# 测试手册

上海岱劲信息技术有限公司

2019/5/15

目录

**[一 概述](#_Toc16846_WPSOffice_Level1)** **[3](#_Toc16846_WPSOffice_Level1)**

**[二 测试流程](#_Toc25821_WPSOffice_Level1)** **[3](#_Toc25821_WPSOffice_Level1)**

**[三 测试阶段](#_Toc8888_WPSOffice_Level1)** **[5](#_Toc8888_WPSOffice_Level1)**

[1单元测试](#_Toc25821_WPSOffice_Level2) [5](#_Toc25821_WPSOffice_Level2)

[2集成测试](#_Toc8888_WPSOffice_Level2) [5](#_Toc8888_WPSOffice_Level2)

[2.1 集成测试的目的](#_Toc24021_WPSOffice_Level2) [5](#_Toc24021_WPSOffice_Level2)

[2.2 集成测试的关注点](#_Toc19743_WPSOffice_Level2) [5](#_Toc19743_WPSOffice_Level2)

[2.3集成测试策略](#_Toc24913_WPSOffice_Level2) [5](#_Toc24913_WPSOffice_Level2)

[2.4集成测试的层级](#_Toc29373_WPSOffice_Level2) [6](#_Toc29373_WPSOffice_Level2)

[2.5 集成测试的产出物](#_Toc14306_WPSOffice_Level2) [6](#_Toc14306_WPSOffice_Level2)

[3系统测试](#_Toc4581_WPSOffice_Level2) [6](#_Toc4581_WPSOffice_Level2)

[3.1功能测试](#_Toc1652_WPSOffice_Level2) [6](#_Toc1652_WPSOffice_Level2)

[3.2性能测试](#_Toc25650_WPSOffice_Level2) [6](#_Toc25650_WPSOffice_Level2)

[4验收测试](#_Toc3878_WPSOffice_Level2) [6](#_Toc3878_WPSOffice_Level2)

**[四 测试策略:](#_Toc24021_WPSOffice_Level1)** **[7](#_Toc24021_WPSOffice_Level1)**

[4.1 WEB功能测试（黑盒测试）：](#_Toc7789_WPSOffice_Level2) [7](#_Toc7789_WPSOffice_Level2)

[4.2 APP测试：](#_Toc27172_WPSOffice_Level2) [8](#_Toc27172_WPSOffice_Level2)

[4.3接口测试：](#_Toc23693_WPSOffice_Level2) [9](#_Toc23693_WPSOffice_Level2)

[4.4接口自动化的持续集成](#_Toc32275_WPSOffice_Level2) [9](#_Toc32275_WPSOffice_Level2)

[4.5 WEB端的UI自动化：](#_Toc749_WPSOffice_Level2) [13](#_Toc749_WPSOffice_Level2)

[4.6 APP端的UI自动化：](#_Toc32009_WPSOffice_Level2) [14](#_Toc32009_WPSOffice_Level2)

**[五 测试文档的管理：](#_Toc19743_WPSOffice_Level1)** **[15](#_Toc19743_WPSOffice_Level1)**

## 一 概述

## 编写此手册是为了明确测试各阶段工作，让开发了解测试的流程。提高测试效率，保证产品测试质量，从而保证产品开发工期与质量，统一测试思想是十分必要的。

## 二 测试流程

流程图

## 

## 测试流程

需求评审结束后，测试人员根据需求文档,原型和接口文档开始测试用例的编写而后进行测试用例的评审。测试用例的评审应该有两轮。一轮组内的，主要参与人员主要是负责相关模块的测试人员。一轮组外的，主要的参与的人员是整个项目组的人员包括开发，产品，测试。测试用例评审后提交文档管理工具。开发提测后开始执行测试用例，如果发现BUG，通过BUG管理工具提交到相应开发，开发修复后进行反测，通过关闭，不通过继续修复，直到BUG的修复。最后还要写一份测试报告抄送给相关负责人和领导。

