软件工程基础

第五章 软件配置管理

苏 生
83201311(Tel)
susheng@uestc.edu.cn

信息与软件工程学院
电子科技大学

本章学习目标

1. 掌握软件配置管理的作用
2. 掌握软件配置管理中心存储库的概念与作用
3. 掌握软件配置管理的过程

第五章 软件配置管理

5.1 软件配置管理的缘由

5.2 软件配置管理中心存储库

5.3 软件配置管理过程

程序员的问题

- 无法找到最新的源程序文件
- 无法找到源文件的历史修改信息
- 多个人修改同一个源文件，有些人的修改被冲掉了
- 程序被误删了，尝试恢复失败，只能重写

项目经理的问题

- 为在项目组成员中共享和隔离资料烦恼
- 无法有效掌握项目过程产生的文件、代码和工作成果
- 调试过程中，项目成员经常为一些问题扯皮，搞不清楚到底是谁产生的错误

产品经理的问题

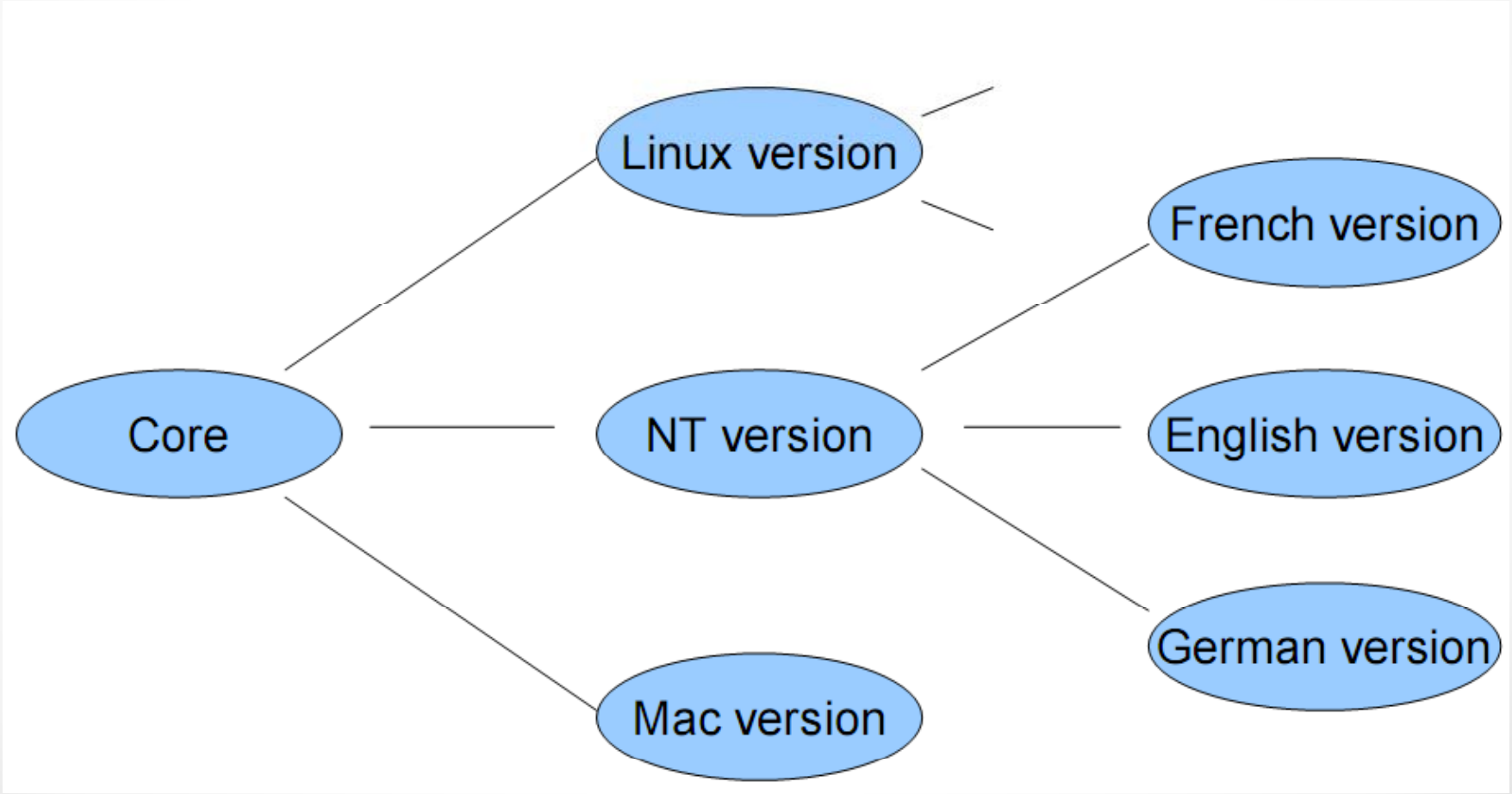
- 交付给用户的软件“缺斤少两”，在安装时出现问题
- 用户使用时发现的问题不能得到及时有效解决

