



SOFT130006

软件工程

3. 版本与开发任务管理



复旦大学软件学院
彭鑫

pengxin@fudan.edu.cn

团队协作开发：需要考虑的问题

开发任务从何而来，有哪些类型？
如何分配和管理开发任务？
如何协调不同开发人员的工作？
如何避免工作内容上的冲突？



开发任务管理
版本管理



软件配置管理

软件配置管理

- 软件开发过程的各种软件制品
 - ✓ 需求模型、设计模型、源代码、可执行文件、测试用例等
 - ✓ 技术文档、计划文档、会议记录等相关文档
- 软件变更不可避免，新的版本不断产生
 - ✓ 客户或用户需求发生变化
 - ✓ 修复软件测试或用户使用过程中发现的缺陷
 - ✓ 软件为了保持市场竞争力需要引入新的特性
- 软件开发自身也经常是增量和迭代的
- 配置管理：确保软件开发和变更有序进行并向客户发布正确的产品版本的一整套管理方法和工具

软件配置管理的主要内容-1

- 版本管理

- ✓ 规范软件版本命名，制定版本发布和迭代计划
- ✓ 跟踪软件的变更与版本历史
- ✓ 确保不同开发人员的修改不会彼此干涉

- 开发任务管理

- ✓ 开发任务包括：由正向需求分解引出的特性开发任务、由软件缺陷引发的缺陷修复任务
- ✓ 对开发任务进行规范描述
- ✓ 追踪开发任务的处理流程
- ✓ 对变更请求进行决策
- ✓ 跟踪变更请求的处理与实施

软件配置管理的主要内容-2

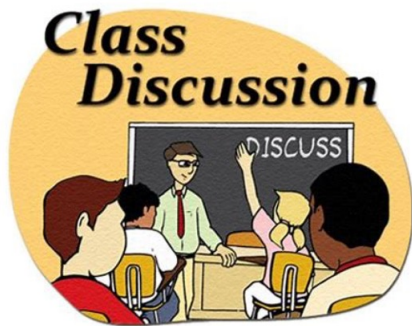
- 构建管理

- ✓ 管理软件的外部依赖，例如第三方库
- ✓ 对代码、数据和外部依赖等软件制品进行编译和链接从而生成可执行的软件版本
- ✓ 运行测试以检查构建和集成是否成功

- 发布管理

- ✓ 在软件构建结果的基础上打包形成可发布的软件版本并进行存档
- ✓ 提供回溯和审查
- ✓ 持续跟踪已经发布供客户和用户使用的软件版本

课堂讨论：协作工作与冲突



课堂讨论：你是否经历过与他人协同工作（不限于软件开发）中的冲突问题，你们是如何解决冲突问题的？

课堂讨论：软件集成与发布

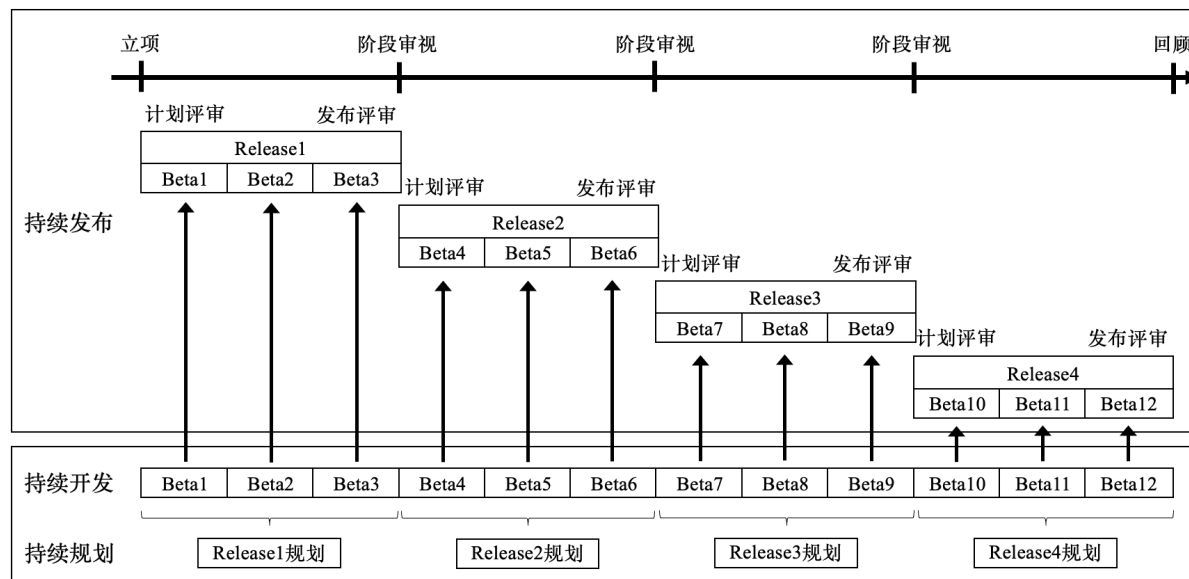


课堂讨论：在你过去的开发经历中，当你完成开发任务后你是如何准备和生成对外的交付物？

软件项目的持续规划、开发与发布

• 持续开展项目的规划、开发与发布

- ✓ 例如，可以规划每月发布一个迭代版本（Beta），每季度发布一个稳定的发布版本（Release）
- ✓ 所有的版本计划和版本发布都需要通过评审
- ✓ 每个稳定版本的发布都需要通过阶段审视
- ✓ 目的：完善并改进团队的管理方法与流程、开发人员的技能、工作量的计划与分配等



版本发布计划

- 明确版本定位、目标、商业价值，规划里程碑，确定版本发布时间
- 估算用户故事的工作量
- 排列用户故事的优先级
- 估算开发团队在每轮迭代中完成的工作量并分配用户故事
- 在此基础上创建版本发布计划

版本管理

- 软件开发和演化过程伴随着各种软件制品的持续变化
 - ✓ 源代码文件、配置文件、文档等原子性的制品文件
 - ✓ 模块、组件等复合性的软件制品乃至整个软件产品
- 版本管理的任务
 - ✓ 将各种软件制品都置于系统性的管理之中，进行版本标识
 - ✓ 追踪演化历史
 - ✓ 确保开发人员在并行协作开发过程中不会相互影响

产品版本号命名

- 每一个软件发布版本都需要分配唯一的版本号以明确标识不同的发布版本
- 常用的点分式版本命名规范：格式为 M.S.F.B([SP][C]) **[]表示可选字段**
 - ✓ 主版本号M：标识产品平台或整体架构
 - ✓ 次版本号S：标识局部架构、重大特性或无法向前兼容的接口
 - ✓ 特性版本号F：标识规划的新特性版本
 - ✓ 编译版本号B：标识编译构建的版本号
 - ✓ 补丁包版本号SP：标识累计一段时间的补丁，即把一段时间的补丁打包出一个补丁包
 - ✓ 补丁版本号C：标识一个补丁

