Day14\_软件测试基础

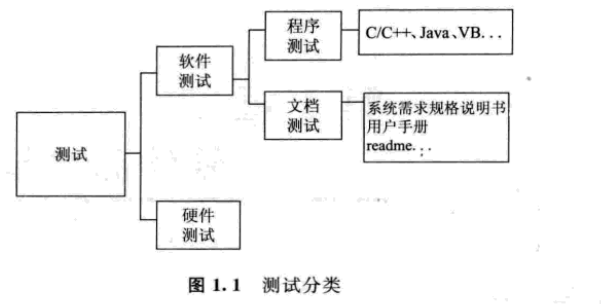
# 软件测试基础

## 什么是软件测试

使用人工或者自动手段，来运行或者测试某个系统的过程。其目的在于检测它是否满足规定的需求或者弄清预期结果与实际结果之间的差别

## 测试包含哪些内容

程序测试包括程序逻辑功能，界面，性能，易用性，兼容性，安装等测试，当然文档测试也算，排版，字体大小，也算软件测试的内容



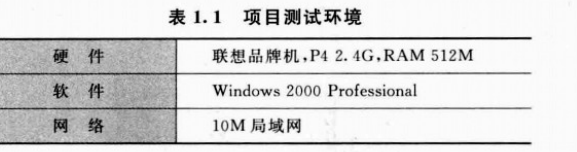
## 测试环境

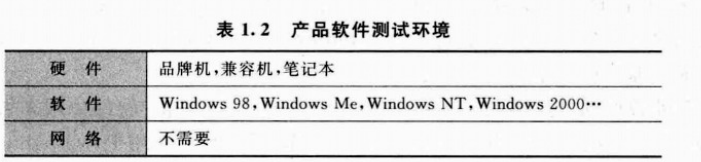
测试环境=硬件+软件+网络

硬件环境：pc机还是笔记本

软件环境：windows10 windows8 windows7 Linux Mac firefox chrom IEtester工具的使用(如何使用:https://jingyan.baidu.com/article/bad08e1e9973ec09c85121f8.html)

网络:局域网还是互联网





S

## 测试流程

产品经理:

提需求

产品原型图

UI:产品效果图

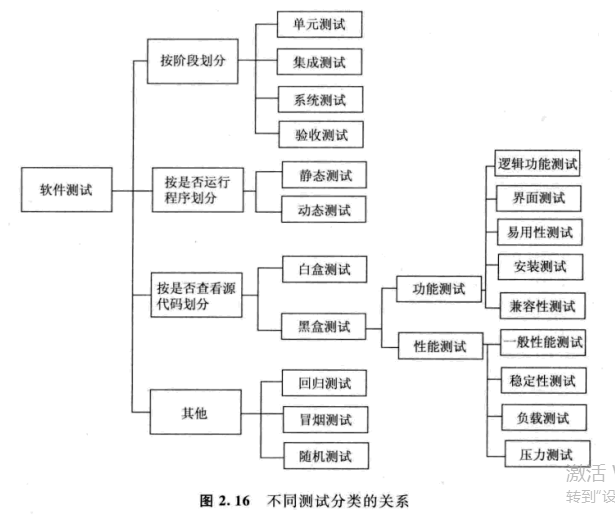
|  |  |  |  |
| --- | --- | --- | --- |
| 需求评审 | 测试计划制定 | 测试计划执行 | 发布与测试报告总结 |
| 1从用户体验角度提供设计建议  2从经验角度，分析设计是否存在风险  3 联合其他模块分析，设计是否存在漏洞 | 1测试用例设计  2测试用例评审，和测试时间估计  3测试资源申请 | 1用例执行  2 Bug修复验证和推动版本进度  3性能监控，压力测试，兼容测试 | 1版本发布和线上质量监控，用户反馈实时响应  2提供版本最终测试报告，包括用例覆盖率，bug数据分析等 |
| 全程跟进需求变更，与产品无缝沟通，在测试阶段有需求变更要第一时间了解改动范围，如果影响版本的质量要说明风险，评估需求是否必须更改以及是否影响版本发布上线的时间线 | 规划测试项目需要的功能开发和自动化开发人员比例，规划整个测试流程需要的时间，要预留处理紧急事件的缓冲 | 执行  协调测试资源，部署测试环境，督促开发和产品提供一切需要的测试工具，测试数据等，推动版本进度，每日进行bug review(复盘)，标识出bug解决的优先级和提测的时间点，每日提供当日产品质量报告 | 报告  项目发布上线后，对整个版本的bug进行数据分析，总结出用例的覆盖率，对于没有覆盖到用例的bug，转化成用例，同时测试人员之间进行分享，针对新接触的测试方法测试工具和有价值的bug进行经验总结 |

