# 匡信科技开发流程规范

# 目录

# 相关支持环境

在日常开发的流程管理中，需要多个支持环境协作管理流程中的各个节点和相关物件。具体支持环境运行和使用规范详见《匡信科技日常开发支持环境》。

## 代码仓库

负责管理所有源代码和原始资源的存储、共享、版本管理、分支管理、访问权限控制。当前使用 subversion 作为代码仓库，可以通过命令行、客户端、集成开发环境（IDE）、浏览器访问。

## 产品仓库

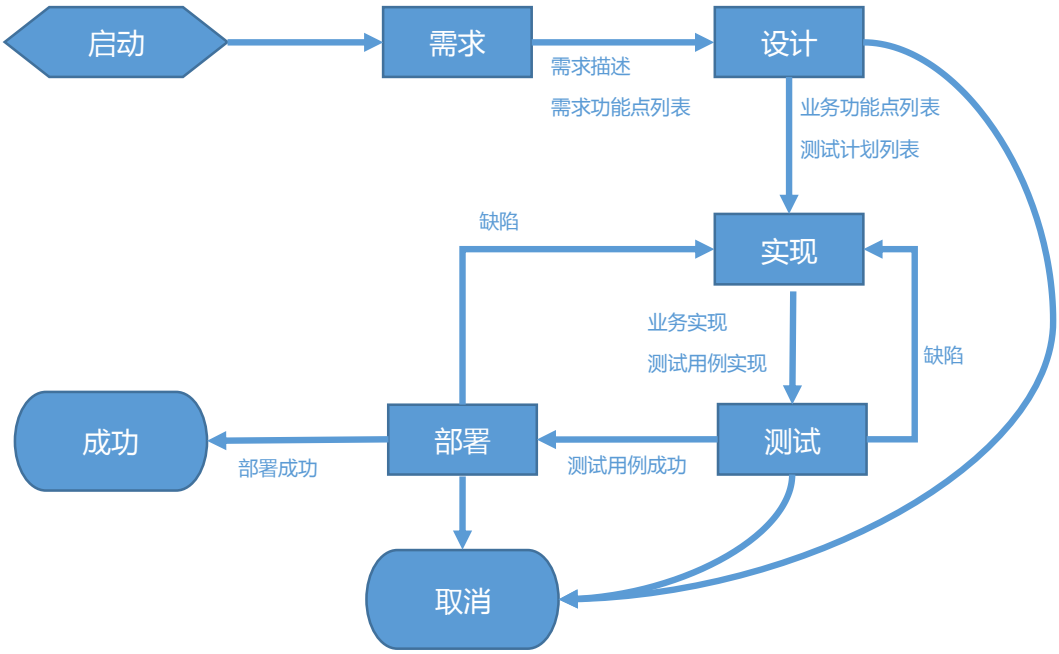
负责管理所有编译构建产出物的存储、共享、版本管理、权限控制。当前使用基于 Sonatype Nexus 的 Maven 仓库，可以通过命令行、集成开发环境（IDE）、浏览器访问。

## 事务跟踪系统

负责管理完整开发生命中期中所有事务流程、节点、时间、成本。

# 流程综述

一个完整的开发流程，覆盖产品在公司技术团队控制中一次版本迭代的完整生命周期，由多个节点以层次形式组成。各个节点以确定的方式发生和执行，以固定的前承/后继关系相互链接，组成一次完整、可控、灵活的开发过程。一个完整的开发流程以某一确定产品的一个版本开发启动为起点，以该产品版本在生产环境上的成功和部署使用为成功终点，以该产品版本取消为失败终点，包括需求、设计、实现、测试、部署等基本节点。节点间流程关系如下图所示：



