### 软件测试基础

#### 软件测试的概念

**定义：**

* 使用人工和自动手段来运行或测试某个系统的过程，其目的在于检验它是否**满足规定的需求或是弄清预期结果与实际结果之间的差别**；

通俗讲：

* 找系统的Bug
* 保障产品质量
* 人工测试和自动化测试
* 根据软件需求判断产品是否满足找出预期和实际的区别；

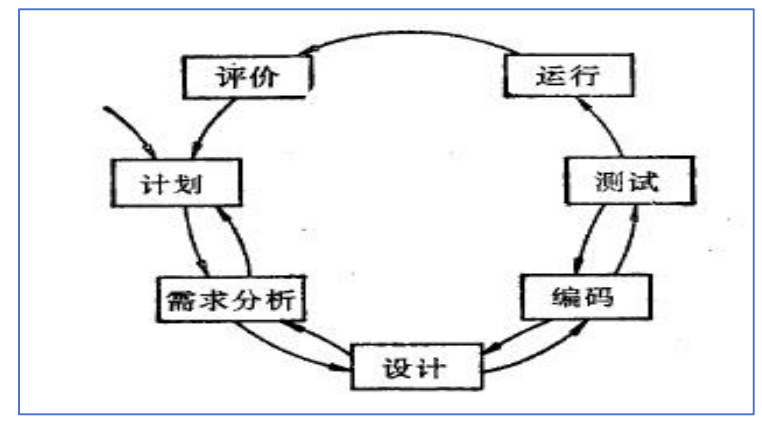
**目的：**

* 找出产品Bug
* 提高产品质量
* 规避风险

#### 软件生命周期

概念

一个软件产品从无到有的研发过程就是软件生命周期；



生命周期是一个从无到有的过程， 主要包括：计划、需求、设计、开发、测试、运维；

在计划到测试的过程中，任何环节出现问题，都可以往前追溯，问题解决后，可以继续往后推进；

在运维阶段存在的需求问题，可以收集整理后，进入新一轮的研发周期；

* 计划：立项：明确：产品方向、产品组织架构、产品周期和资源；可行性分析
* 需求：业务：明确产业的业务模块流程和细节；包括功能需求，非功能需求；输出需求文档，软件需求规格说明书，SRS/ 产品需求文档 PRD ；
* 设计：结构框架设计；对产品功能模块的交互进行技术设计；包括概要设计HLD和详细设计LLD；
  + 概要设计：产品模块的梳理以及之间的交互；
  + 详细设计：功能之间的交互；
* 开发：使用编程语言，代码实现功能逻辑；包括：前端开发、后端开发（接口开发、数据库）
* 测试：验证产品需求是否满足；包括单元测试、集成测试、系统测试；
* 运维：交付维护；保障产品交付后正常运行所做的技术支持；包括：服务器、网络、硬件、软件、数据库等方面的维护；

举例说明软件研发的生命周期？

思路：  
+ 描述整个生命周期过程；  
+ 具体针对整个过程进行举例说明，描述业务细节；  
+ 明确业务之间的关联关系；

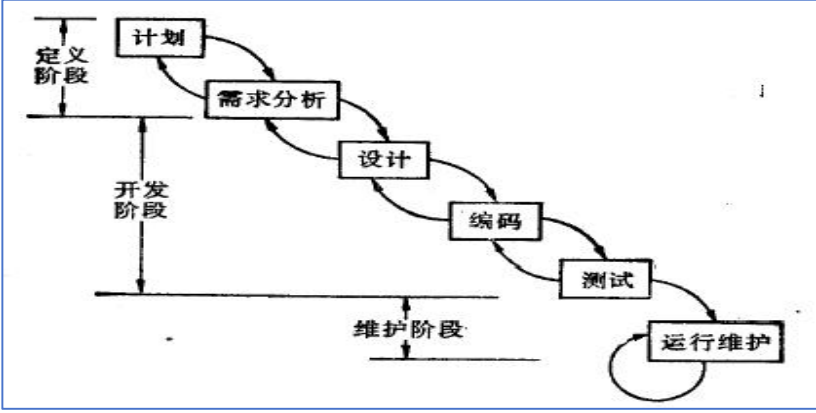
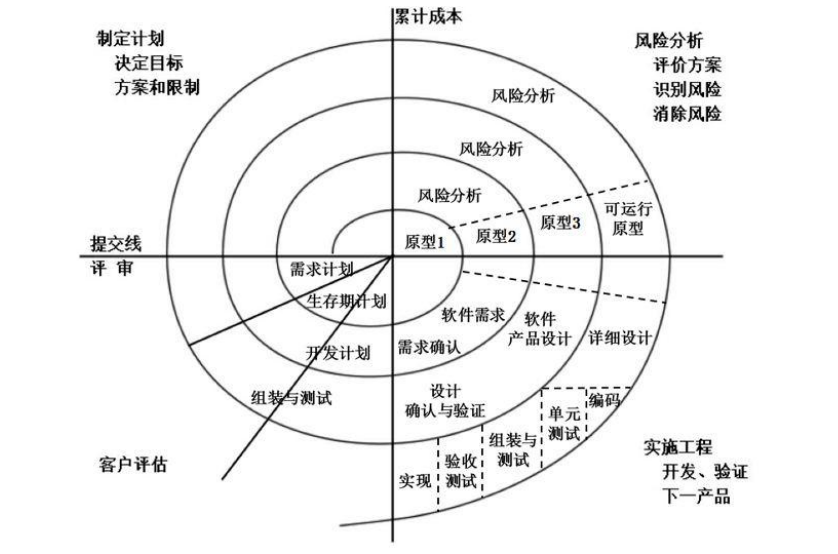
#### 软件研发组织架构

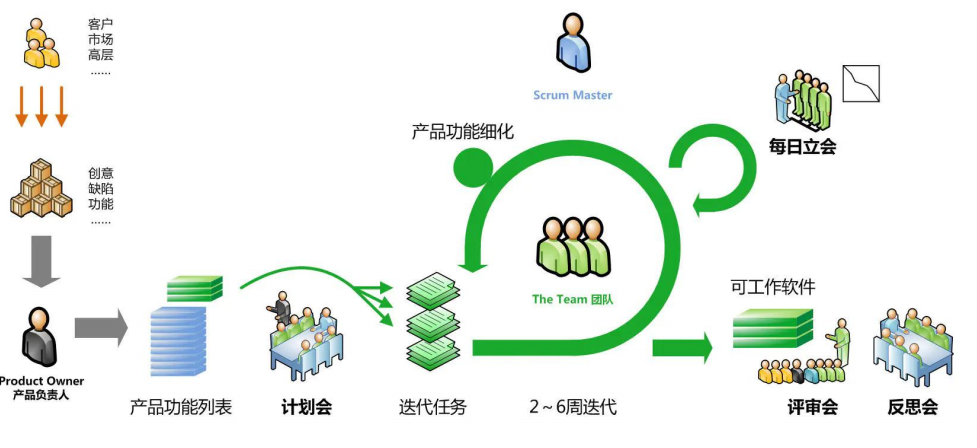
项目经理：PM，负责项目整体的综合管理，是项目的实际负责人和决策人；

* 开发组：写代码；有开发经理/开发组长负责；
  + 开发经理 ： 统筹协调开发的进度，任务分配，时间安排，资源协调；
  + 架构师：负责度产品整体进行框架结构设计；
  + 开发工程师：负责代码实现具体业务模块的功能逻辑；分前端开发：UI界面的实现；后端开发：底层交互逻辑实现；
  + 发布员：负责将代码库中的代码进行版本管理，统筹发布到指定的服务器环境进行运行；
* 产品组
  + 产品经理：PD/pm, 负责业务需求的梳理，挖掘，分析、整理，评审；细化为具体的业务功能；
  + UI设计师：负责产品 原型设计和交付效果图平面设计；
  + 产品助理：协助产品经理负责需求的初步分析和整理，产品完成后的说明书和用户手册的编写；
* 测试组：
  + 测试组长：负责统筹协调测试的人员安排，任务分配，资源协调等；
  + 测试工程师：负责具体页面模块的功能验证，Bug提交和回归测试；
  + 自动化测试：负责代码实现对业务的验证，具体包括脚本编写、自动化框架搭建等；
* 组建一个30人的软件研发团队，各岗位需要多少人？
* 开发：测试=4-5:1  
  产品：测试=1:2-3  
  前端：后端=1:4  
  初级：中级：高级=1:3:1  
  相关岗位可以兼任；