产品原型图--产品经理

产品效果图--UI

产品需求-产品经理

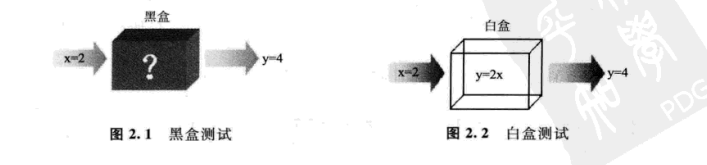
# 软件测试分类—记住



## 黑盒测试和白盒测试

黑盒测试(Black Box -Test)指的是把被测试的软件看做一个黑盒子，我们不去关心盒子里边的结构是什么样子，只关心软件的输入数据和输出结果

白盒测试(White Box Testing),指的是把盒子盖打开，去研究里边源代码和程序结构。



## 静态测试和动态测试🡪白盒测试

静态测试，是指不实际运行被测试软件，而只是静态的检查程序代码、界面或者文档中可能存在的错误的过程。

动态测试:是指实际运行被测程序，输入相应的测试数据，检查实际输出结果和预期结果是否相符的过程。(单元测试:断言)

## 功能测试和性能测试

### 功能测试

是黑盒测试的一部分，它检查实际软件的功能是否符合用户的需求。

功能测试可以细分逻辑功能测试，界面测试，易用性测试，安装测试和兼容性测试。

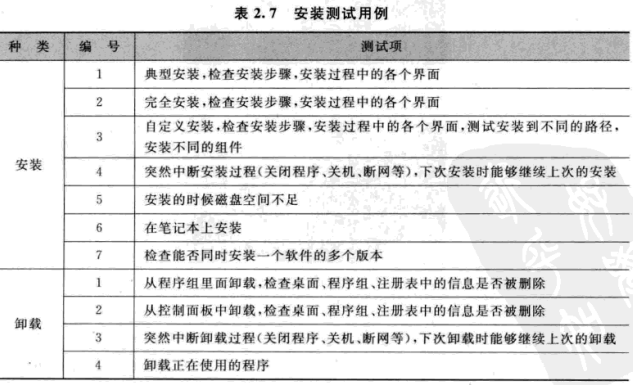
还有一些其他名词:恢复测试，确认测试，接口测试，数据库测试，安全测试，配置测试....

逻辑功能测试:测试应用是否符合逻辑，比如应该先注册账号之后，才能进行登录,登录之后才能看我的购物车

界面测试:窗口大小，按钮大小，点击按钮弹出什么样的提示框，是否有滚动条，下拉菜单是否有展示内容...

易用性测试:从软件使用的合理性和方便性等角度对软件系统进行检查,比如，软件窗口长宽比例是否合适，颜色色彩是否赏心悦目，字体大小是否合适

安装测试:



兼容性测试:硬件兼容性测试和软件兼容性测试

硬件兼容性:比如一款软件在pc机，笔记本上是否兼容

软件兼容性测试:比如一款软件在windows8和windows10上是否兼容

### 性能测试

--转账- --张三给李四转账1000---

--张三转出1000---

--李四收到1000---

--事务是一个完整的单元，要么全部成功，要么全部失败--

时间性能:软件的一个具体事务的响应时间。比如点击一个登陆按钮，到登录成功(失败)的反应时间，浏览器非常常见，ANR（Application not responding 应用程序无响应）

