数据的专用格式，文件或外设不存在或者未准备好的错误情况发生时没有相应处理，未以预期的方式处理预计错误，错误提示信息不正确或不准确。

(8) 其他检查：包括软件是否使用其他外语，处理字符集的范围(ASCII 或 Unicode)，是否需要移植，是否考虑兼容性，编译时是否产生警告或提示信息，等等。

4) 静态结构分析

静态结构分析主要是以图形的方式表现程序的内部结构。例如，函数调用关系图、函数内部控制流图。其中，函数调用关系图以直观的图形方式描述一个应用程序中各个函数的调用和被调用关系；函数内部控制流图显示一个函数的逻辑结构，它由许多节点组成，一个节点代表一条语句或数条语句，连接节点的叫边，边表示节点间的控制流向。

5) 代码质量度量

ISO/IEC 9126 国际标准所定义的软件质量包括 6 方面：功能性、可靠性、易用性、效率、可维护性和可移植性。软件的质量是软件属性的各种标准度量的组合。

2. 动态测试

动态测试直接执行被测试程序以提供测试支持。一般情况下，动态测试在完成静态测试

之后进行，包括功能确认与接口测试、覆盖率分析、性能分析、内存分析等。

覆盖率分析可以对测试质量提供定量的分析，即覆盖率分析对所涉及的程序结构元素或数据流信息（如对变量的定义及其使用途径）进行度量，以最终确定测试运行的充分性。

根据动态测试在软件开发过程中所处的阶段和作用，动态测试可分为如下几个步骤。

1）单元测试

单元测试是对软件中的基本组成单位进行测试，其目的是检验软件基本组成单位的正确性。

2）集成测试

集成测试是在软件系统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。在实际工作中，把集成测试分为若干次的组装测试和确认测试。

（1）组装测试：是单元测试的延伸，除对软件基本组成单位的测试外，还需增加对相互联系模块之间接口的测试。

（2）确认测试：是对组装测试结果的检验，主要目的是尽可能地排除单元测试、组装测试中发现的错误。

3）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和性能等是否满足其规约所指定的要求。系统测试应该按照测试计划进行，其输入、输出和其他动态运行行为应该与软件规约（即软件的设计说明书、软件需求说明书等文档）进行对比，同时测试软件的强壮性和易用性。

4）验收测试

验收测试是软件在投入使用之前的最后测试，是购买者对软件的试用过程。在公司实际工作中，通常采用请客户试用或发布 Beta 版软件来实现。

5）回归测试

回归测试即软件维护阶段，其目的是对验收测试结果进行验证和修改。

3.2.2 白盒测试与黑盒测试

静态测试强调不运行程序，通过人工对程序和文档进行分析和检查，实际上是对软件中的需求说明书、设计说明书、程序源代码等进行评审。动态测试是通过人工或使用工具运行程序进行检查、分析程序的执行状态和程序的外部表现。动态测试一般分为白盒测试和黑盒测试。

1）白盒测试

白盒测试又称结构测试，是清楚地了解程序结构和处理过程，检查所有的结构及路径是否都是正确的，检查软件内部动作是否是按照设计说明的规定进行的。白盒测试作为测试人员常用的一种测试方法，越来越受到测试工程师的重视。白盒测试并不是简单地按照代码设计用例，而是需要根据不同的测试需求，结合不同的测试对象，使用适合的方法进行测试。因为对于不同复杂度的代码逻辑，可以衍生出许多种执行路径，只有通过适当的测试方法，才能帮助我们从代码的迷雾森林中找到正确的方向。

白盒测试与程序内部结构相关，需要利用程序结构的实现细节等知识，才能有效进行测试用例的设计工作。白盒测试方法有程序控制流分析、数据流分析、逻辑驱动测试、域测试、符号测试、路径测试、程序插桩及程序变异等。

2）黑盒测试

黑盒测试也称功能测试或数据驱动测试，是把测试对象看成一个黑盒子，完全不考虑程序的内部结构和处理过程，通常在程序界面处进行测试。黑盒测试只是检查程序或软件是否按照需求规格说明书的规定正常运行，程序是否能适当地接收输入数据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、边界值分析、因果图、错误推测等，主要用于软件确认测试。黑盒测试法着眼于程序外部结构，不考虑内部逻辑结构，针对软件界面和软件功能进行测试。黑盒测试法是穷举输入测试，只有把所有可能的输入都作为测试情况使用，才能以这种方法查出程序中的所有错误。实际上测试情况有无穷多种，人们不仅要测试所有合法的输入，而且还要对那些不合法但是可能的输入进行测试。

3.3 软件测试过程

在软件测试的过程中，没有最好的过程可以遵循，但可以选择通过实践检验证明是更佳的测试过程。掌握并应用测试过程可以带来以下的诸多益处。

（1）测试的一致性。有了规范的过程，可以一致的方式从一个测试过程过渡到下一个测试。过程的运用有效地减少了测试活动的不确定性。

（2）可持续地改进测试过程。在过程实施过程中，通过过程数据的收集、分析和处理，可以发现过程的优缺点。一旦发现了过程的不足和缺点，就可以持续地改进测试过程。

（3）便于管理。测试经理可以通过测试过程进行有效管理。从控制的角度来看，管理一个过程比管理一个人要简单得多。

软件测试是软件开发过程的一个重要环节，是在软件投入运行之前，对软件需求分析、实际规格说明书和编码实现的最终审定，贯穿于软件定义和开发的整个过程，它们是应相辅相成和相互依赖的。其整个测试过程如图 3-1 所示。

