# <项目名称>

**配置管理计划**

**版本 <1.0>**

# 修订版历史

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2023.10.22 | <1.0> | <第一版> | <狄锦文> |
|  |  |  |  |

**CCB委员会评审**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CCB评审** | **检查项描述** | **要求满足情况** | **是否满足情况** | **备注** |
| A001 | 配置项命名规范是否符合要求 | 是 | 是 |  |
| A002 | 配置项文档是否完整 | 是 | 是 |  |
| A003 | 有无修订历史表 | 是 | 是 |  |

**目录**

1. 简介
   1. 目的
   2. 范围
   3. 定义、首字母缩写词和缩略语
   4. 引用
   5. 概述
2. 软件配置管理
   1. 组织、职责和接口
   2. 工具、环境和基础设施
3. 配置管理活动
   1. 配置标识
      1. 标识方法
      2. 项目基线
   2. 配置和变更控制
      1. 变更请求的处理和审批
      2. 变更控制委员会 (CCB)
   3. 配置状态统计
      1. 项目介质存储和发布进程
      2. 报告和审计
4. 里程碑
5. 培训和资源
6. 分包商和厂商软件控制

# 配置管理计划

## 简介

### 目的

**同时更新**

当两个或更多的角色分别对同一个工件进行操作时，最后进行变更的那个角色将破坏前一个角色的工作。根本的问题是：如果系统不支持同时更新，则会导致系列变更并使开发流程减慢速度。但如果使用“同时更新”，那么挑战将在于检测更新已同时进行，而且要在并入变更时解决任何与集成有关的问题。

有限通知

由多个开发人员共享的工件中的某个问题得到了纠正，但只有相关的开发人员收到变更通知。

多个版本

多数大程序是以演进式发布版的方式开发的。一个发布版可能为客户所用，另一个发布版正在测试，而第三个发布版则尚在开发之中。如果在这些版本的任何一个版本中发现了问题，则需要将修复方案通报给所有这些版本。除非对变更进行慎重的控制和监测，否则可能会引起混乱，并且由于混乱而需要进行修复和返工，而这往往成本很高。

### 范围

配置管理计划是项目整体计划的一部分，适用于项目的配置管理活动。

配置管理计划的相关模型可以包括版本控制模型（如分支模型、标签模型）、构建模型（如持续集成、持续交付）、依赖管理模型（如依赖传递、依赖冲突解决）、配置管理模型（如环境变量、配置文件加载顺序）等。

此文档可能会影响其他事物，例如：开发团队的工作流程和规范。

构建和部署流程的标准化。

系统的配置管理和环境部署策略。文档管理和协作方式的规范化。

测试数据和用例的创建、维护和执行。

### 定义、首字母缩写词和缩略语

在配置管理计划中，常见的术语、首字母缩写词和缩略语的定义：

CCB：变更控制委员会 SCM：配置管理

其他术语在《项目词汇表附录》中可查找

### 引用

附录引用：

《项目词汇表附录》

《产品目录结构参考》

### 概述

配置管理计划其他部分：配置库权限:

工作库：项目组所有成员均有读写权限。 。

基线库：配置管理员有读写权限，其他人员经授权可调阅。

发布版本控制：使用版本号作为配置计划的基线标识。版本号采用数字，例如 1.0； 1.1。每当配置计划进行重大更新或变更时，可以增加主版本号或次版本号来标识新的基线。

组织方式如下：

$/

1. 初始配置
2. 启动
3. 需求分析

子系统一子系统二子系统三

1. 设计

子系统一子系统二子系统三

1. 编码
2. 测试
3. 安装
4. 总结
5. 变更
6. 项目管理

10 环境配置文档提交 个人工作区

狄锦文

…….