#### 软件研发模型

1. 研发模型：对软件生命周期按照某种设计模式进行研发的过程；
2. 常见的研发模型：瀑布模型、螺旋模型、敏捷模型；

* **瀑布模型：** 把软件生命周期按照流水线的模式进行组合，整个过程分为三个阶段：定义（计划、需求）、开发（设计、开发、测试）、维护（运维）
  + 特点：分成3个阶段保障产品顺利完成，且每个阶段**设置检查点**，确保当前阶段任务完成再进入后一阶段；
  + 不足：周期比较长，产品只有在后期看到结果，**风险比较大**，不适应需求的变更；
  + 适用于需求确定、周期长的项目，比如：军工，金融类项目；
  + 
* **螺旋模型**：把一个大项目拆分为若干小项目，逐步迭代完成；
  + 特点：引入风险分析，强化风险管控；拆分为向项目，便于**进行需求变**更管理；过程中可以看到产物质量有保障；
  + 不足：引入风险分析需要专业人员评估，否则风险更大；**需求变革导致周**期变长；过程交付产物为半成品，不能直接使用；周期长导致成本比较大；
  + 适用于需要需求变更，看到中间产物，且周期较短的项目，比如：科研创新，医疗，电子产品类项目；
* 
* **敏捷模型**：把大项目拆分为若干小项目进行迭代，每**次迭代交付一个可用的产**品
  + 特点：大项目拆分小项目，便于需求管理变更；迭代交付可用产品，降低研发周期和成本；
  + 不足：迭代周期短，频率比较高，对团队的配合要求较高；
  + 适用于互联网中小型项目；比如：电商、教育等类型项目；



问题：说说对软件研发模型的理解？说说他们的区别？

针对不同类型的项目，对软件生命周期的过程进行不同的过程设计。

| 模型 | 项目周期 | 需求变更 | 风险 | 迭代产品 | 适用项目 |
| --- | --- | --- | --- | --- | --- |
| 瀑布模型 | 周期长 | 不适合 | 比较大 | 产品最后一次性交付 | 需求稳定、周期长；金融类、军工类项目 |
| 螺旋模型 | 周期较长 | 适合 | 存在风险 | 迭代交付不可用的半成品 | 需求变更、周期长；科研类、创新类的项目； |
| 敏捷模型 | 周期较短 | 适合 | 较小 | 迭代交付可用的产品 | 需求变更频繁，迭代周期短，频率快；互联网电商、游戏、社交等项目 |

#### 软件测试工作内容

* 功能测试：
  + 测试执行：根据需求设计出来的测试用例进行执行，验证产品的功能是否实现，对存在Bug进行记录跟踪；
  + 设计测试用例：根据需求设计测试用例，灵活运用各种设计方法，确保软件需求被覆盖到；
  + 搭建测试环境：根据软件产品运行的要求，搭建符合要求的测试环境来运行待测软件产品；
  + 制定测试计划：根据产品测试要求，合理安排测试的资源、环境，进度等，确保产品顺利完成测试；
  + 撰写测试报告：根据测试结果反馈，分析测试过程中的风险，评估软件是否可以交付上线；
* 非功能测试
  + 自动化测试：编写脚本代码，设计自动化用例，对业务流程进行回归测试，提高测试效率；
  + 性能测试：使用工具进行压力测试、负载测试，判断产品是否好用，并发量、响应时间等指标较好；
  + 安全测试：借助工具分析产品是否存在信息泄密，权限漏洞、数据加密，信息丢失，网络攻击等问题；

##### 软件测试的原则

* 测试要尽早介入，越早越好，可以从需求阶段介入；
* 测试不能穷尽，灵活应用各种设计方法设计测试用例数据，覆盖更多的情况；
* 用尽可能少的用例覆盖尽可能多的需求，确保需求的覆盖率；
* 测试过程中存在需求问题，需要及时沟通确认，不可臆断；
* 测试缺陷不可能完全测全，要尽可能测试到位，产品的质量与缺陷的遗漏达到平衡；

##### 测试的职业发展

* 技术路线：主要子技术上不断精进，从初级（功能）->中级（自动化）->高级（性能/安全）->资深（测开）；
* 管理路线：技术满足的情况下，根据自身综合能力进行发展：测试组长-> 测试经理 -> 测试总监；
* 测试可以转岗：开发、产品、运维、项目管理等；

##### 测试过程缺陷产生的原因

* 产品需求不清晰，对需求理解存在偏差；
* 开发技术能力不足，导致存在代码方面的错误；
* 产品研发过程中环境工具不统一，导致存在偏差；
* 团队沟通交流不畅，存在理解和表达问题；

#### 总结

1、软件测试的概念和目的；

2、软件生命周期及研发模型；

3、软件研发组织架构及岗位职责；

4、软件测试的工作内容及职业路线；

5、软件测试的原则及缺陷的原因；

##### 面试问题：

1、说说对软件生命周期的理解

2、说说对软件研发模型的了解

3、说说项目组的人员构成？

4、测试在项目中具体负责那些事情；

### 软件测试过程方法

#### 软件测试过程

软件测试过程分为四个阶段：单元测试、集成测试、系统测试、验收测试；

##### 单元测试（Unit Testing， UT）

* 从代码层面测试，属于白盒测试的一种；
* 对开发好一个具有完整逻辑的函数或者方法进行测试，测试代码的逻辑分支是否覆盖；
* 考察代码覆盖率，常见问题：语法问题、逻辑问题；

##### 集成测试（Integration Testing ，IT）

* 从接口层面测试，属于灰盒测试
* 对产品的功能之间、所组成的模块之间、子系统的之间的测试，测试接口的数据传递；
* 考察接口覆盖率

##### 系统测试（System Testing，ST）

* 从整体全局层面测试，属于黑盒测试；
* 对研发好的软件产品，及所需要硬件、软件、文档、数据、工具等环境进行整体的测试；
* 考察需求覆盖率

##### 验收测试（User Accept Testing , UAT）

* 交付测试，在产品交付用户使用前进行的一次全面测试；包括：α测试、β测试；
* 阿尔法测试和贝塔测试的区别
  + 阿尔法测试：在内部测试环境下由公司内部人员进行的一种模拟测试，验证业务功能是否存在问题，也称为内测；
  + 贝塔测试：在上线后挑选部分真实用户在生产环境进行的实际的业务测试，验证产品业务流程是否存在问题，公测；灰度测试；

|  | α测试 | β测试 |
| --- | --- | --- |
| 环境 | 内部模拟环境 | 线上真实环境 |
| 用户 | 内部用户 | 真实用户 |
| 数据 | 模拟数据 | 真实数据 |
| 可控性 | 可控 | 不可控 |

* 验收的标准；
  + 项目的业务流程100%实现；
  + 所有需求都满足要求，包括显性需求和隐性需求，需求覆盖率达到99%；
  + 不存在3级以上的Bug（致命、严重、一般、轻微）
  + 测试用例的通过率达到95%以上；

##### 冒烟测试

* 概念：在**系统测试之\***\*前**对产品进行的一种快速测试的方法，**验证产品是否可测；\*\*
* 是判断开发是否完成，能否进入全方面的系统的，是开发转入测试的必然路径；**由开发和测试共同参与**；
* 冒烟测试用例从系统测试用例中挑选具有**代表性的用例来进行测试**，功能的正向用例、流程用例；
* 冒烟测试的标准
  + 主要业务流程能够实现
  + 核心功能主要功能均已实现；
  + 所有功能中80%的功能都已实现；
  + 冒烟测试用例通过率85%以上；
  + 不存在致命性的Bug，影响产品使用；

##### 回归测试

* 概念：对开发修复好的Bug再次进行验证的过程，确认Bug是否修复，且对其他模块是否产生影响；
* 存在于单元测试、集成测试、系统测试过程中；
* 回归测试策略：完全重复测试、选择性重复测试；
  + 完全重复测试：对Bug所产生的影响比较大或者不确定，对所有功能进行全面的系统测试；
  + 选择性重复测试：根据Bug影响的范围有选择性的进行测试
    - 覆盖修改法：只测试Bug本身对应的功能模块，明确Bug的修改只对自身有影响，不影响其他模块；
    - **周边影响法**：对Bug本身及可能影响的模块进行测试，明白Bug的影响范围；推荐使用；
    - 指标达成法：结合缺陷的影响情况指定Bug修复率的指标，适用于团队稳定，流程规范，经验丰富的团队；