# 启动阶段

开发流程启动阶段，由产品决策管理者决定启动版本开发，即启动一个项目开发流程。需要做一下准备工作：

1. 在 JIRA 中创建项目（project）和初始版本（version），或创建该项目一个新的版本（如果项目已经存在）。
2. 若尚未创建，在项目下创建“项目管理”、“测试”、“缺陷”组件（component）。
   a) 项目管理组件
      i. “需求描述”任务
      ii. “需求功能点分析”任务
      iii. “业务功能点设计”任务
      iv. “测试用例设计”任务
      v. “部署”任务
   b) 测试组件（每轮测试一个测试组件）
      i. “测试用例实现”任务
      ii. 根据测试用例列表，多个测试用例任务。
   c) 缺陷（多轮测试统一使用一个缺陷组件）
      i. 根据测试用例执行结果，多个缺陷任务。
   d) 其他项目开发组件（根据项目设计划分多个开发组件）
3. 创建代码仓库分支

   项目根目录（例如：http://svn.hzcominfo.com/xxx/），其中包含子目录：

   - 文档分支（/doc）
   - 主开发分支（/trunk）
     - 组件分支（/trunk/yyyy）
     - 测试用例分支（/trunk/testcase）
   - 并行分支（/branches）
   - 标签分支（/tags）

   代码分支建立操作：

   - 初始版本：创建主开发分支（trunk）
   - 版本替代：将 trunk 分支备份（tag 操作）到 SVN 的 tags 目录作为备份，然后修改 trunk 中的版本号，提交后 trunk 作为新版本的主开发分支。
   - 并行版本 :将 trunk 分支( branch 操作 )到 SVN 的 branches 目录 ,并升级该 branch 的版本号 ,该 branch 版本作为并行版本的主开发分支。

# 开发流程

## 需求节点

## 需求采集

由项目经理负责完成。
负责向客户或其他需求提供方获取、整理、具体化需求，形成需求概念描述。
产出：**《*XXX* 项目需求描述》**，由需求提供方确认。
完成以上工作后，项目经理将 JIRA 项目“需求描述”任务为“Resolved”。

## 需求分析

由项目经理负责完成。
根据**《*XXX* 项目需求描述》**和当前产品架构/选型/预研结论，将需求概念性描述具体化为需求功能点列表，从而定义系统的范围（需要完成和不需要完成）和组件划分。
需求功能点要求：

- 完整：需求功能点应包含且仅包含一个完整需求流程的所有业务实现和用户交互。
- 耦合：需求功能点实现不应依赖其他业务功能点的实现。
- 输入输出：需求功能点自身输入/输出数据格式明确，稳定。

产出：**《*XXX* 项目需求功能点列表》**，由需求提供方确认，XXX 对应 JIRA 项目名称。
**《*XXX* 项目需求功能点列表》**中应包含两个维度的需求功能点分类：

1. 组件：项目的物理分类维度，每个组件应为一个物理部署单位。对应 JIRA 项目的组件划分。
2. 模块：需求功能点的逻辑分类维度，每个模块标识一个完整的需求业务，对协作完成同一模块的开发人员交流、协作有较高要求。

组件和模块以正交关系组织一个项目内的功能点（JIRA 中的 ISSUE）。
需求功能点是基于需求逻辑的，所以需求功能点应属于模块，可以跨组件（例如：一个查询需求可以跨前端组件和后端组件，但是应该属于某一指定模块）。
组件和模块规划应保证持续性，即：保证在将来可以预见的项目版本迭代中，项目组件/模块划分和交互、依赖架构稳定。
亦即：

- 对于新的项目应保证组件/模块规划稳定性
- 对于原有项目新版本应使用原有版本的组件/模块规划
- 对于需求的扩展和变更，应增加新的组件/模块以避免原组件/模块定义变更

项目经理根据**《*XXX* 项目需求功能点列表》**在 JIRA 中录入项目的组件规划（compoonent）。

**《*XXX* 项目需求功能点列表》**应包含其对应的开发分支信息（可以在文档标题命名中携带开发分支版本号）。项目经理根据**《*XXX* 项目需求功能点列表》**定义代码仓库分支。
在 trunk 分支中建立尚未建立的组件目录，作为开发项目（project in ide）目录。
完成以上工作后，项目经理标记 JIRA 项目“需求功能点”任务为“Resolved”，并标记“需求”任务为“Resolved”。