代码版本管理场景1：代码演化历史跟踪

- 软件开发过程中新增或修改代码，例如
 - ✓ 可能花了2周时间，写了6000行代码来实现一个客户或产品经理所要求的新特性
 - ✓ 花了2天时间，改动了30个文件来重构代码以提高软件的可维护性
 - ✓ 花了3个小时，改动了4个方法来修复了一个测试人员报告的缺陷
- 如果没有代码版本管理
 - ✓ 不记得为何、在何处以及如何修改的代码
 - ✓ 导致自己以及他人难以理解代码的变动过程、开展代码评审以及及时发现修改过程中引入的缺陷

代码版本管理场景2：历史版本回退

- 开发人员希望回到此前的一个稳定的代码版本上，例如
 - ✓ 新特性上线后不符合客户或市场要求，需要去除这一新特性
 - ✓ 由于增加新特性、代码重构或其他原因而导致重要的功能不可用
- 如果没有代码版本管理
 - ✓ 很难准确而快速地回退到指定的代码版本，从而增加了代码维护的难度

代码版本管理场景3：多人开发协作冲突

- 多人协作开发可能出现冲突，例如
 - ✓ 花了很长时间修改了一个代码文件，然后发现这个文件已经被其他开发人员修改或者删除了
 - ✓ 修改完代码提交后过了一段时间发现被其他开发人员覆盖了
- 如果没有代码版本管理
 - ✓ 协作冲突造成修改内容丢失
 - ✓ 协作冲突难以被察觉和及时处理
 - ✓ 最终导致开发上的混乱

代码版本管理场景4：代码质量检查

- 开发人员可能会将有质量问题（功能缺陷或可维护性问题）的代码合入到版本库中
 - ✓ 造成潜在的缺陷隐患
 - ✓ 影响其他开发人员的代码理解和修改
- 需要根据规范要求检查需要合入的代码
 - ✓ 通过代码自检、同行评审以及门禁检查等手段
 - ✓ 通过检查的代码才允许提交或合入主干分支
- 如果没有代码版本管理
 - ✓ 难以形成代码质量管控的“关口”
 - ✓ 难以有效地管控开发人员所提交的代码质量

版本控制系统

- 实现代码版本管理目标的一种有效途径
 - ✓ 存储、追踪代码修改的完整历史记录（即版本库）
 - ✓ 提供多种机制帮助开发人员进行协同开发
 - ✓ 实现代码合入前的质量检查等门禁检查功能
- 两类版本控制系统
 - ✓ 集中式版本控制系统：版本库集中存放在中央服务器上
 - ✓ 分布式版本控制系统：每个开发人员的客户端机器上都存储着完整的版本库

集中式版本控制系统

- 使用方式

- ✓ 开始工作时，先从服务器拉取工作文件的最新版本
- ✓ 完成工作后，将工作文件的更新提交到服务器

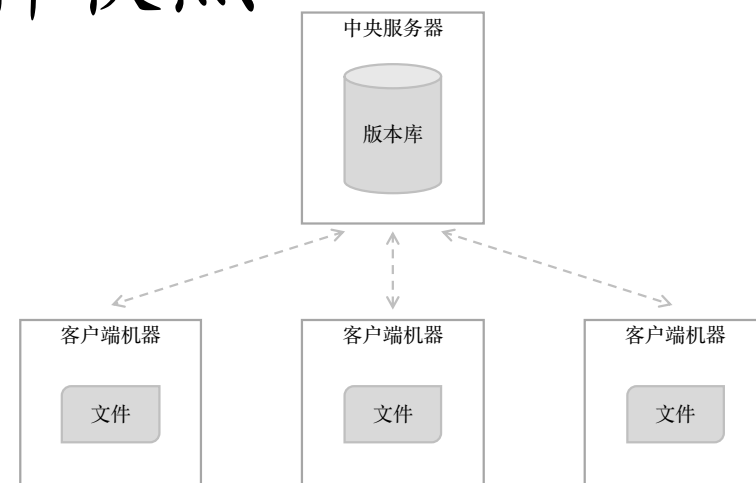
- 客户端机器只存储所拉取的文件快照

- 典型代表：CVS、SVN

- 主要缺点

- ✓ 必须联网才能工作
- ✓ 服务器单点故障影响整个团队
- ✓ 容易发生版本数据丢失

- 已经逐渐被分布式版本控制系统所取代



分布式版本控制系统

- 使用方式

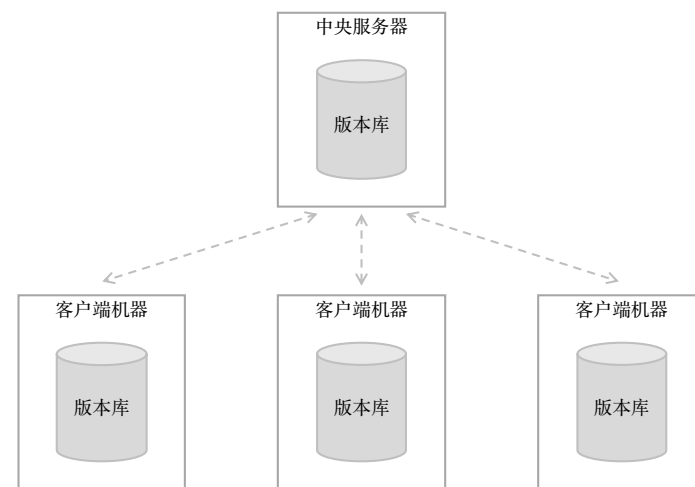
- ✓ 每个开发人员维护着自己的本地版本库（克隆）
- ✓ 完成工作后可以将更新提交到本地版本库（无需联网）
- ✓ 可以拉取中央服务器上的最新版本库，也可以将本地版本库的更新推送到中央服务器上的版本库