## 三 测试阶段

### 1单元测试

又称为模块测试，是针对软件设计的最小单位程序模块进行正确检测的测试工作，多模块可以平行的独立进行单元测试，单元测试为开发自测。

主要目的：

确保每个模块能正常工作。

2集成测试

在单元测试的基础上把已经测试过的模块组装起来，相当于接口测试。

它的最简单的形式：两个已经测试过的单元组合成一个组件，并且测试他们之间的接口。从这一层意义上讲，组件是指多个单元的集成聚合。

2.1 集成测试的目的

验证代码接口的正确性

2.2 集成测试的关注点

1将各个具有相互调用关系的模块组装起来时，检查穿越模块接口的数据是否会丢失。

2判断各个子功能组合起来是否能够达到预期要求的父功能。

3检查一个模块的功能是否对其他模块的功能产生不良影响。

4检查全局数据结构是否正确，以及在完成模块功能的过程中是否会被异常修改。

#### 2.3集成测试策略

使用一次性集成的集成策略。首先对每个模块分别进行模块测试，然后将所有模块集成起来在一起进行测试，最终得到符合要求的软件系统。

#### 2.4集成测试的层级

1 模块内的集成测试

2 子系统之内的集成测试

3 子系统之外的集成测试

#### 2.5 集成测试的产出物

Postman接口测试脚本

3系统测试

将通过集成测试的系统，作为计算机系统的一部分，与系统中其他部分结合起来，在实际运行环境下对计算机系统进行的一系列严格有效的测试，以发现软件潜在的问题，保证系统的正常运行。

3.1功能测试

主要使用黑盒测试*，对软件需求文档中的功能需求逐项的测试，验证功能是否满足需求。测试用例编写时应更新需求追踪矩阵。*

3.2性能测试

主要包括压力测试和负载测试。

**3.2.1什么时候做性能测试？**

凡涉及到高并发的系统都是需要做性能的。在做性能的时候要考虑到系统是内部使用还是对外，系统使用的人数是多少。如果使用的人数较为少，那么无论系统多大，业务逻辑多么的复杂，都是没有必要的。当然除非在功能测试阶段发现非常明显的性能问题，使得用户体验较差的，此时可以做性能测试

**3.2.2性能测试的流程:**

1首先确定具体的非功能性指标，关键的指标包括响应时间，错误率，吞吐量，cpu，闪存等。

2准备性能测试案例，主要针对核心功能，准备测试数据。

3搭建测试环境，必须保证性能测试环境干净，否则会造成测试结果的偏移。常用的性能测试工具包括jemter，loadrunner等

4 性能测试采用的是逐步施压的机制，在性能指标都比较稳定的情况下在进行施压，在遇到测试软件本身的瓶颈或者测试机硬件性能瓶颈时可以采用分散式施压的方式

5负载测试时间上采用1\*24、3\*24、7\*24的原则

4验收测试

UAT，英文User Acceptance Test的简写，也就是用户验收测试。有专门的UAT测试小组，主要成员包括产品经理，用户验收人员，测试人员。如果用户验收人员需要技术部提供UAT测试用例，需要在技术部相关人员提供用例后，对用例进行确认。

#### 4.1 UAT测试条件：

1 UAT环境又叫预生产环境，环境的软硬件配置要和生产环境一致，且在开始UAT测试之前，产品经理需要进行确认测试，主要验证UAT环境的正常使用，以及主功能的实现

2 UAT测试用例准备完毕，只覆盖主功能即可，需要用户方确认。

3 开发的代码处于冻结状态

#### 4.2 UAT测试的产出物：

1用户确认验收书,需要交付纸质确认文件

2 问题记录文档，主要记录在UAT测试中发现的问题。产品经理需要以需求的方式记录在RedMine，便于后期的追踪和优化。

3 UAT测试用例

## 四 测试策略:

### 4.1 WEB功能测试（黑盒测试）：

从理论上来看，黑盒测试只能采用穷举输入测试，把所有可能的输入都作为 测试情况考虑，才能查出程序中所有的错误，但是实际测试情况有无穷多个，这个不可能测试完的，所以我们要进行有针对性的测试黑盒测试行为必须能够加以量化才能保证软件质量，测试用例就是将测试行为具体量化的方法之一，通过测试用例的执行，保证软件测试有序进行。

#### 常用的黑盒测试方法;

1等价类：

分为有效等价类和无效等价类。

有效等价类：是指对于程序的规格说明来说是合理的，有意义的输入数据构成的集合.利用有效等价类可检验程序是否实现了规格说明中所规定的功能和性能。

无效等价类：和有效等价相反

2边界值：

边界值分析是通过选择等价类边界的测试用例。边界值分析法不仅重视输入条件边界，而且也必须考虑输出域边界。它是对等价类划分方法的补充。

3因果图：

前面介绍的等价类划分方法和边界值分析方法，都是着重考虑输入条件，但未考虑输入条件之间的联系，相互组合等。 如果对输入条件之间进行相互组合，可能会产生一些异常的情况。因此必须考虑采用一种适合于描述对于多种条件的组合，相应产生多个动作的形式来考虑设计测试用例。

4错误推断法

是测试人员基于经验和直觉推测程序中所有可能存在的各种错误，从而有针对性的设计测试用例的方法.

