第三章 软件测试流程

1. 软件测试流程

1.1 测试流程的阶段

1. 测试计划阶段：需求分析、指定测试计划
2. 测试设计阶段：制定测试方案、设计测试用例
3. 测试实现阶段：编写测试用例
4. 测试执行阶段：执行测试用例、提交缺陷报告、编写测试报告

1.2. 测试的基本流程

1. 需求分析 （评审完善）
2. 制定测试计划（评审完善）
3. 制定测试方案
4. 用例设计和编写（评审完善）
5. 搭建测试环境
6. 执行测试用例
7. bug跟踪处理（评审完善）
8. 测试报告输出

1.3. 主要的测试文档

- 测试计划：指明测试范围、方法、资源，以及相应测试活动的时间进度安排表的文档。
- 测试方案：指明为完成软件或软件集成特性的测试而进行的设计测试方法的细节文档。
- 测试用例：指明为完成一个测试项的测试输入、预期结果、测试执行条件等因素的文档。
- 缺陷报告：指明软件缺陷的现象和复现条件
- 测试报告：指明执行测试结果的文档
- 工作日报：每天测试执行情况的记录和总结。

1.4. 需求分析与评审

- 强调做什么 (what)，不是如何做 (how)

目的：从源头把握软件质量，确保开发结果于实际需求相一致

角色与职责

- 需求人员：《需求规格说明书的编写》，以及软件开发过程中《需求规格说明书》的修正
- 评审人员：评审《需求规格说明书》，从全面性、完整性、正确性、一致性等方面进行检查，将需求缺陷提交给需求人员，并跟踪需求缺陷直至需求缺陷验证关闭

启动标准：《需求规格说明书》编写完成

输入/输出

- 输入：《需求规格说明书》
- 输出：《需求缺陷》

1.5. 制定测试计划

测试计划描述所有要完成的测试工作

- 叙述了预定的测试活动范围、途径、资源及进度安排的文档
- 确定了测试项、被测特征、测试任务、人员安排以及与计划相关的风险

目的

- 明确测试内容、测试任务安排、测试进度、测试资源、风险控制;
- 保持测试过程的顺畅，有效控制和跟踪测试进度，应对测试过程中的各种变更。

角色与职责

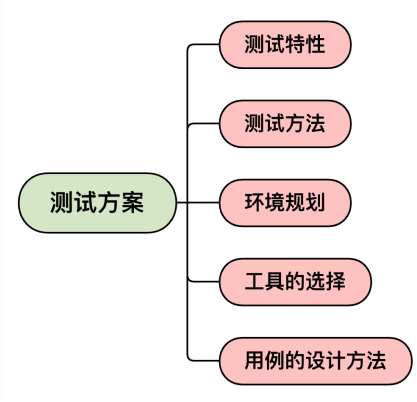
- 测试负责人:根据《项目整体计划》、《需求规格说明书》编制《测试计划》，以便测试工作正常开展。

启动标准：需求评审完成

输入/输出

- 输入：《需求规格说明书》、《项目整体计划》
- 输出：《测试计划》

1.6. 制定测试方案



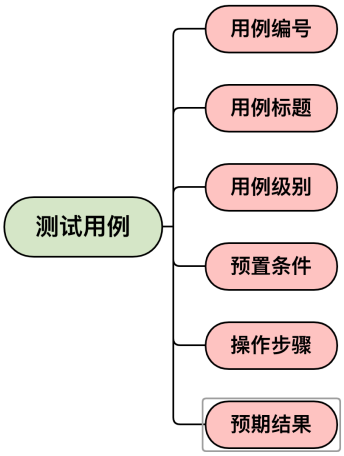
测试方案是从技术上对要测试的系统进行分析和测试设计，根据测试计划中的计划的测试范围、类型来确定测试采用的技术、方法和测试用例的目录大纲，来保证测试的覆盖度。

启动标准：测试计划编写完成，任务分配完成

输入/输出

- 输入：《需求规格说明书》、《测试计划》
- 输出：《测试方案》

1.7. 设计和编写测试用例



目的

以最少的测试用例,实现最大的测试覆盖，保证软件功能的正确性，从而提升软件质量。

角色与职责

- 测试人员
  - 采用多种测试方法编写有效的测试用例，并对遗漏/错误的测试用例进行修正。
- 评审人员
  - 相关的开发和测试人员，对测试人员编写的测试用例进行评审，提出遗漏/错误的用例缺陷，并跟踪直至用例缺陷的验证关闭。