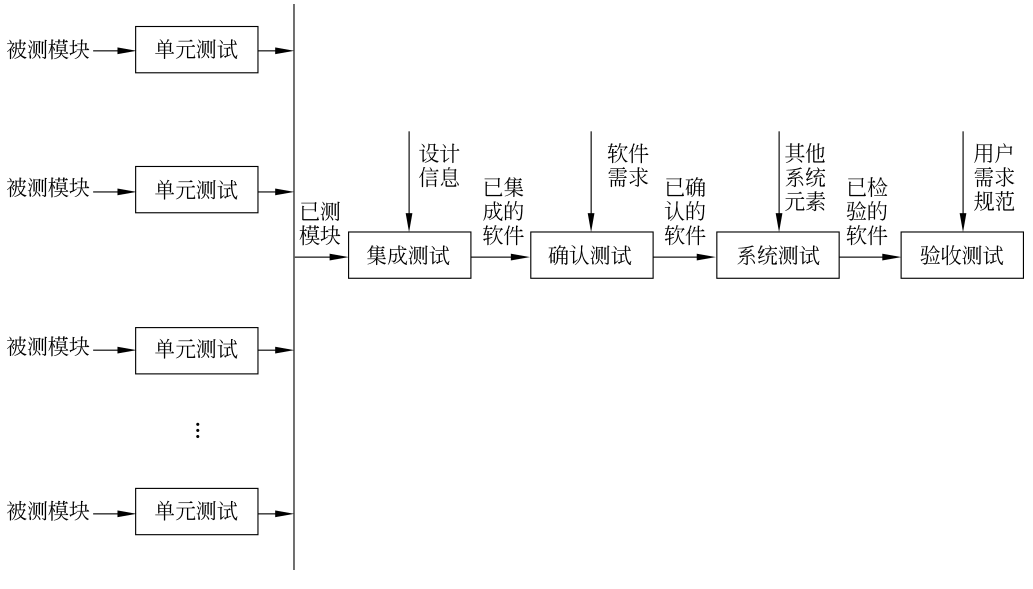


图 3-1 软件测试过程

软件测试由一系列不同的测试阶段组成，即单元测试、集成测试、确认测试、系统测试和验收测试。软件开发是一个自顶向下逐渐细化的过程。软件测试则是自底向上逐步集成的过程。低一级的测试为上一级的测试准备条件。

(1) 单元测试：又称模块测试，其目的在于检查每个单元能否正确满足详细设计说明书中的功能、性能、接口和设计约束等要求，发现单元内部可能存在的各种缺陷。单元测试大多采用白盒测试的方法。

(2) 集成测试：又称组装测试，主要测试单元之间的接口关系，逐步集成为符合概要设计要求的整个系统。集成测试大多采用黑盒测试的方法。

(3) 确认测试：以规格说明书规定的要求为尺度，检验开发的软件能否满足所有的功能和性能需求。

(4) 系统测试：在真实或模拟系统运行的环境下，为验证和确认是否达到需求规格说明书规定的要求，而对集成的硬件和软件系统进行的测试。

(5) 验收测试：按照项目任务书或合同、供需双方约定的验收。根据文档进行的对整个系统的评测，决定是否接收或是拒绝系统。

在系统测试阶段，软件开发项目和测试之间的交互，主要出现在被测件被提交给测试小组进行测试时。若缺乏系统测试计划，将经常看到这个关键的交接点变成混乱的现象。例如，每当开发小组的任何一个成员修复了一个缺陷，软件开发经理就将被测件的新版本发送给整个开发部和测试小组，这样测试小组在一天之内将收到很多测试版本。为规范测试版本的管理和提高测试效率，采用多个测试循环来组成某个阶段的系统测试，如图 3-2 所示，即每隔一定周期发布一个新的软件版本。任何特定的测试阶段至少包含一个测试循环。测试循环的周期和软件的规模大小、测试过程和测试自动化水平等有关，如可采取 1～3 周为一次循环。

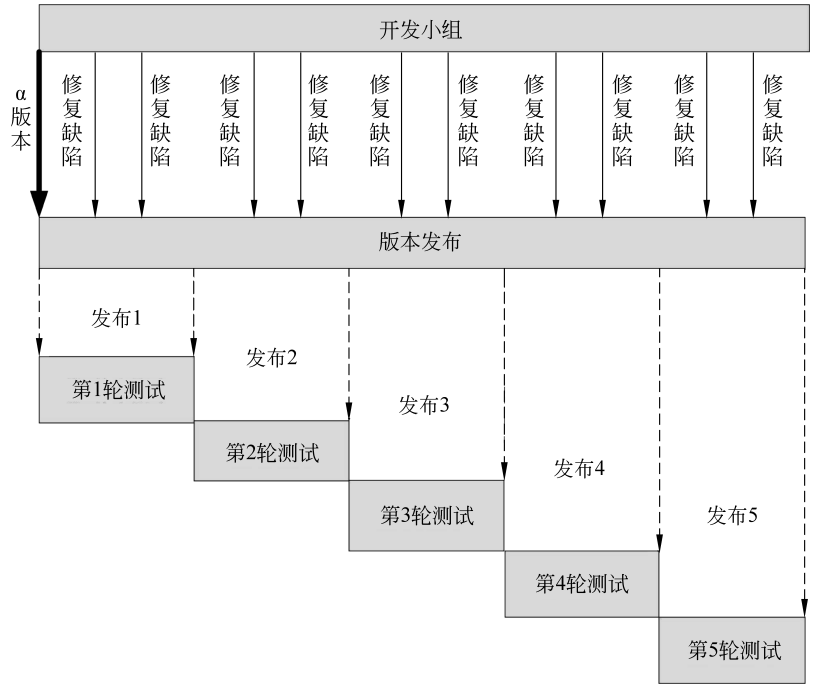


图 3-2 测试循环

3.4 软件测试与软件开发过程的关系

软件测试贯穿在软件开发过程中，在每个开发阶段具有不同的任务。在需求分析阶段，主要进行测试需求分析、系统测试计划的制订；在概要设计和详细设计阶段，主要确保集成测试计划和单元测试计划完成；在编码阶段，主要由开发人员测试自己负责的模块的代码，若项目较大，则由专业人员进行编码阶段的测试任务；在测试阶段，主要对系统进行测试，并提交相应的测试结果和测试分析报告。