- 客户端机器存储着完整的版本库

- 典型代表：Mercurial、Git

- 主要优点

- ✓ 无需联网工作
- ✓ 版本库数据可以从任何本地库恢复，可靠性高
- ✓ 灵活、强大的分支管理

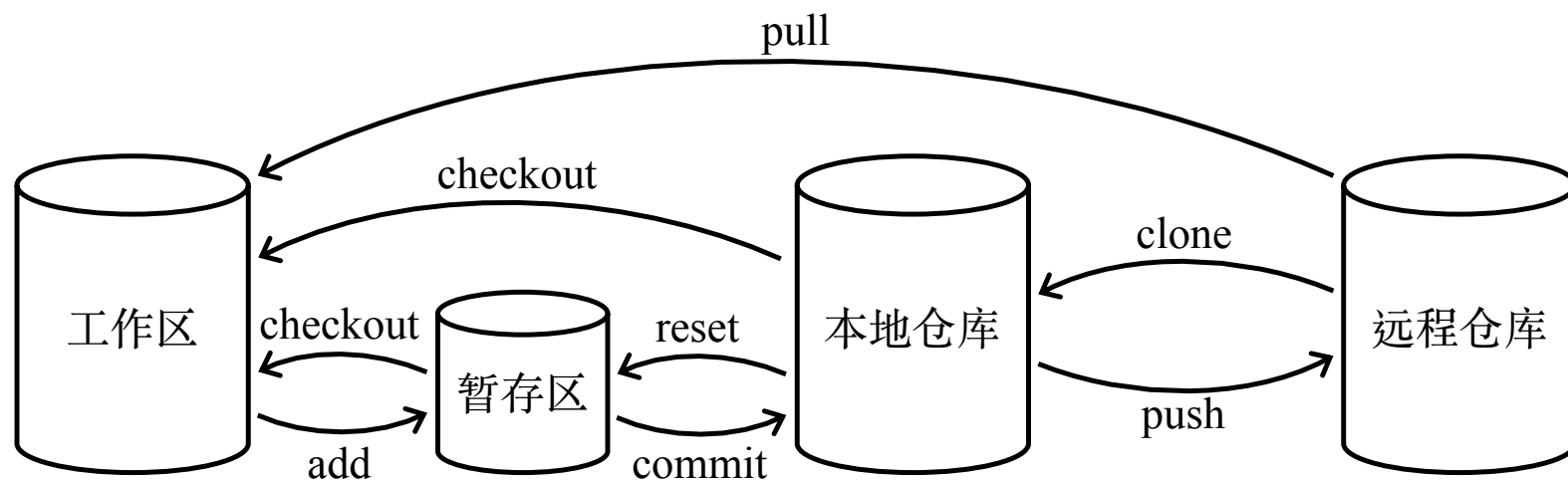


集中式与分布式版本控制系统对比

集中式版本控制系统	分布式版本控制系统
集中式架构	分布式架构
客户端本地 没有完整版本历史	客户端本地 保存完整版本历史
只能联网提交	可离线提交（本地仓库）
以目录的方式管理分支， 不够灵活	强大的分支管理能力

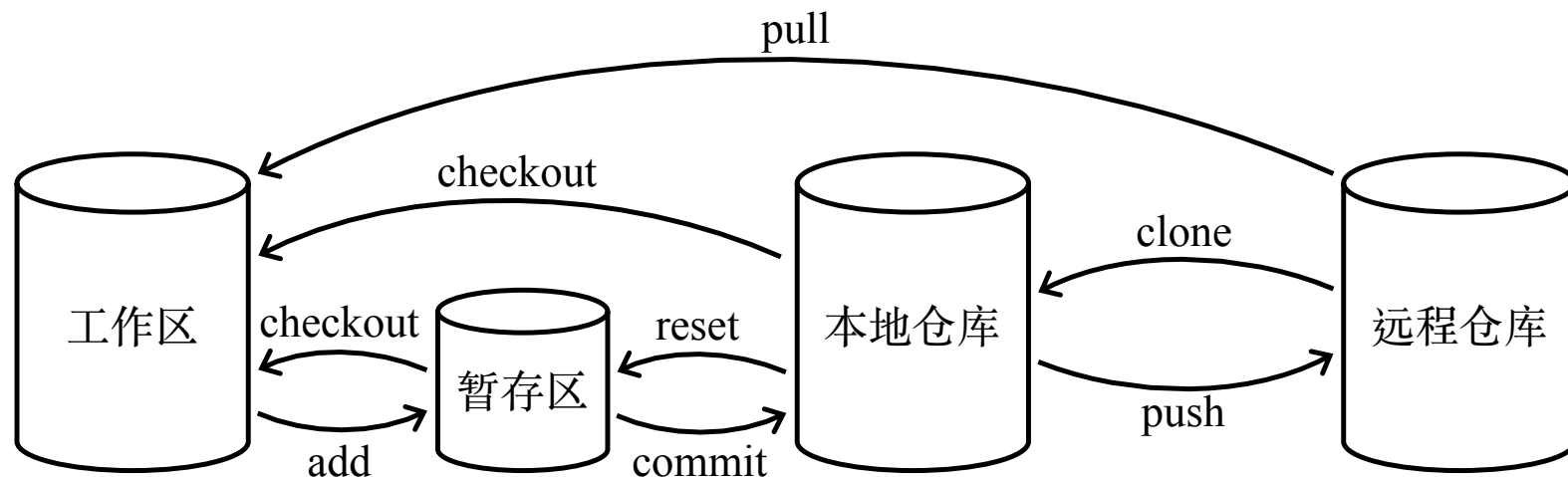
分布式版本控制系统Git工作流程-1

- clone命令：将中央服务器上该项目的远程仓库拷贝到本地机器上
- commit命令：将暂存区中的文件提交到本地仓库
- pull命令：将中央服务器上远程仓库的所有最新提交全部拉取到本地仓库并进行合并
- push命令：将本地仓库中的提交推送到中央服务器的远程仓库中



分布式版本控制系统Git工作流程-2

- checkout命令：将(改乱的)文件重置为暂存区或者本地仓库中的文件内容
- add命令：将指定的文件（更改）保存到暂存区
- reset命令
 - ✓ 为撤销提交：将暂存区重置到这次提交之前的状态，也可以选择是否将工作区也重置到这次提交之前的状态
 - ✓ 为重置暂存区：将暂存区中的文件重置为本地仓库中的文件内容



Git中的提交 (Commit)

- 原子性：提交粒度上要确保每次提交只做一件事情，不要把多件事混在一次提交中
- 在commit命令中指定该提交的描述消息
 - ✓ 类型：新功能、缺陷修复、重构、测试、文档、格式化等
 - ✓ 主题：提交的简要描述
 - ✓ 主体：提交的详细描述
 - ✓ 链接：与其他软件制品（如开发任务、缺陷）的链接关系

fix: couple of unit tests for IE9

Older IEs serialize html uppercased, but IE9 does not...