软件配置管理的缘由

- 软件开发过程中的其他问题
  - 已经发现的**BUG**又重新出现
  - 已经发布的软件不能够再次构建（**Build)**
  - 丢失软件版本对应的源代码
  - 丢失关键文件
  - 文件被“神秘”地修改了
  - 多人协助开发难以继续

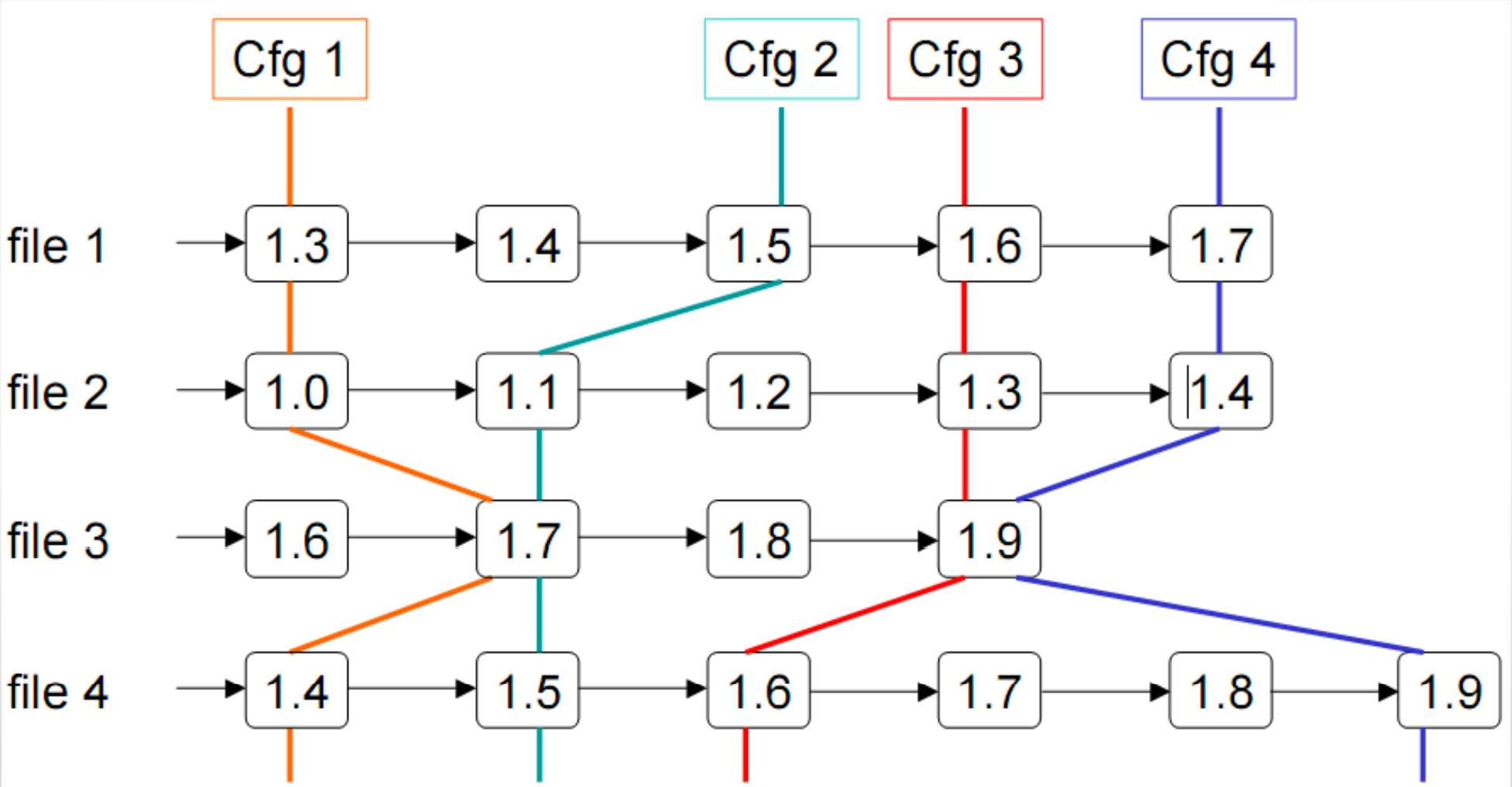
软件配置管理的缘由

- **Netscape**软件开发人员的困扰



软件配置管理的缘由

- 一个包含**4**个源代码文件的软件产品



软件配置管理的缘由

- 软件配置管理好处
  - 源代码、文档管理将更为可靠和有序
  - 多人协作并行开发简单了
  - 系统可以自动构建并且更加快速
  - 已经解决的**BUG**将不会再骚扰程序员
  - 保证软件正确的配置，如兼容性配置

软件配置管理

- 软件配置的定义**1**：
  - 一套应用技术上和管理上的指导和监督方法。
- 该方法用于：
  - 识别和记录配置项的功能特征和物理特征。
  - 控制这些特征的变更。
  - 记录和报告变更的处理和执行的状态。
  - 验证其是否符合特定的需求。

软件配置管理

- 定义2
  - 是用来建立和维护软件项目产品的完整性，并贯穿于软件生命周期的始终，包括确认软件配置项、控制变更、记录和报告变更实现的状态。

    （来自：**IEEE Standard Glossary of Software Engineering Terminology)**
  - 关键词：软件配置项、变更、软件生命周期

软件配置管理

软件配置项：配置管理的对象称为软件配置项

分 类	特　征	举　例
环境类	软件开发环境及 软件维护环境	编译器、操作系统、编辑器、数据库管理系统、开发工具（如测试工具）、项目管理工具、文档编辑工具
定义类	需求分析及定义阶段完成后得到的工作产品	需求规格说明书、项目开发计划、设计标准或设计准则、验收测试计划
设计类	设计阶段结束后得到的产品	系统设计规格说明、程序规格说明、数据库设计、编码标准、用户界面标准、测试标准、系统测试计划、用户手册
编码类	编码及单元测试后得到的工作产品	源代码、目标码、单元测试数据及单元测试结果
测试类	系统测试完成后的工作产品	系统测试数据、系统测试结果、操作手册、安装手册
维护类	进入维护阶段以后产生的工作产品	以上任何需要变更的软件配置项