# 设计节点

## 总体设计

由项目经理完成。
根据**《*XXX* 项目需求功能点列表》**，定义业务功能点。
业务功能点要求：

- 完整：需求功能点应包含且仅包含一个完整业务功能流程的所有业务实现和用户交互。
- 交互：一个业务功能点原则上仅应由一次与用户（业务功能点调用者）的交互。
- 耦合：业务功能点实现不应依赖其他业务功能点的实现。
- 输入输出：业务功能点输入输出数据结构不应依赖其他业务功能点实现。
- 成本：一个业务功能点应由单个开发人员在一周工时内完成。如果超过该限制，应拆分业务功能点。因此业务功能点可以有分级。

一个业务功能点在**运行时**应仅属于一个模块且仅属于一个组件。因此业务功能点运行时可以确定其在组件/模块二维组织中的一个坐标。这里的**运行时**是指，在开发/部署时，多个组件或多个模块可以共享一个业务功能点。
产出：

- **《*XXX* 项目 *YYY* 组件业务功能点列表》**，对应 JIRA 项目某一组件。
  项目经理根据**《*XXX* 项目 *YYY* 组件业务功能点列表》**定义 JIRA 实现任务，作为 JIRA 项目的“实现”任务的子任务录入 JIRA 系统，并指定该子任务所属组件（component）。
- **《*XXX* 项目数据模型设计》**，对应持久层（关系数据库/非关系数据库）中的表结构/文档结构。应涵盖所有项目可能持久化使用的数据结构。供 DBA 完成开发/测试/生产环境数据持久层构建。
  **《*XXX* 项目数据模型设计》**可以携带持久层生成脚本和持久测试数据生成脚本。

完成以上工作后，项目经理标记 JIRA 项目“业务功能点”任务为“Resolved”。

## 业务设计

由开发人员完成，**可选**。
根据**《*XXX* 项目 *YYY* 组件业务功能点列表》**，定义每个业务功能点的详细设计。可以包含文字描述、流程图、数据结构、表结构或其他自定义元素。可以忽略（对于简单业务功能）。
产出：如果未被忽略，产出应该是一个简短文档**《XXX-YYY-1PD》**（One Page Design）。

# 测试用例设计

由测试人员完成。
根据**《*XXX* 项目 *YYY* 组件业务功能点列表》**（多分，覆盖项目各组件），定义测试用例，包括测试输入、测试逻辑、测试期望输出、测试错误定义等。
每一个业务功能点根据需要覆盖的测试场景对应多个测试用例。
每一个需求功能点除其业务功能点的测试用例集合外，还应该包含针对完整需求流程的多业务功能点联合测试用例。
产出：

- **《*XXX* 项目测试用例列表》**，对应 JIRA 中“测试用例设计”任务。
  每个测试用例对应 JIRA 项目中“测试”组件内的一个任务，由测试人员根据**《*XXX* 项目测试用例列表》**录入 JIRA
  **《*XXX* 项目测试用例列表》**对应 SVN 当前主开发分支下的 testcase 子目录

完成以上工作后，测试人员标记 JIRA“测试用例设计”任务为“Resolved”。

# 实现节点

# 业务实现

由开发人员完成。
开发人员检出代码后，每日开发变更以完整变更形式提交。变更提交规范参见**《匡信科技日常开发支持环境》**中代码仓库规范部分。
开发人员完成一项业务功能点开发实现工作，应标记 JIRA 系统中对应业务功能点实现任务为“Resolved”，并附加该项任务相关所有 SVN 提交变更编号（REV No.）。
经代码检视，可以由代码检视者标记为“Close”。