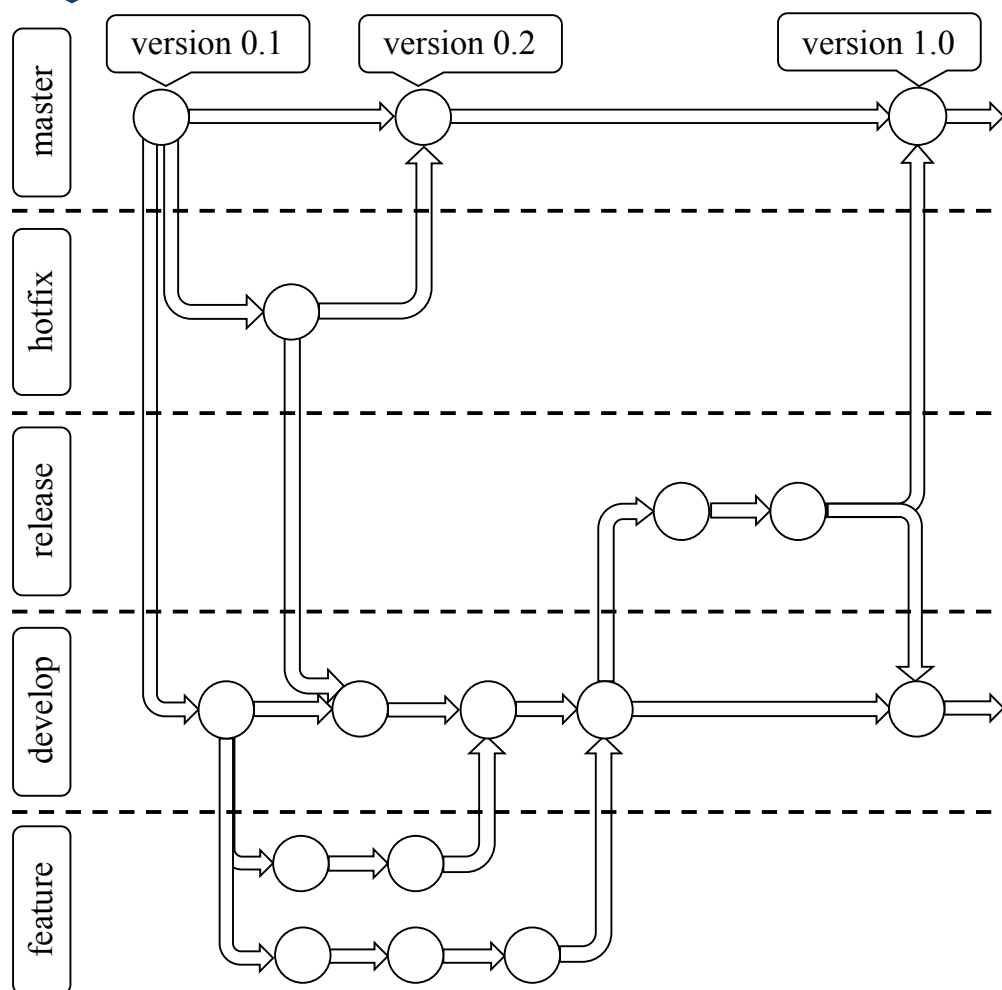
Would be better to expect case insensitive, unfortunately jasmine does not allow to use regexps for throw expectations.

Closes #392

代码分支管理

- 如果所有开发人员都在一个版本基础上进行开发，那么不同开发人员的提交会混杂在一起，版本会变得混乱，导致项目难以进行持续集成和发布
 - ✓ 由于开发人员持续面向不同任务进行开发，整体代码处于一种不稳定状态，难以持续集成和发布
 - ✓ 由于一个未开发完成的特性导致一个紧急的缺陷修复无法及时集成和发布
- 分支管理用于支持多个并行的互不干扰的分支

Git分支管理



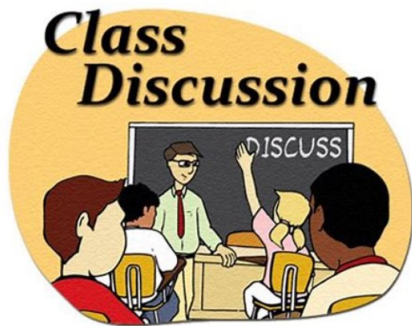
常用分支类型

- ✓ 主分支 (master) : 对应版本发布
- ✓ 开发分支 (develop) : 对应开发环境中的代码
- ✓ 特性分支 (feature) : 对应新功能
- ✓ 发布分支 (release) : 为版本发布做好准备
- ✓ 补丁分支 (hotfix) : 为修复缺陷

分支合并

- ✓ 发生合并冲突的文件中标记出不同分支中的内容
- ✓ 开发人员之间对合并冲突进行协商

课堂讨论：代码提交与分支管理



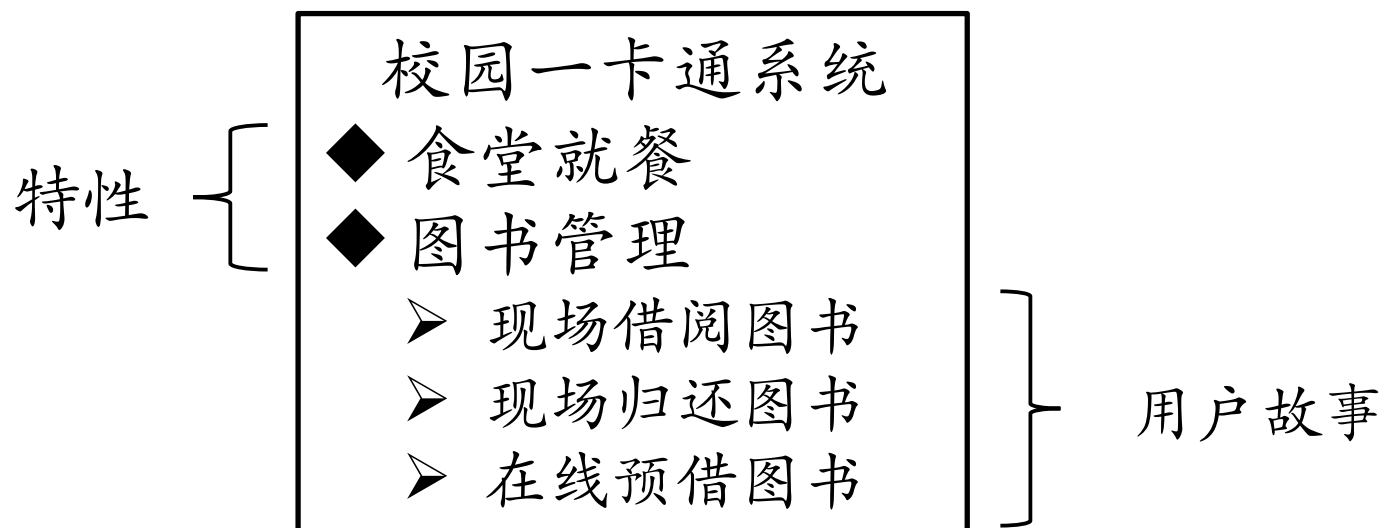
课堂讨论：在本学期的课程项目中，你们小组在代码提交和分支管理方面是否制定了一些规范，效果如何？

基线管理

- 一个分支被称为一条**代码线** (Codeline)
- 一条代码线的上级称为它的**基线** (Baseline)
- **主线** (Mainline) 是没有基线的代码线
- 版本管理就是管理代码线、基线和主线的过程
 - ✓ 主线管理版本发布
 - ✓ 代码线支持并行的独立开发，确保不同开发人员的修改不会彼此干涉
 - ✓ 基线可以重现上级代码线对应的系统版本，可以作为评估系统开发状态的依据