第五章 软件配置管理

5.1 软件配置管理的缘由

5.2 软件配置管理中心存储库

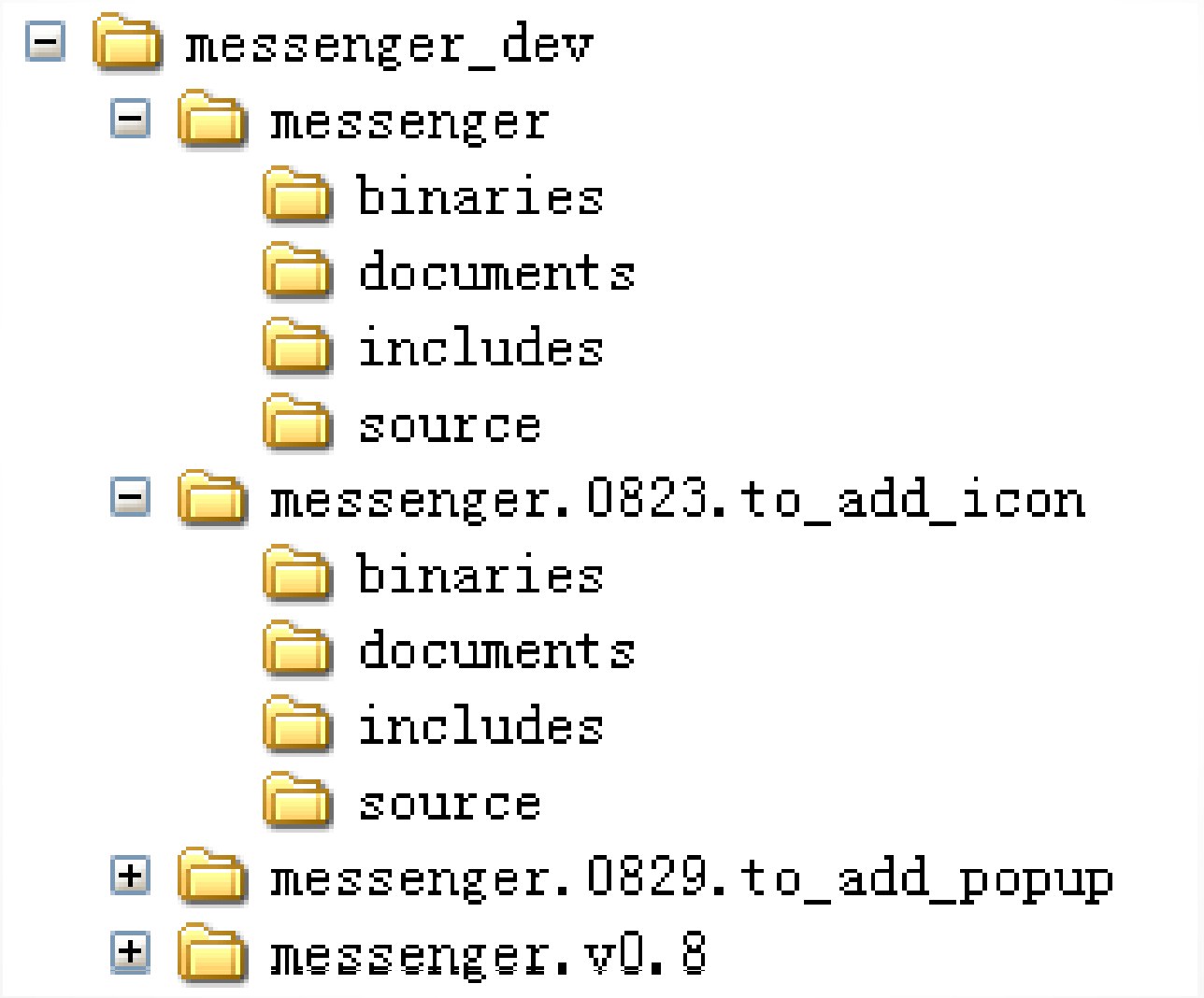
5.3 软件配置管理过程

简单版本控制

- 简单版本控制方式（**ctrl+s**）
  - 定期保存内容，将失误造成的损失降到最小。
  - 作大的修改前备份，修改不成功时可以退回。
  - 以时间和修改内容命名源代码
- 简单版本控制的不足
  - 适合单个开发人员，对与多开发人员不适合。
  - 如果开发时间周期长，版本的修改自己要糊涂的！

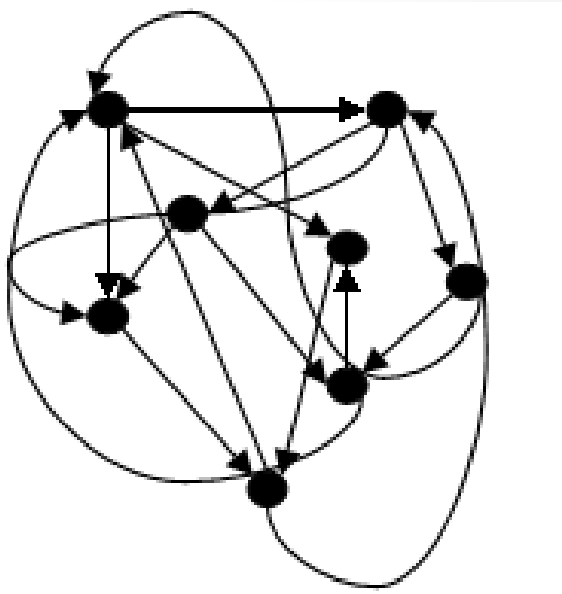
    如：拷贝保存源代码目录为：**2009.1.20_sort**，**1**个月之后自己还记得具体作了什么内容？
  - 是全备份模式，浪费空间
  - 可靠性，存储设备失效如何处理？

简单版本控制


messenger_dev
messenger
binaries
documents
includes
source
messenger.0823.to_add_icon
binaries
documents
includes
source
messenger.0829.to_add_popup
messenger.v0.8

软件配置管理中心存储库

- 多开发人员的软件开发模式：
  - 每个程序员负责一个专门的模块。
  - 修改自己的代码，假设不存在多个程序员修改同一处源代码的问题。
  - 各个程序员相互传送代码
- 问题：
  - 修改之前从哪里获得新版本？
  - 修改后的结果提交到哪里？

