**一、测试基础**

软件测试是属于研发流程中的一个环节

**软件生命周期**

计划：项目经理确定项目的目标、时间、预算等，输出《项目计划》；

需求：需求分析师与用户进行沟通，将用户需求转化为软件需求，输出《需求规格说明书》SRS；

设计：系统架构师设计软件的内部结构(框架)，输出《概要设计说明书》HLD，某个角色编写《详细设计说明书》LLD；

编码：开发工程师进行编码，用代码来实现软件的功能；

测试：测试工程师对研发的软件进行验证，确保软件实现了软件需求、满足用户需求；

维护：运维工程师把软件部署到正式的运行环境，处理软件在运行时出现的异常，收集问题进行二次开发。

一个软件的研发流程，从开始到结束的过程；对应软件的一个版本。

**软件测试的价值**

通过软件测试能发现软件存在的Bug；

通过软件测试能够提升软件的质量；

通过软件测试能够对软件的优缺点有更清楚的认识。

软件测试的工作需要有专业的工作流程，需要有独立的分析与设计能力，软件对质量的要求越高，对测试工程师的需求量就越大。

**软件测试的目的**

验证软件的基本功能和业务流程，证明软件是可以使用的；

尽可能的测试到用户能做的所有操作场景，证明软件是存在Bug的；

尽早的介入测试，能够降低Bug修复的成本，避免Bug的产生。

在不同的研发阶段，软件测试的目的并不相同，先证明软件可用 再尽可能多的发现Bug。

**什么是Bug？**

一切影响软件被正常使用的原因或现象都叫Bug

错误：人为主观判断

缺陷：软件的功能存在缺失或者漏洞，缺陷是引起故障的原因（登录代码的密码验证有问题）

故障：故障是软件存在问题的表现形式（错误密码也能进行登录）

失效：软件在运行时，表现的一个不可接受的现象（用户账号安全得不到保障）

软件因为存在缺陷所以会引起故障，故障如果没有合理解决会导致软件的功能失效

很多时候测试工程师发现的Bug仅是一个错误、故障或者失效，开发修复Bug需要定位到产生故障的原因上才行。

**研发的组织架构**

* 开发部门：主要负责产品的设计与开发

需求分析师：负责产品的需求分析与需求制定；

系统架构师：负责产品的架构设计；

UI前端工程师：负责产品的界面和外观实现；

开发工程师：负责产品的逻辑和功能实现；

* 测试部门：主要负责产品的验证与确认

测试工程师：负责需求的验证，产品的检测；

测试开发工程师：负责实现复杂的测试场景，或者提高测试效率；

QC：产品质量评测

* 质量部门：主要负责产品的研发进度与研发质量

管理岗位：保证研发进度

QA：监督软件研发过程中输出的文档的规范性与及时性

CMO：配置库管理员，管理研发的配置库(文件库)