启动标准：测试方案编写完成，任务分配完成

输入/输出

- 输入：《需求规格说明书》、《测试方案》
- 输出：《测试用例》

1.8. 搭建测试环境

一般来说，软件产品提交测试后，开发人员应该提交一份产品安装指导书，在指导书中详细指明软件产品运行的软硬件环境。

1.9. 执行测试用例

目的

验证软件功能和性能与需求的实际匹配程度

角色与职责

- 测试人员
    - 按照测试用例对软件功能进行测试。对于发现的缺陷必须记录，并且跟踪缺陷的状态，直至缺陷的验证关闭。在测试执行过程中发现的遗漏测试用例必须补充至测试用例，保证测试用例与实际测试的一-致性。
- 开发人员：对于测试人员提交的缺陷进行确认、修复。
- 项目经理：对测试人员与实际开发人员意见不一的问题进行裁决。

启动标准：测试用例编写、测试环境搭建完成，程序准测

输入/输出

- 《测试用例》、程序
- 《问题单》

1.10. BUG 提交与跟踪

测试执行过程中，及时确认发现的问题:

- 如果确认发现了软件的缺陷
    - 毫不犹豫的提交问题报告单!
- 如果发现了可疑问题/难复现问题
    - 要保留现场，然后找相关开发人员到现场定位问题。
- 如果开发人员在短时间内可以确认
- 如果开发人员定位问题需要花费很长的时间?
    - 可以让开发人员记录重现问题的测试环境配置，然后,回到自己的开发环境上重现问题，继续定位。

1.11. 编写测试报告

目的

真实客观地对测试过程中各测试阶段、测试项的情况;版本发布的依据。

角色与职责

- 测试负责人
    - 把测试的过程和结果写成文档，并对发现的问题和缺陷进行分析，为纠正软件的存在的质量问题提供依据，同时为软件验收和交付打下基础。

启动标准：各阶段测试完成

输入/输出

- 输入：各测试阶段、测试项实际测试情况
- 输出：《测试报告》

---

2. 需求分析

2.1. 需求概述

软件需求概念

- 解决用户问题或达到用户目标所需的条件或能力
- 用户对目标软件在功能、性能、约束等方面的期待和要求、编写需求规格说明书

重要性

1. 需求是整个测试过程的基础
2. 制定测试计划的基本依据
3. 编写测试用例的重要依据

目的： 生成需求矩阵和测试要点进而得到测试用例

主要内容

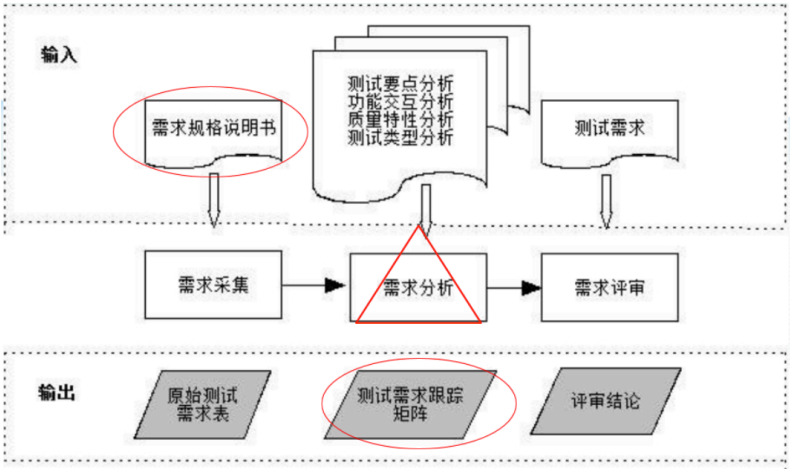
- 功能需求：系统必须完成的那些事.即系统要完成用户提出的各种功能要求
- 非功能需求：软件必须具备的品质或属性，如可靠性、系统响应时间、容错、系统可扩展性等
- 设置约束：客户提出的一-些补充约束说明如系统必须采用Oracle数据库、必须采用商用服务器、必须采用Unix服务器等技术要求.

