目录

[什么是软件测试？ 1](#_Toc468529806)

[软件测试部门如何汇报工作量，如何量化工作？ 1](#_Toc468529807)

[web测试用例如何写？ 2](#_Toc468529808)

[如何写出高效的软件测试用例？ 8](#_Toc468529809)

[网站的易用性测试如何进行？都包括哪些内容？ 9](#_Toc468529810)

[怎样正确做 Web 应用的压力测试？ 10](#_Toc468529811)

## 什么是软件测试？

A 发现软件问题

B 确认软件质量（没有发现bug也是有意义的）

C 提高软件体验（不是bug而是产生improvement甚至new feature）

## 软件测试部门如何汇报工作量，如何量化工作？

这个问题很经典，我在一些书里也没看到让人信服的回答，也许并没有标准答案。

当初拿到微软的软件测试KPI，大概也就统计的是用例、缺陷、自动化率和会议时间等等。

用例里一般是执行情况，缺陷会有缺陷发现率DDP，并且会区分用例缺陷率和随机测试缺陷率。其他两个不言自明。

从我个人来说，我还会看中用例执行效率，例如单位复杂度的用例设计时间、执行时间、缺陷加权和回归测试时间等等。

这些数据并不能简单的作为测试团队考核的标准，但是可以通过一些目标的达成间接的评估，例如自动化率达到多少，用例评审通过率提高多少，DDP提高多少等等。

最常见的衡量指标如下：

Bug数，外网事故数，用例数，完成的测试任务数。

## web测试用例如何写？

任何测试包括Web测试写测试用例都可以先把框架搭好，个人比较喜欢测试大纲法，比较全面。就类似一棵树，先把树干找好，再就每一个树干去发散找树叶（测试用例）。Web测试的树干大致如下：

1. 浏览器兼容性测试
2. 界面测试，UI修正，对照开发出来的网页与设计稿，提出改进方案；

（下面这几个是功能测试大类里面的细分类）

2. 输入框测试（文本、数字、日期、特殊字符等）

3. 搜索测试

4. 注册登陆模块测试

5. 图片上传下载测试

6. 数据增删查改测试

7. 链接测试

8. 特殊键如Back、Tab、Enter、Refresh等测试

9. 易用性测试

10. 兼容性测试（系统、浏览器）

11. 安全测试 （SQL注入、XSS等）

12. 性能测试. （响应时间、吞吐量、资源占用）

除了通用型的，有时候还会根据功能的不同，一些数据一致性测试、cgi测试、通过改包的破坏性测试等

真实用户来测，来体验，用户测试访谈，模拟真实情景测试。

１。检查每个连接是否都有对应的页面；检查页面之间可以正确切换。

　　２。检查按钮的功能，如Add， update， delete， save， cancel.

　　３。检查输入框：

　　１）字符串长度，检查输入超出需求所说明的长度的内容，会不会出错。

　　２）字符类型检查：输入指定类型的地方输入其他类型的内容，检查是否报错。

　　３）回车键检查：在输入结束后直接按回车键，看系统处理如何，会否报错。

　　４。上传下载文件检查上传下载文件的功能是否实现，上传文件是否能打开。

　　５。必填项：检查应该填写的项没有填写时系统是否都做了处理，对必填项是否有提示信息，如在必填项前加\*.

　　６。检查多次使用back键的情况：在有back的地方，back，回到原来页面，再back，重复多次，看会否出错。

　　７。Search功能：在有search功能的地方输入系统存在和不存在的内容，看search结果是否正确。如果可以输入多个search条件，可以同时添加合理和不合理的条件，看系统处理是否正确。

　　８。检查删除功能：在一些可以一次删除多个信息的地方，不选择任何信息，按“delete”，看系统如何处理，会否出错；　然后选择一个和多个信息，进行删除，看是否正确处理。

