浅谈测试和管理

谭莉

目录

三个方面的内容：

- bug相关（重点是怎么回归bug和回归bug后的维护管理）
- 测试用例（里面会介绍到怎么需求变为测试用例）
- 版本进度跟进（里面会介绍到怎么制定版本计划和跟进）

bug相关

- bug类型
- bug生命周期
- bug回归
- bug维护和管理
- bug数据引导测试

bug类型

- 可结合自己项目定义，常见的分类：
  - 问题类型：崩溃问题、功能问题、性能问题（效率，cpu，内存）、界面问题（界面，外观，本地化）、需求问题、建议问题（改进，优化，易用性）等
  - 严重程度：高、中、低等 ，有的会区分为非常严重、比较严重、一般严重，还有的会分为致命错误，严重错误，一般错误等
  - 修改级别：立即修复A、必须修复B、计划修复C、暂不修复D

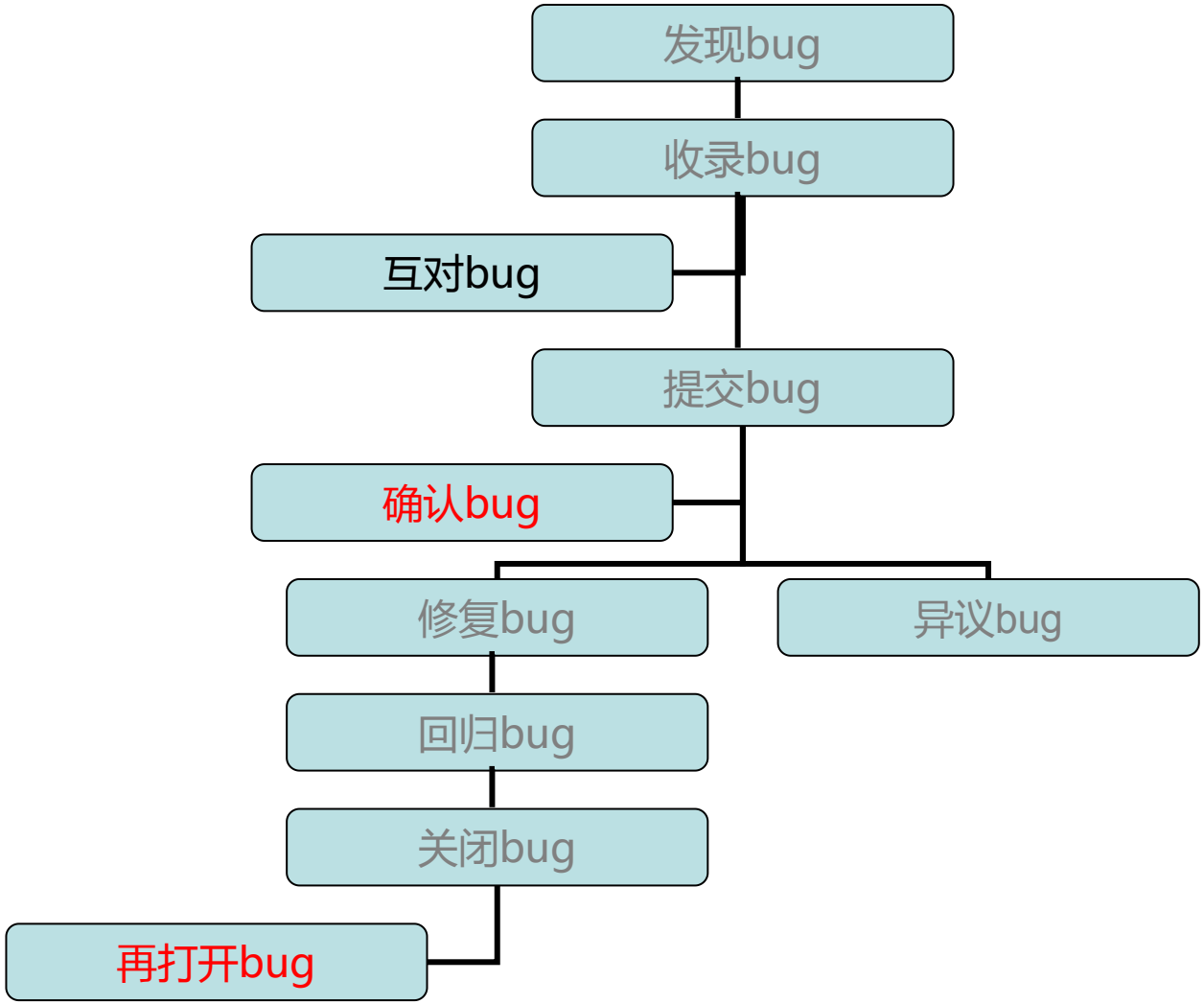
bug类型-举例

- 崩溃问题
- 功能问题：正点闹钟，到时未响等
- 性能问题：启动程序，很长时间不能相应等
- 界面问题：比如对话框大小不合适，控件不美观等
- 需求问题：希望实现某功能等
- 建议问题：建议a提示表达改为b提示，更容易用户理解等