5场景分析法

站在用户的角度模拟可能产生的场景

#### Web的兼容性：

主要检查主流浏览器的兼容性如IE Firefox chrome。确保系统能在不同的浏览器上进行使用。

#### 做功能测试主要的关注点：

***功能的正常使用，异常处理，前端兼容性，业务逻辑。***

### 4.2 APP测试：

#### 和WEB测试的相同点和异同点：

##### 相同点：

***它们的测试方法都一样。***

##### 不同点：

***1 app是基于手机进行展示，web是基于浏览器进行展示***

***2 app测试还需要测试：***

1.不同的系统（Android和ios）

2.不同的品牌的手机

3.不同的系统版本

4.不同分辨率

5.弱网测试：可以使用fiddler 或者 Charles抓包工具模拟

6.中断测试：当APP在运行时，打入电话或者其他程序提示等会不会对APP运行是否造成影响。

7.安装和卸载

### 4.3接口测试：

4.3.1接口测试流程：

在获取接口文档后，根据接口文档进行分析，然后开始编写测试用例，使用postman进行测试。在配置请求时，主要关注接口的协议，ip，路径，端口，请求的方式，以及请求正文的配置等等，根据接口文档中请求示范添加对应的参数，执行后关注接口的返回字段名，字段值，code码，格式以及返回的内容是否与预期相符等等

#### 4.3.2接口测试的意义：

--接口测试实施在多系统的平台架构下，有着极为高效的成本收益比

--接口测试天生为高复杂性的平台带来高效的缺陷检测和质量监督能力，平台复杂，系统越庞大，接口测试的效果越明显。

#### 4.3.3在做接口测试的时候主要关注：

-- 接口的联通性 接口的异常性 接口的组合性 接口的安全性

### 4.4接口自动化的持续集成

#### 4.4.1接口自动化测试的必要性

1--随着公司业务的增长，业务规模越来越大，各个系统以及各个子系统间的交互越来愈多，业务耦合越来越复杂。怎样保证大量业务系统之间的正常交互，以及最大限度的保障系统在不断的开发优化过程中已有的功能不受到影响，不单单是对测试人员的挑战，更是对我们团队的挑战。

2--如果按照价值排序，那么接口自动化能发挥的价值是最高的，单元次之，UI最低。所以综合下来我优先选择了接口自动化。

3--随着系统的日渐庞大，在做回归测试时，需要投入较大的人力。

#### 4.4.2做接口自动化的持续集成前提条件：

1--项目时间长，需求变动不频繁，测试用例的稳定性决定了自动化测试的维护成本。如果软件需求变动过于频繁，那么我就要根据需求来变更测试用例以及自动化脚本，这也是我选择接口自动化的一个原因，接口要比UI要稳定，

2--对项目中稳定的模块相对稳定的可以做自动化，而某些需求变动比较大的还是优先使用手动测试

#### 4.4.3接口自动化测试的可选方案

A- Postman+node.js+newman+Jenkins

B- Robot Framework

C-python+requests

选择Postman+node.js+newman+Jenkins

原因 公司使用postman做接口测试，对postman的熟悉程度较高，且postman+node.js+newman+Jenkins 容易搭建，操作方便。

#### 4.4.4框架介绍

Postman+Node.js+Newman+Git+Jenkins

搭建环境之前需要先理清楚各个环境的依赖关系，jenkins只支持windows命令行跟linux shell环境执行构建命令，而postman的测试脚本不能直接在命令行或shell环境执行，postman脚本需要有自己的执行环境，叫newman；而postman是基于javascript语言，所以newman环境还依赖于javascript的环境node.js。所以整个过程是先搭建nodejs构造javascript环境，在搭建newman构造postman脚本运行环境，再集成到jenkins上。

|  |  |
| --- | --- |
| 软件名称 | version |
| JDK | 1.8.0\_131 |
| postman | 7.0.6 |
| Newman | 4.4.0 |
| Node | 10.15.3 |
| Tomcat | 9.0 |
| Git |  |
| Jenkins | 2.168 |

