## 软件测试理论课程

**课程大纲**

**一、软件的生产过程**

**二、软件测试概述**

**三、软件测试目的**

**四、软件测试原则**

**五、软件测试分类**

**六、软件的开发与测试模型**

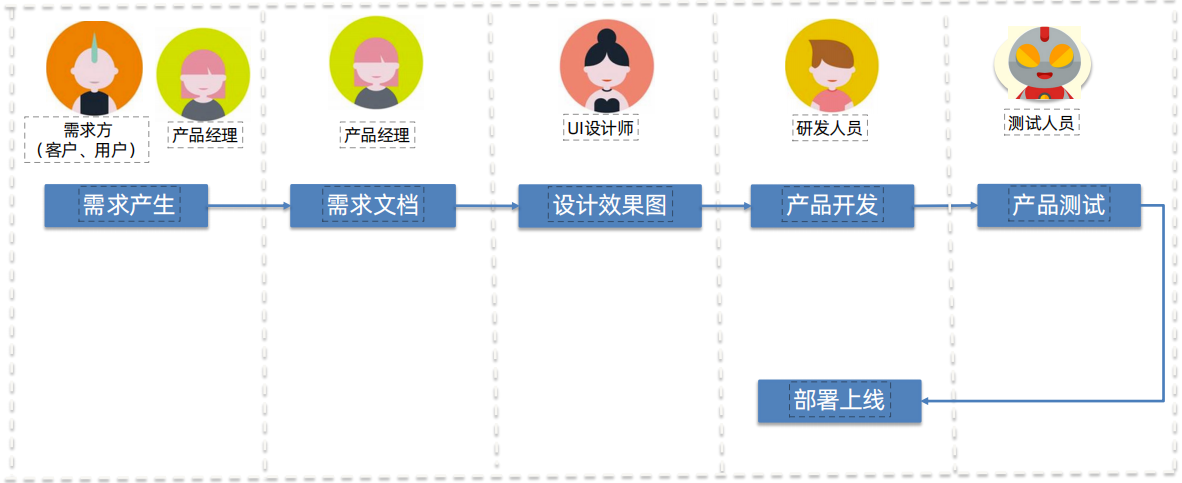
**七、B/S与C/S架构**

**八、软件质量模型**

**九、测试用例**

**十、缺陷概述**

### 一、软件的生产过程



#### 1、软件的生命周期

生命周期的概念应用很广泛，特别是在政治、经济、环境、技术，社会等诸多领域经常出现，其基本涵义可以通俗地理解为“从摇篮到坟墓’的整个过程。

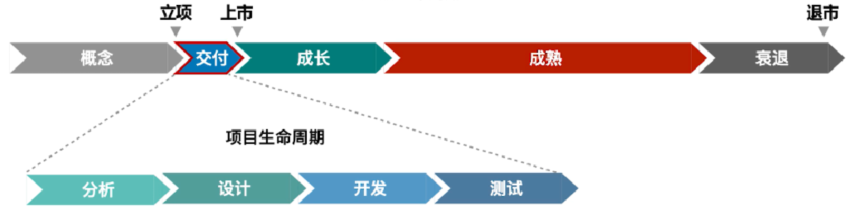
**人的生命周期是什么样的?**

出生、婴儿、儿童、青年、中年、老年.....



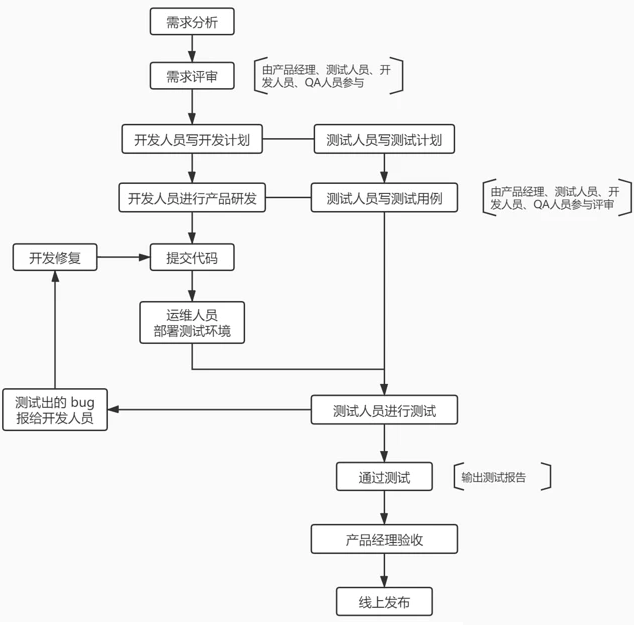
**软件的生命周期**是从概念提出的那一刻开始，软件产品就进入了软件生命周期。在经历需求、分析、设计、实现、部署后，软件将被使用并进入维护阶段，直到最后由于缺少维护费用而逐渐消亡。这样的一个过程被称为“软件的生命周期”。

简单来说：**软件从产生直到报废的过程。**

/



#### 2、软件测试工作流程



**软件测试工作流程（自述）**

项目经理先立项，然后产品开始编写相关的需求文档。写好之后把相关需求文档下发到技术部，测试经理以及开发经理分配好相应的前端开发，后端开发，测试人员。分配到人之后，产品之后开始召集所有的技术人员召开需求分析会议。会议上如果对需求有疑问，及时提出，并讨论好疑问点，最好把需求定稿，然后确定好提测时间以及上线时间。会议结束之后，开发开始写代码，测试开始写测试用例。用例编写完之后需要进行用例评审，用例评审是我这边主导，邀请对应的开发，产品，测试经理来参与，我这边对我的用例进行讲解，会上有提出疑问点的地方，我会做好记录，然后会后进行优化。用例评审完之后，先进行一轮冒烟测试，冒烟测试通过的话，开始根据用例进行全面测试；如果有BUG，就把BUG通过禅道提交给对应开发人员，开发人员进行确认和修复。修复完成之后，再进行复测，如果没有问题，就关闭BUG，继续下面的测试，直到整个版本全部测试结束。上线前，我们测试组会整体的做一遍回归测试，确保老功能没啥问题。测试环境没有问题之后，我们上UAT进行再次回归验证，没有问题了之后再上生产环境，通知产品验收，我们一起对一些新上的功能和老核心功能进行回归验证，没有问题之后，我们会输出相应的测试报告，最后监控和跟进线上的问题情况，大概一个这样的流程。