#### 软件测试方法

测试方法按照不同的方式进行划分；

* 测试方式：黑盒测试、白盒测试、灰盒测试；
  + 黑盒测试：不考虑内部的具体实现逻辑，只关注业务整体的实现；比如：功能测试/系统测试
  + 白盒测试：关注程序内部的代码逻辑实现，考虑内部代码逻辑分支、语法规范等要求；比如：单元测试；
  + 灰盒测试：同时考虑底层代码逻辑，和整体业务流程；比如：接口测试/集成测试
* 运行方式：静态测试、动态测试；
  + 静态测试：不运行产品，通过观察分析程序存在的问题。比如：代码走读、文档评审
  + 动态测试：通过运行产品，分析产品存在的问题；比如：功能测试，性能测试；
* 操作方式：人工测试、自动化测试；
  + 人工测试：通过人为操作来进行测试，特指测试执行通过人工来验证；
  + 自动化测试：通过工具代码来进行测试，特别是测试执行部分通过工具代码来实现；

功能测试属于什么测试方法；

#### 软件测试类型

概念：针对软件的某一特性进行的专项测试；

常见的测试类型：

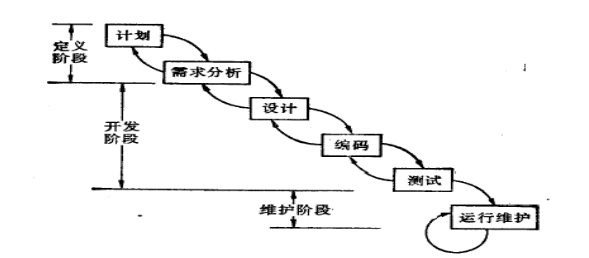
* 功能测试：测试软件**需求是否满足**；主要考虑：需求遗漏错误、显性需求是否满足、隐性需求是否满足；
* GUI测试：用户界面测试，测试用户**界面显示**是否完整统一；主要考虑：页面排版布局，文字颜色、字体大小、按钮控件等是否协调统一；
* 性能测试：测试产品**是否好用，**验证常规性能指标（并发量2000、响应时间2/5/8 \1/3/5、吞吐量、TPS-每秒事务数、资源消耗、CPU/内存80-85%）是否满足：
  + 压力测试：通过不断加载负荷，判断系统所能承受的**最大极限值**；\*\*
  + 负载测试：通过更换不同的负载，判断系统在不同负载下的承受能力，找到系统运行的**最佳负载；**
  + 容量测试：**大数据量测试，**在不同的数据量的情况下判断产品的使用情况，保证产品正常运行；
* 安全测试：测试产品的数据加密、权限访问、网络协议，数据备份、算法等方面是否存在**漏洞风险**；
* 兼容测试：测试产品在**不同的**硬件、软件、数据、网络的等**环境下能够正常**使用；
  + 硬件兼容：主要指硬件设备，比如CPU、主板、内存条等硬件型号；
  + 软件兼容：主要指系统软件，比如：操作系统、浏览器等版本；
  + 数据兼容：主要指产品的新老数据之间的兼容，数据库存储之间的兼容；比如：历史数据与当前数据的兼容，MySQL数据库与Oracle数据库兼容；
  + 网络兼容：主要指产品在不同网络环境下能够正常使用：比如：4G、5G、WiFi
* 安装卸载测试：测试软件能否正常**安装、卸载、更新，**不存在多余的数据或者遗留文档；
* 文档测试：对软件研发**过程中所涉及的所有文档**进行测试，确保文档不存在问题；包括：需求文档、开发文档、设计文档、测试文档等；
* 易用性测试：测试产品是否符合用户的**行为习惯**，方便用户理解、学习和使用，对用户带来较好的**体验**；
* 稳定性测试：测试产品**特定条件长时间运行**，能够正常提供服务，全天候不停歇的运行，不影响产品的使用；
* 可靠性测试：测试产品在**规定的时间规定的条件下完成规定任务**，产品不存在特殊情况；
* 异常测试：测试产品在**人为干预**的情况，验证产品是否具有容错、排错和恢复的能力；
* 网络测试：测试产品在不同**网络**下进行来回**切换**，验证产品是否能够正常运行使用；

记忆口诀：功能性能、界面安全、网络兼容、异常稳定、文档易用、安装可靠；（功G性、网安易、可兼文、异安稳）；

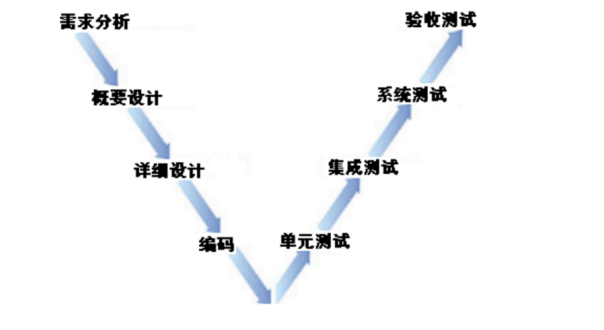
#### 软件测试模型：

##### 瀑布模型

* 特点：在软件生命周期中线性呈现了测试过程，体现测试的必要性，起到了质量保障的作用；
* 局限性：测试后期介入，质量保障作用有限；测试过程中测试内容不明确；

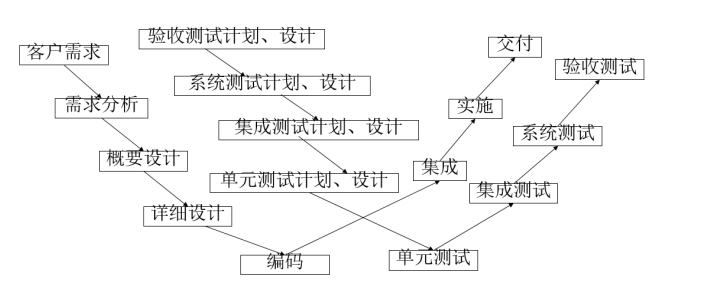


##### V模型

* 特点：测试过程内容细化为单元、集成、系统、验收测试四个阶段；与前面的开发过程建立了对应关系；
* 局限性：测试过程介入较晚，质量保障作用有限；测试周期变长，成本增加；
* 

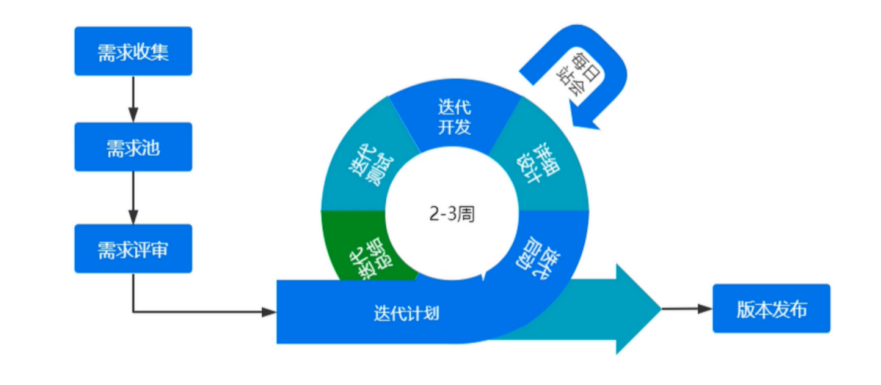
##### W模型

* 特点：每阶段的测试细化为（计划、设计、实现、执行）；测试工作（计划、设计、实现）前移**，实现与开发同步进**行；缩短测试周期，降低了成本；
* 局限性：产品最后一次性交付，工作量大，质量保障有限；



##### 敏捷模型

* 特点：产品划分为若干部分，进行迭代开发测试；缩短交付周期，降低成本；
* 局限性：迭代周期短，频率高，灵活性高，对人的要求较高；



短 平 快

#### 软件测试流程

##### 传统测试流程

* 拿到需求(SRS /PRD、原型设计、开发文档、设计文档)- - > 进行需求分析 - > 制定测试计划（单元测试、集成测试、系统测试）
* 设计测试用例 -> 测试用例评审 - > 拿到版本后进行冒烟测试 - > 冒烟通过进行系统测试执行
* 测试执行及Bug回归测试 - >系统测试报告 - > 验收测试 - > 发布上线；

##### 敏捷测试流程

* 内部测试迭代：
  + 根据需求文档进行测试需求分析 - > 制定测试计划 - > 设计测试用例及评审 -> 拿到版本进行冒烟测试
  + 冒烟通过进行系统测试执行 -- > Bug提交及回归测试 -> 迭代完成进入下一轮迭代
  + 多轮迭代完成 - > 验收测试 - > 发布上线
* 版本交付迭代：
  + 根据需求文档进行测试需求分析 - > 制定测试计划 - > 设计测试用例 - >冒烟测试 - >系统测试- >Bug提交及回归 - >验收测试- >交付上线 - > 进入下一次迭代