大体来说，软件测试阶段和开发阶段的对应关系如图 3-3 所示。

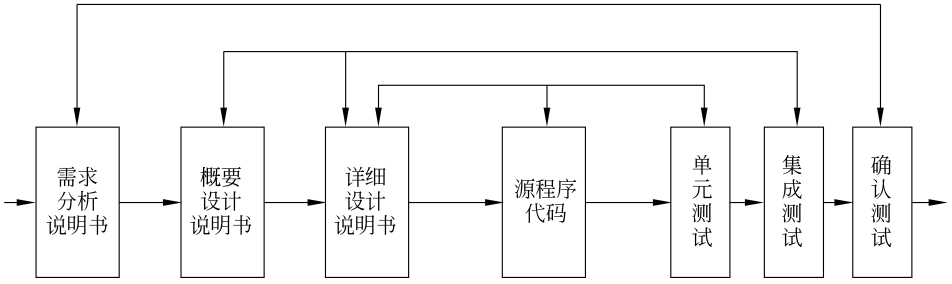


图 3-3 软件测试阶段和软件开发阶段的对应关系

3.4.1 软件开发过程

在软件工程领域，正规的软件开发过程一般包括制订计划、需求分析、软件设计、程序编码、软件测试以及运行和维护 6 个阶段。这 6 个阶段构成了软件的生命周期。

1）制订计划

制订计划要对所要解决的问题进行总体定义，包括了解用户的需求，从技术、经济和社会因素 3 方面研究并论证本软件项目的可行性，编写可行性研究报告，探讨解决问题的方案，并对可供使用的资源（如计算机硬件、系统软件、人力等）、成本、可取得的效益和开发进度做出估计，制订完成开发任务的实施计划。此阶段由软件开发方与需求方共同讨论，主要确定软件的开发目标及其可行性。

2）需求分析

软件需求分析就是回答做什么的问题。它是一个对用户的需求进行去粗取精、去伪存真、正确理解，然后把它用软件工程开发语言（形式功能规约，即需求规格说明书）表达出来的过程。本阶段的基本任务是和用户一起确定要解决的问题，建立软件的逻辑模型，编写需求规格说明书文档并最终得到用户的认可。需求分析的主要方法有结构化分析方法、数据流程图和数据字典等方法。本阶段的工作是根据需求说明书的要求，设计建立相应的软件系统的体系结构，并将整个系统分解成若干子系统或模块，定义子系统或模块间的接口关系，对各子系统进行具体设计定义，编写软件概要设计和详细设计说明书、数据库或数据结构设计说明书，组装测试计划。

在确定软件开发、可行的情况下，对软件需要实现的各个功能进行详细分析。需求分析阶段是一个很重要的阶段，这一阶段做得好，将为整个软件开发项目的成功打下良好的基础。同样，需求在整个软件开发过程中也是不断变化和深入的。因此，必须制订需求变更计划来应付

这种变化，以保证整个项目的顺利进行。

3）软件设计

软件设计可以分为概要设计和详细设计两个阶段。实际上，软件设计的主要任务就是将软件分解成多个模块。模块是指能实现某个功能的数据和程序说明、可执行程序的程序单元，可以是一个函数、过程、子程序、一段带有程序说明的独立的程序和数据，也可以是可组合、可分解和可更换的功能单元。概要设计就是结构设计，其主要目标就是给出软件的模块结构，常用软件结构图表示。详细设计的首要任务就是设计模块的程序流程、算法和数据结构，其次是设计数据库，常用结构化程序设计方法实现。

4）程序编码

程序编码是指把软件设计转换成计算机可以识别的程序代码，即写成以某一程序设计语言表示的“源程序清单”。充分了解软件开发语言、工具的特性和编程风格，有助于开发工具的选择以及保证软件产品的开发质量。

5）软件测试

软件测试的目的是以较小的代价发现尽可能多的错误。要实现这个目标，关键在于设计一套出色的测试用例（测试数据和预期的输出结果组成了测试用例）。如何才能设计出一套出色的测试用例，关键在于理解测试方法，不同的测试方法有不同的测试用例设计方法。白盒测试法的对象是源程序，依据的是程序内部的逻辑结构来发现软件的编程错误、结构错误和数据错误。结构错误包括逻辑、数据流、初始化等错误。用例设计的关键是以较少的用例覆盖尽可能多的内部程序逻辑结果。黑盒测试法依据的是软件的功能或软件行为描述，发现软件的接口、功能和结构错误，其中接口错误包括内部或外部接口、资源管理、集成化及系统错误。

6）运行和维护

维护是指在已完成对软件的研制（分析、设计、编码和测试）工作并交付使用以后，对软件产品所进行的一些软件工程的活动，即根据软件运行的情况，对软件进行适当修改，以适应新的要求，以及纠正运行中发现的错误，并编写软件问题报告、软件修改报告。软件维护是软件生命周期中持续时间最长的阶段。