**工作流程各阶段补充说明：**

**需求分析（评审）阶段**

前提：阅读1遍需求文档，记录不明确之处。

参与人员：前端、后端、测试、产品等项目相关人员。

目的：

1、确保各部门需求理解一致；

2、各角色对需求进行查漏补缺；

3、了解软件有些功能。

提示：需求分析阶段软件还未实现(刚立项)。

**测试计划阶段**

说明：指导测试执行的文档(重要)。

测什么？(目标、范围)

谁来测？(人员进度及安排)

怎么测？(测试工具、测试策略)

**用例设计阶段**

说明：保证能准确验证软件测试点执行的文档。

1、分析需求；

2、提取测试点；

3、设计用例覆盖测试点。

**用例执行阶段**

说明：实施测试。

**缺陷管理阶段**

提交-->验证-->关闭。

**测试报告阶段**

1、bug分析及统计；

2、测试中遇到的问题；

3、测试总结(本次测试中的优点和不足)。

### 二、软件测试概述

#### 1、软件测试的起源与历史

早期软件开发没有测试的概念，开发所做的是调试，目的是发现并纠正软件中的故障。

1957 年，测试与调试被区别开来。但认为测试工作应该往后推，潜意识里认为，测试的目的是验证产品能工作。

1972 年，在美国北卡罗来纳大学举行了首届软件测试正式会议，Bill Hetzel（比尔•海泽尔）在会上正式定义了软件测试的概念。

1979 年，Glenford J.Myers（迈尔斯）在《软件测试艺术》一书中给出测试的经典定义：测试是为了发现错误而执行程序的过程。

1983 年，IEEE 给出了软件测试的标准定义，并制定了测试的标准。

1996 年，Kent Beck（肯特•贝克）在极限编程 XP 方法论中提出 TDD 测试驱动开发理论。

2009 年，James A.Whittaker（惠特克）提出了探索式测试理论。

#### 2、软件测试的定义

通过**手工**或者**测试工具**按照**测试方案**和**流程对**对**被测对象**进行**功能**或**性能**的检测，从而验证**实际结果**与**预期结果**之间是否存在差异。



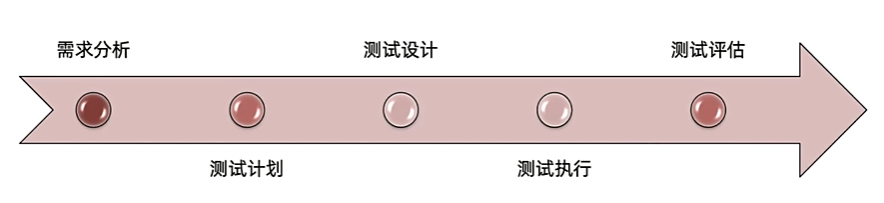
#### 3、软件测试的对象

软件测试对象就是我们的测试工作的主体，在当下的行业里我们绝大多数时候都是在测试某个具体的软件（主体功能），但是我们需要明白一个软件也不仅仅只有功能需要测试，我们可以将软件分为三个部分组成：**功能集合 + 使用说明书 ＋ 配置数据**。



#### 4、软件测试的生命周期

从领取测试任务到本轮测试结束为一个生命周期。



需求分析：

分析需求的正确性和合理性，细化需求，根据需求来提炼测试点。

测试计划：

确定测试范围、目标、人员、工具、时间、环境等。

测试设计：

选择合适的测试方法，完成测试用例的编写。

测试执行：

开发人员已经提交代码，执行测试，提交BUG。

测试评估：

对本次测试情况进行分析和总结。

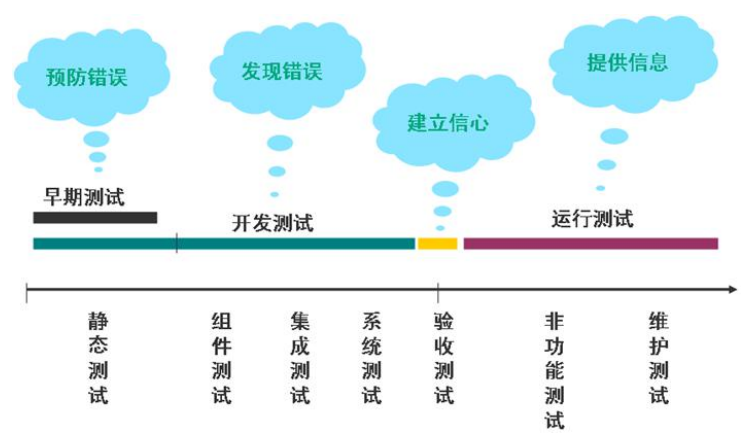
如：写了多少测试用例，执行了多少，发现多少BUG，剩余BUG的解决方案，测试的覆盖率等。