##### 总结

1、软件测试的过程（4）

2、软件测试方法（3）

3、软件测试的常见类型（12）

4、软件测试的模型（4）

5、软件测试的流程；（2）

##### 面试问题：

1、你们测试是怎么开展的，说说你们项目的测试流程（敏捷流程）；

2、软件测试的常见类型有哪些（给你一个软件从哪些方面进行测试）？

3、说说对测试模型的理解（你们项目用到的哪种测试模型说说它的特点）

4、常用的测试方法有哪些，如何进行测试？

### 软件测试需求分析

#### 软件需求的概念和分类

* 软件需求：软件产品未实现但需要满足的业务功能或应该具备的能力；
* 软件需求分类：功能需求、非功能需求；
  + 功能需求：产品能够独立完成，且具有一定的业务逻辑的需求；比如：登录、注册、搜查、查看等；
  + 非功能需求：功能需求之前的要求，比如：性能需求、安全需求；

#### 软件测试需求的概念

* 测试需求：对产品测试提出的要求的总和；
* 测试需求分析：通过划分需求来源、分解测试需求类型，并分析测试需求的确定性、可测性、测试次序、重要性、稳定度、工作量等活动，来定义出测试需求的测试范围、优先级、测试风险、关系及约束的过程。
* 测试需求分析：根据产品提供的需求文档（SRS/PRD 、原型设计、开发文档、设计文档）进行分析->功能矩阵（单个功能）、业务流程图（多功能组合）
* 测试需求分析的产物：《功能矩阵》、业务流程图；
* 功能矩阵包含：功能模块、功能点、测试点、优先级；
  + 功能模块：把业务相似的功能整合在一起形成的板块；
  + 功能点：能够独立完成一件具有业务逻辑（通过循环、条件判断是否存在与、或、非）、UI交互、数据变化的任务。**动词、动名词**
  + 测试点：验证功能好坏的各种情况，用于进行用例设计。
  + 优先级：用于判断功能需求的测试点中，需要测试的先后顺序，根据功能需求使用频率、重要程度来进行标识级别：高、中、低；
  + 举例：

| 功能模块 | 功能点 | 测试点 | 优先级 |
| --- | --- | --- | --- |
| 用户模块 | 登录 | 账号密码正确登录 | 高 |
|  |  | 账号不存在登录 | 高 |
|  |  | 密码错误 | 高 |
|  |  | 未输入密码 | 中 |
|  |  | 未输入用户名 | 中 |
|  |  | 登录连续失败锁定 | 低 |

#### 测试点梳理思路

* 先分析正向（合法）情况：功能点可用的情况；
* 再分析反向（非法）情况：导致功能不可用情况，逐一考虑；
* 最后分析异常（特殊）情况：考虑非正常情况，比如网络异常等特殊情况；

#### 软件测试需求分析过程

1、通读需求理解业务逻辑；

2、整理需求中的疑问，找产品去确认；

3、提取功能点，分析对应的测试点，输出功能矩阵；

4、根据业务功能逻辑，画出业务流程图；

(1) 根据需求梳理业务功能或者关键步骤；

(2) 分析业务判断逻辑；

(3) 画出业务流程；

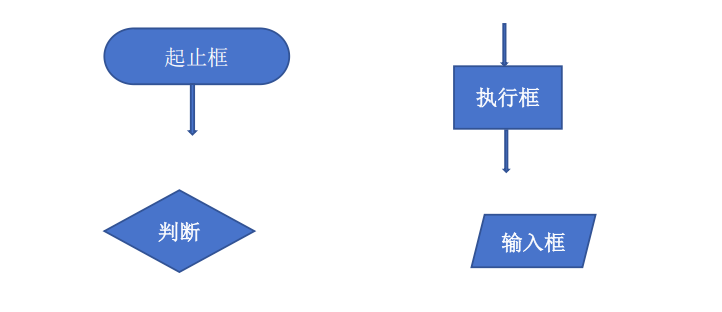
问题：给你一段需求如何进行测试需求分析？

1. 拿到需求（拿到需求文档和原型设计）之后，先熟悉需求，理解业务逻辑；  
2. 整理需求中存在的疑惑，跟产品进行确认；  
3. 从需求中提取功能点，分析每个功能点对应的测试点，输出功能矩阵；  
4. 根据功能点，画出项目的业务流程图；  
5. 对功能矩阵和业务流程图进行评审；

举例说明登录功能的测试点有哪些？（登录功能是怎么测试分析的？）

| 功能点 | 测试点 | 优先级 |
| --- | --- | --- |
| 登录 | 正常登录（手机号、账户、邮箱） | 高 |
|  | 记住密码自动登录 | 高 |
|  | 三方登录（微信、QQ） | 中 |
|  | 账户错误（账户不存在） | 高 |
|  | 账户为空 | 高 |
|  | 密码错误 | 高 |
|  | 密码为空 | 高 |
|  | 登录连续错误2次验证码校验 | 中 |
|  | 登录连续错误5次锁定账户 | 中 |
|  | 多设备同时登录 | 中 |
|  | 多端登录（手机端和PC端） | 中 |
|  | 异地登录 | 低 |
|  | 登录SQL注入安全校验 | 低 |

#### 业务流程图：

使用图形（椭圆、矩形、菱形、平行四边形）和有向线条（直线、折线）把业务逻辑梳理呈现出来，形成的图像；

* 椭圆：起止框，代表业务流程开始、结束；
* 矩形：执行框，代表业务的功能操作，比如支付，点击按钮；
* 菱形：判断框，代表业务的条件判断，一般Yes/No, 是/否两种情况，分别表达两条流程分支；
* 平行四边形：输入框框，代表业务数据的输入，比如登录需要输入用户名、密码；
* 有向线条：通过代有单向箭头的直线或折线，表达业务的流程转；箭头一般从上往下、从左往右画；

#### 业务流程图画法：

1、梳理出业务的所有功能、关键操作；

2、整理出可能的分支判断；

3、根据功能和分支判断画流程图；

4、根据业务检查流程图；

**注意事项：**

1、矩形只能有一个出口，可以有多个入口；

2、菱形只能有两个出口，可以有多个入口；

3、相同的判断只能出现一次；

4、矩形有多个出口，通过连续判断来实现；

5、线条只能直线或折线，尽量避免交叉重叠；

6、整体排版布局，从上往下，从左往右排版，尽量在一屏内显示；

给你一个购物车模块，如何进行测试需求分析？

### 测试用例设计

测试用例：一组用来验证产品功能是否满足的数据集合，包含测试标题、测试编号、测试模块、测试输入、测试步骤、预期结果、优先级、实际结果；

#### 测试用例的要素

测试标题、测试编号、测试模块、测试输入、前置条件、测试步骤、预期结果、优先级、实际结果

##### 用例编号

* 用来唯一标识一条测试用例，每条用例具有**唯一编号；**
* 编号格式：由英文字母、下划线（\_） 、中横线（ - ） 数字组成；不含汉字、空格、其他特殊字符; 避免使用英文单词和拼音混搭；
* 构成要求：项目名称-功能模块-功能点-测试类型-编号 ；产品名称-功能模块-功能点-测试类型-编号
  + 测试类型：UT 、IT、ST 或者 Func、Prof、 Safety、interface 、API
  + 举例：Taobao-Users-login-ST-001

##### 用例标题

* 用来表明测试用例所要验证的测试情况，要体现出测试点和功能点；
* 要求：简洁命明了，用例标题不能重复；
* 格式：在....... 情况下，测试...... 功能； 验证.....功能\_ .......情况
  + 举例：在用户名密码合法的情况下，测试登录功能；
  + 举例：验证注册功能\_用户名已注册

##### 测试模块

* 当前用例所对应的功能模块；
* 格式：大模块/子模块； 模块/功能
  + 举例：会员管理/会员中心 ； 会员管理/ 注册

##### 前置条件

* 为执行当前用例所需要满足的条件；
* 要求：不要把测试过程中的步骤当做前置条件；
* 举例：1. 系统正常运行；2.当前用户已注册；

##### 测试输入

* 测试用例执行过程中所需要的测试数据；
* 格式要求：采用键值对格式， 字段名： 字段值；测试输入不能空白，若没有测试数据，则填入 N/A；若测试数据有特殊要求，则在数据中使用（备注说明）
  + 举例：用户名：张三，密码：123qwe
  + 举例：用户名： （为空）
    - 商品描述：商品介绍...(输入1000个字符)

##### 操作步骤

* 描述测试用执行的具体过程；
* 要求：用例步骤要求有序、连贯、完整；用例步骤不用带具体的数据值，用字段名来代替；步骤中操作按钮使用 【】 进行标识；
* 举例：1. 打开登录页面；2. 在用户名输入框中输入用户名；3. 在密码输入框中输入密码；4. 点击【登录】按钮；