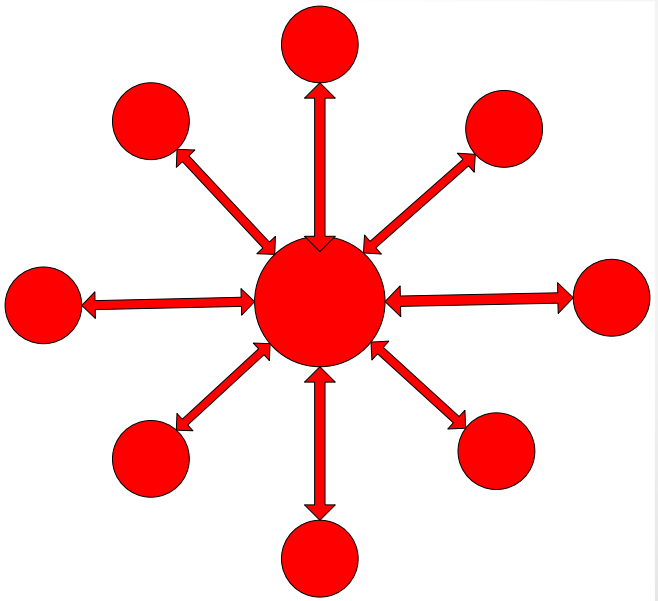


软件配置管理中心存储库

- 问题一造成的结果：
  - 某程序包括**A**，**B**，**C**三个模块。
  - 张三负责**A**，王二负责**B**，李四负责**C**。
  - **A**对外提供了一个**Time**函数，该函数提供定时功能，传入参数为需要定时的时间值，单位为毫秒。
  - 张三将**Time**的参数单位改为秒。
  - 王二的**A**模块是在张三修改之前拷贝过来的，模块**A**的**Time**函数参数还是毫秒。王二将参数设置为**5000**
  - 张三接收王二的模块**B**，集成测试并等待**5000**秒。
- 问题原因？ 版本不一致了！！

软件配置管理中心存储库

- 问题二造成的结果：
  - 程序员**A**修改**Bug1**并提交给用户。
  - 程序员**B**修改**Bug2**并提交给用户。
  - **Bug1**再次出现。
- 解决办法：
  - 避免相互拷贝代码。
  - 将源代码流转的渠道从网状改为心型结构。即设立一个软件配置管理**(SCM)**中心存储库。



软件配置管理中心存储库

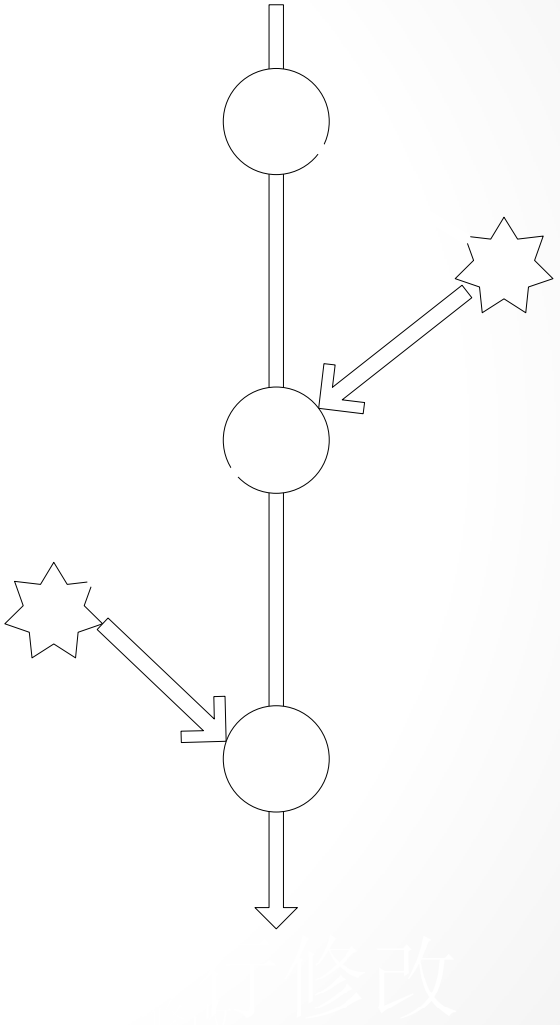
- 对于问题一，如果王二是从**SCM**中心存储库获得代码，则他会知道参数为秒。
- 对于问题二，程序员修改完代码后，并不是提交给客户而是提交到**SCM**中心存储库，在**SCM**中心存储库编译成可运行程序后再交给客户。
- 不足之处：
  - 多个程序员不修改同一源文件的假设不成立，会导致更为复杂的问题！