bug严重级别-举例

- 高
  - 崩溃问题，简单操作导致的性能问题，明显界面问题，基本功能问题，版本信息问题
- 中
  - 普通但需修改的功能问题，影响用户使用感受的问题，自身产品的数据兼容和历史版本兼容问题
- 低
  - 3步以上操作且不是非常严重的功能问题，细微不易发现的界面问题，极限变态的操作导致的性能问题，和其他产品的兼容问题

bug生命周期

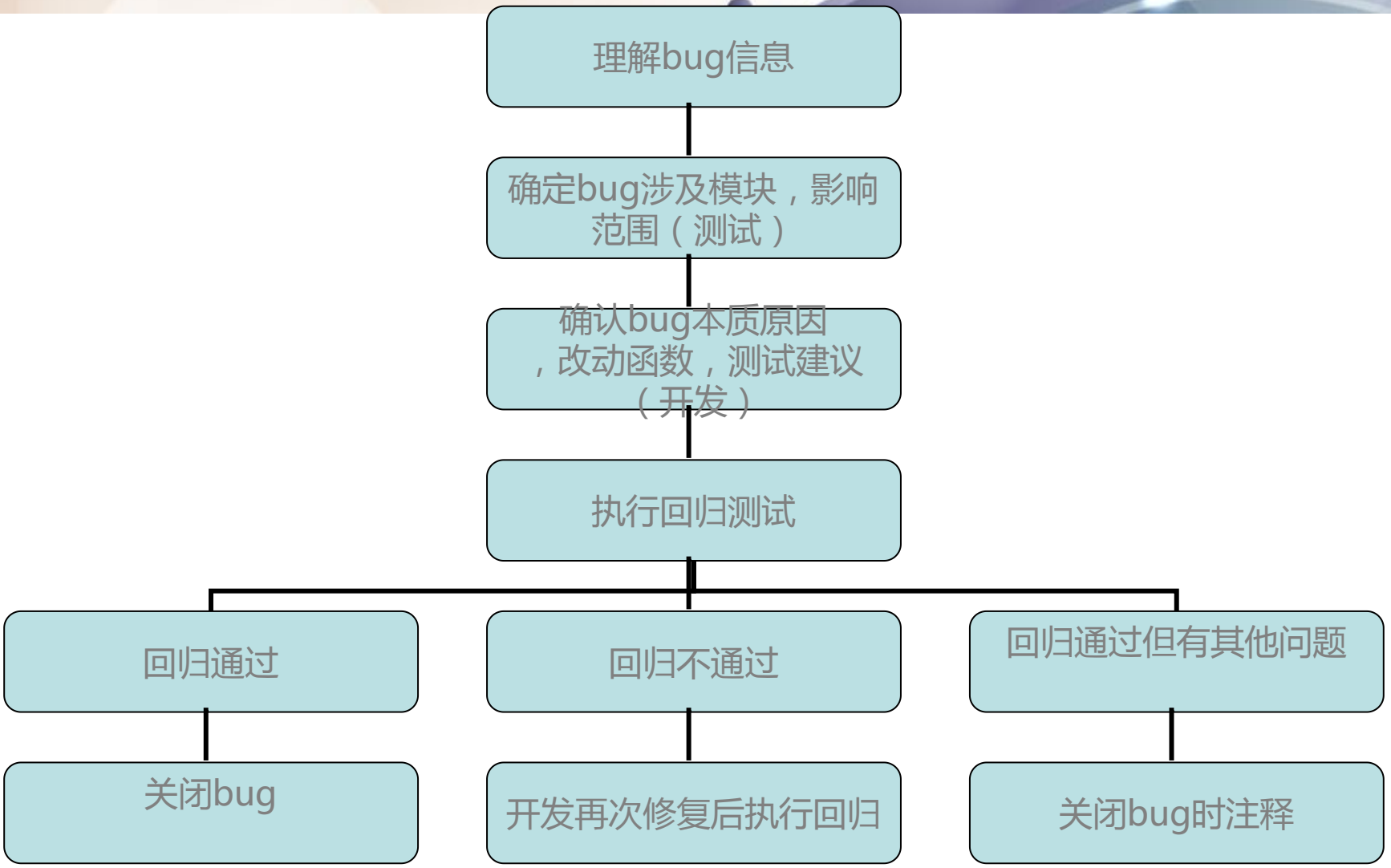


- 灰色内容是一个bug的必经周期

bug生命周期-说明

互对bug	建议正式提交bug该开发修改前的一个流程，避免太多设备的干扰，或者个人因素的干扰
提交bug	bug正式生效提交给开发修改
确认bug	比较适用于版本发布或者测试周期较紧张时期的控制
回归bug	及时回归已修复BUG，一般来说有新构建就需要回归bug
异议bug	定期对一些无效或有异议的问题进行处理，确保意见和开发一致且不放过任何一个严重或者影响用户使用的问题 常见异议问题有，不是问题，问题重复，暂缓处理，是问题但不必修复等
再打开bug	bug一个生命周期结束后，也有可能再开始第二个生命周期，尤其在产品核心重构的时候最容易出现

回归bug流程


理解bug信息
确定bug涉及模块，影响范围（测试）
确认bug本质原因，改动函数，测试建议（开发）
执行回归测试
回归通过
回归不通过
回归通过但有其他问题
关闭bug
开发再次修复后执行回归
关闭bug时注释

回归bug流程

- 执行回归测试
  - 按原bug重现步骤过一遍，保证不重现
  - 围绕bug做扩展测试
    - 对bug的开发改动和测试建议进行测试
    - 提取bug关键字，周边测试bug涉及模块，影响模块
  - 若有对应bug用例，执行bug用例，无bug用例则补充bug用例
  - 其他bug维护，见后面介绍
  - 如果建立在测试人员对功能逻辑，底层函数吗，API都掌握的基础上做效果会更好

bug维护管理

- 入库把关严
- bug必经周期：开发修复，测试回归。能否重现是bug周期中的重要环节bug现象
  - 本质和重现步骤