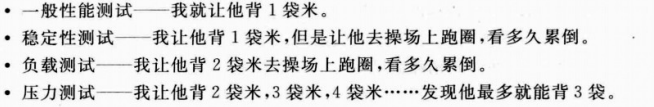
空间性能:软件运行时所消耗的系统资源，比如对内存和cpu的消耗

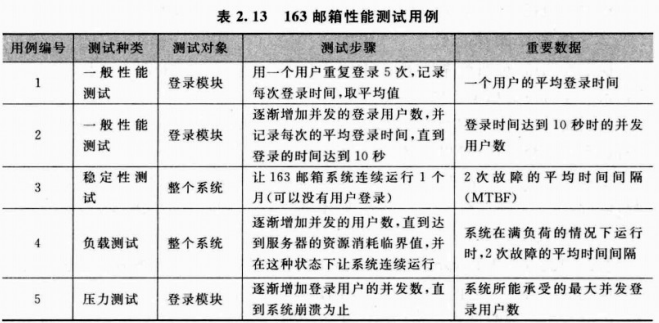
一般性能测试:软件正常运行，不向其施加任何压力的测试

稳定性测试:也叫可靠性测试，是指连续运行被测系统，检查系统运行时的稳定程度。

负载测试:让被测系统在其能够忍受的压力范围之内连续运行，来测试系统的稳定性。

压力测试:持续不断的给被测试的系统增加压力，直到被测试的系统压垮为止，用来测试系统所承受的最大压力。





## 回归测试、冒烟测试、随机测试

### 回归测试

是指对软件的新版本进行测试时，重复执行上一个版本测试时的用例，比如在1.0版本中，有一个bug，到了2.0版本中,再重新测试1.0中这个bug

### 冒烟测试

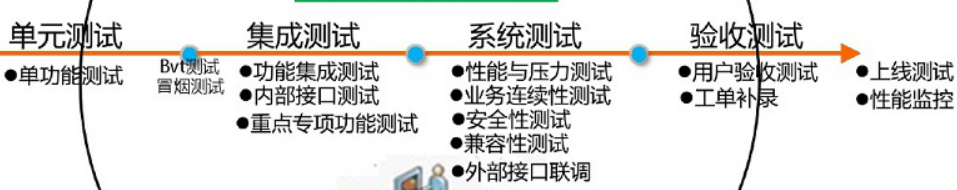
指对一个软件进行系统大规模的测试之前，先验证一下软件的基本功能是否实现，是否具备可测性。

测试小组在正式测试一个新版本之前，先指派一两个测试人员测试一下软件的主要功能，如果没有实现，则打回开发组重新开发，这样做可以节省大量的时间成本和人力成本。

### 随机测试

是指测试中所有的输入数据都是随机生成的，其目的是模拟用户的真实操作，并发现一些边缘性的错误。

## 单元测试、集成测试、系统测试和验收测试



### 单元测试

单元测试，又可以理解为单功能测试

是指对软件中最小可测试单元进行检查和验证

单元测试当一段代码完成之后，是由白盒测试工程师或者开发人员自行测试，比如java中执行单元测试叫做junit测试。

目前大部分公司单元测试由开发人员简单编译和调试一下自己的程序，没有相应的单元测试计划。

单元测试方式:先静态地观察代码是否符合规范，然后动态地运行一下代码,检查运行的结果。

### 集成测试

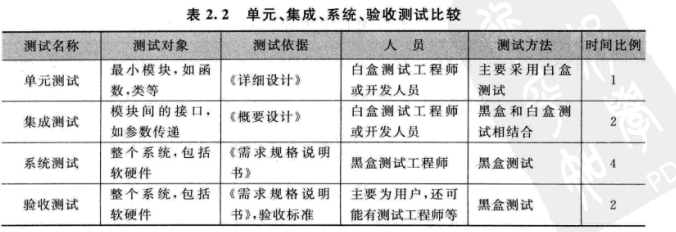
集成测试是单元测试的下一个阶段，是指将通过测试单元模块组装成系统或者子系统，再进行测试，重点测试不同模块的接口部分。

### 系统测试和验收测试

集成测试完成之后，就是系统测试和验收测试。

系统测试:指的是将整个软件系统看做一个1个整体进行测试，包括对功能、性能，以及软件所运行的软硬件环境进行测试。

系统测试由黑盒测试人员在整个系统集成完毕后进行测试，前期主要测试系统的功能是否满足需求，后期主要测试系统运行的性能是否满足需求，以及系统在不同的软硬件环境的兼容性等。



## 测试案例



### **需求：**

测试一个带广告图案的花纸杯

### 功能测试

能否装水，

除了装水， 能否装其他液体。比如可乐，酒精

能装多少ML的水

杯子是否有刻度表

杯子能否泡茶，跑咖啡

杯子是否能放冰箱，做冰块

杯子的材质是什么（玻璃，塑料,黄金做的）

### 界面测试

外观好不好看。

什么颜色

杯子的形状是怎么样的。

杯子的重量是多少

杯子的图案是否合理

### 性能测试

能否装100度的开水 (泡茶)

能否装0度冰水

装满水，放几天后，是否会漏水

杯子内壁上的涂料是否容易脱落。

杯子上的颜色是否容易褪色或者脱落

风吹是否会倒，摔一次是否会摔坏，摔多次是否会摔坏

### 安全性测试

制作杯子的材料，是否有毒

放微波炉里转的时候，是否会熔化。

从桌子上掉到水泥地上是否会摔碎。

杯子是否容易长细菌

杯子内壁上的材料，是否会溶解到水中

装进不同液体，是否会有化学反应。

### 易用性测试

杯子是否容易烫手

杯子是否好端，好拿

杯子的水是否容易喝到

杯子是否有防滑措施

是否能接受杯子的广告内容与图案

## 易用性测试 & 逻辑功能测试

页面,页面元素、功能部分、提示信息、容错性、权限部分、键盘操作

页面，页面元素部分

1. 页面清单是否显示，是否显示完整（3） 页面在窗口中的显示是否正确、美观（4） 页面特殊效果（如特殊字体效果、动画效果）是否显示(5)页面元素是否显示正确(6)页面元素的容错性(容忍错误的性能,可以容许错误,但是提示)是否存在