### 三、软件测试目的

提高软件的质量，保证软件的安全，降低软件开发成本；

降低因软件缺陷带来的商业风险；

树立用户对软件的信心。

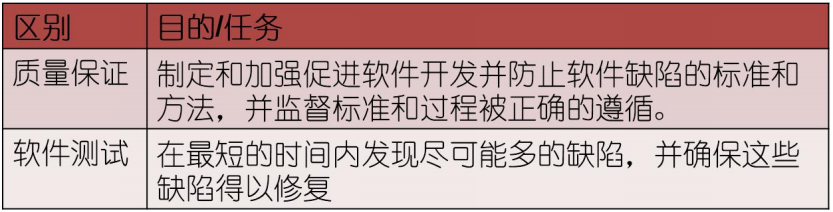


**区分测试 & 调试：**

在目前的开发和测试中，谁去寻找软件中潜在的问题？发现缺陷后谁去修改？



**区分软件质量保证 & 软件测试：**



### 四、软件测试原则

原则，是每个人在行事中所遵循的准则。

软件测试也有一套原则让每位软件测试从业人员去遵循。

**测试只能证明软件存在问题，不能证明不存在问题；**

**测试工作要尽早的介入，降低修复成本；**

**不能进行穷尽测试，应分类别测试；**

**缺陷存在集群现象，通常大部分BUG发生在一小部分核心模块中；**

**杀虫剂现象，尽量多些测试手段以便发现更多BUG；**

**测试依赖环境，对软件开展测试前应先准备好测试的环境；**

**不存在缺陷是谬论。**

### 五、软件测试分类

按软件**产生阶段**划分：单元测试，集成测试，系统测试，验收测试。

按**代码可见度**划分：黑盒测试，灰盒测试，白盒测试。

按**是否运行**划分：静态测试，动态测试。

按**测试手段**划分：人工测试，自动化测试。

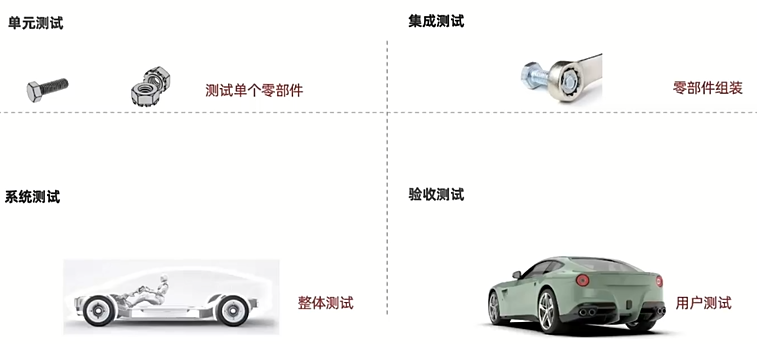
**其他**测试：冒烟测试，回归测试。



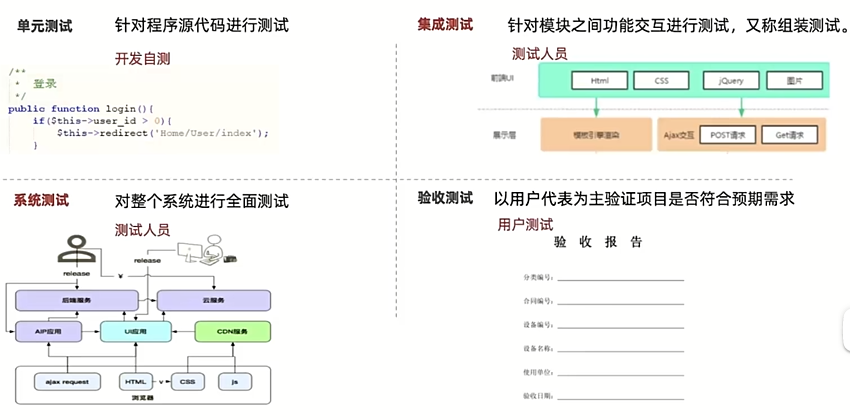
#### 1、按软件生产阶段划分

**单元测试，集成测试，系统测试，验收测试。**

例：生产一辆汽车的过程？



**软件生产阶段是什么样呢？**



例：某电商系统按阶段测试

1、张三开发实现完成注册功能，针对自己开发的该功能代码进行测试属于（）？

A:单元测试 B:集成测试 C:系统测试 D:验收测试

2、从注册、登录、下单联动一起的测试属于（）？

A:单元测试 B∶集成测试 C:系统测试 D:验收测试

3、项目相关开发人员完成了全部系统的核心业务实现，最后提交给测试全面测试，属于（）？

A:单元测试 B:集成测试 C:系统测试 D:验收测试

4、整个电商系统测试通过后，最后交付给用户正式使用，一般需要完成（）？

A:单元测试 B:集成测试 C:系统测试 D:验收测试

**思考：公司验收测试怎么做的？**

#### 2、按代码可见度划分

**黑盒测试，灰盒测试，白盒测试。**



**白盒测试：**

这种测试的主体就是软件的底层代码，不会在意外在的界面是否OK，只要求底层功能实现的同时代码以及逻辑正确。

**黑盒测试：**

又称数据驱动测试，完全不考虑程序内部结构和内部特性，只关心软件的输入数据和输出数据。注重于测试软件的功能以及性能是否满足用户需求**。**

**灰盒测试：**

介于白盒测试与黑盒测试之间的一种测试，既可保证黑盒的关注点又可掌控白盒的内部结构，但不会去对内部程序功能和运作做详细了解，灰盒测试结合了白盒测试和黑盒测试的要素。