特性开发任务描述

- 特性是可以给客户带来价值的产品功能
- 一个特性可以分解成多个用户故事：从用户角度对产品功能的详细描述



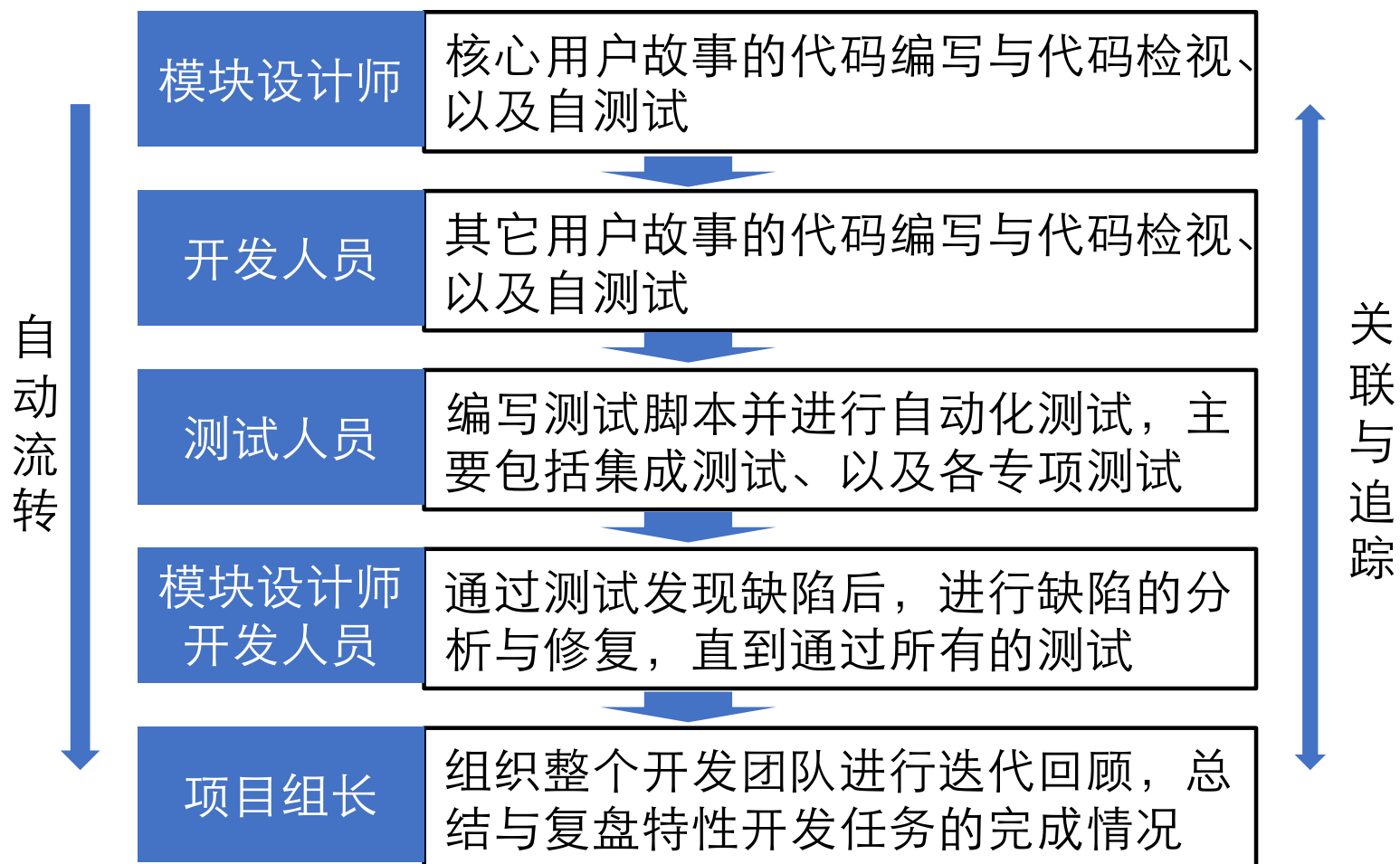
华为软件开发云中用户故事的基本信息-1

标题	对用户故事的简要描述
描述	对用户故事的详细描述
编号	用户故事的标识符
业务编号	所属特性的标识符
状态	用户故事的处理状态：新建、进行中、测试中、已解决、已关闭
模块	用户故事所属的产品模块
迭代	用户故事所处的迭代
特性组长	用户故事所属的特性的项目组长
开发状态	用户故事所处的开发状态：待启动、已启动、已完成

华为软件开发云中用户故事的基本信息-2

开发负责人	用户故事的开发负责人
开发开始时间与结束时间	用户故事开发的开始时间与结束时间
开发预估工作量	完成用户故事开发所预估的工作量
测试状态	用户故事所处的测试状态：启动、已启动、已完成
测试负责人	用户故事的测试负责人
测试开始时间与结束时间	用户故事测试的开始时间与结束时间
测试预估工作量	完成用户故事测试所预估的工作量

特性开发任务管理流程



特性开发任务管理流程

• 开发任务流程管理

- ✓ 分配负责开发任务的测试和开发人员
- ✓ 制定开发任务的日程规划
- ✓ 监控开发任务的进度
- ✓ 提供开发任务完成情况统计

• 交流与沟通

- ✓ 与开发人员讨论、协商和评审开发任务的解决方案
- ✓ 将交流记录以邮件的方式通知相关开发人员，加快沟通与处理速度
- ✓ 代码管理
- ✓ 关联开发任务与代码提交
- ✓ 便于进行针对开发任务的代码评审与责任追溯，实现代码到原始需求的反向追溯

变更管理

- 软件开发过程的变更几乎总是无法避免
 - ✓ 定制化软件的客户需求变化
 - ✓ 面向市场的产品发布计划因为市场或技术因素发生变化
- 开发人员不能随意地进行软件变更，以避免由此引发的技术或商业问题
 - ✓ 变更都会涉及一定的开发工作量
 - ✓ 对软件的稳定性造成影响
 - ✓ 可能涉及商业问题（例如所提出的变更是否符合与客户签订的合同或产品的商业策略）
- 需要一种规范、系统和可控的方式来管理软件变更流程，确保**规范性**和**可追踪性**

变更管理流程

• 提交变更请求

- ✓ 变更来源：是谁发起了变更，如客户、产品经理
- ✓ 变更原因：为什么需要这次变更，如项目进度有延迟
- ✓ 变更内容：对什么进行变更，如版本计划变更、版本号变更、用户故事变更等

