# 智网源代码管理与发布规范

文档历史

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 作者 | 描述说明 |
| 2019.4.8 | V0.01 | 祁宇 | 初始版本 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 文档最后一次修改时间 | | 2019-04-08 14:22:00 | |

## 目的

本规范详细规定源代码管理的对象、存储目录、分支、权限、维护和发布规范等内容，使软件项目源码管理流程化并规范化，确保在系统开发和实施过程中项目的完整性和一致性。

## 适用范围

所有系统开发及实施项目的软件项目都应进行版本管理。项目中所有正式文档和代码都应纳入配置库（可使用工具建立配置库，本文所述使用的是gitea）进行版本管理。

## 职责

配置库管理员：负责配置库的日常维护和管理；监督开发及测试部门及时提交版本管理对象（即配置项）。 此岗位可由开发或测试人员兼任。

## 内容

### 源码管理对象

包括但不限于：

*  项目总体计划
*  可行性研究报告
*  开发计划
*  需求说明书
*  需求设计原型
*  设计说明书
*  系统开发变更申请单
*  系统管理手册
*  用户操作手册
*  培训计划
*  培训记录
*  源程序
*  支持系统运行的配置文件
*  存储过程脚本
*  测试计划
*  测试用例
*  测试脚本
*  测试报告
*  上线计划
*  上线申请
*  版本维护日志

### 源码目录结构

每个项目在配置库中应拥有唯一的项目名称。配置库目录结构与项目内部的目录结构建议按下列格式创建。

配置库目录结构规划：

项目名称

include

src

third

doc

example

test

performance

源码目录结构说明：

include: 放项目的头文件，如.h或.hpp文件

src：放项目的实现文件，如.cpp文件

third：放第三方库

doc：放项目的文档，比如使用文档，测试报告等

example：放项目的示例代码

test: 放项目的单元测试和测试用例

performance：放项目性能测试的代码

### 源码管理

项目解决方案建立三个分支：主分支、开发分支和发布分支。

* 主分支：所有分支版本的基础版本，主版本的开发分支，开发部门开发使用
* 开发分支：主版本的分支版本，共开发部门使用，开发工程师如果以主版本为基准，进行软件项目开发，要先将主分支目录下的代码分支到branches目录的一个子目录，在那里对代码进行开发。多个主版本的分版本可通过在branches顶级目录创建多个分支目录来区分。
* 发布分支：测试和发布专用分支，该分支代码不允许任何形式的修改。每个经过测试后的不同版本的代码做快照放到此分支文件夹下。

### 权限管理

应对配置库的访问权限进行管理，确保软件系统的完整性和安全性。按如下方式进行管理。

#### 开发工程师

仅拥有自己所属项目的add file、delete file、check out、check in权限，无目录创建和删除权限。开发工程师若想创建目录，需向配置库管理员申请。

#### 测试工程师

拥有每个项目的测试分支的add file、delete file、check out、check in权限，无目录创建和删除权限，对于其他分支只有只读权限。

#### 配置库管理员

拥有全部权限，但增删项目和增删目录需要有项目负责人批准。

#### 其他人员

若需要配置库访问权限，需经CTO批准，由配置库管理员分配权限。

### 版本管理

应对软件系统的版本进行管理，确保版本的准确性和可追溯性。建议按如下方式进行管理。

#### 版本维护

软件工程各阶段产生的各种文档和代码，应及时并统一上载到配置库由配置库管理员统一管理。对于要修改的配置项，应从配置库中clone后修改，修改完毕后及时commit，并填写修改的原因和内容。配置项的历史版本应保存在配置库中。