#### 4.4.5接口自动化测试流程

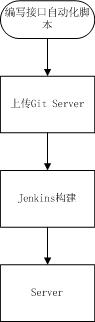
1.使用postman编写自动化脚本（包含接口测试用例），并且导出脚本，必须在本地node.js环境下执行命令行成功。

2.使用git推送至远程仓库相对应的分支

3.jenkins拉取对应git分支，开始构建，即对接口所在服务器发送请求

4.构建后，如果失败jenkins会已邮件的方式发送测试报告到测试人员邮箱

备注：已在jenkins进行相关配置主要包括安装git插件 ，email插件等以及各个插件的安装和配置



#### 4.4.6资源需求

系统ubuntu

硬件

|  |  |
| --- | --- |
| ***硬件名称*** | ***型号*** |
| CPU | I5-6400 |
| 内存 | 2G |
| 硬盘 | 100G |

#### 4.4.7接口自动化覆盖的计划：

1 首先对单个接口进行调试，保证单个接口的正确和通畅，本次检测主要是对参数进行测试。并且集成到jenkins上实现持续集成。在集成到jenkins上时优先覆盖1级用例，为每个版本后期回归测试节约时间，计划在5月31日完成。

2明确数据流和业务流，将相关业务关联的接口组合起来，本次检测主要是对业务进行检测，计划在7月31日完成。

### 4.5 WEB端的UI自动化：

UI自动化测试框架：

Python3+Selenium

UI自动化的测试方法：

主要利用页面标签的元素定位。用代码模拟用户的操作。

总共有八种方法：

Id,name,class,xpath,tag,link,css,jquery

UI自动化的前提条件：

1--项目时间长，需求变动不频繁，测试用例的稳定性决定了自动化测试的维护成本。如果软件需求变动过于频繁，那么我就要根据需求来变更测试用例以及自动化脚本，

2--对项目中稳定的模块相对稳定的可以做自动化，而某些需求变动比较大的还是优先使用手动测试。

UI自动化的优缺点：

优点：

1.提高测试执行效率，节约时间成本；

2.解放人力去做更重要的工作；

3.可重复利用，减少对人的依赖；

4.可大幅度减少兼容性测试的工作量；

缺点：

1.开发测试脚本需要花费较大的时间成本，拉长周期;

2.产品的快速迭代，自动化脚本也将不断迭代，时间成本很高

3.不同的项目之间自动化脚本的复用度很低;

4.对短期型项目产品实现自动化价值不高

5.自动化无法完全代替手工测试找到bug，实现100%覆盖

6.自动化更多的适用于回归测试;

总结：

UI自动化无论是编写自动化脚本，还是维护自动化脚本都会花费非常大的时间成本，并且UI自动化只能在版本的相对后期才能做，不能像接口自动化那样提前发现BUG，并且没有接口那样稳定。

建议：

在后期测试资源比较宽松的时候在做UI自动化。

### 4.6 APP端的UI自动化：

APP自动化测试框架：

Python+Airtest

APP自动化测试的方法：

Airtest既可以使用元素属性定位，也可以通过截图的方式来定位。

APP自动化的优缺点：

优点：

1.重复性的功能测试及验证

2.避免疲惫操作时的人为测试遗漏

3.快速检测软件的兼容性

缺点：

缺点也是比较明显，随着敏捷迭代的速度越来越快，UI控件的频繁变更导致控件定位不稳定，提高了用例脚本的维护成本。

总结：

App自动化和WEB端的UI自动化一样，无论编写自动化脚本还是维护脚本，都要随着页面的变化而改动，因此维护的成本比较高。但是app自动化可以在测试机较多的情况下，很好的检测软件的兼容性，测试机越多，被测软件越复杂，自动化的产出就越多，当然需要的时间和人力成本越大。

建议:

在后期测试资源比较宽松的时候在做UI自动化。

## 五 测试文档的管理：

1功能测试用例以增量的方式上传至文档管理工具。

2接口的自动化脚本（包含接口测试用例），需要使用git上传至远程仓库。

3对未能解决的BUG或则未能执行的测试用例。

以及对app测试未能覆盖的机型等风险要在测试报告中体现出来。