• 变更决策

- ✓ 由CCB（变更控制委员会）执行，决定是否同意变更
- ✓ 讨论并评审变更的合理性、影响范围、实现工作量

• 变更的实施与跟踪

- ✓ 发起特性开发任务，或事务性任务

• 变更归档


- ✓ 记录变更从请求到实施与跟踪的整个流程
- ✓ 便于变更的问题追踪

缺陷修复过程管理

- 软件开发和使用过程中经常会发现缺陷
 - ✓ 开发人员或测试人员在开发或测试过程中发现的缺陷
 - ✓ 用户在使用过程中发现的缺陷
- 需要提交缺陷报告（问题单）并进行跟踪
 - ✓ 描述缺陷的现象、复现缺陷的步骤等
 - ✓ 评估缺陷的优先级和重要程度
 - ✓ 指派修复缺陷的负责人
 - ✓ 与相关开发人员讨论缺陷的修复方案
 - ✓ 跟踪缺陷的处理状态
 - ✓ 关联修复缺陷的代码提交
- 需要实现缺陷修复的有效管理和追踪，否则缺陷修复过程将变得混乱，例如
 - ✓ 缺陷的职责分配不清
 - ✓ 缺陷修复的进度无法监控等

缺陷描述

常用的缺陷追踪系统包括JIRA、Bugzilla等

 Hadoop Map/Reduce / MAPREDUCE-7329 2 of 5371

HadoopPipes task may fail when linux kernel version change from 3.x to 4.x

Export

Details

Type: Bug

Priority: Major

Affects Version/s: 2.6.0

Component/s: pipes

Labels: patch pull-request-available

Hadoop Flags: Reviewed

Status: RESOLVED

Resolution: Fixed

Fix Version/s: 3.4.0, 3.3.1

People

Assignee: chaoli

Reporter: chaoli

Votes: 0 Vote for this issue

Watchers: 3 Start watching this issue

Dates

Created: 15/Mar/21 06:42

Updated: 2 days ago

Resolved: 2 days ago

Time Tracking

Estimated: Not Specified

Remaining: 0h

Logged: 7h 20m

Description

Hadoop Pipes Ping implement has a bug. Recently, we upgrade linux kernel version from 3.x to 4.x. And we find hadoop pipe task exit with connect timeout which is implemented by PingThread in HadoopPipes.cc.

```
I0309 17:16:06.333662 91442 doc_index_and_profile_reducer.cc:138, Reduce] finish one ba
I0309 17:16:06.395869 91442 doc_index_and_profile_reducer.cc:138, Reduce] finish one ba
I0309 17:16:06.453284 91442 doc_index_and_profile_reducer.cc:138, Reduce] finish one ba
I0309 17:16:06.506747 91442 doc_index_and_profile_reducer.cc:138, Reduce] finish one ba
I0309 17:16:06.569262 91442 doc_index_and_profile_reducer.cc:138, Reduce] finish one ba
I0309 17:16:06.634552 91442 doc_index_and_profile_reducer.cc:138, Reduce] finish one ba
Hadoop Pipes Exception: in ping problem connecting command socket at port35713, error:
I0309 17:16:06.702098 91442 doc_index_and_profile_reducer.cc:138, Reduce] finish one ba
```

After a deep research, we finally find that current ping server won't accept ping client created socket, which may cause critical problem:

JIRA中的问题单示例

JIRA问题单中的信息-1

标题	对缺陷的简要描述
描述	<p>对缺陷的详细描述</p> <ul style="list-style-type: none">• 说明错误码来辅助开发人员分析、定位和修复缺陷• 详述发生缺陷的环境信息，是开发环境、测试环境还是生产环境• 列举软件栈信息，包括操作系统及其版本、数据库及其版本等• 说明缺陷是否可以复现并详述复现的步骤• 附上相关的测试脚本、补充截图、日志等信息
状态	新提交（New）、已分配（Assigned）、未解决（Reopened）、已解决（Resolved）、已验证（Verified）、已关闭（Closed）
优先级	Trivial、Minor、Major、Critical、Blocker

JIRA问题单中的信息-2

处理意见	已修复 (Fixed) 、不是问题 (Invalid) 、无法修复 (Wontfix) 、无法重现 (Worksforme) 、以后版本解决 (Later)
影响组件	缺陷所影响的软件组件
影响版本	缺陷所影响的软件组件版本号
修复版本	缺陷修复所在的软件组件版本号
负责人	负责修复缺陷的开发人员
评论	关于缺陷的评论和讨论，便于分析和评审缺陷的解决方案