防止版本覆盖

- 版本覆盖问题：
  - **SCM**中心存储库有**A**，**B**，**C**三模块。
  - 张三要修改**A**，**B**。同时王二要修改**B**，**C**。
  - 张三先修改完成并提交到**SCM**中心存储库。
  - 王二完成后提交。
  - 张三的修改内容被覆盖。

防止版本覆盖

- 解决办法：
  - 串行方法
    - 修改前先上锁，修改完成后提交并解锁。
    - 锁的粒度越小越好，因为可以提高并行度。
  - 并行方法
    - 记录每个人修改前的版本和修改后的版本，在将来的某一个时刻把他们合并。

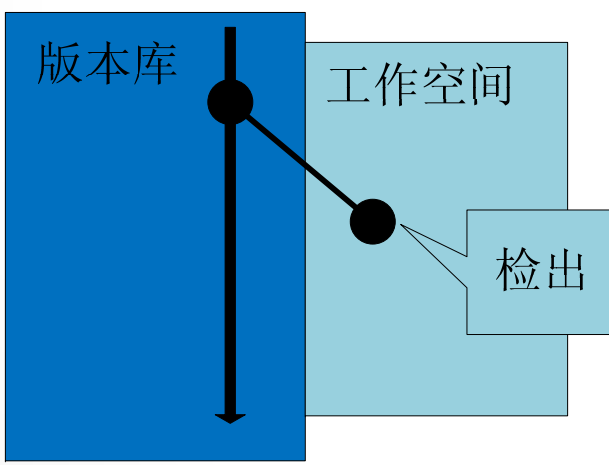


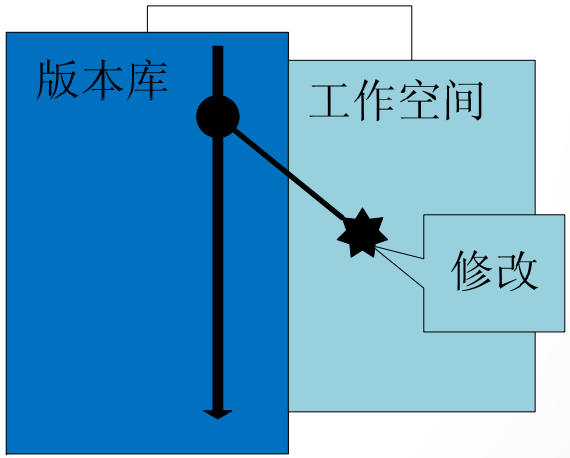
专业术语

- 版本库：软件配置管理中心存储库
- 增量存储：在版本库中只存储的源代码的各个版本差异部分
- 工作空间：每个程序员工作的地方。程序员从版本库中取出源代码放到工作空间，在这里查看，修改，编译，运行和调试。完成后再把新版本的代码放回版本库中。

专业术语

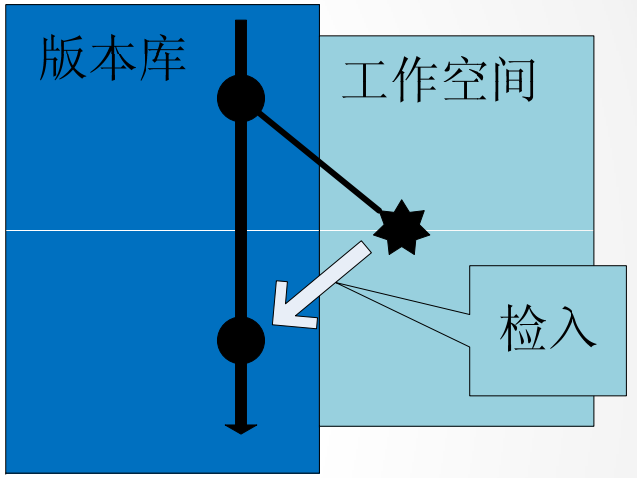
- 检出：在修改代码之前，告知版本控制工具的操作过程。





专业术语

- 检入：
  - 修改完成后，告知版本控制工具的操作过程。
- 软件配置管理（CM）：
  - 版本控制管理工具的选择，安装，设置，培训，疑难解答等一系列工作



常用版本控制软件

- **Tortoise SVN：**
  - 支持串行，并行修改方法，对分支、标签、版本保存、配置变更都有较好的支持，是比较受欢迎的一款软件，并且开源，支持二次开发。
- **Git**
  - 免费、开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。
- **ClearCase：**
  - **IBM**公司收购的大型商用软件配置管理工具，功能强大，非常适用于大型软件系统开发，但配置管理复杂，使用成本较高
- 其它工具：
  - **Cvs**等

