# 需求管理及非功能测试的相关知识

## 第一部分 测试的需求管理

## 相关概念

**（一）、什么是软件需求？**

（1）**用户**解决某一问题或达到某一目标**所需**的软件功能。

（2）系统或系统构件为了满足合同、规约、标准或其他正式实行的文档而必须满足或具备的**软件功能**。

**（二）、什么是软件测试需求？**

（1）测试需求是根据程序文件和质量目标对软件测试活动所提的要求。

（2）软件测试需求是**开发测试用例的依据**。

（3）详细的测试需求还是衡量测试覆盖率的重要指标。

**（三）、什么是软件测试需求管理？**

（1）通过人为的和技术的手段、方法和流程，以**保证和监督**测试团队达到测试软件产品的**目标**。

（2）应对软件需求、软件测试需求及相关需求的问题，并有效地分析出测试的具体需求，并以此为软件测试设计提供尽可能准确的信息作为参考。

**总的来说:**

根据用户的需要，编写开发相关软件；又根据用户的需求，采取人为或技术的手段或方法对软件进行不断地测试，以保证和监督测试团队达到测试软件产品的目标。

## 二、测试需求分析

|  |  |
| --- | --- |
| **项目** | **过程** |
| 目标 | 对要解决的问题进行详细的分析，弄清楚参与软件测试活动的干系人对软件测试活动和交付物的**要求**，包括需要输入什么数据，要得到什么结果，最后应输出什么。 |
| 方法 | 从软件需求推导软件测试需求的方法  （1）根据软件开发需求说明书逐条列出软件**开发需求**，并判断其可测试性。  （2）形成可测试的描述并界定出**测试范围**。  （3）根据质量标准，逐条制定**质量需求**，即测试通过标准。  （4）分析测试执行时需要实施的**测试类型**。  （5） 建立测试需求**跟踪**矩阵，并输入测试需求管理系统，对测试需求实施严格有效的管理。 |
| 结果及评审 | （1）评审内容  包括完整性检查和准确性检查。  （2）评审的方法  相互评审、交叉评审  轮查（分配审查）  走查  小组评审 |

## 测试需求跟踪

|  |  |
| --- | --- |
| 需求跟踪的目的 | 保证需求跟实际开发任务的关联性，这会提高开发费用，降低运维费用。 |
| 需求跟踪的内容 | 包括编制每个需求同系统元素间的联系文档。 |
| 变更的代价 | 需要作出影响分析，例如成本、进度、收益、风险等 |
| 和其它系统的联系 | 最普遍方式是使用需求跟踪矩阵 |

**什么是需求跟踪矩阵？**

需求跟踪矩阵是一种主要管理需求变更和验证需求是否得到了实现的有效工具，借助RTM，可以跟踪每个需求的状态。

**作用：**

（1）检查需求是否被实现了，是否被测试了，执行需求的验证，进行功能审计；

（2）在发生需求变更时，通过检索需求跟踪矩阵发现需要修改的需求、设计及测试用例等。本例即证明了需求跟踪矩阵的第一个作用。

## 第二部分 非功能测试

## 一、相关概念

非功能测试包括性能、负载、安全、可靠性和其他很多方面，有时也被称作行为测试或质量测试。非功能测试的众多属性的一个普遍特征是**一般不能直接测量**。这些属性是被间接地测量，例如用失败率来衡量可靠性或圈复杂度，用设计审议指标来评估可测性等。

## 二、相关属性

|  |  |
| --- | --- |
| **属性名称** | **相关定义** |
| 可靠性 | 软件使用者期望软件能够**无误运行**。可靠性是度量软件如何在主流情形和非预期情形下维持它的功能，有时也包括软件出错时的自恢复能力。例如，自动定时保存现行文件。 |
| 可用性 | 如果用户不明白应该**如何使用**，那么，即使是零差错的软件也会变得毫无用处。可用性测量的是用户学习和控制软件以达到用户需求的容易程度。进行可用性研究、重视顾客反馈意见和对错误信息和交互内容的检查都能提高可用性。 |
| 可维护性 | 可维护性描述了**修改软件而不引入新错误所需的工作量**。产品代码和测试代码都必须具备高度的可维护性。团队成员对代码的熟悉程度、产品的可测性和复杂度都对可维护性有影响。 |
| 可移植性 | 可移植性指一种计算机上的软件**转置到其它计算机上的能力**。从狭义上讲，是指可移植软件应独立于计算机的硬件环境；从广义上讲，可移植软件还应独立于计算机的软件，即高级的标准化的软件。 |