  - 还要关注设备环境，软件版本，相关附件
  - 支持尤其是移动产品类型很多，提供充分的环境帮助开发和回归测试的人做好基础，可以节省很多周转，确认，沟通的时间，这是值得的ps：一般来说bug最好是至少两台设备固定重现，才容易回归，特定设备或特定情况的问题要备注上统一字样并和开发达成一致

bug维护管理

- 回归更严谨
  - 一个bug并非回归通过了生命周期就停止了，以后还有可能再重现或者开始一个周期
  - bug本身维护，bug描述和本质差异较大，bug有更简洁重现方法，bug影响面和预期出入较多，多个bug相关有联系等，需要修改或补充。举手之劳，但是后期查询和管理方便会更方便
  - bug记录和信息统计，bug影响面，新功能实现，新需求改动等，很多信息是通过bug体现的
  - 根据bug本质和影响，进行补漏测试和测试文档，测试用例完善，指导下一次的测试

bug数据引导测试

- 经验之谈：
- 80-20原则，也就是二八定理，80%的缺陷聚集在20%的模块中，经常出错的模块改错后还会经常出错
  - 虽然比重不一定正确，但指向无异议。往往版本在经历了一定的测试周期后，不稳定都是一些新功能或改动较多的地方。
- 测试应当循序渐进，不要企图一次性干完，注意“欲速则不达”
  - 受测试经验和不同人测试方法所限，测试是一个循序渐进的过程，不可能一次性发现完所有的bug，那么怎么循序渐进？用什么来指导循序渐进？

bug数据引导测试

- 阶段性分析bug的修复情况和走势，可以：
  - 汇总bug的集中模块和影响模块，进行针对性测试
  - 确认bug类型的关注度和修复级别是否合理
  - 查看开发的工作粒度，开发改bug的进度是否匹配版本发布周期等等。测试应更多的参与和开发的配合协作，测试开发进度一致，才能更准确的把握产品和版本的质量和进度
  - 查看开发的工作质量，比如回归未通过的bug统计

bug数据引导测试

- 阶段性分析bug的生长情况和走势，可以：
  - 找到每次测试不彻底的症结所在，才能在之后的测试中预防和进步
  - 查看测试的工作效率和工作质量
  - 进行阶段性版本的稳定性评估