功能部分

1. 数据初始化是否正确(默认数据展示)
2. 数据操作(增删改查)是否正确

提示信息

（1） 操作页面成功、失败提示   
（2） 危险操作、重要操作提示(比如删除某些重要的信息)

容错性  
（1） 为空、非空,唯一性  
（3） 特殊字符 、双引号，符号

权限部分

功能权限： 指定用户可以使用那些功能，不能使用那些功能  
数据权限： 指定用户可以处理那些数据，不可以处理那些数据。  
操作权限： 在逻辑关系上，操作前后顺序、数据处理情况。

## 响应时间测试

看页面的 渲染时间

看数据的返回时间等等

# 测试分类占比🡪时间/过程占比

# 软件测试的原则

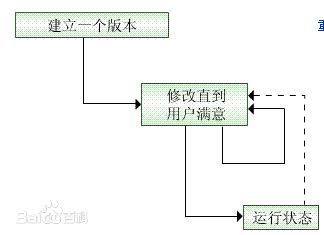
|  |
| --- |
| 所有的测试都应该追溯到用户的需求 |
| 应当尽早、不断的测试 |
| 测试发现的错误很可能起源于20%的模块中 |
| 设计测试用例时，应该考虑各种情况 |
| 对测试出的错误结果一定要有一个确认的过程(描述缺陷报告) |
| 制定严格的测试计划 |
| 完全测试是不可能的，测试需要终止 |
| 妥善保管一切测试资料 |

# 软件生命周期模型

软件生命周期 同任何事物一样，一个软件产品或软件系统也要经历孕育、诞生、成长、成熟、衰亡等阶段，一般称为软件生命周期（软件生存周期）[1]  。软件生命周期模型是指人们为开发更好的软件而归纳总结的软件生命周期的典型实践参考。

## 边做边改模型

许多产品都是使用“边做边改”模型来开发的。在这种模型中，既没有规格说明，也没有经过设计，软件随着客户的需要一次又一次地不断被修改。



在这个模型中，开发人员拿到项目立即根据需求编写程序，调试通过后生成软件的第一个版本。在提供给用户使用后，如果程序出现错误，或者用户提出新的要求，开发人员重新修改代码，直到用户满意为止。

这是一种类似作坊的开发方式，对编写几百行的小程序来说还不错，但这种方法对任何规模的开发来说都是不能令人满意的，其主要问题在于：

（1） 缺少规划和设计环节，软件的结构随着不断的修改越来越糟，导致无法继续修改；

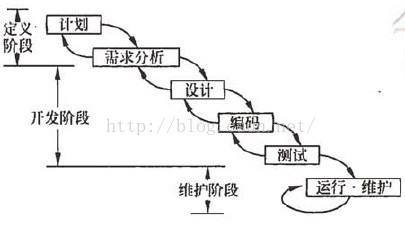
（2） 忽略需求环节，给软件开发带来很大的风险；

（3） 没有考虑测试和程序的可维护性，也没有任何文档，软件的维护十分困难。

## 瀑布模型

瀑布模型是最早出现的[软件开发模型](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91%E6%A8%A1%E5%9E%8B" \t "https://baike.baidu.com/item/%E7%80%91%E5%B8%83%E6%A8%A1%E5%9E%8B/_blank)，在[软件工程](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B" \t "https://baike.baidu.com/item/%E7%80%91%E5%B8%83%E6%A8%A1%E5%9E%8B/_blank)中占有重要的

地位，它提供了软件开发的基本框架。其过程是从上一项活动接收该项活动的工作对象作为输入，利用这一输入实施该项活动应完成的内容给出该项活动的工作成果，并作为输出传给下一项活动。同时评审该项活动的实施，若确认，则继续下一项活动；否则返回前面，甚至更前面的活动。对于经常变化的项目而言，瀑布模型毫无价值。