**思考：黑盒测试与白盒测试有什么区别呢？**

#### 3、按是否运行划分

**静态测试：**

指不实际运行被测软件，而只是静态地检查程序代码、界面或文档中可能存在的错误过程（文字、界面、展示 等）。

**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。

#### 4、按测试手段划分

**人工测试：**

也叫做手工测试，测试人员手动去进行的测试。

优点就是可以灵活的改变操作步骤及测试环境。

**自动化测试：**

利用代码或者工具帮助人工进行测试。

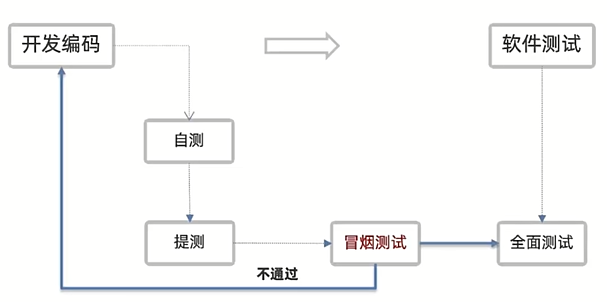
优点就是节省时间、效率高。

#### 5、其他测试

**冒烟测试，回归测试。**

**冒烟测试：**

对核心功能按照全部**正确的数据**或者**流程**，对系统进行最基本功能的测试，保证软件基本的功能和流程能走通。

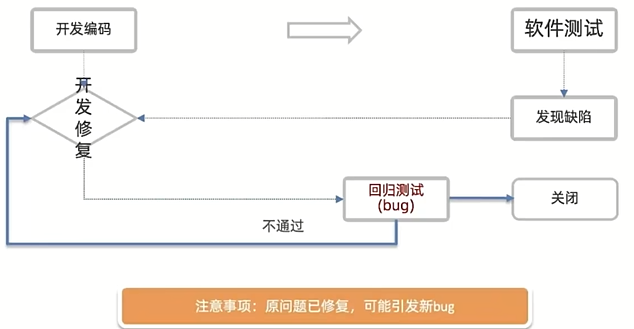


**回归测试：**

1.对已修复Bug，再次测试。

2.软件版本更新/迭代后，对老版本的内容再次测试。

作用：保证Bug修复或确保新功能对旧功能没有影响。





### 六、软件的开发与测试模型

#### 1、软件开发模型

在软件开发的几十年实践中，人们总结了很多软件开发模型用来描述和表示一个复杂的开发过程,如：瀑布模型，快速原型模型，螺旋模型,敏捷开发等。

软件测试与软件的开发模式有着紧密的联系，作为一名测试人员，应该充分理解软件的开发模式，以便找准自己在其中的位置，从而发挥自身的价值。

**瀑布模型**

**需求分析--设计--编码--实现--软件测试--完成--维护。**

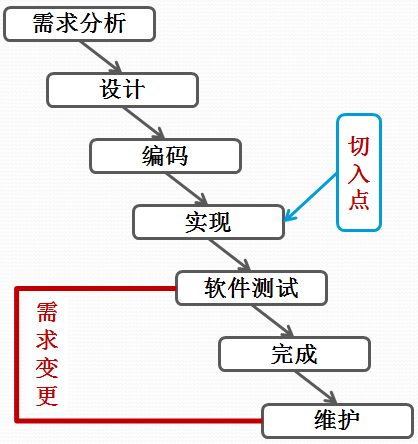
**瀑布模型特点：**

瀑布模型是线性模型的一种，每一个阶段执行一次。

**测试的切入点：**

测试阶段处于软件实现后，必须在代码完成后留出足够的时间给测试活动，否则将导致测试不充分，很多问题到项目后期才暴露。

瀑布模型示意图：



**瀑布模型的优缺点**

**优点：**

开发的各个阶段比较清晰，当前阶段完成后，只需关注后续阶段；

强调早期的计划与需求调研；适合需求稳定的产品。

**缺点：**

不适应需求的变化，依赖早期的需求调研；

风险延至后期才显露，失去了及早纠正的机会。

**改良：**

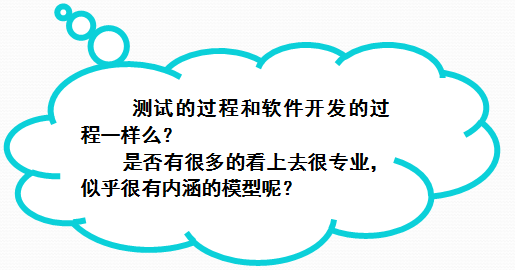
虽然瀑布模型的各个阶段非常清晰，但很容易把前期的风险拖到后期，导致项目延期或失败。