一个中等规模的软件，如果研制阶段需要 1～2 年的时间，在它投入使用以后，其运行或工作时间可能持续 5～10 年。那么，它的维护阶段也是运行的这 5～10 年期间。在这段时间，人们几乎需要着手解决研制阶段所遇到的各种问题，同时还要解决某些维护工作本身特有的问题。做好软件维护工作，不仅能排除障碍，使软件能正常工作，而且还可以使它扩展功能，提高性能，为用户带来明显的经济效益。然而遗憾的是，对软件维护工作的重视往往远不如对软件研制工作的重视。事实上，和软件研制工作相比，软件维护的工作量和成本都要大得多。

在实际开发过程中，软件开发并不是从第一步进行到最后一步，而是在任何阶段，在进入下一阶段前一般都有一步或几步的回溯。测试过程中的问题可能要求修改设计，用户可能会提出一些需要来修改需求说明书等。

完整的软件开发流程如图 3-4 所示。

3.4.2 软件测试在软件开发过程中的作用

从软件开发的生命周期可以看出，在经过了软件制订计划、需求分析、软件设计阶段之后，才进入到软件编码阶段。显然，在软件整个的生命周期中产生的故障或缺陷并不一定是软件编码阶段所引起的，很可能是制订计划、需求分析、概要设计、详细设计阶段所产生的问题引起

