第5章

软件测试流程

本章详细介绍了软件测试的整个过程，包括测试计划、测试设计、测试执行、回归测试以及测试评估。测试执行分为单元测试、集成测试、系统测试和验收测试等。

5.1 测试流程概述

软件测试流程与软件开发流程类似，也包括测试计划、测试设计、测试开发、测试执行和测试评估几个部分，如图 5.1 所示。

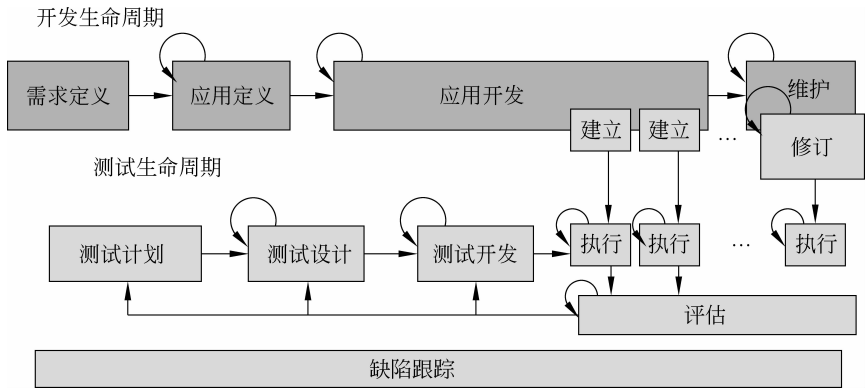

图 5.1 测试生命周期

软件测试生命周期如下所示。

1. 测试计划

根据用户需求报告中关于功能要求和性能指标的规格说明书，定义相应的测试需求报告，使得随后所有测试工作都围绕着测试需求来进行。同时适当选择测试内容，合理安排测试人员、测试时间及测试资源等。

2. 测试设计

测试设计是指将测试计划阶段制定的测试需求分解、细化为若干个可执行的测试过程，并为每个测试过程选择适当的测试用例，保证测试结果的有效性。

3. 测试执行

执行测试开发阶段建立的自动测试过程,并对所发现的缺陷进行跟踪管理。测试执行一般由单元测试、集成测试、系统测试、验收测试以及回归测试等步骤组成。

4. 测试评估

根据缺陷跟踪报告,对软件的质量和开发团队的工作进度及效率进行评价。

5.2 测试需求

测试需求根据市场/产品需求定义、分析文档和相关技术文档,输出《可测试性需求说明书》和《测试规格》等报告。

5.2.1 检查需求文档

需求文档检查步骤如图 5.2 所示,具有如下步骤。

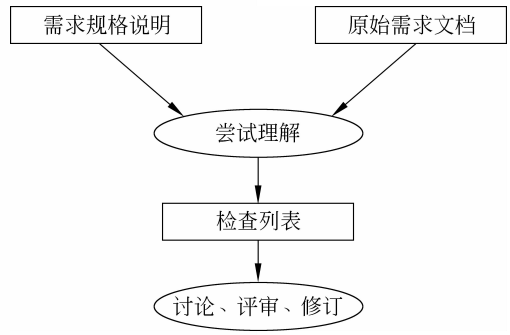

图 5.2 需求文档的检查流程

步骤 1:获取最新版本的软件需求规格说明书,同时尽量取得用户原始需求文档。

步骤 2:阅读和尝试理解需求规格说明书中描述的所有需求项。

步骤 3:对照需求规格说明书检查列表进行检查并记录。

步骤 4:针对检查结果进行讨论,修订需求规格说明书,回到第一步,直到检查列表中的所有项通过。

需求文档的检查需要填写表 5.1,用于检查结果的确认。从需求的完整性、明确性、必要性、可测性、一致性、可修改性和优先级出发,对表 5.1 中的每一个检查项解释如下。