所以我们公司在前期一些重要的阶段之间掺入迭代的思想，尽量避免把风险拖至后期。

#### 2、软件测试模型

随着测试过程的管理和发展，测试人员通过大量的实践，从而总结出了不少测试模型。如：V模型、W模型、H模型、敏捷测试等。

这些模型与开发紧密结合，对测试活动进行了抽象，成为了测试过程管理的重要参考依据。



**V模型**

**需求分析--概要设计--详细设计--编码--**

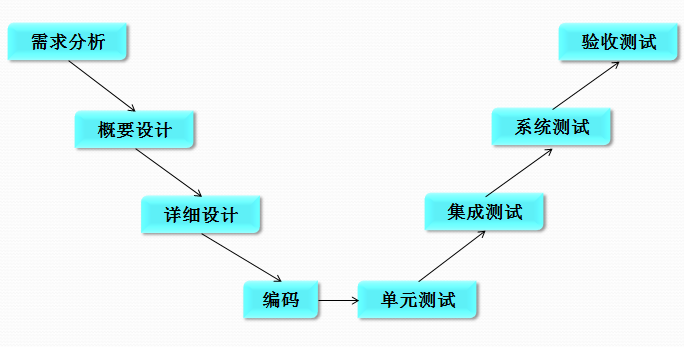
**单元测试--集成测试--系统测试--验收测试。**

V模型是最具有代表意义的测试模型，最早由英国国家计算机中心在20世纪80年代后期提出，旨在改进软件开发的效率和效果。

V模型本身是软件开发中瀑布模型的变种，它反映了测试活动与分析和设计的关系。

V模型标明了测试过程中本身存在的不同阶段，从左到右，描述了开发过程和测试过程间的阶段对应关系。

V模型示意图：



**V模型的优缺点**

**优点：**

包括底层测试（单元测试）以及高层测试（系统测试）；

每个阶段的工作都很明确。

**缺点：**

实际开发工作中，需求经常变更，返工量大，灵活度低。

**思考：V模型能不能像瀑布模型那样改良一下呢？**

**W模型**

测试伴随着整个软件开发周期，并且测试的对象不仅仅是程序，需求和设计同样要测试。

**开发**一个V：

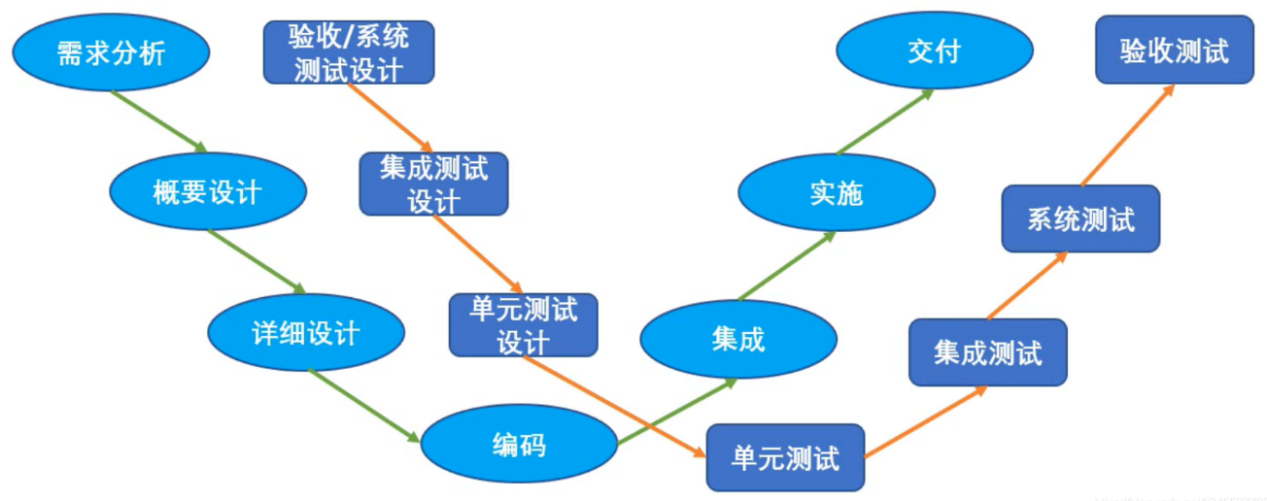
**需求分析--概要设计--详细设计--编码--集成--实施--交付；**

**测试**一个V：

**验收测试设计--系统测试设计--集成测试设计--单元测试设计--**

**单元测试--集成测试--系统测试--验收测试。**

W模型示意图：

**W模型的优缺点**

**优点：**

开发强调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，需求和概要设计同样要测试；

更早的介入测试，以便更早的发现缺陷，降低修复成本；

阶段明确，便于把控项目进度。

**缺点：**

开发与测试依然保持一前一后的线性关系，无法支持迭代、自发性和需求等变更调整；

使用起来技术复杂度很高，对于需求和设计的测试要求很高，实践起来困难。

### 七、B/S与C/S架构

架构可以理解为是用来指导软件产品成型的一种思想，当前软件行业最常见的两种架构分别是：**B/S架构 和 C/S架构**。

**B/S架构（浏览器与服务器）**

**B/S架构的优点：**

分布性强，客户端零维护；开发简单，共享性强；

只要有网络、浏览器即可实现业务处理；

维护方便，只需要改变网页，即可实现所有用户的同步更新。

**B/S架构的缺点：**

在跨浏览器上，存在一定的兼容性；

客户端服务器端的交互是请求--响应模式，响应速度低；

在速度和安全性上需要花费巨大的设计成本。

**C/S架构（客户端与服务器）**

**C/S架构的优点：**

能充分发挥客户端PC的处理能力，很多工作可以在客户端处理后再提交给服务器，所以C/S客户端响应速度快；

操作界面漂亮、形式多样， 能够实现复杂的业务流程；

安全性能更容易保证。

**C/S架构的缺点：**

需要专门的客户端安装程序，不方便实现快速部署安装和配置；

兼容性差，不同的操作系统需要开发不同的程序；

开发、维护成本较高，需要具有一定专业水准的技术人员才能完成，发生一次升级，则所有客户端的程序都需要改变。

**B/S架构与C/S架构的区别？**

**效率**：b/s每次都要加载最新的数据；

c/s效率高，某些内容已经安装在系统中了；

**升级**：b/s可实现无缝升级,通常刷新页面即可完成；

c/s通常要删除老版本再安装新版本；

**安全**：b/s有浏览器即可使用，安全程度低；

c/s更安全，需要安装、注册、登录；

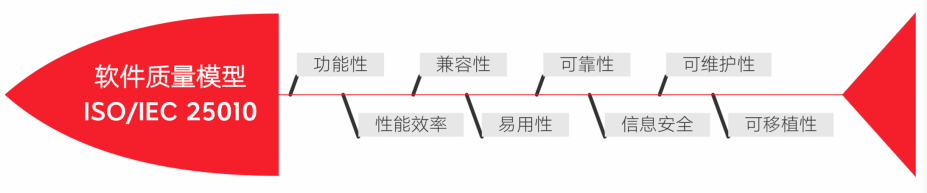
**成本**：b/s有固定的HTML格式，通常开发成本低；

c/s不同的系统需要不同的开发人员，开发成本高。

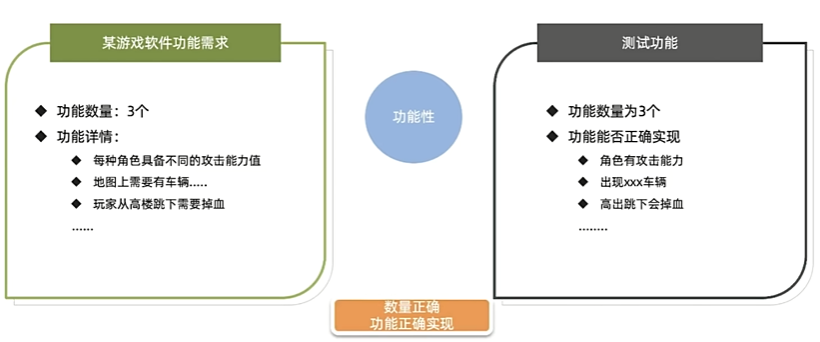
### 八、软件质量模型

衡量一个软件质量的维度。可以理解为软件要达到的质量标准。

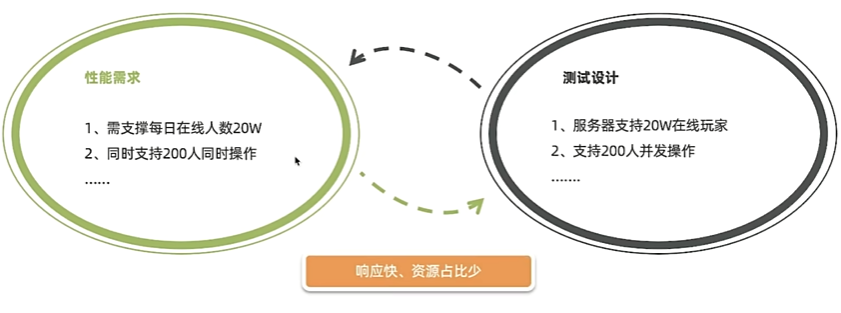
ISO25010软件质量模型是评价软件质量的国际标准，这个模型是软件质量标准的核心，对于大部分的软件，都可以考虑从这这8个特性去测试、评价一个软件。