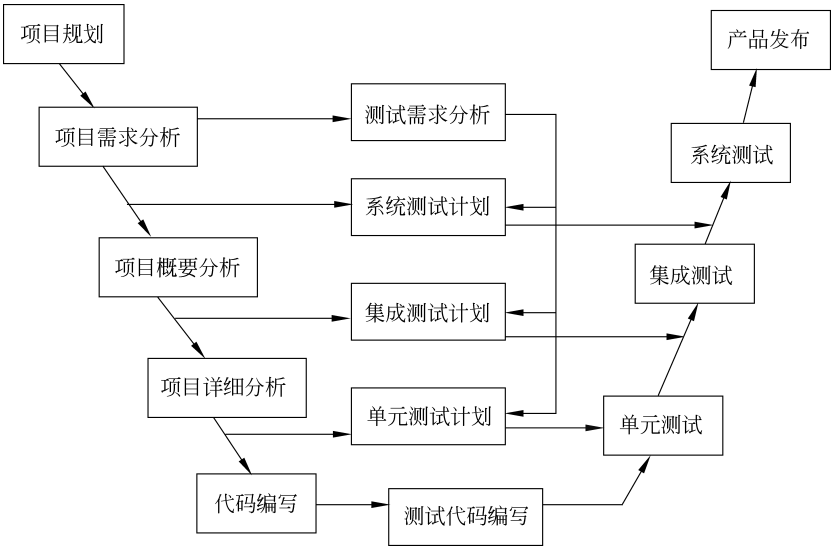


图 3-4　完整的软件开发流程

的。因而，旨在排除故障完善缺陷的软件测试，应该追溯到软件开发生命周期前期的阶段之中。将软件测试贯穿于整个生命周期是必要的。

完整的软件测试流程如图 3-5 所示。

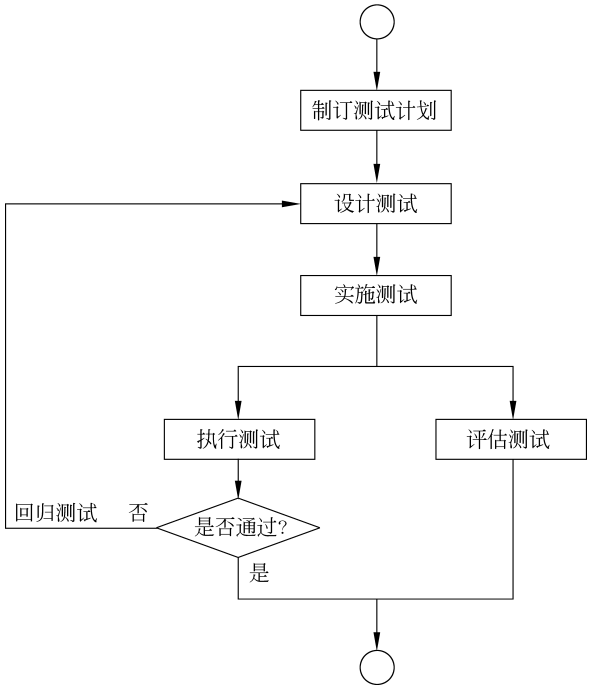


图 3-5　完整的软件测试流程

软件测试在软件开发各阶段的作用如下。

(1) 项目规划阶段：负责从单元测试到系统测试的整个测试阶段的监控。

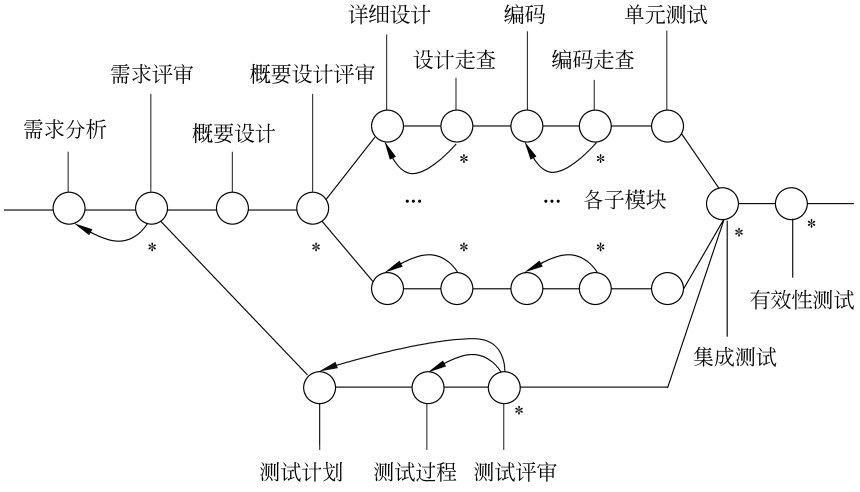
(2) 需求分析阶段：确定测试需求分析、系统测试计划的制订、评审等。

(3) 详细设计和概要设计阶段：确保集成测试计划和单元测试计划的完成。

(4) 编码阶段：由开发人员进行自己负责部分的测试代码。在项目较大时，由专人进行编码阶段的测试任务。

(5) 测试阶段(单元、集成、系统测试)：依据测试代码进行测试，并提交相应的测试状态报告和测试结束报告。

软件测试的目标是致力于“揭示至今为止尚未发现的错误”，从这方面看来，软件测试工作和软件开发工作并行进行既能揭示不同类型的错误，又能耗费最少的时间和最少的工作量。图 3-6 是软件测试与软件开发二者并行进行的示意图。



注：*项目阶段任务的里程碑

图 3-6 软件测试与软件开发二者并行进行

3.5 软件自动化测试

3.5.1 软件自动化测试概述

通常，软件测试的工作量很大。据统计，测试会占用 40%的开发时间；一些可靠性要求非常高的软件，测试时间甚至占到开发时间的 60%。而测试中的许多操作是重复性的、非智力性的和非创造性的，只要求做准确细致的工作，软件自动化测试就适合代替人工去完成这样的任务。

软件自动化测试是相对手工测试而存在的，主要是通过所开发的软件测试工具、脚本等来实现的，具有良好的可操作性、可重复性和高效率等特点。

1. 手工测试的局限性

手工测试主要有以下局限性。

(1) 通过手工测试无法做到覆盖所有代码路径。

(2) 简单的功能性测试用例在每一轮测试中都不能少，而且具有一定的机械性、重复性，工作量往往较大。

(3) 许多与时序、死锁、资源冲突、多线程等有关的错误，通过手工测试很难捕捉到。

(4) 进行系统负载、性能测试时,需要模拟大量数据或大量并发用户等应用场合时,很难通过手工测试来进行。

(5) 进行系统可靠性测试时,需要模拟系统运行十年甚至几十年,以验证系统能否稳定运行,这也是手工测试无法模拟的。

(6) 如果有大量的(如几千个)测试用例,需要在短时间(如 1 天)内完成,手工测试几乎不可能做到。

2. 自动化测试的好处

自动化测试有如下好处。

(1) 缩短软件开发测试周期,可以让产品更快地投放市场。

(2) 测试效率高,充分利用硬件资源。

(3) 节省人力资源,降低测试成本。

(4) 增强测试的稳定性和可靠性。

(5) 提高软件测试的准确度和精确度,增加软件信任度。

(6) 软件测试工具使测试工作相对比较容易,能产生更高质量的测试结果。

(7) 手工不能做的事情,自动化测试能做(如负载、性能测试)。

软件测试实行自动化进程,绝不是因为厌烦了重复的测试工作,而是因为测试工作的需要,更准确地说是回归测试和系统测试的需要。

3. 自动化测试的原理和方法

软件测试自动化实现的原理和方法主要有直接对代码进行静态和动态分析、测试过程的捕获和回放、测试脚本技术等。

1) 代码分析

代码分析类似于高级编译系统,一般针对不同的高级语言去构造分析工具,在工具中定义类、对象、函数、变量等定义规则、语法规则;在分析时对代码进行语法扫描,找出不符合编码规范的地方;根据某种质量模型评价代码质量,生成系统的调用关系图等。

2) 捕获和回放

代码分析是一种白盒测试的自动化方法,捕获和回放则是一种黑盒测试的自动化方法。

(1) 捕获。捕获是将用户每一步操作都记录下来。这种记录的方式有:程序用户界面的像素坐标或程序显示对象(窗口、按钮、滚动条等)的位置,以及相对应的操作、状态变化或属性变化,将所有的记录转换为一种脚本语言所描述的过程,以模拟用户的操作。

(2) 回放。回放时,将脚本语言所描述的过程转换为屏幕上的操作,然后将被测系统的输出记录下来,同预先给定的标准结果作比较。回放可以大大减少黑盒测试的工作量,在迭代开发的过程中,能够很好地进行回归测试。

目前的自动化负载测试解决方案几乎都是采用“录制—回放”的技术。所谓“录制—回放”技术,就是先由手工完成一遍需要测试的流程,同时由计算机记录下这个流程期间客户端和服务器端之间的通信信息,这些信息通常是一些协议和数据,并形成特定的脚本程序(script)。然后在系统的统一管理下同时生成多个虚拟用户,并运行该脚本,监控硬件和软件平台的性能,提供分析报告或相关资料。这样,通过几台机器就可以模拟出成百上千的用户对应用系统进行负载能力的测试。