　　９。相关性检查：删除/增加一项会不会对其他项产生影响，如果产生影响，这些影响是否都正确。

　　１０。 检查带出信息的完整性：查看信息和update信息时，查看所填写的信息是不是全部带出。，带出信息和添加的是否一致。

　　１１。 快捷键检查：是否支持常用快捷键，如Ctrl+C Ctrl+V Backspace等，。

功能测试：1.链接测试。2.表单测试。3.cooking测试。4.数据库方面的功能测试。

性能测试：1.连接速度测试（响应时间）。2.负载测试（模拟大量用户同时操作）。3.压力测试（模拟一定数量的用户访问）。4.数据库的性能测试。

WEB测试的常用的检查点

1：页面连接检查每一个连接是否都有对应的页面并且页面之间切换正确。

2：相关性检查删除/增加一项是否会对其他项产生影响如果产生影响这些影响是否都正确。 3：检查按扭的功能是否正确如updatecanceldeletesave等功能是否正确。

4：字符串长度检查输入超过需求说明的字符串长度的内容看系统是否检查字符串长度会不会出错。

5：字符类型检查在应该输入指定类型的内容的地方输入其他类型的内容如在应该输入整形的地方输入其他字符类型看系统是否检查字符类型是否报错。

6：标点符号检查输入内容包括各种标点符号特别是空格各种引号回车健看系统是否处理正确。

7：中文字符处理在可以输入中文的系统输入中文简体或繁体看是否会出现乱码或出错。

8：检查带出信息的完整性在查看信息和update信息时查看所填写的信息是否全部带出带出信息和添加的是否一致。

9：信息重复检查在一些需要命名且名字应该唯一的信息输入重复的名字或id看系统有没有处理是否报错重名包括是否区分大小写以及在输入内容的前后输入空格系统是否作出正确处理

10：检查删除功能在一些可以一次删除多个信息的地方

经验：先满足于开发环境的用户体验，再来做其他浏览器的兼容性。

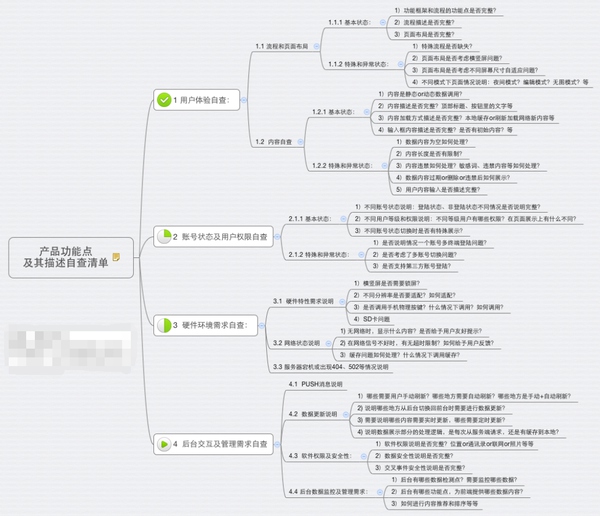
功能性测试就要有逆反心理，反着来。比如表单如果使用了默认值，则需要检验默认值的正确性。若只能接受指定的某些值，则测试时可以跳过这些字符，看系统是否会报错。

后来找了点资料，主要通过这篇文章：

http://developer.51cto.com/art/200807/83518.htm

整理了如下的Web测试思维导图，欢迎拍砖。





## 如何写出高效的软件测试用例？

信的意思就是，用例要忠实的反应产品的功能，包括产品说明书、包括常识、包括你的发散性思维探索性的可能、系统性的思考。

要做到这个可不简单。

1、首先必须了解需求，吃透产品/需求说明书

2、你得有常识。什么叫有常识，举个例子：百度输入框，输入之后，可以鼠标点搜索，也可以直接按enter键搜索。这就是常识，通常产品说明书可不会告诉你，这个按钮要对enter做响应

3、发散性是什么呢？就是你能从产品说明中挖掘隐性的需求。比如说上面的这个框。产品可能只是说，输入什么，就在引擎查找并返回什么。发散一下就是，我输入特殊符号呢？中文呢？英文呢？西藏文？还能不能正常工作？