##### 预期结果

* 每一操作步骤按照需求所应该实现的效果；
* 要求：预期结果要与操作步骤一一对应；若一个步骤存在多个结果，则在一条中体现；
* 举例：1. 进入到登录页面；2. 用户名输入框中显示输入的用户名；3. 密码输入框中隐藏显示输入的密码；4. 提示“登录成功”，页面跳转到首页；

##### 优先级

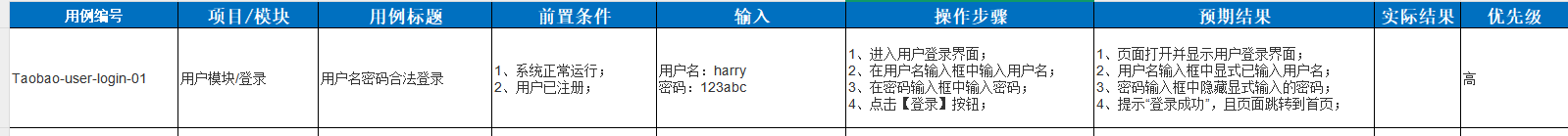
* 代表用例执行的先后顺序；
* 要求：用：高、中、低来表示，优先级高的用例先执行，优先级低的用例最后执行；

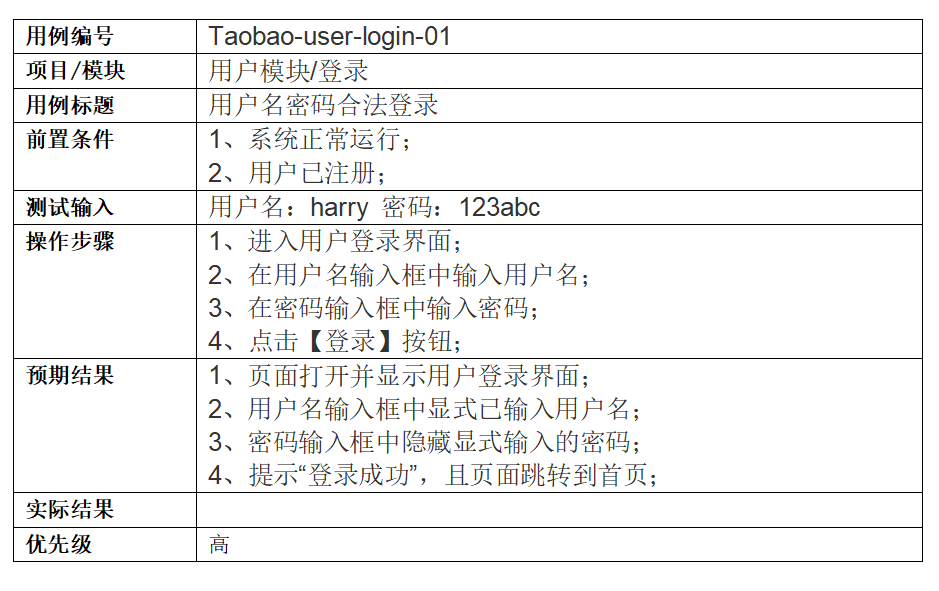
##### 实际结果

* 当前用作执行后的结果，用：通过Pass 、不通过Fail、未执行N/A
* 通过Pass ：代表当前测试用执行满足需求；
* 不通过Fail：测试用例未通过，不满足需求，存在Bug；
* 未执行N/A: 当前用例因特殊原因无法执行；

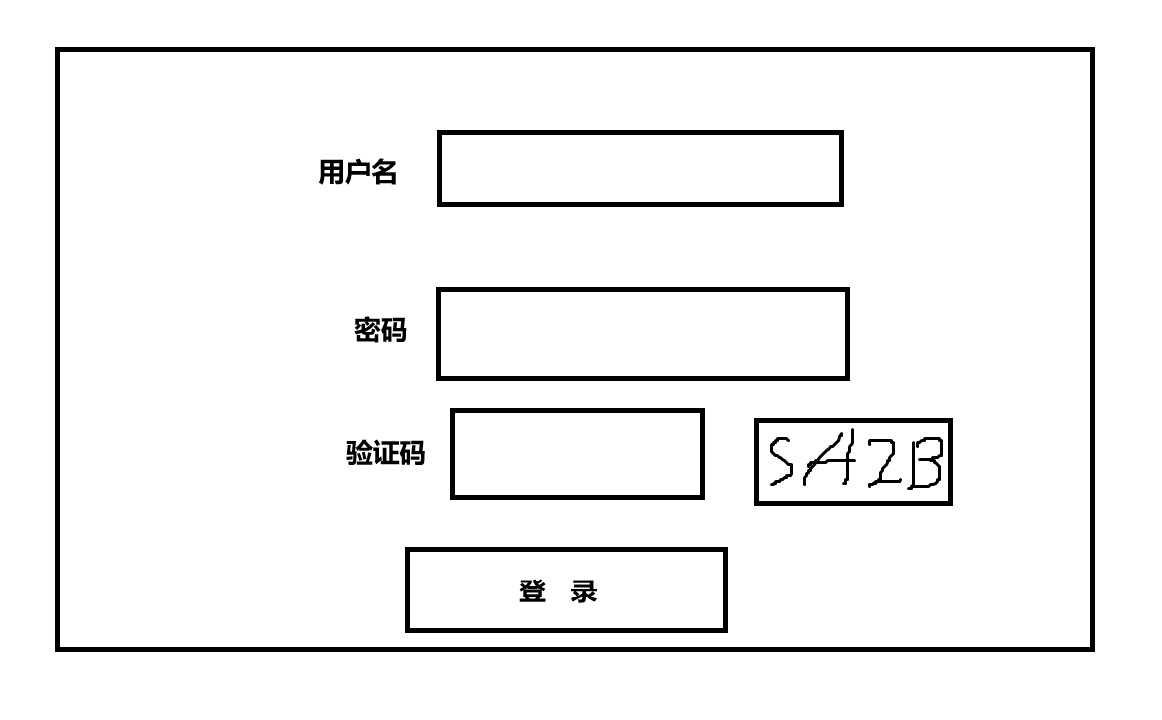
##### 其他要素：

* 测试环境：产品运行所需要的软件、硬件、网络等环境；
* 用例编写人：当前用例的owner；
* 用例执行人：本轮测试执行的操作人员；
* 用例执行日期：什么时间执行的
* 用例执行的轮次；第XX次执行；





**案例：对用户登录模块进行需求分析，并设计测试用例；**



#### 测试用例设计方法

**常用黑盒测试用例设计方法包括**：

等价类划分法 、边值分析法、因果图法、判定表法、状态迁移图法、流程分析法、正交实验法、异常分析法、错误猜测法等。

* 针对单个输入条件的取值：等价类、边界值；
* 多条件的组合的情况：因果图、判定表；
* 业务流程的场景：状态迁移、流程分析；

#### 等价类划分法

##### 概念

* 概念：将庞大的测试数据，按照类别划分若干部分，从每个部分中挑选具有代表性的数据来进行测试，挑选的数据在该类别中具有等效性；
* 等价类按照逻辑通常划分为正向和反向两类，即有效等价类和无效等价类；
  + 有效等价类：确保业务功能是合法、正确的数据；
  + 无效等价类：证明业务功能是不可用的，非法、错误的数据；

##### 等价类划分的原则

* 输入数据是一个取值范围：一个有效、两个无效；
* 输入数据是一个取值集合：一个有效、一个无效
* 输入数据是一个布尔量：一个有效、一个无效；
* 输入数据是一组遵守的规则：一个有效、若干个无效；
* 等价类可以根据业务进一步划分；

##### 等价类划分的用例设计步骤

* 根据需求找到输入条件；
* 根据输入条件，判断等价类划分；
* 列出等价类划分表，明确每个条件的有效和无效等价类；
* 组合测试情况，生成测试数据
  + 先组合有效等价类：尽可能所有条件的有效等价类全部进行组合，若有效等价类存在条件并列，或者条件互斥，则分开覆盖；
  + 再组合无效等价类：依次组合，一个条件的无效项与其他条件的有效项进行组合；
* 根据生成的测试数据设计测试用例

#### 边界值分析法

##### 边界值分析法概念：

对于带有边界范围的业务，错误往往发生在边界附近，若边界附近不存在错误，则范围区间也不会出现错误；

* 对于范围：开区间和闭区间，开区间不包含边界 ( ) ，闭区间包含边界 [ ]；
* 上点：边界上的点
* 离点：边界附近离上点最近的一个点，遵循“**开内闭外**”；
* 内点：范围内的任意一点；