第一项需要检查需求规格说明书是否满足了用户提出的每一项需求,实现需求的完整性。

第二项需要检查需求文档的用词、用语问题,实现需求的明确性。

第三项检查的是需求规格说明书对需求覆盖是否准确,实现需求的必要性。

第四项检查的是软件使用环境的描述是否清晰,实现需求的完整性。

第五项检查的是需求规格说明书中的需求编号是否正确,实现需求的可修改性。

第六项主要检查需求是否是自相矛盾的，实现需求的一致性。

第七项主要检查软件系统允许的输入与预期的输出，实现需求的可测性。

第八项检查的是软件系统的性能需求有没有得到清晰的描述，实现需求的完整性。

第九项检查的是需求的关注重点和实现的先后顺序是否清晰地被描述出来，实现需求的优先级。

第十项检查是对软件系统的约束条件是否完整进行描述，实现需求的可测性。

表 5.1 测试需求检查项

序号	检 查 项	检查结果
1	是否覆盖了用户提出的所有需求项	是[ ] 否[ ] NA[ ]
2	用词是否清晰，语义是否存在有歧义的地方	是[ ] 否[ ] NA[ ]
3	是否清楚地描述了软件系统需要做什么及不做什么	是[ ] 否[ ] NA[ ]
4	是否描述了软件使用的目标环境，包括软硬件环境	是[ ] 否[ ] NA[ ]
5	是否对需求项进行了合理的编号	是[ ] 否[ ] NA[ ]
6	需求项是否前后一致、彼此不冲突	是[ ] 否[ ] NA[ ]
7	是否清楚系统的输入、输出格式，以及输入与输出之间的对应关系	是[ ] 否[ ] NA[ ]
8	是否清晰描述了软件系统的性能要求	是[ ] 否[ ] NA[ ]
9	需求的优先级是否合理分配	是[ ] 否[ ] NA[ ]
10	是否描述了各种约束条件	是[ ] 否[ ] NA[ ]

5.2.2 测试用例编写

测试用例编写的具体流程如下所示。

步骤 1：在分析清楚需求的前提下对测试活动进行计划和设计。

步骤 2：按既定的策划执行测试用例和记录用例。

步骤 3：对测试的结果进行检查分析，形成测试报告。

步骤 4：使用测试结果和分析报告又能指导下一步的测试设计。

步骤 5：形成了一个质量改进的闭环。

【例 5.1】 测试用例举例。

测试用例编号：input_001。

测试优先级：中等。

估计执行时间：2 分钟。

测试目的：验证业务单据数据的查询正确性。

标题：业务单据查询。

步骤如下：

步骤 1：打开查询界面。

步骤 2：输入查询条件。

步骤3：确定并提交查询。

步骤4：查看并验证返回信息。

【分析】 编写的测试用例回答如下问题。

(1) 可输入的查询条件包括哪些。

(2) 提交查询之前是否会验证输入数据的正确性。

(3) 输入数据的单位、范围有无限制。

(4) 所有条件都不输入是否意味着查询出所有业务单据。

5.3 测试计划

测试计划以测试需求为基础，分析产品的总体测试策略，输出《产品总体测试策略》等报告。

5.3.1 测试计划要点

测试计划规定测试任务、安排人员、预见风险，指导测试，实现测试的目标，一般测试计划要点如下所示。

1. 测试范围

确定各阶段的测试范围、技术约束等，以及测试成功的标准和要达到的目标。

2. 测试策略

开发有效的测试模型，决定黑盒测试和白盒测试、人工测试和自动化测试的比重等。

3. 测试资源

确定测试所需要的时间和资源，对人员、硬件和软件等资源进行组织和分配。

4. 进度安排

分解项目工作结构，并采用时限图、甘特图等方法制定时间/资源表。

5. 风险及对策

测试可能存在的风险分析，对风险进行回避、监控、管理，采用变更管理和控制等。

5.3.2 测试计划步骤

测试计划一般具有如下步骤。

步骤1：明确测试对象。

首先要明确测试的对象，有些对象是不需要测试的。有时候，测试的范围是比较难判断的。例如，对于一些整合性系统，它把若干个已有的系统整合进来，形成一个新的系统，那么就需要考虑测试的范围是包括所有子系统，还是仅仅测试接口部分，需要结合整合的

