|  |  |  |  |
| --- | --- | --- | --- |
| **客户端代码管理及出包规范文档历史** | | | |
| **文档版本** | **修改时间** | **修改内容描述** | **修订人** |
| V1.0 | 2018/08/18 | 管理规范初始草稿 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**一 版本代码控制**

1. 正常版本迭代



由上图可知：

1. 所有的提测，都从release分支上出包
2. 开发和修改BUG都在feature分支上进行
3. openbeta分支保留了当前的公测版本
4. master分支保留了当前的线上版本
5. 合并到openbeta和 master分支的时候，需要打标签，标签格式：

**版本号\_code号­\_年月日时分**

1. 线上BUG修复



线上BUG的修复，在hotfix分支上进行。修复完毕，需要将修改合并到openbeta，feature和release分支上。

临时的紧急需求，处理流程同线上BUG修复一样。

1. 已经提测的某个需求临时撤下

有时会遇到这种情况，两个需求feature1 feature2同时开发和提测，但是feature2的测试始终不通过，为了不影响发布，决定把feature2从本次发布计划中撤下，流程如下：



**二 不同环境的出包处理**

项目一共有四个环境（dev/sgage/pre/生产），为了避免针对不同的环境重复出包，客户端需要设计出功能菜单，让测试人员能够自主选择环境。

从release分支上出包的时候，出**默认是生产环境**的包。

**三 分支权限管理**

采用Gitlab的权限管理机制来进行分支权限管理。

角色：

客户端总监 owner

三个开发组长 master

开发人员 developer

将maseter openbeta release 分支设置成**保护分支**，开发人员只有通过merge request的方式，请开发组长审查后，才能提交代码到这三个分支上。

**四 版本号**

版本号的格式如下：



**五 出包的流程**

**手动出包流程（Jenkins工具未完善之前）**

1 三个客户端各自有一台独立的出包服务器，所有提测包必须从出包服务器上出。

2 出包前，服务器上的代码删除干净，编译环境中的临时文件、中间文件全部删除干净，从 release分支拉取新代码，不可依赖编译环境的clean命令。

3开发人员手动编译打包，将提测包上传至PMS系统，同时发提测邮件。

**Jenkins出包的流程：**

1 开发人员输入出包描述，版本号， Jenkins自动从Gitlab上拉取代码，自动编译打包，自动将打好的提测包放置在公司内部服务器上，将地址链接通过邮件发送给开发人员，开发人员发提测邮件。

2 Jenkins需要做好限制，只有release分支可以出包。

3 Jenkins设置每次构建前删除本地workspace。

**Jenkins每日构建**：

1 Jenkins每天晚上自动从feature分支上构建包，并运行自动测试脚本，将相应的出错信息以邮件的形式发送给开发人员。

2 自动构建的workspace跟正式出包的workspace要隔离。

**五 提测邮件格式**

提测邮件的格式

标题：【提测】Android 学生端\_v3.08.01.111 老师端\_v3.08.01.211 通用环境

邮件内容：

客户端 Android学生端\_v3.08.01.111 （CODE 30801）

老师端\_v3.08.01.211 （CODE 30801）

已修复bug:

1 bug编号 bug标题

2 bug编号 bug标题

未修复bug（注：只列出上个提测版本提出的，而本次未修复的bug）

1 bug编号 bug标题 未修复原因

2 bug编号 bug标题 未修复原因

版本重点测试内容：

提测包地址：

邮件接收人:测试主管

邮件抄送人：测试人员，客户端开发组长，开发总监，项目经理，技术VP，产品经理，产品VP，本次提测功能的共同开发人员

注意：环境统一用以下名词：通用环境，dev环境，stage环境， pre环境 ，生产环境。

**六 代码review**

代码比较工具，采用beyond compare；

当feature分支要合并到release 分支上的时候，由于release分支已经被设为保护分支，所以开发人员需要通过Gitlab的 merge request向开发组长提交合并申请，同时@一个同级的组员，开发组长和同级的组员要对提交的代码进行一次code review，通过后才能merge代码。

代码review的checklist:

1 是否符合技术文档的设计？

2 版本号和code 是否正确？

3 配置文件是否正确（如默认环境是否是线上环境）？

4 是否符合代码规范？

5 对数组的访问，是否考虑了边界的情况？

6 对循环的访问，是否考虑了边界的情况？

7 对输入的参数，是否考虑了值为空的情况？

8 对接口的访问，是否考虑了接口返回延时较长的情况？

9 是否存在代码修改了之后，相应注释没有修改的情况？

10 是否存在重复调用，影响效率的情况？