4、系统性的思考。何为系统性的思考，通常不会都测试都是新的东西，有可能是运营的小需求。流程加了状态，加了按钮，要从整个系统层面考虑这样的改变，会带来什么影响，从而得到新的测试用例。这个思考的方式在另外的领域叫沙盘推演

达这个的意思就是你用例的粒度，不能太粗也不能太细。要自然。就是同级别或者低一级别的QA看你的用例，也知道这个用例要做什么，达到什么效果，写的让别人还要再挖掘一轮的，都不是好用例

雅 这个就是纯粹的用例排版或者一些基本套路的表达。包括用例集的管理，比如按照功能划分用例集、按照测试类型划分用例集等等

## 网站的易用性测试如何进行？都包括哪些内容？

通常对“网站易用性”有如下定义：

易理解Easy to discover：单单凭观察，用户能否在几秒钟知道网站是做什么的。以及知道可以采取的行动。

易操作Easy to use：初次接触网站功能或界面时，用户完成的难易程度，是否可不用帮助文档就能使用网站。

易学习Easy to learn：通过简单的帮助文件，用户能否能解决遇到的问题

效率性（efficiency）：用户熟悉网站功能或界面后，完成任务的速度

出错率（mistake）：在使用网站过程中，用户操作出现了多少错误，这些错误有多严重，从错误中恢复容易吗？

满意度（satisfaction）使用这个设计的用户满意度。

“网站易用性”一般的执行步骤如下：

第一步：前期准备，搭建测试环境

第二步：招募测试对象，找到典型用户

第三步：设定任务（目标），观察（录制）其操作轨迹

第四步：采访测试对象使用感受

第五步：对观察结果和采访结果进行综合分析

同时也可以结结合GA或者其他软件进行网站分析，通过热力图，页面停留时间，某个按钮停留时间，用户的行为轨迹，来获得一定的信息，从而进行网站的改善。

## 怎样正确做 Web 应用的压力测试？

一个完整的压力测试需要关注三个方面：如何正确产生压力、如何定位瓶颈、如何预估系统的承载能力

(1) 首先说一下如何产生压力，产生压力的方法有很多，通常可以写脚本产生压力机器人对服务器进行发包和收包操作，也可以使用现有的工具(像jmeter、LoadRunner这些），所以说产生压力其实并不难，难点在于产生的压力是不是真实地反映了实际用户的操作场景。举个例子来说，对游戏来说单纯的并发登陆场景在整个线上环境中的占比可能并不大(新开服等特殊情况除外)，相反“登陆-开始战斗-结束战斗”、不同用户执行不同动作这种“混合模式”占了更大的比重。所以如何从实际环境中提炼出具体的场景比重，并且把这种比重转化成实际压力是一个重要的关注点。

(2) 产生压力之后，通常我们可以拿到TPS、响应时延等性能数据，那么如何定位性能问题呢？TPS、响应时延只能告诉你服务器是否存在问题，但不能帮助你定位问题。这些表面背后是整个后台处理逻辑综合作用的结果，这时候可以先关注系统的CPU、内存、IO、网络，对比在tps、时延达到瓶颈时这些系统数据的情况，确定性能问题是系统哪一部分造成的，然后再回到代码的逻辑中逐个优化这些点。

(3) 当服务器的整体性能就可以相对稳定下来，这时候就需要对自己服务器的承载能力有一个预估，通过产生真实压力、对比系统数据，大致可以对单套系统的处理能力有个真实的评价，然后结合业务规模配置服务器数量。可以看下腾讯wetest的这个压测工具http://wetest.qq.com/gaps/

总结一下：通过产生真实压力来发现问题、结合系统性能来解决问题

应该了解产品

a 最频繁使用功能

b 最耗资源功能

c 设计峰值并发用户数

等等与客户行为真正相关的东西