方式、系统之间通信的方式。

步骤 2 ：制定测试策略。

测试的策略包括宏观的测试策略和微观的测试策略战术。其中，测试战略和测试技术的关系如图 5.3 所示。

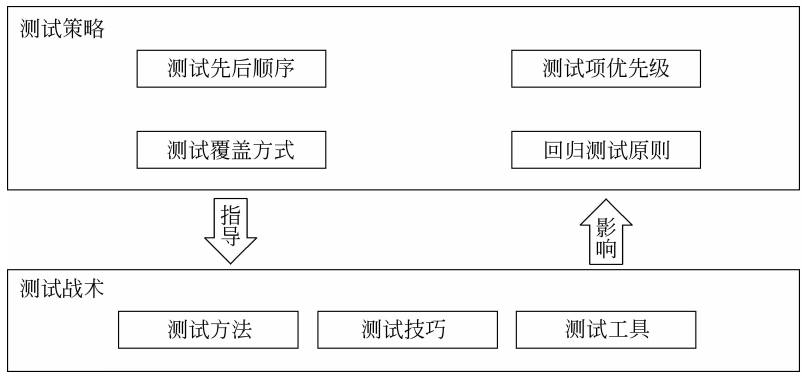

图 5.3　需求文档的检查流程

为了设计出好的测试策略，需要了解软件的结构、功能分布、各模块对用户的重要程度等，从而决定测试的重点、优先次序、测试的覆盖方式等。设计测试用例时，应尽可能用最少的测试用例发现最多的缺陷，尽可能用精简的测试用例覆盖最广泛的状态空间，还要考虑哪些测试用例使用自动化的方式实现，哪些使用人工方式验证等。

步骤 3：决定测试战术。

根据软件采用的技术、架构、协议等，考虑采用的测试方法和手段，是否需要进行白盒测试或黑盒测试，采用什么测试工具进行自动化测试等。制定好测试策略后，需要安排测试资源，通过充分估计测试的难度、测试的时间、工作量等因素，决定测试资源的合理利用方式。

步骤 4 ：安排测试进度。

测试的进度安排需要结合项目的开发计划、产品的整体计划进行考虑，还要考虑测试本身的各项活动进行安排。把测试用例的设计、测试环境的搭建、测试报告的编写等活动列入进度安排表，如图 5.4 所示。

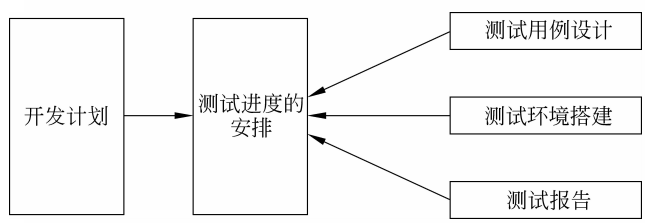

图 5.4　需求文档的检查流程

步骤 5：估计计划风险。

一般可能碰到的风险是项目计划变更、测试资源不能及时到位等问题。制定测试计

划时应根据项目的实际情况进行评估,并制定出合理、有效的应对策略。对于项目计划的变更,可以考虑建立更加流畅的沟通渠道,让测试人员能及时了解到变更的情况,以及变更的影响,从而可以作出相应的改变。

5.4 测试设计

测试设计建立在测试计划书的基础上,根据测试大纲、测试内容及测试的通过准则,将测试需求转换成测试用例的过程,用于描述测试环境、测试执行的范围、层次和用户的使用场景以及测试输入和预期的测试输出等信息,输出《产品或者版本总体测试方案》等报告。