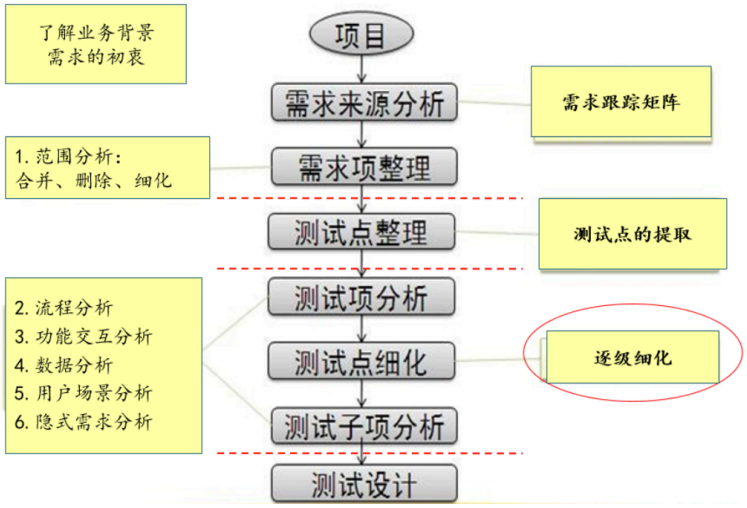
对测试的意义

- 制定测试计划的基本依据
- 编写测试用例的重要依据
- 测试人员工作中非常重要的环节

- 需求分析越详细精准、越有利于设计测试用例

2.2. 需求分析过程





需求规格说明书

- 按照需求分析方法进行需求分析
- 得到需求跟踪矩阵（测试要点）
- 需求评审
- 得到最终的测试要点

在绘制需求跟踪矩阵时候看，可以在表格中直接完成，也可以借助于思维导图工具实现

2.3. 需求分析方法

1）范围分析法

根据需求说明，把功能模块的一段文字，通过删减、合并等方法转化为测试要点的过程。

明确测试范围，输出测试需求列表

1. 列出所需的功能和功能子项

2. 列出性能测试的关注项

把需求文档中的文字转化为一条一条测试要点的过程

我们默认都会用这种方法去进行需求分析，提取测试要点。

除了范围分析法外，还有其他几种方法，可以用来对需求做补充分析，看能否发现新的测试要点。

2）流程分析法

明确每一个功能的业务处理流程

- 常用的或规定的业务流程
- 各业务流程分支的遍历
- 明确规定不可使用的业务流程
- 没有明确规定但是不可以执行的业务流程

3）数据分析法

明确各测试模块的数据流

- 数据的输入
- 已显数据的来源
- 数据的输出
- 数据关联

从输入/输出数据上、数据显示、数据关联上考虑测试要点的过程。

4）功能交互分析法

不同功能点间业务的结合

- 侧重点是功能的实现
- 操作入口明确、合理
- 实现功能的步骤简洁明确
- 交互执行的结果正确完整

从功能交互上细化测试要点的过程。

5）用户场景分析法

模拟用户实际业务场景

- 事件触发控制流程，事件触发形成了流程
- 同一事件不同触发顺序和处理结果形成了事件流

6）隐式需求分析法

挖掘显式需求背后的隐式需求。

需求分析方法总结

拿到需求文档，首先要对文档进行范围分析：---借助表格或者思维导图工具，对文档中给出的显式需求，通过删减、合并等方法提取功能子项，设计测试要点；

此外，可以用其他方法对测试要点补充：

- 流程分析：----画流程图
- 数据分析：----分析数据
- 功能交互分析：---功能实现
- 用户场景分析：---通过事件流/场景
- 隐式需求分析：---经验挖掘隐式需求

基本流程

- 需求分析 -----输出测试要点
- 测试计划-
- 测试方案-
- 用例设计和编写-
- 环境搭建-
- 用例执行-
- bug跟踪-
- 测试报告输出，过程中会有不断的评审和完善