大公司才会岗位职责明确，分工细致；小公司一般都是身兼多职。

**研发流程**

* 瀑布模型：按照软件生命周期的顺序从前往后进行

阶段明确，便于分析问题

适用于传统的项目，一般1-3个月更新一次版本

* 螺旋模型：将需求分段实现，每段需求的实现都是一个瀑布模型，每段需求实现完成之后都会进行风险评估

需求分段实现能实时的发现项目风险，降低项目风险

适用于大型项目的研发过程，一般会经历半年以上的研发时间

* 敏捷迭代：

1、将需求进行细化与拆分（理论上不允许有13天工作量以上的需求）

2、将各部门拆散，需求、设计、开发、测试组成小的团队来认领需求

3、省去大部分的编写文档的工作，强调沟通与交流

适用于互联网类的软件研发，1-2周更新一次版本，能够快速的响应市场需求。

对员工的个人能力、沟通交流能力，对团队的默契要求较高。

**二、测试过程**

**软件测试的4个阶段**

* 单元测试 Unit Testing

测试软件的最小单元(函数)，验证软件是否满足LLD（详细设计说明书）的要求

例如：网购的衣柜，第一步需要检查零件是否完好和齐全

* 集成测试 Intergration Testing

测试软件组装之后的模块，以及模块与模块之间的接口，验证软件是否满足HLD（概要设计说明收）的要求

例如：衣柜的底座与四根支柱的组合，支架与衣柜罩子是否能够良好组合

* 系统测试 System Testing

测试软件的整体外部特性(功能、易用性、性能等)，验证软件是否满足SRS（需求规格说明书）的要求

例如：衣柜能否打开与关闭、衣柜长宽高、外观、负重等

* 验收测试 Acceptance Tesing

从用户的角度来验证软件是否满足用户的需求

α测试：在受控制的环境进行验收测试，例如游戏内测

β测试：在不受控制的环境进行验收测试，例如游戏公测

正式验收测试：在客户现场由客户代表进行确认结果的验收测试

UT --> IT --> ST --> AT

UT --> SIT --> UAT

当前IT行业，UT测试一般由开发自测，测试进行UT投入成本较高。

**回归测试**

验证Bug是否被正确修复的过程

所有的测试阶段都有可能发现Bug，都需要进行回归测试

* 回归测试需要测试什么?回归测试策略

完全回归测试很好，消耗时间和人力，一般不推荐

选择性测试：

1、已发现的Bug被正确修复，并且没有引入新的Bug

2、用户频繁使用的功能

3、软件的整体业务流程通畅

**软件测试的4个活动**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **计划** | **设计** | **实现** | **执行** |
| **测试经理**  **测试组长** | 1.输出《测试计划》  人员分工、时间安排等  2.组织需求评审。  3.组织《测试计划》评审。 | 1.输出《测试方案》  制定测试方法、测试工具、测试规程等；  2.组织测试思路评审； | 组织测试用例评审； | 1.搭建测试环境；  2.组织每日例会；  3.汇总日报，编写测试组的测试日报；  4.跟踪小组的工作进度；  5.处理工作阻塞性问题；  6.测试执行完成后，进行测试分析，输出测试报告。 |
| **测试工程师** | 1.参与需求评审。  2.参与《测试计划》评审。 | 1.画流程图理解业务；  2.进行测试需求分析，梳理测试思路；  3.评审测试思路； | 编写测试用例；  用例评审； | 1.冒烟测试；  2.执行测试用例；  3.提交跟踪Bug；  4.选择性回归测试；  5.提交工作日报； |

* 计划 测试经理确认当前版本的工作范围、时间安排、人员分工、风险预估、工作规范，输出《测试计划》

从管理的角度规划版本的软件测试工作，who what when

* 设计 高级测试工程师 根据需求和测试计划 从技术角度规划当前版本的测试工作如何实现，测试关注点、测试方法、测试用例设计、环境安排等，输出《测试方案》

从技术的角度设计软件测试工作如何实现，how

* 实现 测试工程师根据软件需求、测试计划、测试方案，把需求转化为一个可执行的文档《测试用例》
* 执行 搭建测试环境（操作系统、网络、数据库等技术）

冒烟测试 （验证软件最基础的功能，确认当前版本的需求基本实现，保证测试用例可以正常执行）

写好测试用例后，从中抽取20%左右的测试用例进行冒烟测试

执行测试用例（按照测试用例中的操作步骤操作软件，对比软件的实际结果和用例的预期结果，不一致则有可能是Bug）

提交和跟踪Bug（测试人员发现Bug，开发修复Bug，测试人员验证。期间需要测试协助开发重现和定位Bug）

选择性回归测试

写日报，汇报每天的进度和问题（测试经理/测试组长，需要根据组员每天反馈的进度来把控项目进度，需要了解组员遇到的问题及时去协调解决，保证组员能够顺利完成任务）

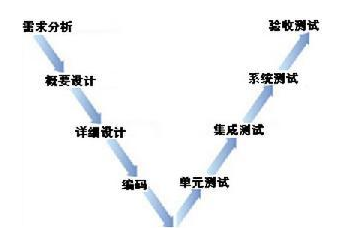
写测试报告 （由测试经理/测试组长编写，汇总本版本的工作任务和成果，确认该版本测试是否通过，项目经理根据测试报告来判断软件是否可以发布）