基于需求的测试用例设计一般有如下方法。

方法一:临时的同行评审。

同行评审尤其是临时的同行评审,应该演变成类似结对编程一样的方式,体现敏捷的"个体和交互比过程和工具更有价值"这一原则,强调测试用例设计者之间的交流,通过讨论、协作来完成测试用例的设计。

方法二:用户参与评审。

用户参与评审体现了敏捷的"顾客的协作比合同谈判更有价值"这一原则,这里顾客的含义和如何定义测试有关。如果测试是对产品的批判,则顾客是指最终用户或顾客代表;如果测试是被定义为对开发提供帮助和支持,那么顾客显然就是程序员。

5.4.1 测试设计内容

软件测试设计主要内容如下所示。

(1) 制定测试的技术方案,确认各个测试阶段采用的测试技术、测试环境和平台,以及选择什么样的测试工具。其中,系统测试中的安全性、可靠性、稳定性、有效性等是测试技术方案的内容重点。

(2) 设计测试用例,根据产品需求分析、系统设计等规格说明书,在测试技术方案的基础上设计具体的测试用例。

(3) 根据测试的目的和任务,以及测试用例的特性和属性(优先级、层次、模块等)设计测试用例,从而构成执行某个特定测试任务的测试用例集合(组),如基本测试用例组、专用测试用例组、性能测试用例组、其他测试用例组等。

(4) 根据所选择的测试工具,将自动化测试的测试用例转换为测试脚本。

(5) 根据所选择的测试平台以及测试用例所要求的特定环境,进行服务器、网络等测试环境的设计。

软件测试设计中,需要考虑如下要点。

(1) 所设计的测试技术方案是否可行、是否有效、是否能达到预期的测试目标。

(2) 所设计的测试用例是否完整、边界条件是否考虑、其覆盖率能达到的百分比。

(3) 所设计的测试环境是否和用户的实际使用环境比较接近。

5.4.2 测试用例属性

设计测试用例主要根据测试用例的以下属性，并结合测试用例的编号、标题、描述（条件、步骤、期望结果）等进行测试用例管理。

1）优先级

测试用例的优先级越高，被执行的时间越早、执行的频率越多。由最高优先级的测试用例组来构成基本验证测试，每次构建软件包时，都要被执行一遍。

2）目标性

根据不同的目标设计测试用例。有的测试用例是为主要功能而设计，有的则为系统的负载而设计。

3）所属范围

根据测试用例所属不同的组件或模块进行管理。

4）关联性

测试用例一般和软件产品特性相联系，多数情况下验证某个产品的功能。

5）阶段性

根据不同的测试阶段，如单元测试、集成测试、系统测试、验收测试等设计测试用例，便于得出该阶段的测试覆盖率。

6）状态性

测试用例有不同的状态，只有被激活的测试用例才被运行。

7）时效性

针对同样功能，可能所用的测试用例不同，是因为不同的产品版本在产品功能、特性等方面的要求不同。

8）所有者

测试用例还包括由谁、在什么时间创建，又由谁、在什么时间修改等。

5.5 测试执行

当测试用例的设计和测试脚本的开发完成之后，就开始执行测试。测试执行是对每个测试阶段（单元测试、集成测试、系统测试和验收测试等）测试用例的编写和自动化脚本的编写，保证每个阶段的测试任务得到执行，输出《产品自动化测试用例》和《手工执行测试用例》等报告。