需求和测试用例

- 需求和用例的关系
- 需求细化为用例
- 合格的测试用例

需求和用例的关系

- 需求一般用来指导产品验收，更倾向于产品表现上的东西
- 测试用例一般用来执行测试，不仅要测试产品表现，还要深入到产品内部结构和功能测试
- 测试用例的设计也就是测试需求细化的过程，需求文档够细的话，也可以指导测试，但不能完全依赖于需求文档去执行测试
- 需求文档变测试用例，需要一定的测试方法和测试经验

需求细化为用例

需求：	实现上传图片功能
用例：	1.图片来源，PC？USB设备？手机？ 2.本地上传？网速，网络中断？ 3.图片格式，bmp，png，jpg，gif，emf? 4.图片大小？分辨率？是否上传后的图片有压缩？ 5.一次上传一个多个？ 6.等等

需求细化为用例

需求：	编辑框允许输入数字
用例：	正常： 1.遍历0-9之间的所有数字 2.单位数多位数，1，11等 3.整数小数 4.正数负数 非正常： 1.输入中英文，符号，验证容错处理，错误提示等（等价类划分方法？） 2.输入很大很小的值，验证极限值（边界值方法？） 3.输入为空 4.输入前置，中置，后置空格，对空格的识别（错误推测？）

合格的测试用例

- 这是不是一个合格的用例呢？
- 测试用例（Test Case）是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序路径或核实是否满足某个特定需求。
- 因为用例必须包括的是：执行条件（操作、输入值、测试点）+预期结果（检查点）+
- 一般还会包括实际结果（测试结果，问题记录）甚至bugID等
- 所以刚才的这个表中只有测试点，至少还需增加两列，预期结果和实际结果

合格的测试用例

- 有了需求细化的测试用例后，是不是就够了呢？
- 也不是，还要对产品本身的所有测试内容进行查漏补缺
- 产品本身要测哪些内容
  - 需求测试
  - 界面测试，逻辑功能测试，易用性测试，安装卸载测试，兼容性测试
  - 性能测试，稳定性测试，压力测试
  - 。。。。
- 测试用例评审和及时的维护更新

版本进度跟进

- 制定版本计划
- 版本进度可控

制定版本计划

- 制定版本测试计划
- 一个好的测试计划才能够指导如何进行有效地测试，和确定如何能够放心地结束测试。
- 五W一H原则
  - why，为什么做　　what，要做什么
  - who，投入人力　　when，测试周期
  - where，相关资料
  - how，怎么测试

制定版本计划

- why，明确版本目的
  - 用户群体，了解用户关注或使用率高的功能，用户使用环境等等
  - 发布时间，预算人力时间和工作量，保证版本进度可控
  - 修复问题，当前版本必须解决的问题
  - 发布渠道，预装？网络下载安装？升级？
- what，针对版本目的确认要进行的工作子项
  - 确认重要bug全部修复
  - 用户关注问题是否解决
  - 版本推出的更新信息？
  - 版本测试的内容有哪些？

制定版本计划

- how，每一项任务要怎么去具体的执行
  - 测试的方法，测试的粒度
  - 任务的优先级
  - 可执行的测试计划，可参考的测试文档
  - 可依靠的自动化工具，可节省的手工测试量等
- who，每项任务安排人心里要有轻重主次之分，并给每一项任务找最合适的人来做
  - 工作效率？工作质量？工作态度？细心程度？
  - 明确且合理的分工，或者共同协作时的主导者

制定版本计划

- when，对当前人力时间和工作量做个预算，保证在规定的周期内完成版本测试和发布
  - 时间充裕的情况下，预RC或者过过基础功能，系统测试
  - 时间不够充裕的情况下，如何压缩任务或协调安排
- where，相关资料的路径

版本进度跟进

- 发布前：计划OK，把控版本的进度，做好RC测试前准备
  - 要合主干的分支尽早合
  - 要带的新功能尽早稳定
  - 严重bug尽早修复
  - 至少在版本准备RC测试前几天版本已经稳定或趋向于稳定了
  - RC测试，执行版本测试计划
- 发布后：不能发布了就完事了
  - 对前一次出版本过程中的一些问题进行总结改进
  - 积极处理相应用户反馈，多倾听用户的声音，要黏住用户，每一次发布的版本出去之后让用户看到自己期望的东西，甚至有更高的期望这个是很重要的

THANKS

Q&A：sweet.alicetl@gmail.com