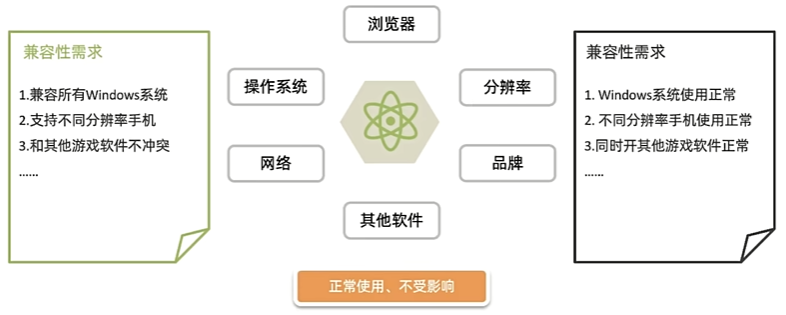
**功能性：**软件是否具备某方面的能力。



**性能效率：**多用户同时使用能否满足要求（时间、资源)。



**兼容性：**在不同的设备/平台上能否正常使用。



**易用性：**易学、易用、用户粘性好。



**安全性：**敏感数据存储/传输安全。



**可靠性：**长时间运行稳定，不出现异常。



**可移植性：**应用系统升级/数据迁移方便。



**可维护性：**运行过程出现问题维护操作是否方便。



**例：如何验证某系统质量呢? 以微信为例**

**1．功能性:与需求数量一致，功能正确；**

**2．性能:响应快、占用资源少；**

**3．兼容性:不同设备平台正常使用；**

**4．易用性:用户体验好；**

**5．安全性:敏感信息无泄密存储有保障；**

**6．可靠性:持久运行无异常；**

**7．可移植性:升级迁移数据不丢失；**

**8．可维护性:出现异常恢复简单、可扩展功能、升级更新便捷。**

**思考：若有一只水杯，你从哪些角度去测它呢？**

### 九、测试用例

**什么是用例？**

用户使用的案例。



**什么是测试用例？**

测试用例(Test Case)是为测试项目而设计的执行**文档**。

解决测什么？怎么测？等此类问题。

**作用:**

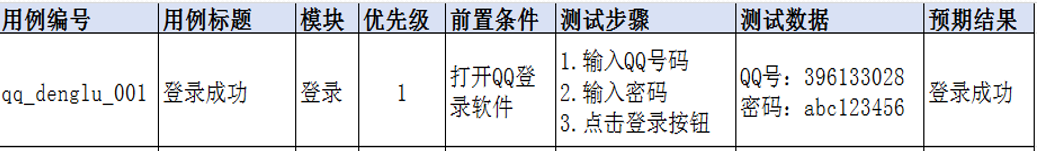
防止漏测、重复测试，实施测试的标准，以便更好的开展测试工作。

#### 测试用例编写格式（要素）★

**通常用Excel文档编写，要素由以下八个要素成：**

**用例编号、用例标题、测试模块、优先级、**

**前置条件、测试步骤、测试数据、预期结果。**



**测试用例编写格式说明：**

**用例编号：**通常由**项目\_模块\_编号**组成；

**用例标题：**预期结果（测试点）；

**测试模块：**所属项目或模块/子模块。

**优先级：**表示用例的重要程度或影响力，

由高至低依次为1、2、3、4（也有称P0、P1、P2、P3）；

**前置条件：**要执行此条用例，有哪些前置操作；

**测试步骤：**描述操作的步骤；

**测试数据：**操作软件需要的数据，没有可以不填；

**预期结果：**期望达到的结果（唯一性）。

**如何划分优先级呢？**

**1级（P0）**为核心功能测试用例，此部分测试用例如果执行部通过会阻碍大部分其他测试用例的验证。

**2级（P1）**为高优先级测试用例，最常执行，以保证软件的基本功能性是可用、稳定的。

**3级（P2）**为中优先级测试用例，全面地验证功能的各中情况；如：异常测试，边界值、执行中断、断网、容错、UI等测试用例。

**4级（P3）**为低优先级测试用例，不常常被执行，如：性能、压力、兼容性、稳定性、安全、可用性等等。

**测试用例的5W1H（了解）**

测试用例必须给出测试目标、测试对象、测试环境要求、输入数据和操作步骤，概括为5W1H。