## 三、测试的类型

|  |  |
| --- | --- |
| **测试类型** | **相关定义** |
| 性能测试 | 目的是验证软件系统**是否能够达到用户提出的性能指标**，同时发现软件系统中存在的性能瓶颈，优化软件，最后起到优化系统的目的。性能测试类型包括压力测试、负载测试，强度测试，容量测试等。 |
| 压力测试 | 一般来说，压力测试的目的是要通过模拟**比预期要大的工作负载**来让只在峰值条件下才出现的缺陷曝光。内存泄漏、竞态条件、数据库中的线程或数据行之间的死锁条件、和其他同步问题等等，都是压力测试能发掘出来的常见缺陷。 压力测试**主要是为了测试硬件系统是否达到需求文档设计的性能目标，**例如，在一定时期内，系统的cpu利用率，内存使用率，网络吞吐量等。 |
| 负载测试 | 负载测试是要探讨在高峰或高于正常水平的负载下，**系统或应用软件会发生什么情况**。例如，一个网络服务的负载测试会试图模拟几千名用户同时连线使用该服务。**测试的主要是软件系统的性能**，例如，软件在一定时期内，最大支持多少并发用户数，软件请求出错率等。 |
| 平均无故障时间测试 | 平均无故障时间测试是测量系统或应用软件**在出错或当机前的平均运行时间**，包括平均无错时间或平均无当机时间。 |
| 低资源测试 | 低资源测试是要**确定当系统在重要资源（内存、硬盘空间等）降低或完全没有的情况下会出现的状况**。重要的是要**预估**将会发生什么，例如，为文件存盘而无足够空间、或一个应用程序的内存分配失败时将会发生什么。 |
| 容量测试 | 容量测试一般是用来执行服务器或服务测试。目的是要**确定系统最大承受量**，例如，系统最大用户数，最大存储量，最多处理的数据流量等。容量模型通常建立在容量测试数据基础上。有了这些数据，营运团队就能定计划什么时候增加系统容量。 |
| 重复性测试 | 重复性测试是**为了确定重复某一程序或场景的效果**而采取的一项技术。这个技术的精髓是循环运行测试直到达到一个具体界限或**临界值**。例如，一个操作也许会泄漏20字节的内存。这并不足以在软件的其他地方产生任何问题，但如果测试连续运行2000次，泄漏就可以增长到4万字节。如果是提供核心功能的程序有泄漏，那么这个重复性测试就抓到了只有长时间连续运行该软件才能发现的内存泄漏。 |
| 兼容性测试 | 兼容性测试是指测试软件**在特定的硬件平台上、不同的应用软件之间、不同的操纵系统平台上、不同的网络等环境中是否能够很友好的运行**的测试。主要测试软件是否能在不同的操作系统平台上兼容，或测试软件是否能在同一操作平台的不同版本上兼容；软件本身能否向前或向后兼容；测试软件能否与其他相关的软件兼容；测试数据能否共享等。 |
| 安全性测试 | 安全性测试是**检查系统对非法侵入的防范能力**。主要包括用户认证、系统网络安全和数据库安全方面的测试。安全测试期间，测试人员假扮非法入侵者，采用各种办法试图突破防线。例如，想方设法截取或破译口令；专门开发软件来破坏系统的保护机制；试图通过浏览非保密数据，推导所需信息等。理论上，只要有足够的时间和资源，没有无法进入的系统。因此系统安全设计的准则是**使非法侵入的代价超过被保护信息的价值**，此时非法侵入者已无利图。 |
| 辅助功能测试 | 辅助功能测试**保证软件公司开发的软件能被伤残人使用**。其中任何应用程序都必须测试的特性包括：操作系统的设置测试、辅助特性的测试（包括Tab 键顺序、热键和快捷键）、编程访问的测试、以及辅助的技术工具的测试。辅助功能测试的一个重要方面就是使用辅助功能工具去测试应用程序， 这些工具包括：屏幕阅读器、放大镜、语音识别或者其他输入程序。 |
| 本地化测试 | 本地化就是将软件版本语言进行更改，测试的对象是软件的本地化版本。测试的目的是**测试特定目标区域设置的软件本地化质量**。测试的环境是在本地化的操作系统上安装本地化的软件。测试方法上可以分为基本功能测试，安装/卸载测试，当地区域的软硬件兼容性测试。测试的内容主要包括软件本地化后的界面布局和软件翻译的语言质量，包含软件、文档和联机帮助等部分。 |
| 配置测试 | 配置测试就是测试软件是否和系统的其他与之交互的元素之间兼容，如浏览器、操作系统、硬件等，**验证被测软件在不同的软件和硬件配置中的运行情况**。配置测试执行的环境是所支持软件运行的环境。测试环境适合与否严重影响测试结果的真实性和正确性。硬件环境指测试必须的服务器、客户端、网络连接设备、打印机等，软件环境指被测试软件运行时的操作系统、软件平台、数据库其他应用软件构成的环境。 |
| 可用性测试 | 可用性测试是在产品或产品原型阶段实施的通过观察或访谈或二者相结合的方法，**发现产品或产品原型存在的可用性问题**，为设计改进提供依据。可用性测试不是用来评估产品整体的用户体验，主要是**发现潜在的误解或功能在使用时存在的错误**。可用性测试适于解决的问题：确定测试产品的可用性水平；与预期目标、与竞争对手、与老版设计相比的可用性水平；比较不同方案，确定哪个方案更加可行。 |

**参考资料：**

<http://blog.csdn.net/seacean2000/article/details/54086954>

<http://blog.csdn.net/dancing_youh/article/details/49184893>

<http://blog.csdn.net/u012938881/article/details/44776093>

<https://www.cnblogs.com/gisen_6/p/3734188.html>