# 测试用例实现

根据《XXX 测试用例列表》完成测试用例实现，提交到代码仓库的 trunk/testcase 目录。
完成测试用例实现后，测试人员标记 JIRA 项目“测试用例实现”任务为“Resolved”。
经代码检视，可以由代码检视者标记为“Close”。

# 测试流程

测试流程分为多个阶段：

## 开发测试

由开发人员独自在开发环境中同步进行开发测试，针对当前开发中的功能点。
当前开发/测试功能点在开发人员开发环境运行，持久数据使用联调数据库。
根据测试用例列表自定义功能内部测试项（无需涵盖全部测试用例）。
完成开发测试可以进入联调测试。

## 联调测试

由开发人员在开发过程中同步联调测试。主要测试当前开发功能点与其他功能点的交互。
当前开发功能点在开发人员开发环境中运行；持久数据使用联调数据库；所交互的相关功能点在联调环境运行。
根据测试用例列表自定义业务功能点联调测试项（无需涵盖全部测试用例）。
完成联调测试可以提交代码。

## 集成测试

由测试人员在开发实现阶段完成后运行。
集成测试应依据当前主分支代码执行标准编译构建流程。
标准编译构建流程如下：

1．检出主分支代码。
2．完成编译构建获得构建产出物。
3．发布到产品仓库（Maven 仓库）。
4．从产品仓库获得构建产出物，发布到集成测试环境。

在集成测试环境/集成测试数据库中运行，覆盖所有测试用例。

1．集成测试产生 BUG：
   a） 测试人员在 JIRA 项目的缺陷组件中建立“BUG”任务。
   b） 项目经理负责分派“BUG”任务给正确的开发人员进行修复。
   c） 开发人员修复完成后，提交修复代码，修改“BUG”任务状态为已修复。
   修改 JIRA 中 BUG 状态时填写修复代码提交变更编号。
   d） 测试人员重新运行该测试用例，验证已修复的 BUG 和可能的新出现 BUG。
   若验证通过，修改“BUG”任务状态为关闭。
2．所有 BUG 关闭后，重复完整测集成测试流程（发布产品仓库）。
3．完成集成测试后可以发布产品正式版本，可以进入验收测试。

a） 检出主分支代码（应保证与成功集成测试基准 REV 一致）。
b） 去除版本号中的“SNAPSHOT”标签。
c） 执行标准化构建，构建产出发布产品仓库，成为 RELEASE 版本。
d） 测试人员提供《XXX 项目集成测试报告》，主要包括测试统计数据如 BUG 规模、BUG 分布、修复时间分布、重现分布等。

# 验收测试

由用户验收人员在完成集成测试阶段后进行。
验收测试应依据此前成功完成的集成测试构建产出物。
1. 从产品仓库获得集成测试成功产出物（根据版本号）。
2. 发布验收环境。
在验收环境/验收数据库中运行。验收环境/验收数据库应完全等同于生产环境/生产数据库。
覆盖所有测试用例。
1. 验收测试产生 BUG：
a） 测试人员在 JIRA 建立“BUG”任务。
b） 项目经理建立代码仓库的 BUGFIX 分支（在 branches 中）。
c） 项目经理负责分派“BUG”任务给正确的开发人员进行修复。
d） 开发人员在 BUGFIX 分支上进行修复。
i. 完成后，提交修复代码，修改“BUG”任务状态为已修复。
ii. 每次 BUG 修复均应仅有一次提交。
iii. 修改 JIRA 中 BUG 状态时填写修复代码提交变更编号。
e） 验收 BUG 修正后：
i. 测试人员在 BUGFIX 分支上开始一个新的集成测试流程。
ii. 若集成验证通过，修改“验收 BUG”任务状态为关闭。
iii. 所有验收 BUG 关闭后
1. 项目经理合并 BUGFIX 分支回到主分支。
2. 清理 BUGFIX 分支。
3. 重新构建，重新发布产品仓库。
a） 覆盖产品原 RELEASE 版本或升级版本（增加版本后缀）。
iv. 测试人员在主分支上开始一个新的验收测试流程。
2. 完成验收测试后可以上线（发布生产环境，开始生产运维）。
a） 测试人员提供《XXX 项目验收测试报告》，主要包括测试统计数据如 BUG 规模、BUG 分布、修复时间分布、重现分布等。