小组工作区

子系统一子系统二子系统三

其他结构参考附录《产品目录结构参考》

## 软件配置管理

* 1. **组织、职责和接口**职责：

1. 配置经理：配置经理是专门负责配置管理，负责制定和执行配置管理计划，确保配置管理活动的顺利进行。通常负责管理版本控制系统、构建和部署流程、依赖管理、配置文件管理等。
2. 项目经理：项目经理负责整个项目的规划和执行，确保配置管理活动与项目目标和进度保持一致。与配置经理和其他团队成员协调合作，确保配置管理计划的执行和效

果。

1. 测试经理：测试经理负责执行软件测试和质量保证活动，可能需要使用配置管理工具来管理测试数据、测试用例和测试环境的配置。
   1. **工具、环境和基础设施**计算机环境：Windows

软件工具：Git过程：

1. 初始化仓库：在项目的根目录下，使用 Git 初始化一个新的仓库。可以通过命令行执行

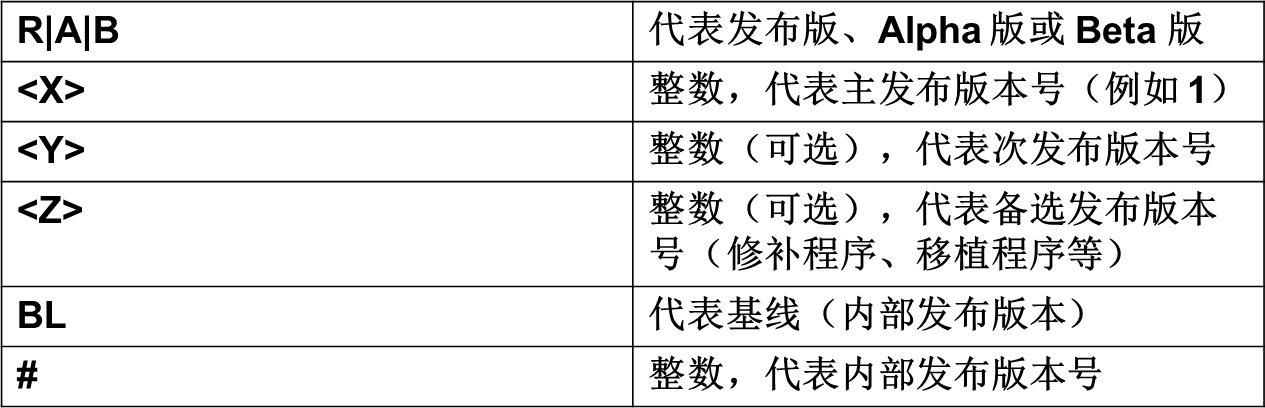
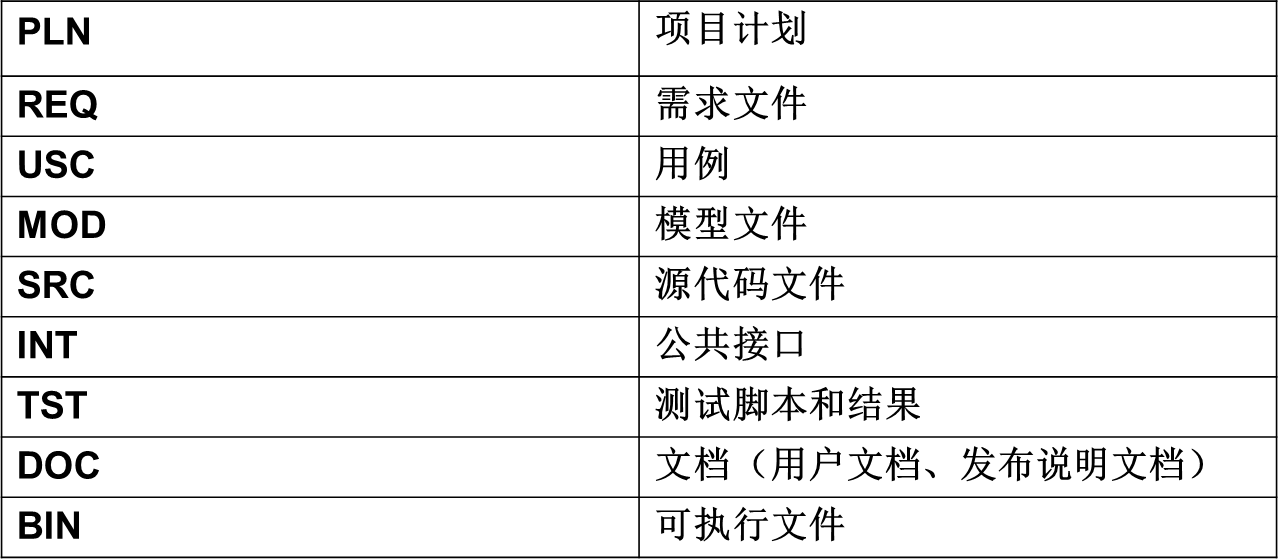
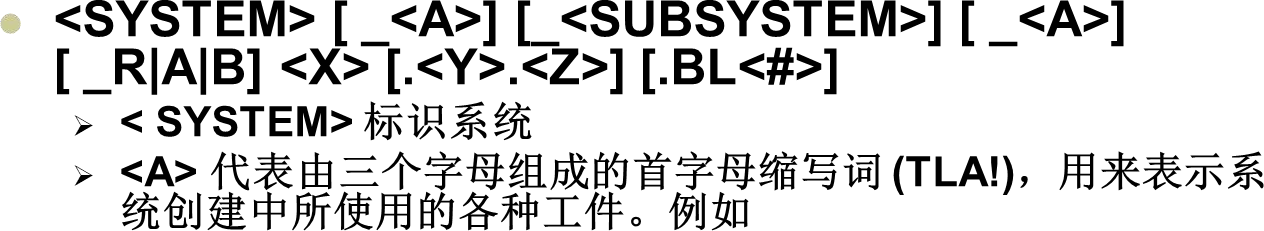
**git init** 来创建一个空的 Git 仓库。