3) 脚本技术

脚本是一组测试工具执行的指令集合,也是计算机程序的一种形式。脚本可以通过录制

测试的操作产生，然后再做修改，这样可以减少脚本编程的工作量。当然，也可以直接用脚本语言编写脚本。脚本技术可以分为以下几类。

（1）线性脚本：由录制手工执行的测试用例得到的脚本。

（2）结构化脚本：类似于结构化程序设计，具有各种逻辑结构（顺序、分支、循环结构），而且具有函数调用功能。

（3）共享脚本：指某个脚本可被多个测试用例使用，即脚本语言允许一个脚本调用另一个脚本。

（4）数据驱动脚本：将测试输入存储在独立的数据文件中。

（5）关键字驱动脚本：是数据驱动脚本的逻辑扩展。

3.5.2 软件自动化测试工具

软件自动化测试通常借助测试工具进行。测试工具可以进行部分的测试设计、实现、执行和比较的工作。部分的测试工具可以实现测试用例的自动生成，但通常的工作方式为人工设计测试用例，使用工具进行用例的执行和比较。如果采用自动比较技术，还可以自动完成测试用例执行结果的判断，从而避免人工比对产生的疏漏问题。下面简单介绍主要的软件自动化测试工具。

1. QuickTest Professional（QTP）

1）QTP 简介

QTP 是新一代自动化测试解决方案，它可以覆盖绝大多数的软件开发技术，简单高效，并具备测试用例可重用的特点。它是一款先进的自动化测试解决方案，用于创建功能和回归测试。它自动捕获、验证和重放用户的交互行为，可为每一个重要软件应用和环境提供功能和回归测试自动化的行业最佳解决方案。

2）QTP 的测试流程

（1）制订测试计划。自动测试的测试计划是根据被测项目的具体需求，以及所使用的测试工具而制订的，完全用于指导测试全过程。

QTP 是一个功能测试工具，主要帮助测试人员完成软件的功能测试，与其他测试工具一样，QTP 不能完全取代测试人员的手工操作，但是在某个功能点上，使用 QTP 的确能够帮助测试人员做很多工作。在测试计划阶段，首先要做的是分析被测应用程序的特点，决定应该对哪些功能点进行测试，可以考虑细化到具体页面或具体控件。对于一个普通的应用程序来说，QTP 应用在某些界面变化不大的回归测试中是非常有效的。

（2）创建测试脚本。当测试人员浏览站点或在应用程序上操作的时候，QTP 的自动录制机制能够将测试人员的每一个操作步骤及被操作的对象记录下来，自动生成测试脚本语句。与其他自动测试工具录制脚本有所不同，QTP 除了以 VBScript 脚本语言的方式生成脚本语句外，还将被操作的对象及相应的动作按照层次和顺序保存在一个基于表格的关键字视图中。例如，当测试人员单击一个链接，然后选择一个 CheckBox 或者提交一个表单，这样的操作流程都会被记录在关键字视图中。

（3）增强测试脚本的功能。录制脚本只是为了实现创建或者设计脚本的第一步，基本的脚本录制完毕后，测试人员可以根据需要增加一些扩展功能。QTP 允许测试人员通过在脚本中增加或更改测试步骤来修正或自定义测试流程，如增加多种类型的检查点功能，既可以让 QTP 检查在程序的某个特定位置或对话框中是否出现了需要的文字，还可以检查一个链接是

否返回了正确的 URL 地址等，还可以通过参数化功能使用多组不同的数据驱动整个测试过程。

(4) 运行测试。QTP 从脚本的第一行开始执行语句，运行过程中会对设置的检查点进行验证，用实际数据代替参数值，并给出相应的输出结构信息。测试过程中测试人员还可以调试自己的脚本，直到脚本完全符合要求。

(5) 分析测试。运行结束后系统会自动生成一份详细完整的测试结果报告。

2. Web 测试工具——Selenium

1) Selenium 简介

Selenium 是 ThoughtWorks 专门为 Web 应用程序编写的一个开源验收测试工具。Selenium 测试直接在浏览器中运行，就像真实用户所做的一样。Selenium 测试可以在 Windows、Linux 和 Macintosh 上的 Internet Explorer、Mozilla 和 Firefox 浏览器中运行。

Selenium 不同于一般的测试工具。一般的脚本测试工具录制脚本，都是通过拦截浏览器收发的 HTTP 请求来实现的，事实上并没有办法录制用户对 HTML 页面的操作。Selenium 的脚本录制工具是通过监听用户对 HTML 页面的操作来录制脚本的。Selenium 是真正能够监听用户对 HTML 页面操作的录制工具。Selenium 完全了解用户操作的 HTML 页面。

2) Selenium 执行原理

(1) SeleniumServer 通过网络与 Selenium 客户端通信，接收 Selenium 测试指令。

(2) SeleniumServer 通过向浏览器发出 JavaScript 调用实现对 HTML 页面的全面追踪，并通过网络把执行结果返回给 Selenium 客户端。

(3) Selenium 客户端一般使用单元测试技术实现，通过判断返回的结果与预期是否一致来决定程序是否运行正确。

(4) Selenium 是通过 JavaScript 来实现对 HTML 页面操作的。它提供了丰富的指定 HTML 页面元素和操作页面元素的方法。

(5) Selenium 打开浏览器时把自己的 JavaScript 文件嵌入网页中，然后 Selenium 的网页通过框架嵌入目标网页，这样就可使用 Selenium 的 JavaScript 对象来控制目标网页。

(6) Selenium 的 JavaScript 对象中，最重要的就是 Selenium 对象。它的作用是代表 Java 中的 Selenium 接口执行一系列的命令，让浏览器执行。

3. 性能测试工具——LoadRunner

1) LoadRunner 简介

LoadRunner 是一种预测系统行为和性能的工业标准级负载测试工具。通过以模拟上千万用户实施并发负载及实时性能监测的方式来确认和查找问题，LoadRunner 能够对整个企业架构进行测试。通过使用 LoadRunner，企业能最大限度地缩短测试时间，优化性能，缩短应用系统的发布周期。