##### 边界值分析的原则：

* 输入条件是取值范围，取值选择：边界上的点和边界附近的点，上点和离点；
* 输入条件是取值个数是范围，取值选择：取值个数，比取值个数小1，比取值个数大1；
* 输入条件是有序集合，取值选择：有序集合的第一个、最后一个；
* 输入条件是数据结构，取值选择：数据结构的最小值和最大值；

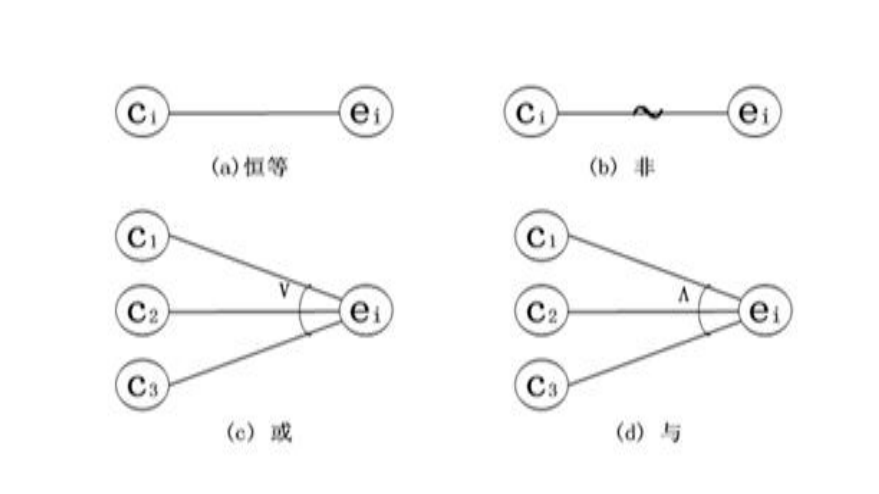
**边界值分析法一般和等价类划分法结合在一起使用，适用于单条件的输入；**

#### 因果图：

##### 概念

找到需求中的原因和结果，利用因果之间的关系，画出因果图来进行分析业务逻辑的方法；

##### 4种因果关系



* 恒等：适用于单个条件和单个结果的对应关系，要求条件成立，结果成立；
* 非：适用于单个条件和单个结果的对应关系，要求与恒等相反，条件成立，结果不成立；
* 或：适用于多个条件和单个结果的对应关系，要求多个条件中至少有一个成立，结果成立；
* 与：适用于多个条件和单个结果的对应关系，要求多个条件要同时成立，结果成立；

##### 原因和原因、结果和结果之间的约束

* 异约束（E）：多个条件之间至多有一个成立；
* 或约束（I）：多个条件之间至少有一个成立；
* 唯一约束（O）：多个条件之间有且仅有一个成立；
* 要求约束（R）：两个条件之间，a成立，b也必须成立；
* 强制约束（M）：两个结果之间，a成立，强制b不成立；

##### 因果图用例设计步骤

1. 根据需求分析出所有的原因（条件）和结果（动作）
2. 根据原因和结果之间的条件关系画出因果图；
3. 根据因果图列出判断表；
4. 根据条件关系列出判断表的所有情况，并进行合并；
5. 根据判断表的每一列生成对应的测试数据；

#### 判定表

##### 概念：

通过明确多个条件之间逻辑关系，进行条件组合的用例设计方法；

* 条件桩：即输入条件的名称，原因；
* 动作桩：即结果的名称，根据条件组合所采取的具体操作；
* 条件项：条件桩的取值，条件项的取值只有真假两种取值，通过1/0 ,T/F , Y /N ；
* 动作项: 动作桩的取值，动作项的取值只有真假两种取值，通过1/0 ,T/F , Y /N ；

##### 判断表用例设计步骤

1. 根据需求分析条件和动作；
2. 明确条件和动作的取值；
3. 根据条件桩确定判定表的规则个数：2的条件次方确定规则个数（列）；
4. 判断表中依次填写条件项的取值：对于每个条件填写一半真，一半假取值；后续其他条件均在上一条件一半真的基础上进行真假填写；
5. 根据条件的组合，结合项目业务逻辑，依次填写每一列动作项取值；
6. 对所有结果进行简化合并：针对结果相同的两列，多个条件中只有一个条件的取值不同其他相同，则可以进行合并；
7. 根据优化后的判断表每一列生成测试用例；

因果图便于对项目也全局进行理解，设计用例时比较繁琐；判断表方便进行测试用例的设计，局限于对业务整体没有全面的理解；

#### 状态迁移图

##### 概念

针对带有状态变化的业务流程，通过明确状态直接的转换关系所形成的路径进行用例设计；

##### 用例设计步骤

1. 明确业务的所有状态；
2. 根据业务确定状态之间的转换关系，形成状态迁移图；
   1. 所有状态横向排列；
   2. 从左往右依次判断每个状态与其他状态直接的关系，存在则用单向箭头标注；
3. 根据状态迁移图生成状态转换树；
   1. 明确根节点：初始状态或者业务的第一状态作为根节点；
   2. 转换树的节点要逐级填写，若同一节点在前一级别中出现过，则后续不用展开；
   3. 相同节点在同一路径中最多出现两次；
4. 根据状态装换树的每一条路径生成测试用例；

#### 流程分析法

##### 概念

将业务操作流程的路径分析处理，根据分析的路径组合为对应的场景进行用例设计；

* 关键要找出基本流（主干流程）和备选流（分支流程）
* 将基本流和备选流进行组合，设计对于的场景用例；

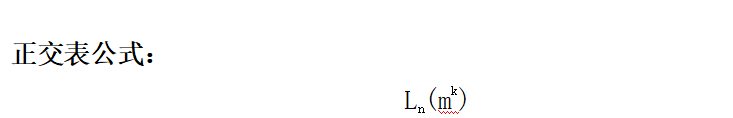
##### 用例设计步骤

1. 根据业务需求梳理业务流程图；
2. 根据业务流程图找出基本流和备选流；
3. 设计场景用例：即基本流和备选流组合；至少保证基本流和任一备选流进行组合一次；
4. 设计场景用例数据，生成测试用例；用例条数至少为：备选流个数+1；

#### 正交实验法

##### 概念：

将多个条件的多种取值进行组合，借助正交实验表公式进行用例设计方法，从中挑选具有代表性的记录做测试用例；

* 因素：条件
* 水平：条件的取值
* 
* 正交实验的特点：整齐可比，均衡分散：每个条件的每种取值的概率是一样，每种情况的组合的概率一样；

##### 用例设计步骤；

1. 根据需求确定因素（条件）和水平（条件的取值）
2. 根据因素和水平选择合适的正交公式；
3. 使用正交表工具生成对应的测试用例；
4. 对生成的用例进行优化完善；

##### 注意事项

* 选正交表时，若存在与因素和水平一致的公式，则直接使用；
* 若每个因素的水平一样，但因素个数不匹配，则选择水平一样，因素稍大的公式；
* 若因素的水平数不同，则选择水平数出现概率最多的公式，未覆盖到的水平进行补充；
* 正交实验只能作为辅助用例设计方法，若存在需求未覆盖的情况，须根据实际情况进行补充完善；
* 正交实验往往用来简化用例设计发复杂度。

#### 异常分析法

##### 概念

通过人工干预进行操作，分析系统可能存在的异常，判断系统是否能够容错、排错的方式设计用例；

##### 常见异常情况

* 人为制造断网、断电、死机情况，判断系统是否正常处理；
* 通过非法更改状态，判断业务流转；
* 同时操作业务，比如同时点击提交按钮；
* 金额、库存数量不足的情况操作
* 协议未勾选，勾选后取消的情况
* 必填项不填写情况
* 构造非法URL进行访问；

#### 错误猜测方法

##### 概念

通过过往项目的知识、经验、直觉，推测系统可能存在错误，进行用例设计；

##### 常见的错误猜测的情况

* 输入类型的业务：输入0、1、空格、特殊字符等容易出现错；
* 带有范围类型业务：边界附近的值容易出现错误；
* 带有数组、列表类型的业务：使用索引进行取值时，取第一个（0/1）或最后一个容易错（为空）；
* 对于线性列表进行处理：列表为空、列表只有一个值、列表中嵌套列表等情况容易出错；
* 时间处理：开始时间、结束时间的第一秒，最后一秒处理时容易出错；
* 数字金额处理：数字的进制，小数的精确度出来容易出现错误；
* 网络处理：网络切换、弱网情况下容易出错；