**测试目标**：Why 为什么而测？功能、性能、可用性、容错性、兼容性、安全性等。

**测试对象**：What 测什么？被测试的项目。如：对象、函数、类、菜单、按钮、表格、接口、整个系统等。

**测试环境**：Where 在哪里测？测试用例运行时所处的环境，包括系统的配置和设定等要求，也包括操作系统、浏览器、通讯协议等单机或网络环境。

**测试前提**：When 什么时候可以测？测试用例运行时所处的前提或条件限制。

**输入数据**：Which 那些数据？在操作时，系统所接受的各种可变化的数据。如：数字、字符、文件等。

**操作步骤**：How 如何测？执行软件和程序的先后次序步骤等。

如：先打开对话框，再点击按钮等。

**练习:QQ登录测试用例**

**1.账号密码正确，登录成功；**

**2.账号错误，登录失败；**

**3.密码错误，登录失败。**





### 十、缺陷概述

#### 1、缺陷定义

软件在使用过程中存在的任何**问题**都叫软件缺陷，简称Bug。

软件缺陷的存在会导致软件产品在某种程度上不能满足用户需求。



#### 2、缺陷判定标准

**少功能**：软件未实现需求说明书中明确要求的功能。

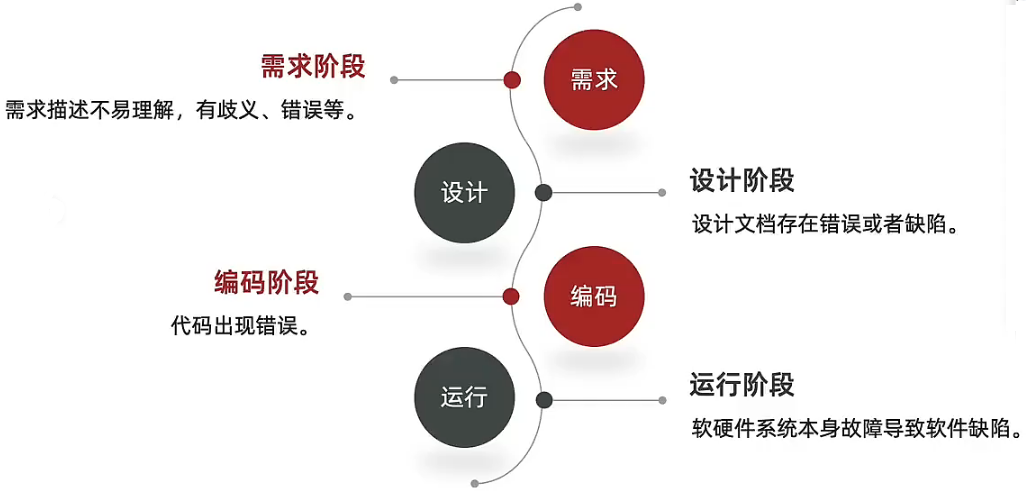
**多功能：**软件实现的功能超出了需求说明书指明的范围。

**功能错误：**软件出现了需求说明书中指明不应该出现的错误。

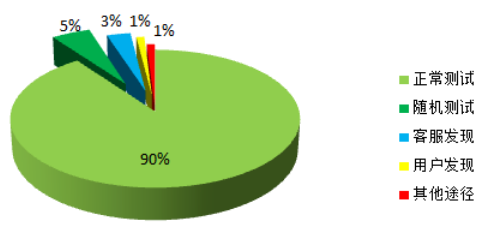
**隐形功能错误：**软件未实现需求说明书中虽未明确指明但应实现的要求。

**不易使用：**软件难以理解，不易使用，运行缓慢，用户体验不好。

#### 3、缺陷产生的原因

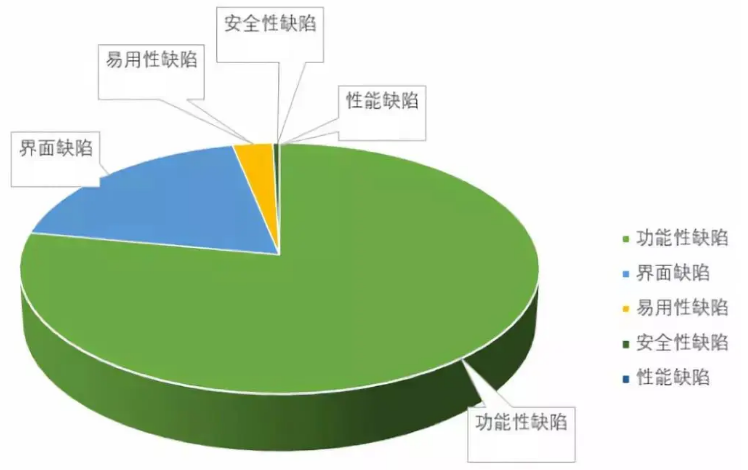


#### 4、缺陷发现的途径

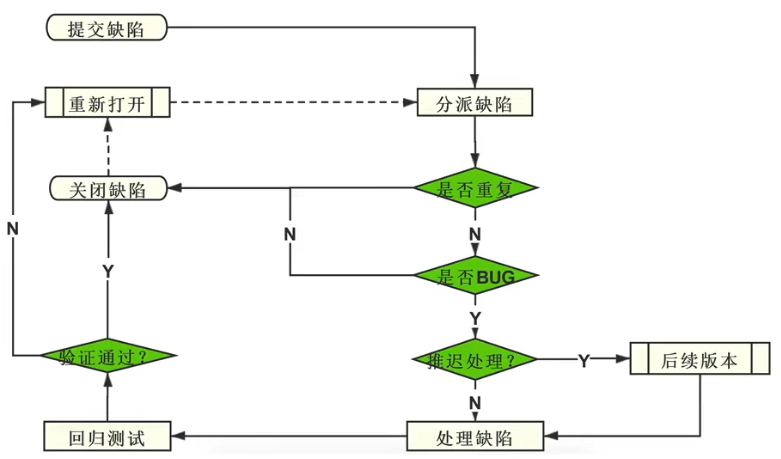


#### 5、缺陷的类型





#### 6、缺陷的跟踪流程 ★



**缺陷的跟踪流程（缺陷的生命周期）自述：**

测试人员提交一个BUG，

对应的开发人员首先判断这个BUG**是否**重复？

如果是重复提交的缺陷则会将其关闭；

如果不是重复提交的，则会复现BUG，以确认测试人员提交的**是不是**一个BUG？

如果不是BUG则会拒绝修改；

如果是一个BUG，会再次判断**是否**推迟处理？

如果不能推迟处理，则立即修复；

对于推迟处理的缺陷，需开会研讨（与会人员通常有项目经理，产品，开发，测试等项目相关人员），会议结果如果认为可推迟则确认推迟处理的日期和版本，待后期跟进；如果会议结果表示不能推迟，则必须修复。