LoadRunner 的测试对象是整个企业的系统，它通过模拟实际用户的操作行为和实行实时性能监测，来帮助更快地查找和发现问题。此外，LoadRunner 能支持广泛的协议和技术，为特殊环境提供特殊的解决方案。

2) LoadRunner 基本原理

使用 LoadRunner 的 Virtual User Generator，能很简便地建立系统负载。该引擎能够生成虚拟用户，以虚拟用户的方式模拟真实用户的业务操作行为。它先记录下业务流程(如下订单或机票预定)，然后将其转换为测试脚本。利用虚拟用户，可以在 Windows、UNIX 或 Linux

系统上同时产生成千上万个用户访问。所以，LoadRunner 能极大地减少负载测试所需的硬件和人力资源。另外，LoadRunner 的 TurboLoad 专利技术能提供很高的适应性。TurboLoad 可以产生每天几十万名在线用户和数以百万计的点击数的负载。

用 Virtual User Generator 建立测试脚本后，可以对其进行参数化操作，利用不同的实际发生数据来测试应用程序，从而反映出本系统的负载能力。以一个订单输入过程为例，参数化操作可将记录中的固定数据，如订单号和客户名称，由可变值来代替。在这些变量内随意输入可能的订单号和客户名，来匹配多个实际用户的操作行为。

LoadRunner 通过它的 Data Wizard 来自动实现其测试数据的参数化。Data Wizard 直接连接数据库服务器，从而可以获取所需的数据，并直接将其输入测试脚本。这样避免了人工处理数据。

为了进一步确定 Virtual User 能够模拟真实用户，可以利用 LoadRunner 控制某些行为特性。例如，只需单击鼠标就能轻易控制交易的数量、交易频率、用户的思考时间和连接速度等。

4. 性能测试工具——JMeter

1) JMeter 简介

Apache JMeter 是 Apache 组织基于 Java 开发的压力测试工具，用于对软件做压力测试。它最初被设计用于 Web 应用测试，但后来扩展到其他测试领域。它可以用于测试静态和动态资源，如静态文件、Java 小服务程序、CGI 脚本、Java 对象、数据库、FTP 服务器等。JMeter 可以用于对服务器、网络或对象模拟巨大的负载，在不同压力类别下测试它们的强度和分析整体性能。

JMeter 能够对应用程序做功能/回归测试，通过创建带有断言的脚本来验证用户的程序是否返回了期望的结果。为了有最大限度的灵活性，JMeter 允许使用正则表达式创建断言。

Apache JMeter 可以用于对静态的和动态的资源（文件、Servlet、Perl 脚本、Java 对象、数据库和查询、FTP 服务器等）的性能进行测试。它可以用于对服务器、网络或对象模拟繁重的负载来测试它们的强度或分析不同压力类型下的整体性能。可以使用它做性能的图形分析或在大并发负载测试服务器、脚本、对象。

2) JMeter 的特性

(1) 能够对 HTTP 和 FTP 服务器进行压力和性能测试，也可以对任何数据库进行同样的测试（通过 JDBC）。

(2) 完全的可移植性和 100％的纯 Java。

(3) 完全的 Swing 和轻量组件支持（预编译的 JAR 使用 javax.swing.*）包。

(4) 完全的多线程框架允许通过多个线程并发取样和通过单独的线程组对不同的功能同时取样。

(5) 精心的 GUI 设计允许快速操作和更精确的计时。

(6) 缓存和离线分析/回放测试结果。

(7) 高可扩展性：①可链接的取样器允许无限制的测试能力；②各种负载统计表和可链接的计时器可供选择；③数据分析和可视化插件提供了很好的可扩展性及个性化；④具有提供动态输入到测试的功能（包括 JavaScript）；⑤支持脚本变成的取样器。

3.6 项目案例

3.6.1 学习目标

制订测试计划的步骤：
(1) 决定系统测试类型。
(2) 确定系统测试进度。
(3) 组织系统测试小组。
(4) 建立系统测试环境。
(5) 安装系统测试工具。

3.6.2 案例描述

本章通过“艾斯医药商务系统”来学习如何制订软件测试计划。

3.6.3 案例要点

制订测试计划要点：
(1) 明确工作的目标。
(2) 工作的范围，计划针对哪些内容。
(3) 工作任务的分派(时间、人力、物力、技术)。
(4) 明确工作完成的标准。
(5) 明确工作中存在的风险。

3.6.4 案例实施

变更记录表如表 3-2 所示。

表 3-2 变更记录表

日　　期	版　　本	变 更 说 明	作　　者
2010-08-09	V1.0	新建	—
⋮	⋮	⋮	⋮

签字确认表如表 3-3 所示。

表 3-3 签字确认表

职　　务	姓　　名	签　　字	日　　期

1. 引言

1.1 编写目的

本测试计划主要用于控制整个“艾斯医药商务系统”项目测试，本文档主要实现以下目标。

(1) 通过此测试计划能够合理、全面、准确、协调地完成整个测试项目控制。

(2) 为软件测试提供依据。

(3) 项目管理人员根据此计划,可以对项目进行宏观调控。

(4) 测试人员根据此计划,能够明确自己的权利、职责,准确定位自己在项目中的任务。

(5) 相关部门可以根据此计划,对相关资源进行准备。

1.2 背景

本测试计划从属于亚思晟科技有限公司,为×××医药公司实现“艾斯医药商务系统”的测试。项目任务的提出者为亚思晟公司项目管理部;系统的开发者为亚思晟公司;系统的使用者为×××医药公司。此测试项目的进行,将在需求确认后开始执行,基准是准确、全面的需求文档。测试重点是对开发实现的功能和性能进行测试。