优点：

1. 为项目提供了按阶段划分的检查点
2. 当前一阶段完成后，只需要去关注后续阶段。

缺点：

1）各个阶段的划分完全固定，阶段之间产生大量的文档，极大地增加了工作量。

2）由于开发模型是线性的，用户只有等到整个过程的末期才能见到开发成果，从而增加了开发风险。

3）通过过多的强制完成日期和里程碑来跟踪各个项目阶段。

4）瀑布模型的突出缺点是不适应用户需求的变化。

## 螺旋模型

1.软件分多个版本开发，每个版本就是一次螺旋。

2.每个版本按照这样的顺序进行：

1）确定软件目标，选取定实施方案，弄清项目开发的限制条件；（图中左上象限）

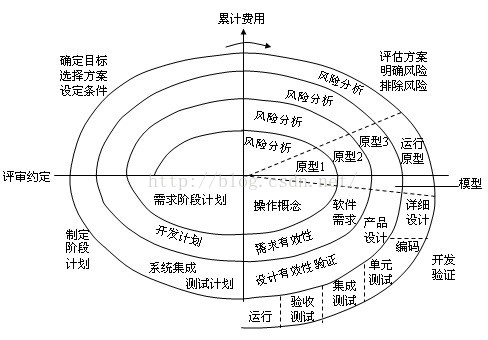
2）分析所选取方案，考虑如何识别和消除风险；（图中右上象限）

3）实施软件开发；（图中右下象限）

4）评价开发工作，提出修正建议，调整计划。（图中右下象限、左下象限）

3.需求不是一次获取和实现的，通过多个螺旋来完善。

4.计划也不是一次成型的，每次螺旋都需要调整。



图中的螺旋线代表随着时间推进的工作进展；开发过程具有周期性重复的螺旋线形状。4个象限分别标志每个周期所划分的4 个阶段：制定计划、风险分析、实施工程和客户评估。螺旋模型要点：统一了瀑布模型与原型模型，与增量模型相似，更强调风险分析。

优点：

1）设计上很灵活，可以在项目的各个阶段进行变更；

2）以小的分段来构建大型系统，使成本计算变得简单容易；(国企项目)

3）客户始终参与每个阶段的开发，保证了项目不偏离正确方向以及项目的可控性；

4）随着项目推进，客户始终掌握项目的最新信息 , 从而能够和管理层有效地交互；

5）客户认可这种公司内部的开发方式带来的良好的沟通和高质量的产品。

缺点：

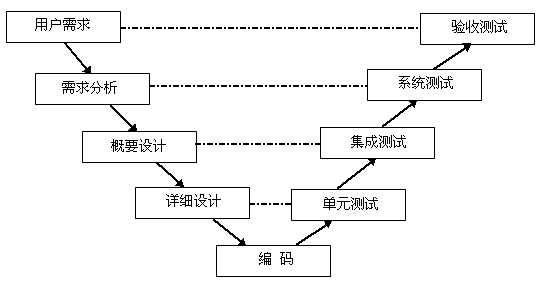
螺旋模型强调风险分析，但要求许多客户接受和相信这种分析，并做出相关反应是不容易的。

因此，这种模型往往适应于内部的大规模软件开发。该模型建设周期长，而软件技术发展比较快，所以经常出现软件开发完毕后，和当前的技术水平有了较大的差距，无法满足当前用户需求。

## Ｖ模型-It公司常用,必须会画

V 模型的左边下降的是开发过程各阶段，与此相对应的是右边上升的部分，即各测试过程的各个阶段。

V 模型的优点在于它非常明确地标明了测试过程中存在的不同级别，并且清楚地描述了这些测试阶段和开发各阶段的对应关系。



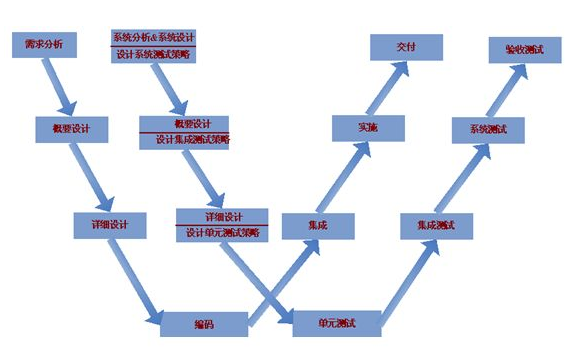
V模型的缺陷及解决思路

V模型仅仅把测试过程作为在需求分析、系统设计及编码之后的一个阶段，忽视了测试对需求分析,系统设计的验证，需求的满足情况一直到后期的验收测试才被验证。

解决的思路是，当一个软件开发的时候，研发人员和测试人员需要同时工作，测试在软件做需求分析的同时就会有测试用例的跟踪，这样，可以尽快找出程序错误和需求偏离，从而更高效的提高程序质量，最大可能的减少成本，同时满足用户的实际软件需求。