开发人员将缺陷修复之后，测试人员对其进行复测；

如果复测通过则关闭BUG；如果复测没有修复，则重新将这个BUG提交给对应的开发人员进行重新修复。

#### 7、缺陷状态

**提交—打开—拒绝—修复—推迟—关闭**

**提交：**已提交的缺陷；

**打开：**确认“提交缺陷”，等待处理；

**拒绝：**拒绝“提交缺陷”，不需要修复或者不是缺陷、重复缺陷、无法重现；

**修复：**缺陷被修复；

**推迟：**可在以后解决，但要确定修复日期或版本；

**关闭：**确认修复的缺陷，将其关闭。

#### 8、缺陷核心内容★

提交Bug单时，对缺陷的描述过程中，这几个元素要重点描述清晰，避免开发人员对你提交的Bug单产生理解上的歧义。

**缺陷标题，缺陷的预置条件，缺陷的复现步骤，**

**缺陷的预期结果，缺陷的实际结果，**

**缺陷的严重程度，修复缺陷的优先级，缺陷的附件。**



#### 9、缺陷的严重程度

通常指缺陷对软件质量的破坏程度，即此缺陷的存在将对软件的功能和性能产生怎样的影响。

共分为4级，由高至低依次为**1、2、3、4（P0、P1、P2、P3）**。

**1级（P0）致命：**死机，非法退出，死循环，数据库发生死锁。

如：软件死机，意外退出，操作系统崩溃。

**2级（P1）严重：**功能不符，严重计算错误，接口数据错误等。

如：主要功能失效或未实现。

**3级（P2）一般：**界面或内容错误，异常操作未给出提示等。

如：非主要功能失效或未实现。

**4级（P3）轻**微：格式不规范，提示窗口文字未采用行业术语等。

如：某个控件没有对齐，某个标点符号丢失等。

#### 10、修复缺陷的优先级

表示处理和修正软件缺陷的先后顺序。即哪些缺陷需要优先修正，哪些缺陷可以稍后修正。

共分为4级，由高至低依次为**1、2、3、4（P0、P1、P2、P3）**。

**1级（P0）紧急：**立即修复。如：1天内修复。

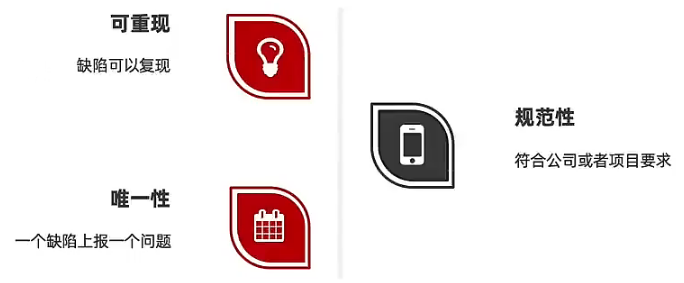
**2级（P1）高级：**几天内修复。如：一周内修复。

**3级（P2）中级：**上线前修复。如：此版本上线前修复。

**4级（P3）低级：**时间允许时再修复。如：在不影响软件主要功能使用的前提下，可考虑在后续版本修复。

一般而言，**严重程度高**的缺陷修复的**优先级也高**。但我们还需要注意，有些BUG严重程度低修复的优先级不一定低。

#### 11、提交缺陷注意事项



#### 12、编写缺陷规范



#### 13、缺陷修改补充说明

并不是所有缺陷都要修复；

有时市场的压力使得产品最终发行有时间限制；

或者测试人员错误理解或者不正确操作引出的缺陷；

错误的修改影响的模块较多，带来的风险较大(遗留)；

修改性价比太低；

缺陷报告中提出的问题很难重现等等，这些原因导致的缺陷会视情况而定。

**为什么开发人员会拒绝修改缺陷？**

程序员无法重现或者现象难以捕捉；

没有明确的报告以说明重现缺陷的步骤；

程序员无法读懂的缺陷报告；

用户很少使用或者不符合用户使用习惯的操作出错；

由不受信任的测试人员提出。

**你提交的BUG开发认为不是个BUG，你会怎么做？**

首先我会看下开发是在什么情况下认为它不是个BUG？

其次我会跟开发核对下需求文档，看是否理解有异；

若开发还是不认可，我会再次把BUG的重现步骤以及相关的截图以及运行日志一并发给开发验证；

实在解决不了我会向经理报风险，通常会在周五下午的周总结会上提出，并解决；若比较紧急，会立即开会解决。

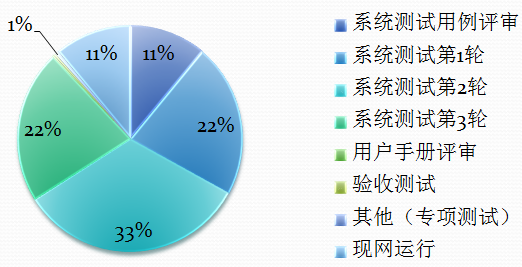
#### 14、缺陷统计

动态采集每个测试周期中发现的缺陷数据，按错误类型、严重程度分类，并有效地控制缺陷的修复率；

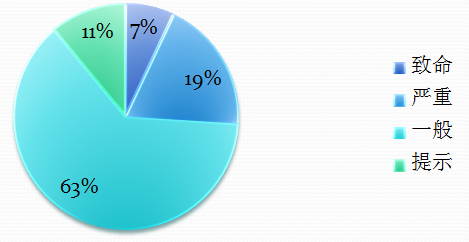
对缺陷的分布进行分析，找出系统的薄弱环节；

对软件缺陷类型进行分析，以便针对各自的特点，先修复严重缺陷。

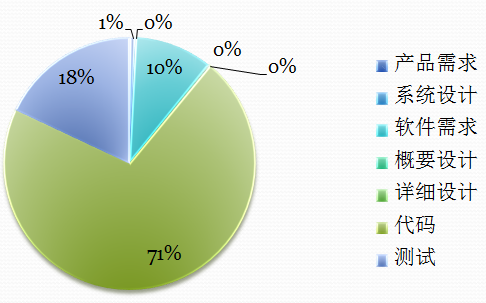
**按活动分布**



**按严重程度分布**



**按引入源分布**



#### 15、缺陷密度计算

进行缺陷密度计算，可有效的进行软件管理。目前行业标准是每一千行代码中存在5个以上Bug。

缺陷总数÷代码行数×1000‰ = 缺陷密度（/KLOC）

如：一个1万行的源程序代码里发现了68个缺陷

则缺陷密度为：6.8/KLOC

计算方式：68÷10000×1000‰=6.8‰

#### 16缺陷数据分析

**缺陷数据分析关注的问题：**

正在测试的软件哪个模块的问题最多？

测试人员中谁报告的软件缺陷最多？

各类缺陷所占的数量百分比分别是多少？

开发人员能及时修复软件缺陷吗？

开发人员一次正确修复缺陷的百分比是多少？

……

**缺陷数据分析的重要性：**

统计未修复的缺陷数目（特别是严重性高的缺陷），预计软件是否可以如期发布？

分析缺陷的类型分布，发现存在较多缺陷的程序模块，找出原因，进行软件开发过程改进。

根据测试人员报告缺陷的数量和准确性，评估测试有效性和测试技能。

根据报告的缺陷修复是否及时，改进软件开发与测试的关系，使测试与开发更有机的配合。

**缺陷数据分析的数据指标：**

每天/周报告的新缺陷数量是多少？

每天/周修复的缺陷数量是多少？

累计报告的缺陷数量是多少？

累计修复的缺陷数量是多少？

不同严重性类型的缺陷数量是多少？

程序模块与发现的缺陷的对应关系？

**小结：**

**1、如何描述软件的生命周期？**

**2、如何描述软件测试的生命周期？**

**3、软件测试的工作流程是什么？**

**4、软件测试的定义是什么？**

**5、软件测试的对象是什么？**

**6、软件测试的目的是什么？**

**7、软件测试的原则是什么？**

**8、软件测试分为哪些类？**

**9、瀑布模型是什么？如何改良？**

**10、V模型与W模型是什么？V模型如何改良？**

**11、B/S架构与C/S架构有什么区别？**

**12、软件测试的质量标准（特性）是什么？**

**13、测试用例的要素有哪些？**

**14、测试用例的作用是什么？**

**15、什么是缺陷？**

**16、发现Bug后，你是怎么确认的？**

**17、缺陷的要素（Bug单）有哪些？**

**18、缺陷的严重程度以及修复缺陷的优先级怎么划分？**

**19、缺陷的处理流程（生命周期）？**

**20、你提交的缺陷开发不认可你怎么办？**

**21、如何提交一条高质量的Bug？**