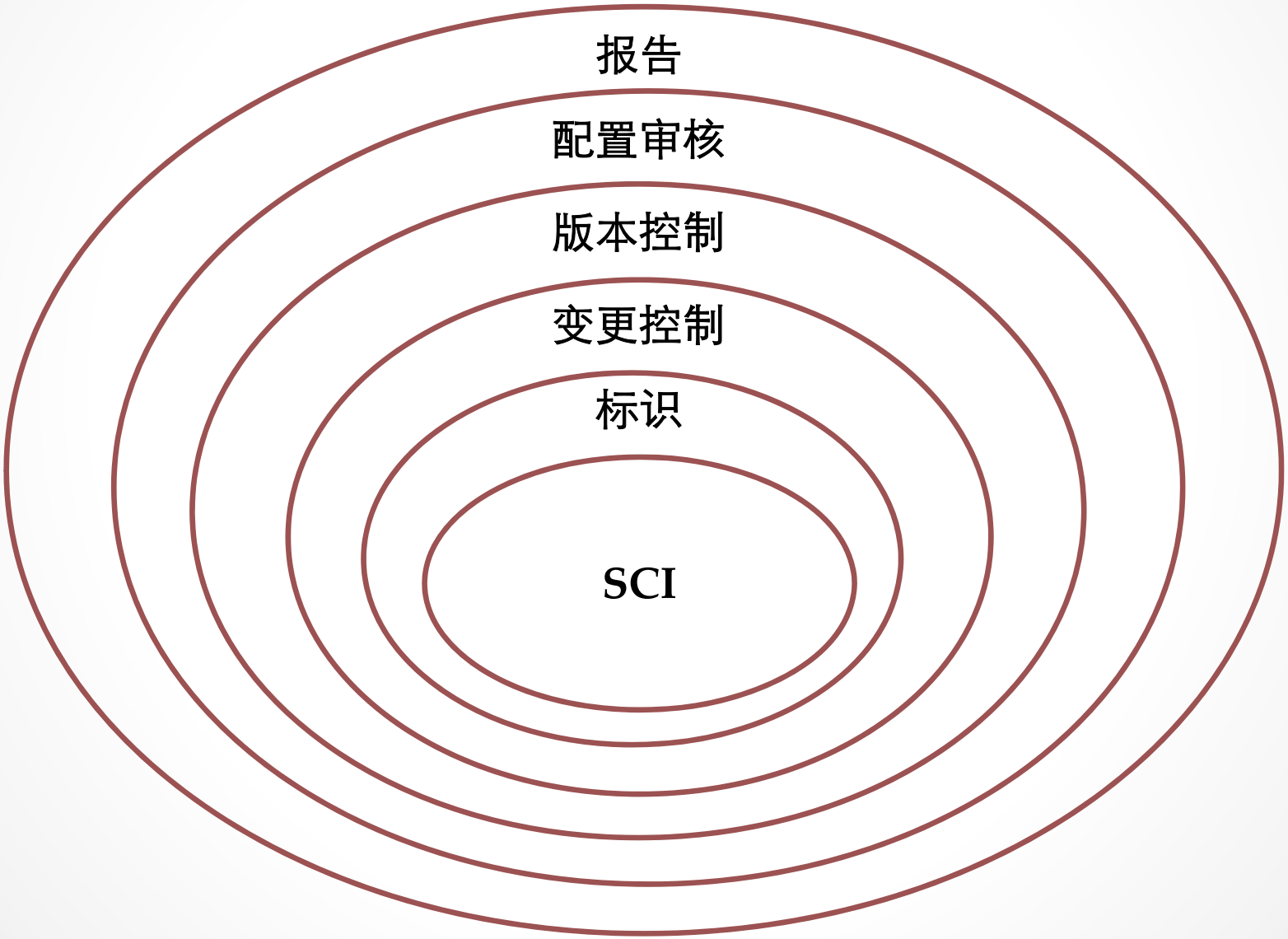
第五章 软件配置管理

5.1 软件配置管理的缘由

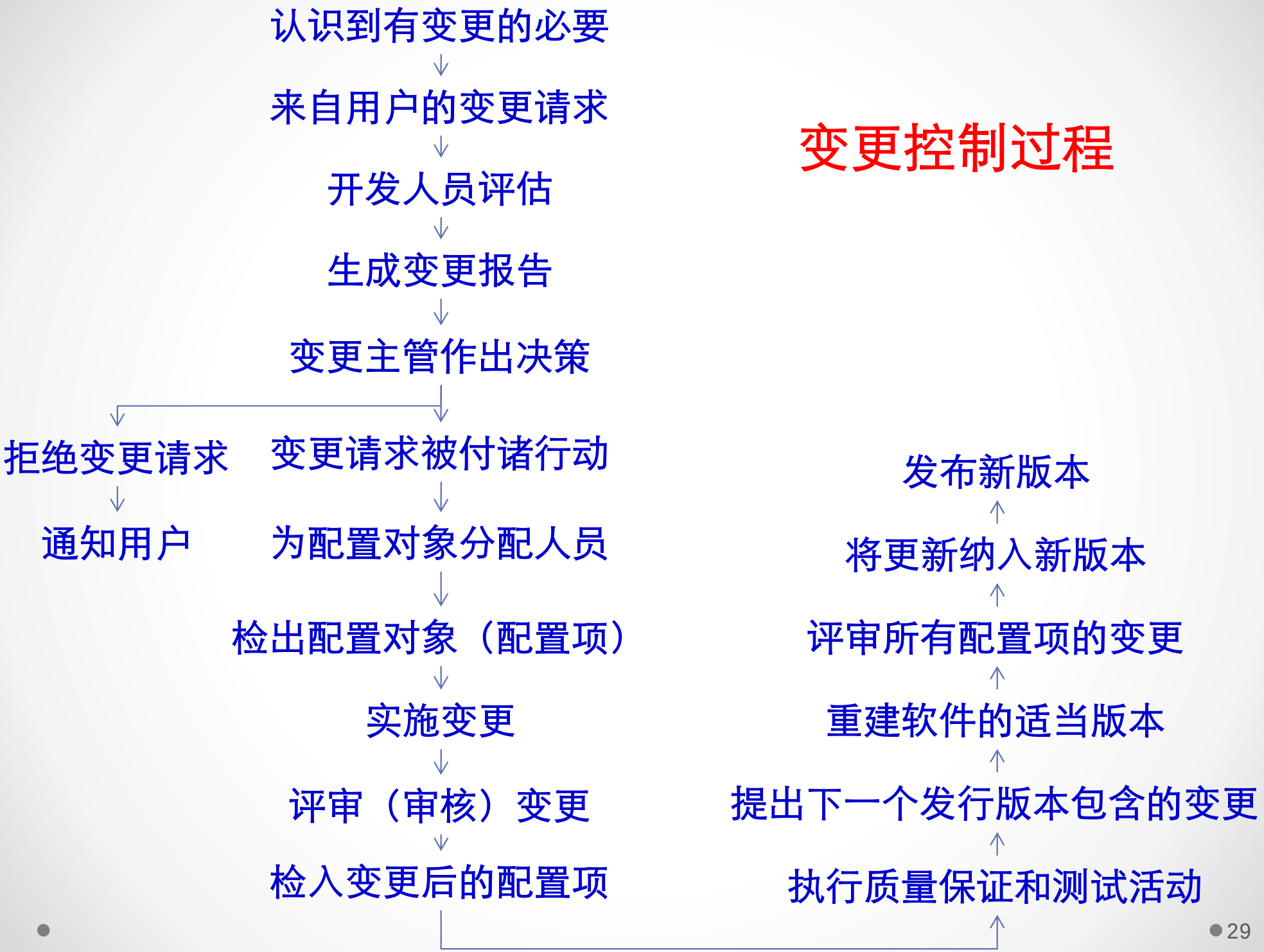
5.2 软件配置管理中心存储库

5.3 软件配置管理过程

软件配置管理过程



变更控制过程


认识到有变更的必要
来自用户的变更请求
开发人员评估
生成变更报告
变更主管作出决策
拒绝变更请求
通知用户
变更请求被付诸行动
为配置对象分配人员
检出配置对象（配置项）
实施变更
评审（审核）变更
检入变更后的配置项
执行质量保证和测试活动
提出下一个发行版本包含的变更
重建软件的适当版本
评审所有配置项的变更
将更新纳入新版本
发布新版本

Question?