#### 分支迁移

从开发分支到测试分支的迁移，由开发工程师操作。迁移的时机有：

1. 当开发负责人提交测试申请时；

2. 开发过程中进行测试，修改好一个或多个bug，需要测试工程师验证时。

从测试分支到发布分支的迁移，由配置库管理员操作。迁移的时机有：

1.当开发组提交上线申请时。

对于每个项目从测试分支到发布分支的迁移，配置库管理员要建立分支迁移日志，并详细记录。

#### 版本升级

软件系统迁移到发布分支后，生成新的版本。

每个系统新的版本不仅以分支形式存在于配置库中，并且要以独立压缩包形式备份。

版本的命名规则为，version N1.N2.N3[.N4][\_][T/R5]\_YYYYMMDD

1. N1是系统编号。当项目整体重新设计时，N1加1，基数为1

2. N2是模块编号。当模块重新设计时，N2加1，基数为0

3. N3是功能编号。当项目增加某一功能，或某一功能需要修改时，N3加1，基数为0

4. N4是BUG编号。当项目的BUG被修复时，N4加1，基数为0

5. T/R5中的T/R分别对应Test/Release。当项目发布时为R，当项目提交测试时为T，T/R5数值基数为0，以发布/测试提交顺序递增加1 。

6. YYYYMMDD代表生成版本的实际年月日，如：20190202

#### 版本基线定义

公司首次采用版本管理规范时，可以采取下列方法定义一个基线版本。

获取各项目最新的源程序、配置文件和文档，形成发布分支、测试分支和开发分支。

对每个项目的提测和发布分支都生成一个版本基线，如：Version1.0.0\_R1\_20190202。

#### 版本提交准则

##### 提交之前先更新

更新的原则是要随时更新，随时提交。当完成了一个小功能，能够通过编译并且自己测试之后，谨慎地提交。

如果在修改的期间其他同事也更改了同一个文件，那么pull更新时会自动进行合并，如果修改的是同一行或者二者修改差异过大，那么合并时会产生冲突。这种情况就需要同之前的开发人员联系，两人一起协商解决合并冲突。解决合并冲突之后，还需要两人一起测试，以保证解决冲突之后，各自的程序不会受到影响。

在更新时注意所更新文件的列表，如果提交过程中产生了更新，则需要重新编译并且再次完成单元测试，再进行提交。这样既能了解别人修改了哪些文件，同时也能避免合并错误导致代码有错。

##### 保持原子提交

为确保在需要时可以随时回溯代码版本，每次提交的代码只能包含实现一个独立、完整功能所必需的代码，不能夹带提交其他与此功能不相关的代码。为尽早提交，也可以将此独立、完整功能分解为若干小细节功能，分别开发并提交所必需的代码，但必须确保多次提交的功能代码组合在一起，完全实现此独立、完整功能。

仅提交自己修改的部分，最好不要一下子将整个项目提交。

每完成一个独立、完整的功能后，最好尽早提交，以免后续更改时出现bug，无法恢复到正常代码。

每次提交的间歇尽可能地短，以几个小时的开发工作为宜。我们提倡多提交，也就能多为代码添加上保险。为做到尽早提交，在开发功能模块的时候，先将功能分解成一个个独立的、不可再分割的小细节功能，分别完成。每完成一个并通过单元测试，就提交一次。在修改bug的时候，每修改掉一个bug并且确认修改了这个bug，也就提交一次。

##### 不要提交本地自动生成的文件

一般配置管理员都会将项目中一些自动生成的文件或者与本地配置环境有关的文件屏蔽提交（例如Eclipse中的.classpath文件等，Visual Studio中的.suo文件，Debug,Release,Obj等编译文件夹及其下文件，以及其他的一些自动生成，同编译代码无关的文件）。

如果项目中没有进行这方面的配置来强行禁止提交这样的文件，请自觉不要提交这样的文件，如果不小心签入了，需要从配置库中删除，以免其他同事在更新后就可能与本地的环境冲突从而影响大家的工作。

##### 不要提交不能通过编译的代码

代码在提交之前，首先要确认自己能够在本地编译通过，并且代码在提交前已经通过自己的单元测试。

如果在代码中使用了第三方类库，要把相应类库文件统一存储在代码相应目录中并提交，以免项目组成员中有些成员可能没有安装相应的第三方类库，从而在更新代码后引起代码运行错误。

##### 不要提交自己不明白的代码

代码在提交之后即被项目成员所分享。如果提交了不明白的代码，自己看不懂，别人也看不懂，如果在以后出现了问题将会成为项目质量的隐患。因此在引入任何第三方代码之前，确保对这个代码有一个很清晰的了解（必要时应有对应文档说明）。

##### 并行开发（同一模块）前沟通

如果开发小组采用并行开发模式开发同一模块功能，在开发前，需要对协作开发进行合理的工作计划与任务分配，让小组成员相互间了解对方的工作计划与工作内容。这样能尽可能的减少在开发过程中可能出现的冲突，提高开发效率。同时也能够在和成员的交流中发现自己之前设计的不足，完善自己的设计。

##### 对提交更新的信息采用明晰的标注

如果提交空的标注或者不确切的标注将会让项目组中其他的成员不了解此次签入动作的背景情况（如新增/修改签入的原因是什么？新增/修改什么内容？），项目经理无法通过提交的标注信息，清晰的掌握开发工作进度细节进度。没有清晰标注，甚至会对回溯代码版本造成影响。所以，在提交工作时，要填写明晰的标注，能够概要的描述所提交文件的信息，让项目组其他成员在看到标注后不用详细看代码就能了解你所做的修改。