1. 添加和提交配置项：将项目中的配置项添加到 Git 仓库中进行版本控制。使用 **git add**命令将配置项添加到暂存区（Staging Area），然后使用 **git commit** 命令提交配置项的更改到仓库中。提交时，应提供有意义的提交消息，描述所做的更改。
2. 创建和切换分支：如果需要并行开发或实验不同的功能，可以创建新的分支。使用 **git branch** 命令创建分支，并使用 **git checkout** 命令切换到相应的分支。开发人员可以在不同的分支上进行独立的工作。
3. 进行更改和提交：在所选分支上进行配置项的更改。使用 **git add** 命令将更改的配置项添加到暂存区，然后使用 **git commit** 命令提交更改到当前分支的仓库中。
4. 合并分支：当在分支上的工作完成时，可以将其合并回主分支或其他目标分支。使用 **git merge** 命令将分支的更改合并到目标分支中。在合并过程中，Git 会自动尝试合并更改，但可能会出现冲突，需要手动解决冲突。
5. 标记版本：在项目的重要里程碑或版本发布时，可以使用标签（Tag）来标记特定的配置项版本。使用 **git tag** 命令创建标签，并为标签提供描述性的名称。标签可以用于方便地跟踪和回溯特定版本的配置项。
6. 回退版本：如果需要回退到先前的配置项版本，可以使用 **git checkout** 命令加上目标版本的标识符或分支名称来切换到该版本。这将恢复项目的状态到指定的版本。
7. 远程仓库同步：为了实现团队协作和备份，可以将本地 Git 仓库与远程仓库进行同步。使用 **git remote** 命令添加远程仓库的 URL，并使用 **git push** 命令将本地仓库的更改推送到远程仓库。使用 **git pull** 命令从远程仓库拉取最新的更改。
8. 冲突解决：当多个开发人员同时修改同一个配置项时，可能会发生冲突。Git 会标记冲突的文件，并提供冲突解决工具来手动解决冲突。开发人员需要仔细审查冲突，编辑文件以解决冲突，并提交解决后的文件。