缺陷追踪系统主要功能

- 缺陷修复流程管理

- ✓ 指定缺陷修复的优先级
- ✓ 分配负责修复缺陷的开发人员
- ✓ 监控缺陷修复的进度
- ✓ 提供缺陷修复的统计报表等

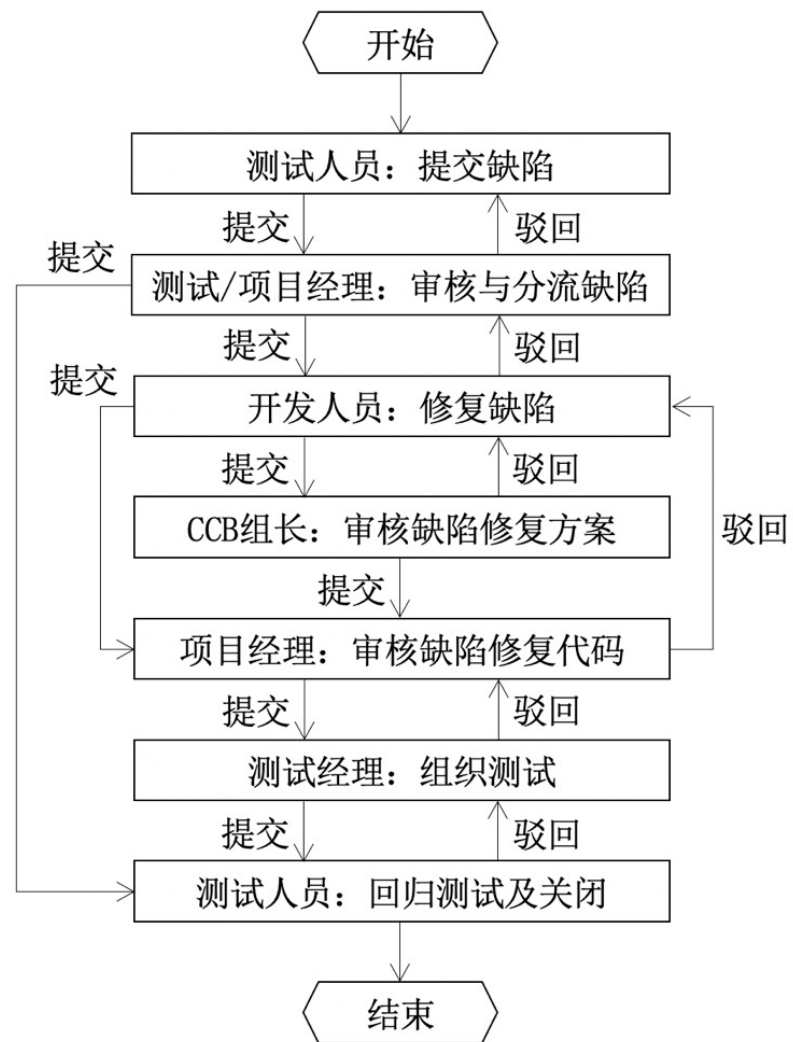
- 缺陷交流与沟通

- ✓ 与相关开发人员讨论、协商和评审缺陷修复的实现方案
- ✓ 将缺陷的讨论记录以邮件的方式通知相关开发人员，提高缺陷修复的沟通与处理效率

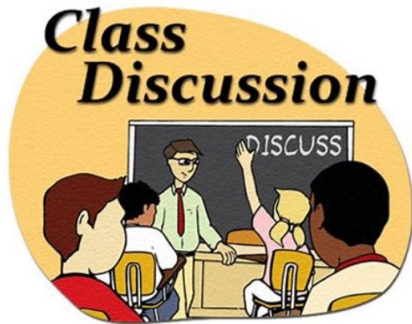
- 代码管理

- ✓ 通过在代码提交中引用缺陷标识符来关联缺陷与修复缺陷的代码提交
- ✓ 便于进行针对缺陷的代码评审与追溯

缺陷修复处理流程



课堂讨论：缺陷管理和处理流程



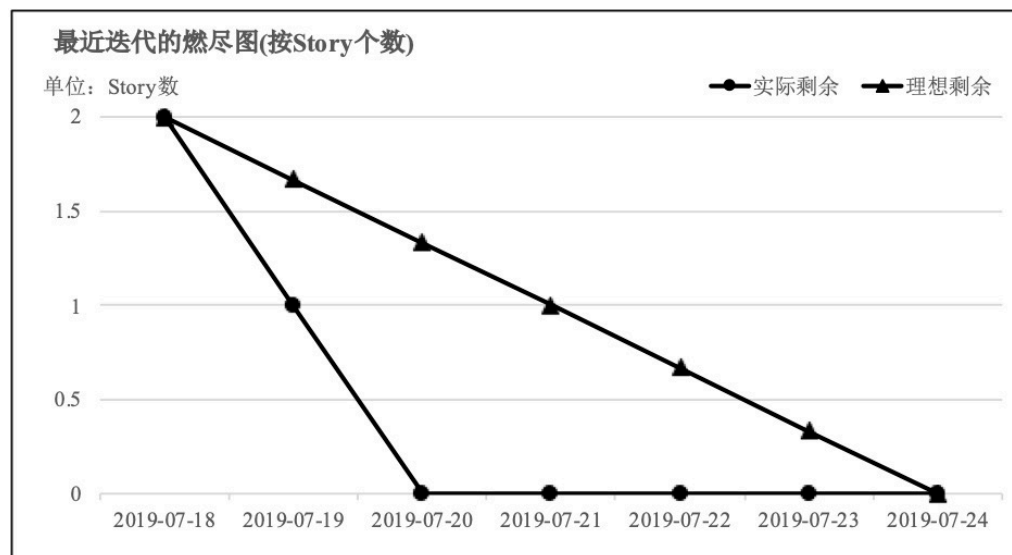
课堂讨论：在本学期的课程项目中，你们小组针对所发现的缺陷是如何进行管理和处理的？是否存在改进空间？

基于追踪与回溯的工作量与质量分析

- 目的：改进软件开发与流程管理过程，以提升软件开发质量
- 手段：基于特性开发任务以及缺陷的各类追踪关系进行多维度的工作量与质量分析
- 实践：对于优先级为Critical及以上的问题单，需要组织相关人员进行回溯分析，寻找流程中的问题与根因并制定对策

基于追踪的分析

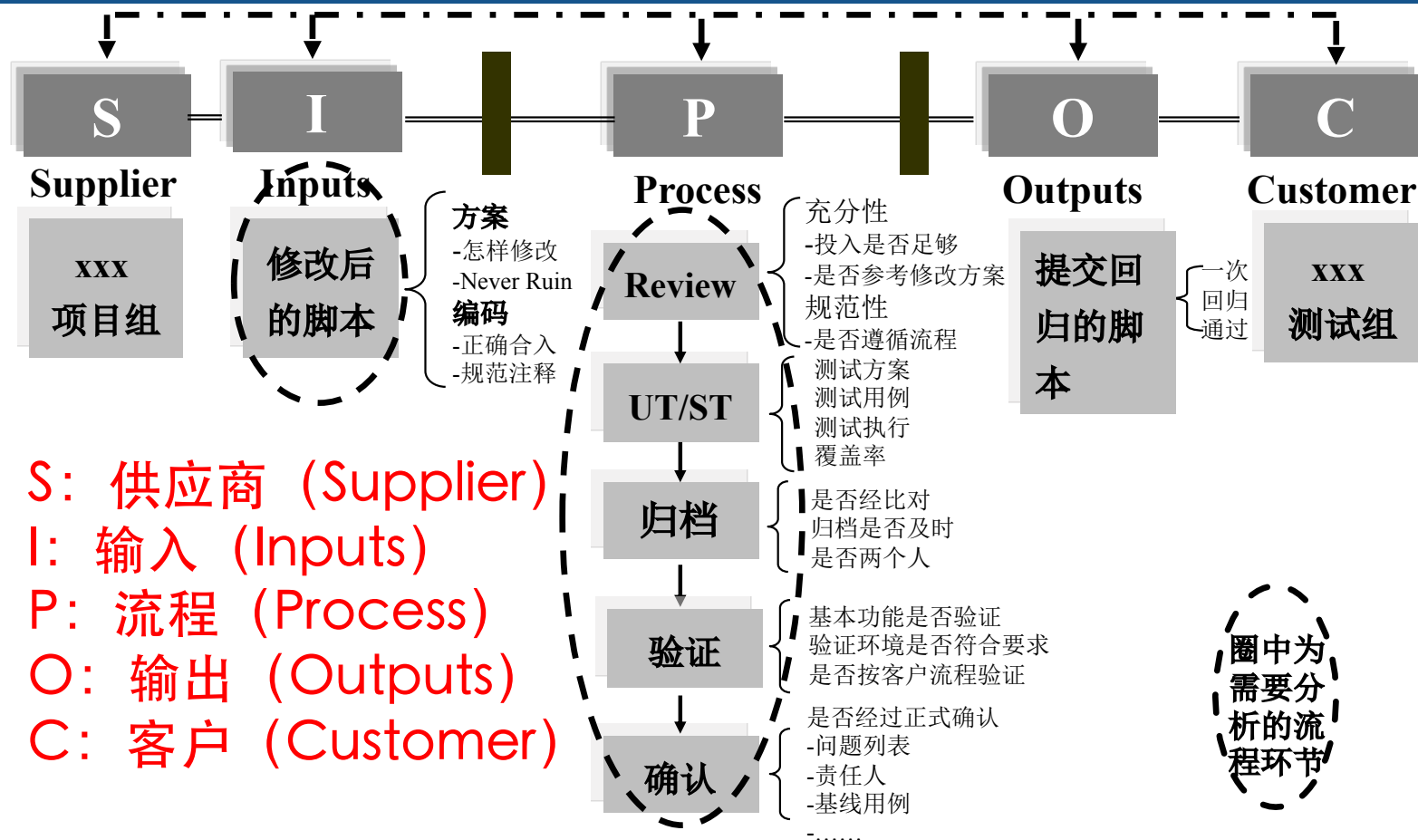
- 在单次迭代中基于预计完成的用户故事数与实际完成的用户故事数进行追踪
- 通过燃尽图分析随着时间的减少工作量的剩余情况，提供迭代进度的持续监控