---

3. 软件测试计划

3.1. 测试阶段组成

1）测试计划

2）测试方案

3）测试用例编写

4）测试执行

5）测试评估

3.2. 测试计划定义

- 一个叙述了预定的测试活动范围、途径、资源及进度安排文档
- 确定了测试项、被测特征、测试任务、人员安排以及与计划相关的风险
- 三要素：时间、资源、范围
- 其它方面：策略、风险控制

3.3. 测试计划目的

- 尽早明确工作内容（范围）、测试工作的方法以及需要的各种资源
- 所有涉及到测试的工作人员，尽快将下一步测试工作需要考虑的问题和准备的条件落实
- 重点在于对当前工作任务的准备和规划以及信息的交流

3.4. 测试计划作用

- 计划能给管理者和被管理者指明前进的方向
- 可以减少不确定性对组织的影响和冲击
- 减少无序和浪费
- 有利于管理和控制

3.5. 测试计划包含的内容

软件测试计划是指导测试过程的纲领性文件，包含了产品概述、测试策略、测试方法、测试区域、测试配置、测试周期、测试资源、测试交流、风险分析等内容。

3.6. 核心活动

- 确定测试策略
- 确定测试系统（软件和硬件）
- 预估工作量（资源和时间进度计划）
- 评估事件进度风险和并准备风险缓解计划
- 准备并复查测试计划文档

3.7. 测试计划的设计与实现

- 获取需求文档
    - 定义测试需求：根据用户需求定义并完善测试需求，作为整个测试的标准
- 制定测试策略
    - 确定测试范围（测什么）
    - 确定测试顺序
    - 测试方法（如何测）
        - 不同测试阶段测用不同方法
    - 需求分析阶段
        - 编码和单元测试阶段
        - 集成测试阶段
        - 系统测试阶段
        - 验收测试阶段
    - 测试标准

        - 入口标准
        - 出口标准
    - 自动化策略
- 确定测试系统
    - 测试架构
    - 测试环境
    - 测试配置
- 预估工作量
    - 确定任务
    - 预估工作量
    - 确定时间进度计划
- 风险管理
- 复查测试计划
    - 编写策略、系统、工作量和时间进度文档
    - 与团队一起复查测试计划

3.8. 避免测试计划束之高阁

- 测试计划缺乏参考价值
    - 当那些会对测试计划产生影响的因素发生变化时，要及时更新测试计划的相关内容
    - 计划不是用来应付领导或客户的，而是用来指导实际工作的，因此，计划的内容要正确、详实、具有可行性
    - 若项目过于庞大，可以尝试着把工作阶段分几个更小的阶段来设计完成。把测试工作控制在自己的能力范围内。

3.9. 测试计划和测试文档的不同

- 测试计划是一份描述软件测试工作的目标、策略、方法和重点的文档
- 测试计划的准备过程是思考检查并确认一个软件产品的可接受性的一个有用的方法

3.10. 如何让测试计划有实际作用

- 测试计划缺乏参考计划
- 解决方法:
    - 当那些会对测试计划产生影响的因素发生变化时，及时更新测试计划的相关内容
    - 计划不是用来应付领导或客户的，而是用来指导实际工作的，因此，计划的内容要正确、详实、具有可行性
    - 若项目过于庞大，可以尝试着把工作阶段分几个更小的阶段来设计完成。把测试工作控制在自己的能力范围内。

3.11. 5W1H、6要素

- why 为什么要进行这些测试
- what 测试哪些方面，不同阶段的工作内容
- where 相应文档的存放位置，测试环境等
- when 测试不同阶段的起止时间
- who 项目有关人员组成，安排哪些测试人员进行测试
- how 如何去做，使用哪些测试工具以及测试方法进行测试

3.12. 测试类型和目的

- 功能测试
- 用户界面测试
- 性能测试
- 兼容性测试
- 安全及访问权限测试

3.13. 测试方法

功能测试

- 测试目标
    - 确保所有被测对象功能正常
- 测试方法
- 系统测试阶段所有的测试用例均采用手工方式通过对用户界面的操作来实现
- 完成标准
    - 测试用例全部被执行，提交的缺陷全部被正确的解决
- 特殊事项的考虑
    - 如果由于某项原因导致测试时间被缩短，考虑按测试用例优先级重新选择测试用例

