软件开发流程及规范

版本	描述	修改人	日期
V1.0	最初版本	徐跃福	
V1.1	添加测试部对 Review 流程的监督 强调研发人员的自测	徐跃福	
V1.2	代码管理工具切换到 Perforce	徐跃福	2011-8-24
V1.3	Branch 后加入 Bug 更详细的描述，加入了各个阶段的输出文档	徐跃福	2011-11-18

目的

本文旨在规范软件开发流程，规范代码管理工具和 Bug 管理工具的使用，以提高代码的可追踪性，减少 Bug 的出现。

有关 urtracker 和 Perforce 使用的几点规则

1) 所有 code 的改动，均需关联到 urtracker 事务，任何人都可以创建 urtracker 事务。
2) 代码提交时的 description 中，需要关联 urtracker ID，并简要描述改动。
   例如:

V2-378, To improve performance of uart data handling.

   如果是从其它版本 merge 过来的 code，提交时把原 Changelist 号也写上。

V2-378, MFM 92,93, To improve performance of uart data handling.

   用 MFXX 的形式去表示从哪个版本上 merge 过来的，比如：
   MFM = Merge From Main
   MF1008.2 = Merge From 1008.2 branch
3) 在开发阶段（尚未建立 Branch 时），代码提交时，需要在 urtracker 中详细描述问题产生的场景，原因，解决方法，需要注意的事项等，对于 Bug，还要指明以后如何避免。同时把 Perforce 提交的 changelist 贴于 urtracker 中。问题的描述一定要清晰，必要的时候可以贴一些图形，达到看 urtracker 就知道解决什么问题，如何解决问题的目的。
   例如:

场景：基站连接三一设备时，程序产生崩溃 Branch: branch-release-sany 原因：memcpy 时，size 错误 解决方法: 用 sizeof(struct)修改 memcpy 的第三个参数 注意事项: 无 Changelist：92

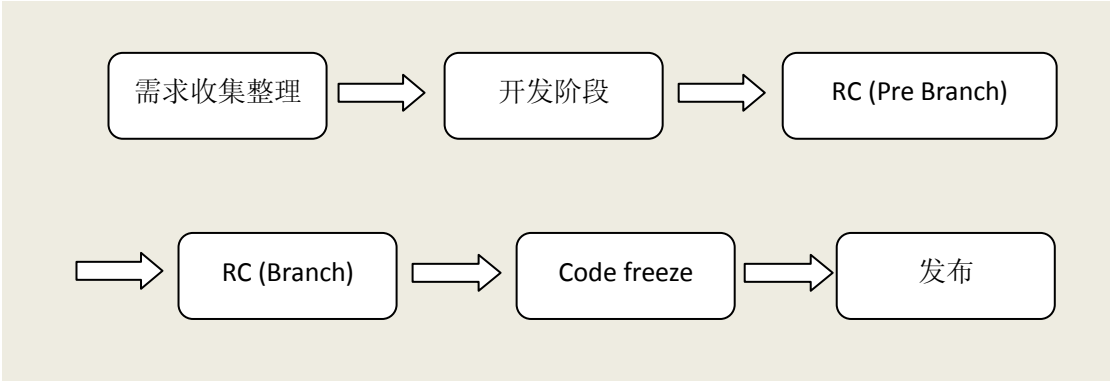
4) 在 release 阶段(新建 branch 后)，所有的改动先在 main 版本上进行，并 merge 到 release 版本中。这样有利于保证主干版本的正确性，一致性。同样在 Perforce checkin 时，标注 urtrackerID 及简要描述，在 urtrackerID 评论中需要同时标注出两个版本上的信息，并给出更详尽的描述。对于 Bug，还需注明引入问题的 cvs/P4 的版本、ChangelistID，urtrackerID，引入人员，Review 人员等详细信息，模板如下：

问题:在井下无法判断掘进机设备是否与基站互通 产生原因：描述问题产生原因(Bug 时需要) 解决方法:用基站相应口的指示灯进行表示，如果连接正常，灯亮 ，否则，灯灭。 main:

```
Changelist 92, 93


branch-release-sany:
Changelist  94
引入问题的 changelistID: 哪个 cvs/p4 的 changelist 引入的
引入问题的 urtrackerID: 以上 changelist 是针对哪个 urtrackerID 做出的修改
引入问题的时间: (RC 版本和具体的时间点)
引入问题的人员:
引入问题时 urtracker 对应的 Review 人员
总结:(避免措施)
```

开发过程及注意事项：



a) 需求整理收集:
release 开始前，由研发，测试，产品部门共同召开会议，确定本次 release 需要完成的事务。所有需要做的事务需要在 urtracker 上有记录。
输出：产品部根据会议内容，输出 PRD
研发部输出资源安排及项目计划

b) 开发阶段：开发人员主导，分为设计阶段和编码阶段
设计阶段：研发部根据会议内容和产品部提供的 PRD，形成设计文档。
编码阶段：编码实现功能
输出：研发部输出设计文档和代码，测试部输出测试用例文档

c) RC (Pre Branch)
功能性需求完成后，测试人员可以介入，开始 RC 测试。一般来讲，1~3 个 RC 后可以开始建立 branch。在 RC 测试开始，建立 Branch 之前的阶段，称之为 Pre Branch 阶段。此阶段应尽可能的发现/修复 Bug，以减少建立 Branch 之后，对代码的维护成本。该阶段研发人员可独自修复 Bug 和进行功能改进。

d) RC (Branch)
建立 Branch 之后，项目负责人需要为每一位开发人员指定一个 Partner，开发人员 checkin 的 code 必须经过 Partner 的 Review。Review 的内容包括编码风格，Bug 是否完全修复，会否引入新的 Bug 等。Review 人员需要在 urtracker 上提交 Review 记

录，对于经过 Review 后还产生的问题，开发者和 Review 人员共同承担责任。测试部在此阶段开始填写 Bug 统计表

e) Code freeze
建立 Branch 并且经过几个 RC 之后，Bug 基本不存在，进入 Code freeze 阶段。此时只有非常严重的 Bug 才会去修复。Bug 的严重等级由部门经理决定，code 改动必须经过项目负责人或部门经理的 Review 方可 checkin。由此产生的问题由项目负责人和部门经理承担。

f) 发布
产品发布，发布之后有可能还会出现一些缺陷，此时可以计划推出 hot fix 版本。Hot fix 版本流程与上同。

g) 测试部负责整个流程的监督，特别要针对 Branch 之后的 Review 流程和 Code freeze 之后对代码的控制，并将监督结果反馈至部门经理。

h) PostRC 阶段，测试部负责填写 Bug 统计表，该 Bug 统计表反映出 Bug 的原因及责任人等情况，将作为评估个人工作情况的依据。

i) 研发部同事对每个问题的解决要进行自测，自测通过后方可提交至下一流程执行人。

有关重要接口部分的改动规则

重要接口部分包括 ARM/ZB 协议，数据库表格等。此部分的修改应限定权限，只有项目负责人可以对此进行修改，并将修改通过 urtracker 和邮件的方式通知相关人员。

2011-11-18

此规范自发布之日起立即执行，执行情况将列入月度/年度考核中。