## 配置管理活动

### 配置标识

* + 1. 标识方法标识方法：



例如：

### T2K\_R1.0 Thorn 2000 系统 1.0 发布版本

* + 1. 项目基线

基线采用以下方式不断更新先启阶段：项目开始前

精化阶段：项目开始实施构建阶段：项目构建成型

产品化阶段：产品开发完成后

配置经理对基线授权基线包含内容：

* + - 1. 需求基线 （系统分析员）软件需求规格说明书
      2. 设计基线 （设计师）概要设计说明书
      3. 编码基线 （配置经理）各发布版本
      4. 测试基线 （测试经理）系统测试用例

系统出场测试报告

* + - 1. 界面基线（界面设计师）界面设计报告
      2. 验收基线 （项目经理）系统初验报告

系统终验报告

### 配置和变更控制

* + 1. 变更请求的处理和审批

1.

变更申请

1) 变更申请人通过提出对配置项的变更请求。

2)

变更申请人负责填写《变更申请表》，并提交配置控制委员会实施变更评估。

2.

变更评估

1. 针对变更申请人提交的变更请求，配置控制委员会在评估后决定是否实施该变更。
2. 配置控制委员会将针对该变更做出的决定（接受或拒绝）通知变更申请人。

3.

变更实施

1) 变更申请获得批准后，配置控制委员会将该变更分配给变更申请人实施。

1. 变更执行人实施该变更；

4.

变更复审

1) 配置控制委员会对变更后的工作产品进行验证，以确定变更是否正确完成。

2)

在变更完成并经过验证后，配置经理将经批准的配置项写入配置库。

* + 1. 变更控制委员会 (CCB)

[说明 CCB 在处理和审批变更请求时所遵循的成员资格标准和过程。]

CCB 成员：

安姿桐（CCB 组长）赵欣宇

郑佳策 孟庆涛 狄锦文 罗香雨。项目经理

安姿桐：负责对项目的总体调控，对项目中计划的变更等进行确认，并对变更所涉及的资源变更进行评估，

系统分析员：

赵欣宇：负责项目的整体需求。设计师：

郑佳策：负责概要设计评估测试经理

孟庆涛：负责评估变更中测试方面的问题。配置经理

狄锦文：负责变更的执行以及配置库日常管理和权限控制。界面设计师

罗香雨：界面设计评估。

### 配置状态统计

* + 1. 项目介质存储和发布进程配置状态发布：

发布名称 发布频率 发布内容 已知问题 安装说明 发送方式基线发布 基 线 生 成 或 基线变更时 版本号 …… …… e-mial

备份：

定期备份配置管理员每周整体备份一次配置库，保留 4 周以内的备份记录。、备份的文件要明确标明备份日期，保存在可靠的 U 盘中

事故处理恢复计划：

出现事故时恢复到最新备份日期存储介质：U 盘

格式：zip 压缩包

* + 1. 报告和审计
       1. 配置项状态报告表格示例：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **配置项名称** | **版本号** | **发布日期** | **最后修改日期** | **状态** | **所有者** |
| 配置项 A | 1.0 | 2021-01-01 | 2021-02-15 | 活动 | 用户 A |
| 配置项 B | 2.3 | 2021-03-05 | 2021-04-20 | 活动 | 用户 B |
| 配置项 C | 1.5 | 2021-02-10 | 2021-03-30 | 过时 | 用户 C |

* + - 1. 变更管理报告表格示例：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **变更编**  **号** | **变更**  **类型** | **变更原**  **因** | **变更目的** | **存在时间** | **危急程度** |
| CR-001 | Bug 修  复 | 修复配  置项 A 中的错误 | 提高系统稳定性 | 1 日 | 低 |
| CR-002 | 功能  增强 | 添加新  功能 | 改善用户体验 | 2 日 | 中 |
| CR-003 | 配置  更改 | 更新配  置项 B 的设置 | 优化性能 | 1 周 | 高 |