**V模型与双V模型**

* V模型 描述了软件研发过程中，每个工作的开展顺序

会让人误以为测试工作是在编码完成之后才开始的

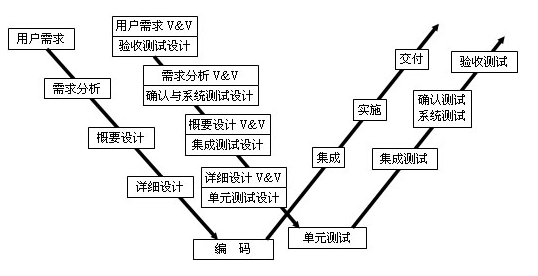
实际测试工作是从需求分析阶段就已经开始



* 双V模型 开发工作和测试工作是并行的

设计顺序和执行顺序是相反的

设计工作与执行工作是分离的，先设计，再执行



设计顺序：ST-->IT-->UT

执行顺序：UT-->IT-->ST

搭建环境就是开发发布的程序包（包含客户端与服务器）进行安装和部署，保证在测试环境中能够正常的使用被测软件

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 需求分析 | 概要设计 | 编码 | 测试 |
| 开发 | 评审需求 | 评审架构  详细设计 | 写代码 | 改Bug |
| 测试 | 评审需求  测试需求分析  编写测试计划 | 写测试方案  写测试用例 | | 搭建测试环境  冒烟测试  执行测试用例  回归测试  提交跟踪Bug  日报  测试报告 |

测试规程：描述测试用例执行的顺序

搭建环境就是开发发布的程序包（包含客户端与服务器）进行安装和部署，保证在测试环境中能够正常的使用被测软件

----------------------------------------------------------------------------------------------------

**一、测试方法**

测试方法 != 测试阶段

**黑白灰盒测试**

* 黑盒测试

不关注内部逻辑，仅关注外部功能的测试（只关注输入和输出）；

系统测试阶段，仅测试软件的外部功能

集成测试阶段，仅测试模块的外部功能

单元测试阶段，仅测试函数的外部功能

* 白盒测试

仅关注内部逻辑，不关注外部功能的测试

单元测试阶段，仅测试函数的内部逻辑与编写规范

* 灰盒测试

既关注内部逻辑，又关注外部功能的测试

集成测试阶段，既测试模块的外部功能，又测试接口的内部逻辑

**动静态测试**

* 动态测试

运行被测试对象所开展的测试

例如：调试代码，运行软件进行功能测试

* 静态测试

不动力被测试对象所开展的测试

例如：走读代码，需求评审

使用工具进行自动化的代码扫描属于静态测试

**人工自动化测试**

* 人工测试

手工执行的测试叫人工测试

* 使用工具、代码、脚本执行的测试叫自动化

1、自动化测试能够缩短测试时间、节省测试人力

2、简单重复的工作可以靠工具，需要智力的工作靠人力

3、人工测试是自动化测试的基础，只有人工测试足够成熟了才可以考虑开始自动化

4、测试工作中（更新测试版本、冒烟测试、回归测试）比较适合做自动化，业界称这一套操作为“持续集成”