1.3 定义

无。

1.4 参考资料

(1)《“艾斯医药商务系统”需求规格说明书》1.0 版本;

(2)《“艾斯医药商务系统”测试计划编写规范》。

1.5 控制信息

本项目测试经理:×××;电话号码:(010)××××××××。

1.6 测试目标

本测试项目将通过设计和执行接受测试、界面测试、功能测试和性能测试,对软件实现的功能,以及软件的性能、兼容性、安全性、实用性、可靠性、扩展性等方面进行全面系统的测试。基于本系统的业务复杂性强和开发周期短的特性,系统测试的重点将放在功能测试和性能测试上。通过测试提高软件的质量,为用户提供更好的服务,并合理地避免软件的风险和减少软件的成本。

2. 计划

2.1 测试过程

测试过程参考本章内容。

2.2 进度安排及里程碑

进度安排及里程碑如图 3-7 所示。

给出进行各项测试的日期和工作内容(如熟悉环境、培训、准备输入数据、实施测试等),如表 3-4 所示。

表 3-4 各项测试的日期和工作内容

里程碑任务	工 作 人 员	开 始 日 期	结 束 日 期
制订测试计划	安××	2020-08-09	2020-08-10
设计测试	安××	2020-08-10	2020-08-13
实施测试	安××	2020-08-16	2020-08-25
对测试进行评估	郭××	2020-08-26	2020-08-27

2.3 测试角色

测试角色如表 3-5 所示。

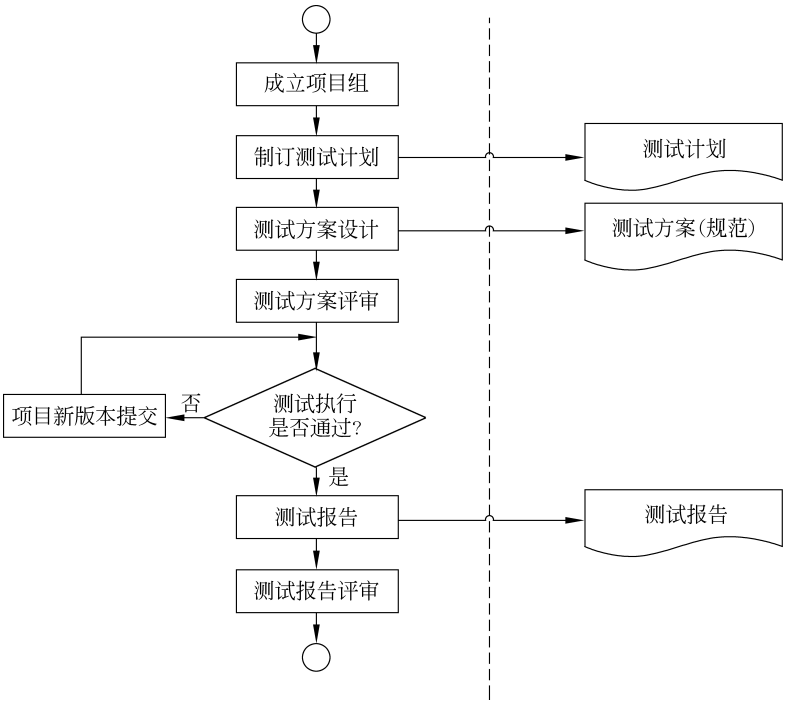


图 3-7　进度安排及里程碑

表 3-5　测试角色

负　责　人	郭××　　其他负责人	职　　责	联 系 信 息
职责：负责制订测试计划；负责编写和验收用例；完成项目实测；负责与外部合作部门交互；负责协调内部人员的工作；负责编写测试报告			

测 试 组 成 员

姓　　名	职　　责	联 系 信 息
安××	负责功能测试用例的编写和实施	
孙××	负责性能和其他非功能测试用例的编写和实施	

2.4 系统资源

表 3-6 列出了测试项目所需的系统资源。

表 3-6　测试项目所需的系统资源

系 统 资 源	
资　　源	名称/类型
数据库服务器	MySQL
网络或子网	
服务器名称	

续表

资　　源	名称/类型
数据库名称	Ascent
客户端测试 PC	IE 8
包括特殊的配置需求	Tomcat
测试存储库	Bug
网络或子网	
服务器名称	
测试开发 PC	Windows XP
硬件环境	Intel Core(TM) CPU 2.0GHz;内存 1GB

2.5 可交付工件

(1) 测试计划：一份。

(2) 测试用例：一份。

(3) 测试缺陷记录：一份。

(4) 测试报告：一份。

2.5.1 测试模型

Ascent 医药商务系统 1.0。

2.5.2 测试记录

采用测试用例的形式提交测试过程，详见《测试用例》文档。

2.5.3 缺陷报告

采用缺陷记录的形式，详见《测试缺陷记录》文档。

2.6 测试资料

(1) 测试文档：测试相关模块。

(2) 需求文档：项目需求文档。

2.7 项目风险分析

项目风险分析如表 3-7 所示。

表 3-7　项目风险分析

风 险 类 型	风 险 综 述
现有人力资源严重不足。在确保质量的前提下，人力资源与项目周期比例失调，因此人员不到位，将存在项目风险	增加人员
测试中使用 IE，因此在 IE 7 等其他环境下运行存在风险	与客户确定为争取时间保证质量仅使用 IE 进行测试
进度存在风险	实际进度将按照开发进度进行，预期度按照开发进度进行，但是实际开发度变更时将按照实际开发进度及时更正测试进度
测试环境各服务器的配置低于实际产品使用时的服务器配置	与客户商议达成一致