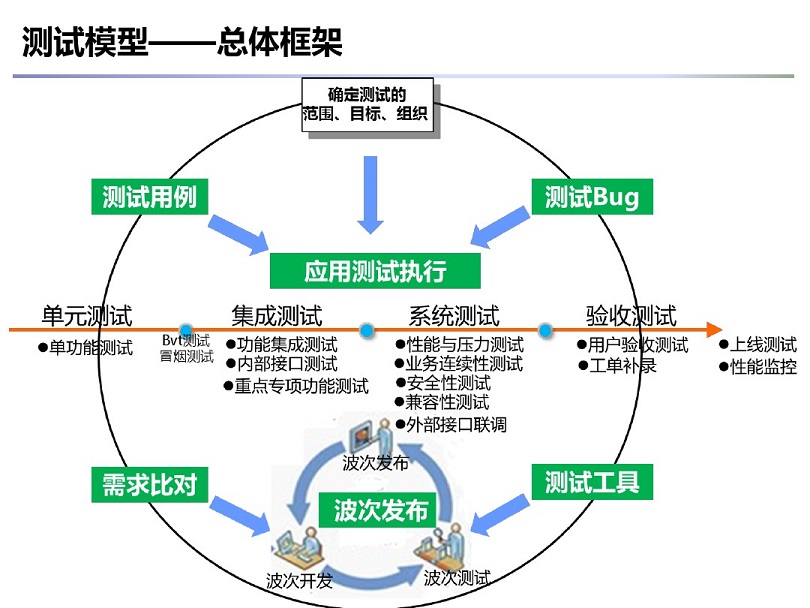
## Ｗ模型!!

相对于V模型，W模型更科学。W模型是V模型的发展，强调的是测试伴随着整个软件开发周期，而且测试的对象不仅仅是程序，需求、功能和设计同样要测试。测试与开发是同步进行的，从而有利于尽早地发现问题。



IMG_256

# 敏捷开发和敏捷测试



敏捷开发

敏捷开发是针对传统的瀑布开发模式的弊端而产生的一种新的开发模式，目标是提高开发效率和响应能力。敏捷开发以用户的需求进化为核心，采用迭代、循序渐进的方法进行软件开发。

由于版本节奏比较快，开发与测试几乎并行，一个版本周期内会有两版在推动，也就是上图中提到的波次发布，波次发布用于尝试新加入的功能，做小范围快速的开发，验证和发布，为下个大版本的功能做实验和调研。快速发版的需求要求测试快速响应，敏捷测试模式适应项目需求。

模型优势：

工作任务划分清晰，工作效率较高

与开发和产品沟通紧密，团队协作性强

测试介入到整个项目的所有会议中，对整体版本信息情况把控全面

模型的缺陷：

模块提交较快，测试时较有压迫感

项目规划要合理，不然测试时会出现复测的现象，加大工作量

## 软件可靠性

### 成熟性

软件系统防止内部错误扩散而导致失效的能力。

▲子系统、模块、单元模块的设计人员应该仔细分析和自身有接口关系的子系统、模块、单元模块，识别出这些接口上可能会传递过来的错误，然后在自己子系统、模块、单元模块内部对这些可能的错误预先进行防范，规避这些错误传递到自身而引起自身的失效。

### 容错性

软件系统防止外部接口错误扩散而导致系统失效的能力。

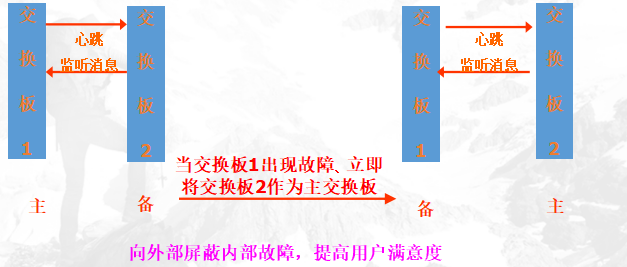
▲设计人员应该充分分析外部接口可能产生的错误，然后在设计上对这些错误一一予以防范，防止这些外部传入的错误波及自身而失效。