**二、软件质量**

质量就是用户对产品的感受。

* **影响质量的因素**

人：研发人员的个人素质与道德修养能影响软件的质量；

技术：采用更成熟、更先进的技术，能够使软件具备更强大的功能与性能；

流程：严谨规范的开发流程，能够保障软件的质量；

* **质量认证**

ISO：全行业的质量认证体系

CMM：软件能力成熟度模型

|  |  |  |  |
| --- | --- | --- | --- |
| **能力等级** | **特点** | **关键过程** | **备注** |
| 第一级 初始级（最低级） | 软件工程管理制度缺乏，过程缺乏定义、混乱无序。成功依靠的是个人的才能和经验，经常由于缺乏管理和计划导致时间、费用超支。管理方式属于反应式，主要用来应付危机。过程不可预测，难以重复。 |  | 无明确计划无规范管理，应急式工作 |
| 第二级 可重复级 | 基于类似项目中的经验，建立了基本的项目管理制度，采取了一定的措施控制费用和时间。管理人员可及时发现问题，采取措施。一定程度上可重复类似项目的软件开发。 | 需求管理，项目计划，项目跟踪和监控，软件子合同管理，软件配置管理，软件质量保障 | 有计划和管理，按照规划工作 |
| 第三级 已定义级 | 已将软件过程文档化、标准化，可按需要改进开发过程，采用评审方法保证软件质量。可借助CASE工具提高质量和效率。 | 组织过程定义，组织过程焦点，培训大纲，软件集成管理，软件产品工程，组织协调，专家审评 | 研发阶段定义明确，项目组成员工作规范有初步的定义，有评审工作 |
| 第四级 已管理级 | 针对制定质量、效率目标，并收集、测量相应指标。利用统计工具分析并采取改进措施。对软件过程和产品质量有定量的理解和控制。 | 定量的软件过程管理和产品质量管理 | 每个工作阶段和工作成果，都有相应的指标来定义其完成质量 |
| 第五级 优化级（最高级） | 基于统计质量和过程控制工具，持续改进软件过程。质量和效率稳步改进。 | 缺陷预防，过程变更管理和技术变更管理 | 有应急预案，有定期度量，有改进计划 |

6Sigma 六西格玛 6σ：

统计软件的错误率

6个西格玛=3.4失误/百万机会―意味着卓越的管理，强大的竞争力和忠诚的客户 0.00034%

5个西格玛=230失误/百万机会－优秀的管理、很强的竞争力和比较忠诚的客户 0.023%

4个西格玛=6,210失误/百万机会－意味着较好的管理和运营能力，满意的客户 0.621%

3个西格玛=66,800失误/百万机会－意味着平平常常的管理，缺乏竞争力 6.68%

2个西格玛=308,000失误/百万机会－意味着企业资源每天都有三分之一的浪费 30.8%

1个西格玛=690,000失误/百万机会－每天有三分之二的事情做错的企业无法生存 69%

* **软件质量模型**

内部质量：产品研发过程中的质量（研发人员、研发技术、研发流程）

可维护性：易分析、易改变、易测试、依从性

外部质量：产品从发布到交付给客户过程中的质量（产品本身、用户手册、技术支持、使用培训）

测试需要重点关注软件的外部质量，学习质量模型为测试需求分析提供**思路**

功能性：适合性、准确性、互操作性、保密安全性、依从性

效率性：时间特性、资源占用、依从性

易用性：易理解、易学、易操作、吸引性、依从性

可靠性：成熟性、容错性、易恢复性、依从性

可移植性：适应性、易安装、易替换、共存、依从性

使用质量：用户在使用产品的过程中的质量（售后服务）

**三、覆盖率**

覆盖率是衡量测试是否充分的一种标准；

覆盖率一般情况下很难达到100%；

测试没有必要追求100%覆盖；

* 需求覆盖率/功能覆盖率

已测试需求/总需求数

一个版本的测试不要求所有的需求都测试到，但是当前版本实现的需求，必须100%覆盖

* 接口覆盖率

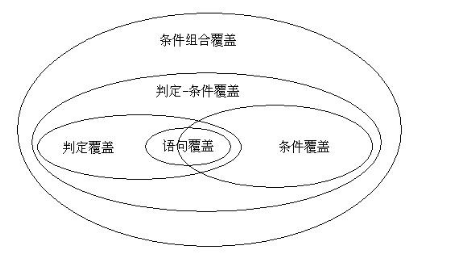
已测试接口/总接口数

当前版本开发涉及到的接口，必须100%覆盖

* 函数覆盖率

当前版本开发涉及到的函数，必须100%覆盖

* 逻辑覆盖率



----------------------------------------------------------------------------------------------------