性能测试

- 测试目标
    - 确保系统在一般和极限状态都能正常运行
- 测试方法

词汇表：避免测试计划文档使用者对同一事件有不同解释

确定测试资源

- 人力资源
- 硬件/软件资源
- 其它资源 （如文档的存放位置

测试环境

时间表：开始时间、需要多少时间完成

定义工作进度

确定测试策略

测试优先级

- 风险
    - 风险评估确定不同测试对象的风险
- 用户协议
- 开发部门的进度安排

---

4. 测试用例

4.1. 概念与意义

概念：在实施测试时，对被测软件提供输入数据、操作和预期结果的集合

目的：确定应用程序的某个特性是否正常工作

测试用例设计是软件测试的核心

意义

1. 测试过程是以测试用例为向导的，并严格按照测试用例对软件项目进行测试的
2. 编写测试用例有利于对测试的组织和管理，避免盲目测试，提高测试效率

4.2. 基本要素与格式

测试用例六要素

1. 用例编号：唯一，格式可以是项目名*测试阶段*测试模块_001
2. 用例级别：1-3级，测试用例执行的优先级，级别高的用例先执行
3. 用例名称：提炼测试步骤，用简洁的语言描述，尽量让人看到用例名称就能知道该用例要做什么工作。20个字内
4. 预置条件：执行该测试用例应该提前准备好的环境和数据
5. 测试步骤：把测试用例要做的具体工作拆分一条一条去编写，傻瓜式的编写，让不懂该软件的人也能根据测试步骤执行测试。
6. 预期结果：根据需求，应该看到的结果。

常遇到问题

- 追求用例设计的一步到位
- 多个用例写成一条

编写依据：依据需求

测试用例要定期的维护和更新，不是写好了就一成不变了。

4.3. 测试用例的编写原则

- 用例设计
    - 用成熟的设计方法
- 测试用例
    - 正确性
    - 代表性
- 测试结果
    - 可判定
    - 可再现
- 测试步骤
    - 足够详细、准确和清晰

测试用例编写依据

- 设计测试用例易犯问题
    - 追求测试用例设计一步到位
    - 将多个测试用例混在一个用例中
- 编写依据
    - 建立在需求的基础之上
    - 依据和参考文档和资料

4.4. 测试用例的评审与完善

测试用例修改原因

- 测试用例考虑不周，设计不全面
- 软件缺陷覆盖不完全
- 软件自身的新增功能和更新测试用例也必须更新

测试用例管理

- 配备测试用例管理软件对测试用例进行管理

测试用例评审

- 测试用例的评审能使用例结构更清晰，覆盖用户场景更全面
- 评审内容
    - 结构安排是否合理
    - 优先级安排是否合理
    - 是否覆盖全部功能点
    - 是否具有很好的可执行性

    - 是否删除了冗余用例
    - 是否包含充分负面测试用例
    - 是否从用户层面设计用户使用场景和使用流程的测试用例
    - 是否简洁、复用性强
- 评审种类
    - 外部评审
    - 内部评审
- 评审方式
    - 会议
    - 邮件

测试用例的执行与跟踪

- 全方位的观察测试用例执行结果
- 加强测试过程记录
- 及时确认发现的问题
- 与开发人员良好的沟通
- 及时更新测试用例
- 提交一份优秀的问题报告单
- 测试结果分析

---

5. 缺陷管理与跟踪

5.1. 缺陷概念

- 产品内部
    - 缺陷是软件产品开发或维护过程中存在的错误等各种问题
- 产品外部
    - 缺陷是系统所需要实现的某种功能的实效
- 缺陷的存在会导致产品在某种功能上不能满足用户的需要

语法的错误（内部）+功能的违背（外部）

5.2. 缺陷的类型

1. 软件未达到产品说明中已标明的功能。
2. 软件出现了产品说明书中指明不会出现的错误。
3. 软件功能超出了产品说明书指明的范围。
4. 软件测试员认为软件难以理解，不易使用，运行速度慢，或者最终用户认为该软件使用效果不好。