基于回溯的分析

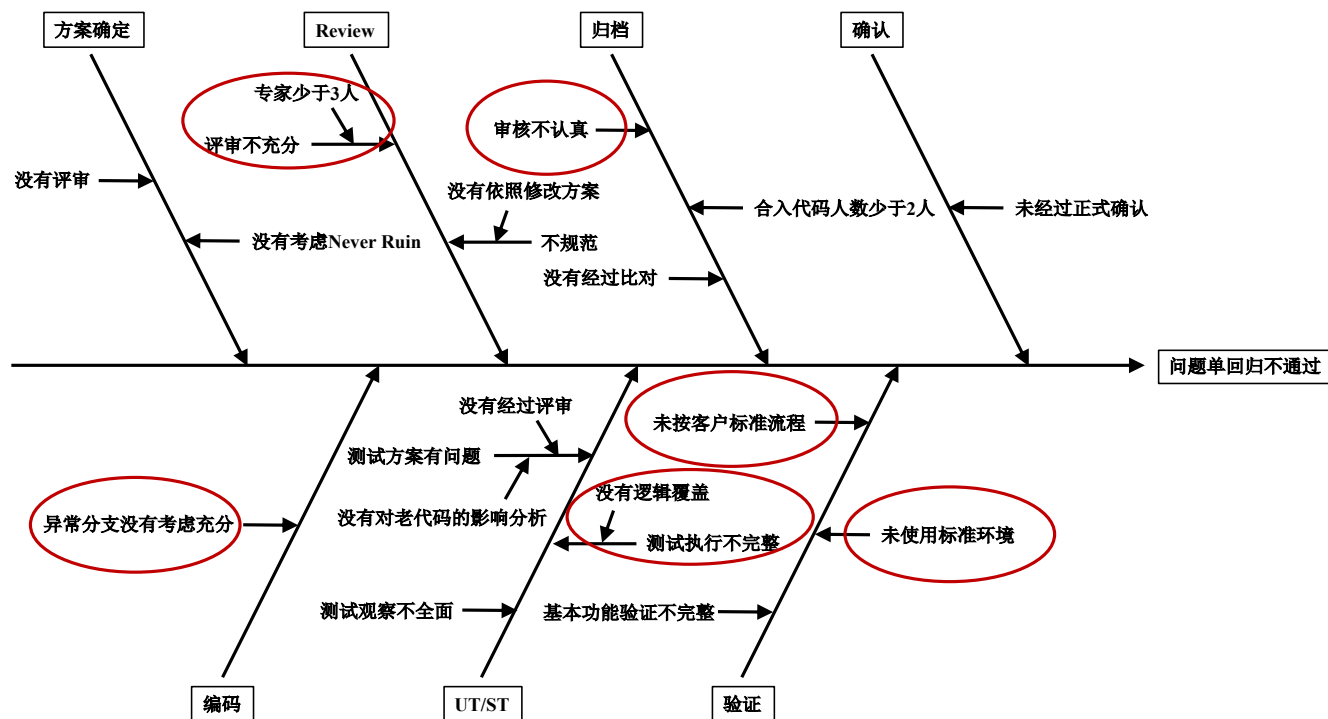
- 对于优先级为Critical及以上的问题单，需要召集相关人员进行回溯分析，从而进行针对性的改进
 - ✓ 描述问题：描述问题、定位结果、以及问题的后果
 - ✓ 成立回溯小组：所有相关人员参加才能全面地展开分析
 - ✓ 分析流程：沿着流程找原因，清晰完整地展现流程细节
 - ✓ 分析原因与确认要因：分析导致环节出现问题的具体原因
 - ✓ 寻找根因：寻找根本原因，从而制定相应的措施进行解决
 - ✓ 制定对策：制定相应的对策，指定责任人和完成时间，并落实到流程中，防止重犯同样的错误

SIPOC分析方法



沿着流程找原因，清晰完整地展现出流程细节，用简洁直观的形式表现出流程的结构概况，为后续的分析奠定基础

鱼骨图分析方法



针对一个问题（作为鱼头），由回溯小组经过充分讨论列明产生问题的大原因（鱼骨主干，如Review、UT/ST、归档等），从大原因继续反复论证，列举出每个大原因产生的中原因，中原因再论证小原因，如此一层层论证分析下去，直到找出所有可能的原因

5why分析方法

要因：为什么ST没有逻辑覆盖？
没有意识到。

为什么没有意识到？
问题单规范里没有明确要求。

为什么没有明确要求？
以为在其他流程中已经要求过，
不需要重复要求。

要因：为什么审核不充分？
员工技能不足。

为什么员工技能不足？
审核环节安排的是新员工。

为什么会安排新员工审核？
在流程中没有人员资格的明确要求。

对一个问题连续问5个“为什么”以追究其根本原因

实际使用时，不限定只问5个“为什么”，主要是必须找到根本原因为止

本章小结

- 软件开发伴随着持续的产品演化和代码修改，同时还需要支持大量开发人员的并行协同开发
 - ✓ 需要通过规范化的版本管理来支持版本变更与开发迭代
 - ✓ 需要对于由新特性开发和缺陷报告驱动的开发任务进行系统性的管理
- 版本管理
 - ✓ 规划产品迭代发布计划，对发布版本进行规范化版本命名
 - ✓ 使用版本控制系统来管理代码版本与代码分支
- 特性开发与缺陷修复任务
 - ✓ 管理特性开发、缺陷修复、变更管理等任务
 - ✓ 实现特性、缺陷、变更的有效追踪，提升开发效率和质量
- 基于追踪与回溯的工作量与质量分析
 - ✓ 基于各类追踪关系支持工作量与质量分析，改进流程管理
 - ✓ 对关键问题进行回溯分析，确认管理中的问题并制定对策

SOFT130006

软件工程

End

3. 版本与开发任务管理