**一、测试用例**

测试用例是依据需求编写的一个可以执行的文档；

测试人员将测试内容的分析设计与执行分离操作，可以避免在执行的过程中出现遗漏；

**测试用例要素**

* 用例编号(用例标识)

用于对测试用例的唯一标识，通过一个用例编号能够查找到一条指定的用例

编号规则：项目+版本号+模块+序列号，例如：FeiQv2.4\_SharedFile\_001

* 测试项目

要测试的模块

* 用例标题

简洁清晰的描述该条测试用例所要测试的内容

在敏捷模式中，测试人员编写测试用例仅需要编写用例标题，省去了很多的写用例的时间

建议大家在写用例时，优先完成用例标题的编写，后面的内容再依据标题的思路来补充完整

* 重要级别

分为高、中、低三个级别

体现测试用例的执行优先级，先执行级别为高的用例

例如：登录成功优先级为高，密码错误的优先级为中

* 预置条件

执行该条测试用例所需要进行的前期准备/必要条件；

该项不是一定要写的内容

* 测试输入

1、将操作步骤中需要输入的内容写到“测试输入”项中，可以避免操作步骤过多而导致用例难以看懂和执行；

2、对于相同的操作步骤，不同的测试输入，预期结果一致的测试用例，可以合并成一条；

3、对于相同的操作步骤，不同的测试输入，预期结果不一致的测试用例，要求分成多条测试用例；

* 操作步骤

将用例的执行操作，分步骤逐条说明

测试人员在执行测试用例时，就会参考操作步骤逐条执行

* 预期结果

依据用例中所描述的操作，对软件将会作出的动作进行的预期

在执行用例时，如果软件的实际结果与用例的预期结果不一致，则有可能是Bug

**二、缺陷管理**

测试人员发现Bug之后，需要将Bug提交给开发人员修改，等开发人员修改完成之后需要测试人员验证；

缺陷的提交需要规范，缺陷的状态需要跟踪，缺陷的流程需要管理

**缺陷要素**

* 缺陷编号

缺陷单的唯一标识，一般由系统自动算成，不需要人为关注；

* 缺陷标题

要求简洁清晰的描述Bug

在XX位置进行XX操作得到XXX不正确的结果

例如：切换到新局域网中，飞秋必须先刷新好友才能看到新局域网内共享的文件

切换到新的局域网中，飞秋无法看到新局域网内共享的文件，除非刷新好友

* 测试环境

测试人员操作出现Bug的操作系统、浏览器、服务器版本等，便于开发重现、定位、分析Bug的原因

* 操作步骤

不准复制测试用例步骤！！！

* 预期结果

针对操作步骤应该出现的正确结果

* 实际结果

软件所体现出的Bug现象

* 重要级别

描述缺陷的影响范围

致命：导致软件崩溃，或者存在影响软件使用的问题；例如：飞秋查找不到局域网中的联系人

严重：软件存在问题影响用户正常操作； 例如：飞秋查看不到共享的文件，收发消息有60秒延时

一般：软件存在问题不影响用户正常操作； 例如：必须通过手动刷新好友，才能获取新的共享文件

提示：主观意识上的错误，一般都是优化性的建议； 例如：FeiQ界面长得丑

* 优先级

优先级是开发修改Bug顺序的依据，分为高、中、低

严重的Bug优先级不一定高，提示的Bug优先级不一定低

* 引入阶段

描述造成缺陷的原因，需求描述不清、设计不合理、代码错误、测试操作错误等；

在测试人员提交缺陷时，往往不能很准确的填写引入阶段，所建议不知道就填代码错误；

* 状态

**缺陷状态**

* 已提交/待确认 New

测试人员发现Bug之后，提交缺陷单，待测试经理/开发经理确认

* 已确认/待处理 Open

测试经理/开发经理确认Bug有效，指派给开发修改

* 已处理/待验证 Fixed

开发修复Bug，提交给测试人员验证

* 已关闭 Closed

测试人员验证Bug已修改，关闭缺陷单，流程结束