5.3. 缺陷产生阶段

- 需求 ⟹ 编写说明书
- 设计 ⟹ 设计阶段
- 编码 ⟹ 编写代码
- 测试
- 发布

5.4. 缺陷属性

1）严重程度 ⟹ 根据对软件的破坏程度，定了5级

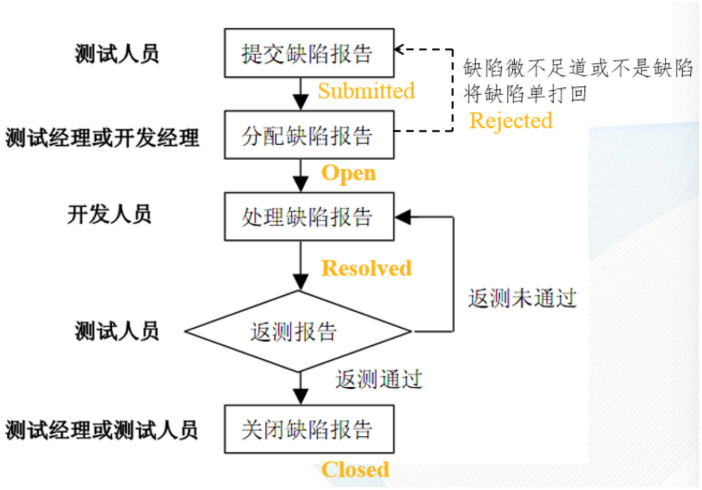
- 5 fatal ⟹ 崩溃、死机，数据丢失、功能丧失
- 4 critical ⟹ 主要功能部分丧失、次要功能全部丧失，致命的错误声明
- 3 major ⟹ 提示信息不准确等
- 2 minor ⟹ 排版、文字不准确
- 1 建议 ⟹ 质疑

2）优先级 ⟹ 开发修复缺陷的先后顺序

- 高 ⟹ 必须立即解决
- 一般 ⟹ 需要正常排队等待修复
- 低 ⟹ 在方便时纠正

正常情况下，严重程度高的优先级也高，但不是绝对的。如果有些严重程度高的缺陷，但是客户群体很窄，可以把优先级调低，规避一下该问题；有些问题严重程度很低，比如公司的logo出错这样的问题，但是要把优先级调高。

缺陷生命周期：



状态

- `New` 新建：测试人员新提交的缺陷，状态是New
- `Open` 打开：测试经理审核新建的缺陷，审核通过之后打开缺陷
- `Rejected` 拒绝：测试组长认为缺陷有问题，或者认为不是缺陷，打回
- `Fixed` 已修复：开发人员修复缺陷后，交给测试人员返测
- `Closed` 关闭：缺陷经过3-5轮回归测试没有问题后，由有经验的测试组长或者测试经理关闭

5.5. 缺陷报告

是描述软件缺陷现象和重现步骤的集合

作用

可以衡量测试人员的工作能力，体现测试价值

- 记录缺陷
- 缺陷跟踪

组成

- 基本信息：缺陷ID，被测模块，测试人，处理人，测试版本
- 属性：严重程度、优先级、状态
- 主要描述：缺陷名称、复现步骤、实际结果、截图和日志

编写准则 $\Longrightarrow$ 5C

- Correct(准确) $\Longrightarrow$ 每个组成部分的描述准确，不会引起误解;
- Clear (清晰) $\Longrightarrow$ 每个组成部分的描述清晰，易于理解;
- Concise (简洁) $\Longrightarrow$ 只包含必不可少的信息，不包括任何多余的内容;
- Complete (完整) $\Longrightarrow$ 包含复现该缺陷的完整步骤;
- Consistent (一致) $\Longrightarrow$ 按照一-致的格式书写全部缺陷报告。

编写的注意事项

- 保证重现缺陷
- 分析故障 $\Longrightarrow$ 使用最少的步骤重现缺陷
- 包含所有缺陷的必要步骤
- 方便阅读
- 一个缺陷一个报告
- 中性的语言
- 相关经验

缺陷跟踪系统

- DTS $\Longrightarrow$ Defect Tracking System
- 优点
    - 保持高效率的测试过程
    - 提高软件缺陷报告的质量
    - 实施实时管理，安全控制
    - 利用该系统还有利于项目组成员间协同工作