3 审计报告表格示例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | | |
| **检查项编号** | **检查项描述** | **要求满足情况** | **实际情况** | **备注** |
| A001 | 配置项命名规范是否符合要求 | 是 | 是 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A002 | 配置项文档是否完整 | 是 | 否 | 部分文档缺失 |
| A003 | 配置项访问权限是否受限 | 是 | 是 |  |

## 里程碑

此项目主要分为以下五个里程碑：

需求分析确立

项目的需求分析全部完成

已形成相应的需求分析说明书及其它附属文档需求分析说明书已通过项目经理确认为需

求分析阶段已结束，可以进入设计阶段

设计完成

已形成相应的概要设计说明书及其它附属文档概要设计说明书已通过项目经理确认为需

求设计阶段已结束，可以进入设计阶段

编码完成

项目的编码全部完成

项目所有程序已经经过调试并确定可以运行已通过项目经理确认为编码阶段已结束， 可以进入测试阶段

1. 测试完成

测试需求已经确定并完成；

已形成相应的测试计划说明书及其它附属文档测试用例已经覆盖所有测试需求

已形成相应的测试用例说明书及其它附属文档

系统测试完成，所发现的所有缺陷已得到妥善处理已完成测试分析报告

1. 项目结束上线成功

已得到客户的确认并通过验收测试与客户一致认为该项目已结束

## 培训和资源

工具：Git

人员：CCB 成员

培训：培训项目组成员、执行日常配置管理工作

## 分包商和厂商软件控制

根据项目的部署计划，将外部软件部署到项目环境中。确保正确配置和设置外部软件的参数，以便与项目环境和其他组件进行正确的交互。

**附录**

《项目词汇表附录》

1. CI/CD：持续集成/持续交付（Continuous Integration/Continuous Delivery）是一种开发实践，通过自动化构建、测试和部署流程，实现快速、可靠的软件交付。
2. VCS：版本控制系统（Version Control System）是一种用于管理和跟踪代码版本的工具，如 Git、Subversion 等。
3. SCM：软件配置管理（Software Configuration Management）是一种管理软件开发过程中的配置项和变更的实践，包括版本控制、构建、发布等。
4. Repository：代码库或存储库，用于存储和管理软件开发过程中的代码、文档、配置文件等资源。
5. Branch：代码分支，是基于主分支创建的独立开发线，用于并行开发、实现新功能或修复 bug。
6. Tag：标签，用于在代码库中标记特定的版本或里程碑，以便于后续追踪和回溯。
7. Build：构建，指将源代码转换为可执行软件的过程，包括编译、打包、测试等。
8. Artifact：构建产物，生成的可执行软件、库文件、文档等。
9. Dependency：依赖项，指项目所依赖的第三方库、组件或模块。
10. Build Tool：构建工具，如 Maven、Gradle 等，用于自动化构建过程。
11. Configuration File：配置文件，包含系统运行所需的各种配置参数和设置。
12. Environment：环境，指开发、测试或生产等不同的运行环境。
13. Deployment：部署，将软件应用程序安装和配置到目标环境中的过程。
14. Continuous Integration（CI）：持续集成，指频繁地将开发人员的代码变更合并到共享代码库中，并进行自动化构建和测试。
15. Continuous Delivery（CD）：持续交付，指将经过集成和测试的软件交付到可部署的状态，以便随时可以进行部署。
16. Test Data：测试数据，用于执行软件测试的输入数据。
17. Test Case：测试用例，定义了对软件系统进行验证的一组输入、操作和预期结果。
18. Documentation：文档，包括技术文档、用户文档、需求文档等。
19. Logging：日志记录，用于记录系统运行时的事件、错误和状态信息。

以上是一些常见的术语、首字母缩写词和缩略语的定义，可以根据具体项目和组织的需求进行补充和调整。

《产品目录结构参考》