#### 总结

##### 用例设计策略

* 结合项目灵活选择合适的用例设计方法；
  + 对单个条件的业务，优先考虑使用等价类和边界值；
  + 对于多个条件进行组合的业务，优先考虑使用因果图和判断表；
  + 对于带有流程性的业务，优先考虑使用状态迁移和流程分析法；
  + 对于多条件组合，业务比较复杂的情况，考虑使用正交实验法；
* 项目业务不局限于使用某种用例设计方法，可以根据情况灵活组合；
  + 带有业务流程的项目业务，可以先考虑使用流程分析，再进行判断表条件组合，最后使用边界值考虑单条件的输入；
* 常规用例设计完后，可以基于个人的经验，使用异常分析、错误猜测进行用例补充和完善；

#### 用例评审

##### 用例评审

* 个人在设计用例中难免存在局限性，必须进行用例评审，加强用例对需求的覆盖率；
* 用例评审采用两种方式：组内评审、组外评审；
  + 组内评审:测试组内部进行评审；侧重评审用例的规范；
  + 组外评审：项目组内部评审，产品、开发、测试共同评审；侧重业务的覆盖；

##### 输入项较多的单功能用例设计

* 先考虑正向的情况，确保功能可用：
  + 所有**必填项全部**填写的正常情况；
  + 所有**必填项和选填项**全部填写的情况；
  + 正向情况中带有区间范围，**要特别考虑边界值**，对区间范围的两个边界需要分别设计覆盖；
* 考虑反向的情况，依次验证各输入项非法；
  + 各输入项非法依次考虑输入项：必填为空、内容格式、内容的长度、含特殊字符等情况；
  + 输入项之间存在关联，组合的情况；比如：密码和确认密码关联；
* 考虑异常情况，，非功能特性的情况；
  + 违反业务逻辑：唯一性要求；用户名唯一、手机号唯一绑定；
  + 非功能特性的情况：网络异常、同时操作，多端操作等；

##### 面试问题：

1、项目中对于测试用例有什么的规范要求？

2、常见的测试用例设计方法有哪些，结合项目举例说明如何设计

3、说说购物模块如何进行测试设计用例？

4、满减活动如何进行用例设计；

5、测试用例如何评审的？

##### 测试用例设计方法对比

| 测试方法 | 特点 | 应用场景 | 举例 |
| --- | --- | --- | --- |
| 等价类 | 把输入数据划分有效和无效等价类； 等价类划分表； | 适用于单个 条件输入； 输入是取值范围、取值集合、布尔值、遵守的规则； | 登录、注册等单个功能 |
| 边界值 | 根据边界附近的点设计用例； 上点、离点、内点； | 适用于带有取值范围的场景； 输入是取值范围、取值个数、有序集合、数据结构； |  |
| 因果图 | 分析原因和结果关系来设计用例； 恒等、非、或、与； | 适用于多个条件组合的场景，梳理复杂的业务逻辑； 条件之间要进行逻辑组合，同时满足或者满足其一； | 多条件搜索、满减活动等多条件组合 |
| 判定表 | 根据条件组合来分析判断结果来设计用例； 条件和动作的取值只有真假两种取值； 根据条件的个数明确规则的个数； | 适用多个条件组合的场景，简单快速设计用例； 复杂的条件叠加，简化用例设计； |  |
| 状态迁移 | 根据流程状态变化来设计用例 状态转换树 | 适用于状态之间的复杂交互的场景； | 购物、审批流程 |
| 流程分析 | 根据流程的操作变化梳理业务流程； 基本流和备选流 | 适用于带有流程走向的业务场景； |  |
| 正交实验 | 多条件组合 正交表 | 适用于复杂的多条件组合场景 | 多条件组合 |
| 错误猜测 | 基于项目经验和常识分析判断设计 | 常见情况： 输入框输入特殊字符、输入内容为空、数据值选择边界附近、时间节点在上下限 |  |
| 异常分析 | 基于人为制造异常的场景进行设计 | 常见情况举例： 人为断网断电、并发同时操作、切换网络等 |  |

#### 缺陷管理

##### 缺陷概念

软件研发过程中存在的问题包括，需求方面的缺陷、代码设计层面的错误、运行过程中的故障等；

##### 缺陷的分类

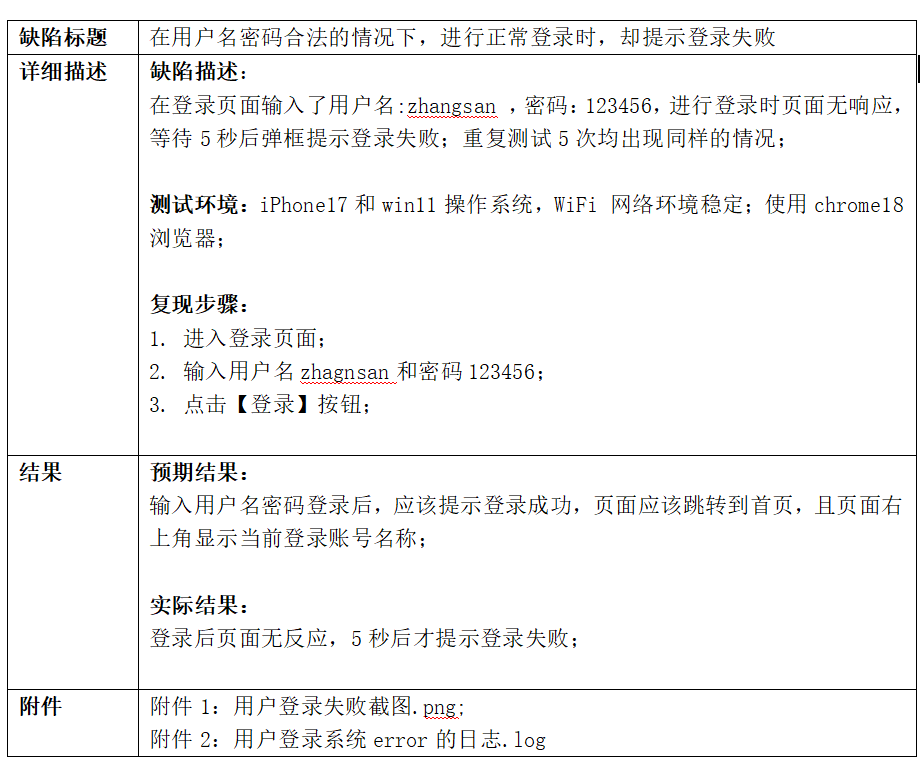
* 遗漏：需求中要求实现，但开发为实现；
* 错误：实现了需求，但与要求的不一样；
  + 显性需求：明确要求实现的功能需求；
  + 隐性需求：辅助性的非功能需求；
* 额外实现：需求中未要求，但实现了；
* 改进建议：总体满足了需求，但从用户体验的角度感受不是很好的地方，可以作为改进建议；

#### 缺陷报告

用来记录发现的问题，对问题进行详细描述，操作复现，数据记录，测试环境复现的文档；用来辅助开发进行问题的定位和解决，便于测试后期的跟踪回归；

##### 缺陷报告的内容

* 缺陷标题 ：简单描述缺陷的现象；要体现出测试的功能点和测试点；
  + 格式：在...... 情况下，测试......功能, 发现.......问题；测试...... 功能时，出现......问题；
  + 举例：在用户名密码合法的情况下，进行正常登录时，却提示登录失败；测试登录时，登录按钮无法点击；
* 缺陷描述：描述缺陷发生的具体情况，包括缺陷的各种现象；
  + 要求：具体呈现缺陷发生时的细节，包括提示信息、页面反馈、操作的前置情况，测试数据等；
  + 举例：在登录页面输入了用户名:zhangsan ，密码：123456，进行登录时页面无响应，等待5秒后弹框提示登录失败；重复测试5次均出现同样的情况；
* 复现步骤：描述发现缺陷的完整过程，详细完整的体现具体操作步骤；
  + 要求：步骤连贯、有序、完整；
  + 举例：1. 进入登录页面；2. 输入用户名zhagnsan和密码123456，3. 点击【登录】按钮；
* 预期结果：按照需求的要求应该实现达到的效果；
  + 要求：不用像测试用例的预期结果，总体体现操作完后的结果；若伴随多种情况，则一起体现出来；
  + 举例：输入用户名密码登录后，应该提示登录成功，页面应该跳转到首页，且页面右上角显示当前登录账号名称；
* 实际结果：实际执行过程中出现的问题现状
  + 要求：总体体现操作后的问题描述；
  + 举例：登录后页面无反应，5秒后才提示登录失败；
* 测试环境：当前缺陷发生时所处的软硬件环境；
  + 要求：描述出当前问题的硬件、软件、网络等环境；
  + 举例：硬件环境：iPhone17和win11操作系统，网络环境：WiFi 环境稳定；软件：使用chrome18浏览器；
* 测试附件：用来辅助描述问题的截图、系统日志；
  + 要求：附件截图直接呈现问题的现象，系统日志能从底层记录问题发生时代码层面的操作；
  + 举例：附件截图：用户登录失败.png; 系统的登录error.log