### 易恢复性

系统失效后重新恢复原有功能、性能的能力

①原有能力恢复的程度

②原有能力恢复的速度



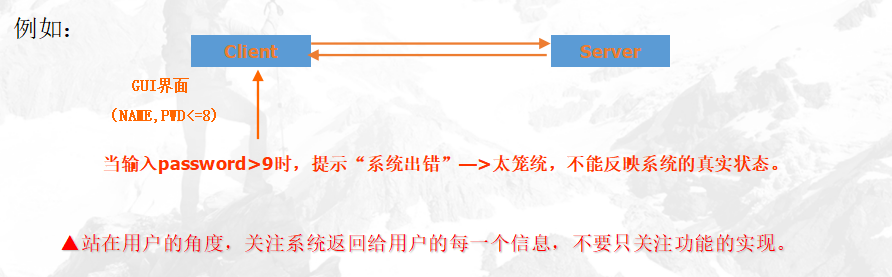
### 可靠性依从性

遵循相关的标准（国际标准、国家标准、行业标准、企业内部规范等)约定或法规以及类似规定的能力。ISO9000

## 软件易用性

### 易理解性

用户在使用软件系统的过程中，系统交互给用户的信息是否准确、清晰、易懂，能帮助用户准确理解系统当前真实的状态，指导其进一步的操作。



### 易学性

软件系统提供相关的辅助手段，帮助用户学习使用它

的能力。

例如：是否有用户手册，用户手册是否有中文版，是否有在

线帮助，界面上控件是否有回显功能等。

### 易操作性

例如：

①Nokia手机和Moto手机在编辑短消息时的方便性差异。

②GUI界面，菜单层次不要太深

③安装软件的过程

错误：给用户大量的安装步骤，每步又有大量分支选项

（把用户当成本软件的专家）

▲测试时应该以非专业的角度来测试过程，往往需要α、

β测试。

### 吸引性

美观：GUI界面、手机外观等

新颖：如夏新手机来电跳舞功能

5、易用性的依从性

遵循相关的标准（国际标准、国家标准、行业标准、企业内部规范等)约定或法规以及类似规定的能力。

## 软件效率（性能测试）

### 时间效率(2-5-10) 358 2 5 8

2 5 10原则:

响应时间在2s内，认为很好

响应时间在2-5s之间，可以接受

响应时间在5-10s，体验下降

超过10s，体验度极差

系统在各业务场景下完成用户指定的业务请求所需的响应时间。

### 资源效率

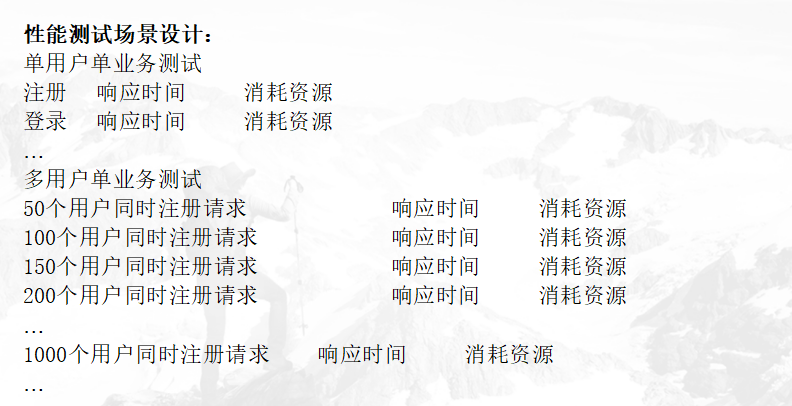
系统在各业务场景下完成用户指定的业务请求所消耗的系统资源，如CPU占有率(70%)、内存占有率、通信带宽占有率、软件内部消息包资源占有率等。