* 已挂起 Delayed

需求、测试、开发等成员承Bug有效，但是当前版本不修改，流程结束

* 已驳回 Rejected

测试经理/开发经理，开发人员认为Bug无效，驳回给测试人员

* 待处理 Reopen

测试人员验证Bug未修改，重新提交给开发处理

new --> open、rejected

open --> fixed、delayed、rejected

fixed --> closed、reopen

closed -->

delayed -->

rejected --> new、open

reopen --> fixed、delayed、rejected

**三、系统测试**

**系统测试类型**

* 功能测试

数据的增删改查，数据的提交审批驳回，互操作性等

* 性能测试

测试软件的性能指标：响应时间、并发用户数、吞吐量(单位时间内处理事务的数量)、资源占用

负载测试：逐渐增加压力，观察软件在不同压力级别下性能指标的变化情况

压力测试：逐渐增加压力，直到软件出现大量报错或者软件崩溃，获取软件所能提供的最大服务级别

容量测试：设置一个不可接受的性能点，测试软件能容纳的最大用户数量；

* 易用性测试

GUI测试：用户界面测试，测试界面布局、字体大小、摆放位置、颜色等

图标是否易懂、操作错误提醒、必填项标识、提示信息

快捷键、右键菜单、全选等能够提高用户操作效率的设置

* 兼容性测试

服务器兼容

操作系统：Windows Server、Windows、Linux、Unix

web兼容

操作系统：Win7、Win10、MacOS

浏览器：Chrome、IE、Firefox、Safari

根据浏览器内核，自己百度

分辨率：1366\*768、1920\*1080

app兼容

操作系统：Android、IOS各版本

机型：手机、平板、智能手表

品牌：苹果、华为、小米、VIVO、魅族、OPPO

屏幕尺寸：6.4英寸、5.5英寸、4.7英寸

信号源：2G/3G/4G/5G，WiFi，移动、电信、联通

* 安全性测试

防攻击：验证码

保护数据：数据加密

* 安装卸载测试

安装前的数据包的完整性

安装过程的正确性

安装后的软件能使用

卸载功能的正确性

* 异常测试

弱网测试：挑选信号不好的地方对软件进行测试

低电量情况下的测试

使用过程中突然断网，数据的可恢复性

使用过程中突然断电，数据的可恢复性

* 稳定性测试/健壮性测试

软件在承受正常压力情况下，长时间运行的资源占用和事务成功率

**系统测试流程**

* 计划

熟悉需求、编写测试计划

* 设计

进行测试需求分析，编写测试点，编写测试方案

* 实现

设计测试用例

* 执行

搭建测试环境

冒烟测试

执行测试用例

提交跟踪缺陷单

回归测试

写测试日报

编写测试报告

**项目一工作任务** 测试笨笨记账本软件

1、熟悉需求，熟悉软件

2、进行测试用例的评审（熟悉测试用例）

3、执行测试用例（建议2人合作完成）

4、将Bug提交到ZenTao项目管理系统 禅道

5、提交测试日报（反馈工作进度和问题）

禅道环境安装与部署

1、运行ZenTaoPMS.8.2.6.exe，释放文件到桌面（不要放在有中文的目录！！！）

2、启动禅道，访问禅道，开源版

3、将禅道设置为开机不自动启动

Win+R --> services.msc --> 将服务apachezt和mysqlzt属性设置为手动

4、使用管理员账号登录 admin/123456

5、添加用户账号

6、添加产品，添加项目

7、设置团队 将添加的账号添加到项目中

8、一个团队仅需要安装一个禅道

http://IP地址/