##### 缺陷的严重等级

为了明确缺陷的严重程度及影响范围，通过严重等级进行标识：致命、严重、一般、轻微；P0 - P4；

* 致命：导致软件出现崩溃、闪退、死机等无法正常使用的情况；
* 严重：导致软件核心功能、主要业务流程无法正常使用、产品数据丢失的情况；
* 一般：导致软件非核心功能，主要功能无法正常使用，存在数据安全风险的情况；
* 轻微：产品出现UI显示错误、界面排版错乱等不影响产品使用的问题；

##### 缺陷的优先级

由开发定义用来确定Bug修改的先后顺序；通过 紧急、高优先级、普通、低优先级；L1-L5 进行标识；

* 紧急：Bug需要立刻马上被解决；
* 高优先级：需要重点安排人员解决；
* 普通：需要正常排队按序解决；
* 低优先级：其他Bug都解决完后最解决；

##### 优先级制定策略

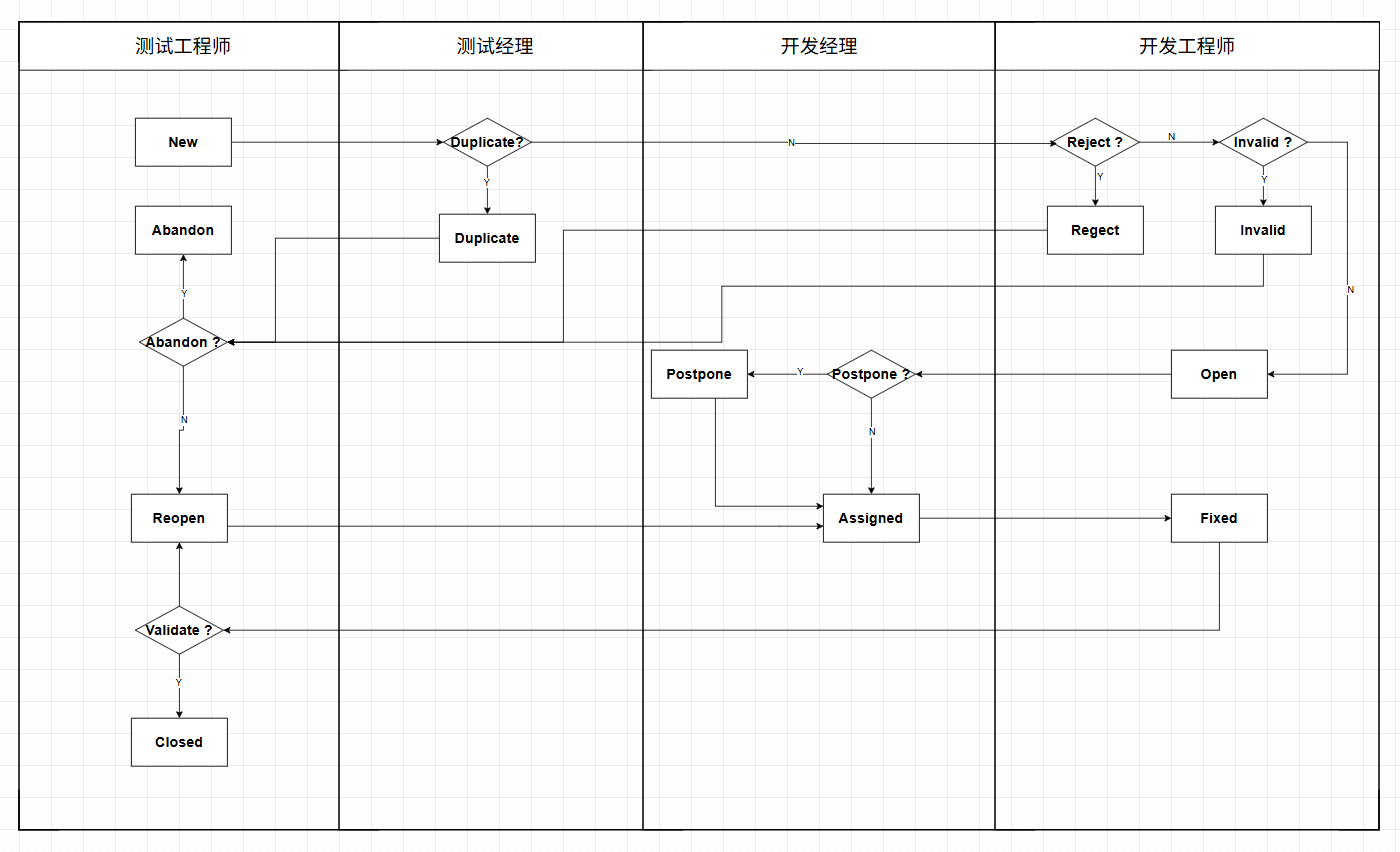
* 由客户提出的Bug要优先解决；
* 线上环境的Bug优先解决；
* 影响产品流程或者阻碍产品研发推进的Bug优先解决；
* 跟金钱相关的业务的Bug优先解决；
* 后台接口或底层代码方面的Bug优先解决
* 与数据库存储相关的Bug优先级解决；

**Bug的优先级与严重等级没有必然联系；**

**Bug的严重等级一旦确定不可随意更改。**  
**Bug的优先级可以根据具体情况进行动态调整；**

##### 缺陷的状态

用来进行缺陷跟踪标识的，明确Bug在不同阶段的处理情况；



#### 测试计划与报告

##### 测试计划

对测试过程进行整体安排，主要包括测试资源、测试任务、测试进度；

##### 制定计划的原则

* 测试计划要**尽早开始**，从全局的角度进行把控设计；
* 制定计划不必过于详细尽善尽美，要保持**足够的灵活性**；
* 制定计划**要切合实际**，根据需要进**行动态调整**
* 制定计划要简洁易读，进行多**渠道评审保**障计划的完整性和全面性；
* 制定计划可以基于5W1H原则进行设计；
* 计划制定的关键在于是否可以落地执行；

##### 测试计划内容

* 测试的需求和范围（测试任务）
  + 测试需求：功能需求（开展功能测试）和非功能需求（开展性能测试、安全测试、兼容测试、易用性测试）；**以功能测试为主，非功能测试为辅**；
  + 功能测试需求：项目**的功能矩阵和业务流程**；
  + 测试范围：明确测试的重点和优先级，比如：功能测试中哪些要测，哪些不测，哪些重点测试，哪些非重点测试；
* 测试的进度和安排
  + 测试进度：明确测试时间周期（测试开始的起止时间），测试的阶段划分（系统测试：计划、设计、实现、执行），时间节点和里程碑（版本迭代关键点）；
  + 测试安排：详细的测试进度推进表；明确什么人什么时间干什么事儿；
* 测试的资源和风险
  + 测试资源：明确测试过程中所涉及的软件、硬件、文档、工具、网络、数据、环境等资源，并进行协调；
  + 测试风险：提前分析预判测试过程中可能存在的影响项目推进和软件质量的各种风险，并给出解决方案；比如：需求风险、人员风险、开发风险、沟通风险，环境风险等；

##### 测试报告

测试报告是对测试过程的总结，对个各个阶段的产物数据进行整理汇总，分析产品可能存在的潜在风险，明确产品的需求覆盖率以及产品交付的质量程度；

##### 测试报告制定原则

* 报告内容**数据化**，用图表的形式直观反应测试的过程；
* 报告内容**个性化**，根据不同阅读对象制定相对应的报告；
* 测试报告**简洁易懂**，便于理解；
* 测试报告要**客观公**正，实事求是；

##### 测试报告的内容

* 测试数据统计分析
  + 主要统计：需求统计分析（测试需求覆盖率）、用例统计分析（用例的覆盖率、用例执行率、用例的通过率）、缺陷的统计分析（缺陷数量、严重等级比例、Bug遗留数量）；
  + 要统计测试过程中**主要阶段，关键版本**的数据；
  + 数据统计以图表进行展示：饼状图、柱状图、曲线图、数据表等；
  + 要对测试过程的数据统计图表进行分析说明，研判测试质量的趋势；
* 测试结论和建议
  + 测试结论：明确测试是否可以进行交付上线的最终结论：建议交付上线、建议延迟交付、暂缓上线；并给出潜在风险分析；
  + 测试建议：针对测试过程中存在问题进行优化改进的具体建议；比如：流程优化、文档标准统一、任务协同的人员时间标准；