# 上线部署流程

部署生产环境应依据此前成功完成的集成测试构建产出物。

1. 从产品仓库获得集成测试成功产出物（根据版本号）。
2. 发布验生产环境。

成功部署后：

1. 备份原主开发分支到 SVN 仓库的 tags 目录。
2. 升级主分支中主版本号并增加 SNAPSHOT，作为下个版本的主开发分支。
3. 项目经理关闭 JIRA 中“部署”任务。
4. 进入下一个版本迭代生命周期。

# 完成阶段

完成项目总结，由整理《项目总结报告》，主要包括：

- 项目概况
- 项目边界（已完成内容和未完成内容）
- 项目过程描述（成本分配如甘特图、测试缺陷报告等）
- 项目结果（成功结果如运行 Benchmark，失败结果如取消原因）

# 流程实例

以“云平台监控”项目为例

## 文档

1. 《云平台监控需求描述》
2. 《云平台监控需求功能点列表》
   其中定义了三个需求功能点。
   - “CPU 监控”
   - “业务接口存活监控”
   - “业务接口流量监控”
3. 《云平台监控交互前端业务功能点列表》
   其中针对“CPU 监控”需求功能点定义了四个业务功能点。
   - “CPU 数据查询结果图形化展现”
4. 《云平台监控业务服务业务功能点列表》
   其中针对“CPU 监控”需求功能点定义了四个业务功能点。
   - “CPU 数据采集”
   - “CPU 数据保存”
   - “CPU 数据查询”
5. 《CPU 监控测试用例列表》
   其中针对“CPU 数据采集”功能点定义了三个测试用例。
   - “CPU 数据采集 - 正常运行 CPU 数据采集”
   - “CPU 数据采集 - 不存在 CPU 数据采集”
   - “CPU 数据采集 - 超负荷挂起 CPU 数据采集”
6. 《云平台监控集成测试报告》
7. 《云平台监控验收测试报告》
8. 《云平台监控项目总结报告》

## JIRA 组织

JIRA 建立项目“云平台监控”项目。

# 组织结构

- 组件

    - 项目管理
    - 云平台监控交互前端
    - 云平台监控业务服务
    - 集成测试
    - 验收测试
    - 缺陷
- 版本
    - 1.0.0

# 任务定义

| 组件 | 项目管理 | 交互前端 | 业务服务 | 集成测试 | 验收测试 | 缺陷 |
|---|---|---|---|---|---|---|
| 任务 | 需求描述 | CPU 数据查询结果图形化展现 | CPU 数据采集 | 测试用例实现 | 测试用例实现 | |
| | 需求功能点分析 | CPU 数据查询结果表格展现 | CPU 数据保存 | CPU 数据采集 - 正常运行 CPU 数据采集 | CPU 数据采集 - 正常运行 CPU 数据采集 | |
| | 业务功能点设计 | | CPU 数据查询 | CPU 数据采集 - 不存在 CPU 数据采集 | CPU 数据采集 - 不存在 CPU 数据采集 | |
| | 测试用例设计 | | | CPU 数据采集 - 超负荷挂起 CPU 数据采集 | CPU 数据采集 - 超负荷挂起 CPU 数据采集 | |
| | 部署 | | | | | |

# 代码仓库

- http://svn.hzcominfo.com/monitor/trunk/monitor-web
- http://svn.hzcominfo.com/monitor/trunk/monitor-service
- http://svn.hzcominfo.com/monitor/trunk/testcase
- http://svn.hzcominfo.com/monitor/branches
- http://svn.hzcominfo.com/monitor/tags