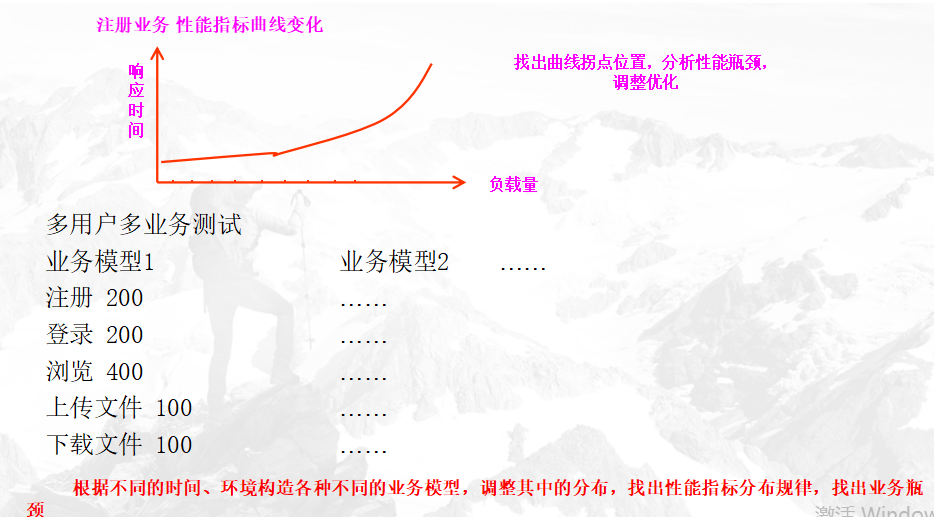
100Mbit 最多理论值 12.5MBit/S

10M

### 效率依从性

遵循相关的标准（国际标准、国家标准、行业标准、企业内部规范等)约定或法规以及类似规定的能力。





## 软件可维护性

### 易分析性

软件系统提供辅助手段帮助开发人员分析识别缺陷、失效产生的原因，找出待修复部分的能力。（降低缺陷定位的成本）

### 易改变性

对软件缺陷的修复容易被实施（降低修复缺陷成本）

▲设计上封装性好、高内聚（同层次设计时，一个实体只完成一个功能）、低耦合，为未来可能的变化留有扩充余地。

### 稳定性

例如：代码中的有物理含义的数字，一定用宏代替。

### 易测试性

（降低发现缺陷的成本）

①软件可控制：

软件系统提供辅助手段帮助测试工程师控制该系统的运行，实现其测试执行步骤的能力（通过打点、改变内部状态、值等手段）

②可观察：

软件系统提供辅助手段帮助测试工程师获得充分的系统运行信息，以正确判断系统运行状态和测试执行结果的力。

a、设计单独的测试模式

b、提供单独的测试版本

▲测试部（一般指测试系统工程师）应该在需求分析阶段就提出可测试性需求，可测试性需求和软件产品其他需求一起纳入需求包被分析设计并实现。

5、维护性的依从性

遵循相关的标准（国际标准、国家标准、行业标准、企业内部规范等)约定或法规以及类似规定的能力。

## 软件可移植性

### 适应性

软件系统无需做任何相应变动就能适应不同运行环境（操作系统平台、数据库平台、硬件平台等）的能力。

▲解决平台无关、可移植性问题的一个常用思路是构造出一个虚拟层，虚拟层将下层细节屏蔽，对上层提供统一口。

### 易安装性

主流平台 全部测试用例

非主流平台 10%测试用例

### 共存性3Q

软件系统和在公共环境与其共享资源的其他系统共存的能力。

▲测试不仅需要关注自身特性的实现，还要关注本软件是否影响了其他软件的正常功能。

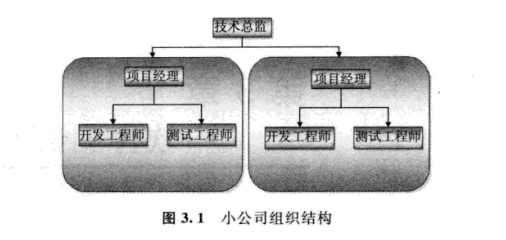
## 易替换性

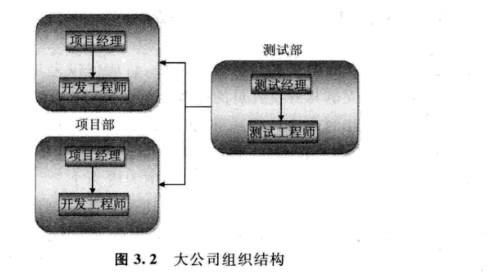
软件系统升级能力（在线升级、打补丁升级等）

可移植性的依从性

遵循相关的标准（国际标准、国家标准、行业标准、企业内部规范等)约定或法规以及类似规定的能力。

## 测试部门的组织结构



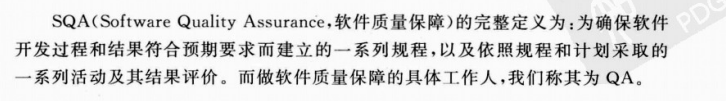


小中型公司:1-2测试，跟着开发的，也有可能是没有测试，独立测试，拓荒者

大型公司:专门的测试部分，测试总监，测试经理，测试工程师

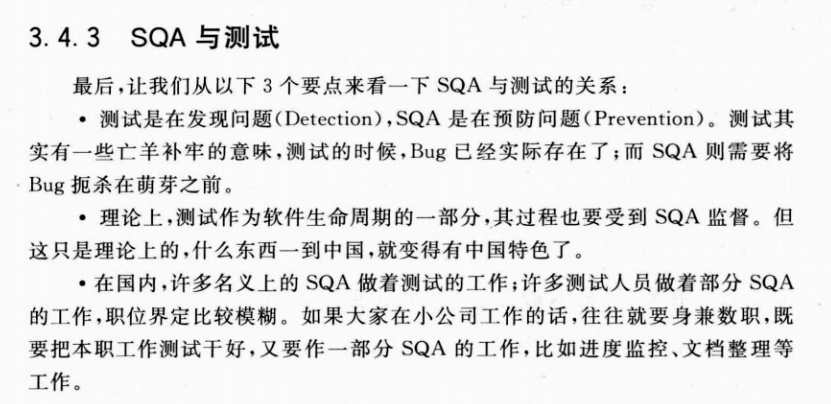
## 软件测试和SQA的关系

### 什么是SQA



Iso9001

### SQA与软件测试工程师



注意：我们国内QA一般就是指的就是软件测试工程师。

QA 质量保证

QC 质量控制