1）单元测试

单元测试的目的在于发现各模块内部可能存在的各种差错，一般由程序员执行，但须提交单元测试用例和测试报告，由测试人员进行审查。

2）集成测试

集成测试的主要目标是发现与接口有关的问题，对于关键模块应尽早测试，并将自顶向下、自底向上两种测试策略结合起来，严格执行各个模块。

3）系统测试

系统测试以用户环境模拟系统的运行，用于验证系统是否达到在概要设计中所定义的功能和性能。

4）验收测试

验收测试是指当技术部门完成了所有测试工作后，由用户参与的软件测试，通常采用α测试和β测试，用于确保产品能真正符合用户需求。

5.5.1 单元测试

单元测试是根据源程序、编程规范、产品规格设计说明书和详细的程序设计文档，以测试软件设计中的最小单位，例如，面向过程语言的函数或子过程。面向对象语言的类或成员函数用于发现语法、格式和逻辑等缺陷，从而判断特定条件或场景下函数的行为，输出缺陷跟踪报告。

单元测试一般有如下优点。

(1) 单元测试是验证行为。

程序中的每一项功能通过测试来验证其正确性，为其后代码的重构提供了保障。

(2) 单元测试是设计行为。

软件的设计考虑如何实现软件的某项功能、用户界面等，单元测试关注于软件的具体功能实现是否符合需求设计，而不仅仅定位于代码的实现运作机制上。

(3) 单元测试是编写文档的行为。

单元测试是表示函数或类如何使用的最佳文档。通过单元测试这份文档的编译、运行，从而保持与代码同步。

(4) 单元测试具有回归性。

自动化的单元测试避免了代码出现回归，编写完成之后，便于随时运行测试。

单元测试针对程序模块进行测试，主要有以下5个任务——模块接口、局部数据结构、边界条件、独立的路径和错误处理，如图5.5所示。

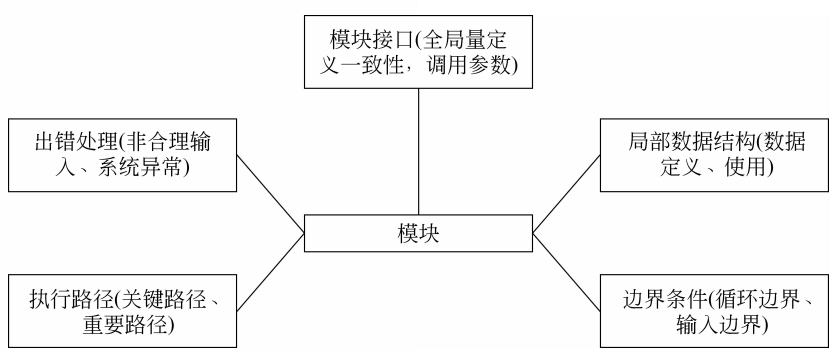

图5.5 单元测试解决的任务

1）模块接口测试

通过对被测模块的数据流进行测试，检查进出模块的数据是否正确。因此，必须对模

块接口，包括参数表、调用子模块的参数、全程数据、文件输入/输出操作进行测试。具体涉及以下内容。

(1) 模块接受输入的实际参数个数与模块的形式参数个数是否一致。

(2) 输入的实际参数与模块的形式参数的类型是否匹配。

(3) 输入的实际参数与模块的形式参数所使用单位是否一致。

(4) 调用其他模块时，所传送的实际参数个数与被调用模块的形式参数的个数是否相同。

(5) 调用其他模块时，所传送的实际参数与被调用模块的形式参数的类型是否匹配。

(6) 调用其他模块时，所传送的实际参数与被调用模块的形式参数的单位是否一致。

(7) 调用内部函数时，参数的个数、属性和次序是否正确。

(8) 在模块有多个入口的情况下，是否有引用与当前入口无关的参数。

(9) 是否修改了只读型参数。

(10) 全局变量是否在所有引用它们的模块中都有相同的定义。

如果模块内包括外部 I/O，还应该考虑下列因素。

(1) 文件属性是否正确。

(2) OPEN 与 CLOSE 语句是否正确。

(3) 缓冲区容量与记录长度是否匹配。

(4) 在进行读写操作之前是否打开了文件。

(5) 在结束文件处理时是否关闭了文件。

(6) 正文书写/输入错误。

(7) I/O 错误是否检查并做了处理。

2) 模块局部数据结构测试

测试用例检查局部数据结构的完整性，如数据类型说明、初始化、缺省值等方面的问题，并测试全局数据对模块的影响。

(1) 在模块工作过程中，必须测试模块内部的数据能否保持完整性，包括内部数据的内容、形式及相互关系不发生错误。

(2) 局部数据结构应注意以下几类错误：不正确的或不一致的类型说明；错误的初始化或默认值；错误的变量名，如拼写错误或书写错误；下溢、上溢或者地址错误。

3) 模块中所有执行路径测试

测试用例对模块中重要的执行路径进行测试，其中，对基本执行路径和循环进行测试往往可以发现大量路径错误。测试用例必须能够发现由于计算错误、不正确的判定或不正常的控制流而产生的错误。

(1) 常见的错误如下所示。

误解的或不正确的算术优先级、混合模式的运算、错误的初始化、精确度不够精确、表达式的不正确符号表示。

(2) 针对判定和条件覆盖，测试用例能够发现如下错误。

不同数据类型的比较；不正确的逻辑操作或优先级；应当相等的地方，由于精确度的错误而不能相等；不正确的判定或不正确的变量；不正确的或不存在的循环终止；当遇到

分支循环时不能退出;不适当地修改循环变量。

4) 各种错误处理测试

检查模块的错误处理功能是否包含有错误或缺陷。例如,是否拒绝不合理的输入;出错的描述是否难以理解、是否对错误定位有误、是否出错原因报告有误、是否对错误条件的处理不正确;在对错误处理之前,错误条件是否已经引起系统的干预等。

(1) 测试出错处理的重点是模块在工作中发生了错误,其中的出错处理设施是否有效。

(2) 检验程序中的出错处理可能面对的情况如下。

① 对运行发生的错误描述难以理解。

② 所报告的错误与实际遇到的错误不一致。

③ 出错后,在错误处理之前就引起系统的干预。

④ 例外条件的处理不正确。

⑤ 提供的错误信息不足,以至于无法找到错误的原因。

5) 模块边界条件测试

(1) 边界测试是单元测试的最后一步,必须采用边界值分析方法来设计测试用例,为限制数据处理而设置的边界处,测试模块是否能够正常工作。

(2) 一些与边界有关的数据类型,如数值、字符、位置、数量、尺寸等特征。

(3) 在边界条件测试中,应设计测试用例检查以下情况。

① 在 n 次循环的第 0 次、1 次……n 次是否有错误。

② 运算或判断中取最大值、最小值时是否有错误。

③ 数据流、控制流中刚好等于、大于、小于确定的比较值是否出现错误。

在源程序代码编制完成,经过评审和验证,确认没有语法错误之后,开始设计单元测试的测试用例。由于模块并不是一个独立的程序,考虑测试模块时,同时要考虑它和外界的联系,因此使用一些辅助模块去模拟与被测模块相关的其他模块。辅助模块分为驱动模块和桩模块两种。

① 驱动模块。

驱动模块用来模拟被测试模块的上一级模块,相当于被测模块的主程序,用于接收测试数据,并把这些数据传送给被测模块,启动被测模块,最后输出实测结果。

② 桩模块。

桩模块用来模拟被测模块工作过程中所调用的模块。桩模块一般只进行很少的数据处理,不需要把子模块所有功能都带进来,但不允许什么事情也不做。

被测模块、驱动模块及桩模块共同构成了一个测试环境,如图 5.6 所示。

5.5.2 集成测试

时常发生这样的情况,每个模块都能单独工作,但将这些模块组装之后却不能正常工作。导致这种情况的可能原因如下所示。

(1) 模块相互调用时引入了新的问题,例如数据可能丢失、模块之间的相互影响等。

(2) 子模块分别实现了子功能,但组合后无法实现主功能。