第二节 敏捷开发与项目协作

詹文翰

zhanwenhan@163.com

课程安排

1. 软件工程与软件建模
2. 敏捷开发与项目协作
3. 开发环境、规范与测试
4. 需求、设计与程序框架
5. 打包解包及其关键技术
6. 压缩解压及其关键技术
7. 加密解密与系统集成
8. 软件发布与项目收尾

传统软件开发过程的问题

软件开发实质为一种手工过程。为保证项目成功，传统的软件过程模型均对软件开发过程进行过度规范（如能力成熟度模型CMMI），其强调：

1、严格的流程管理；
2、复杂的管理和开发工具；
3、详尽的文档；
4、和甲方的合同谈判；
5、严格的执行计划；

然而……

敏捷开发的诞生

《敏捷软件开发宣言》

我们一直在实践中探寻更好的软件开发方法，身体力行的同时也帮助他人。由此我们建立了如下价值观：

个体和互动	高于	流程和工具
工作的软件	高于	详尽的文档
客户合作	高于	合同谈判
响应变化	高于	遵循计划

也就是说，尽管右项有其价值，我们更重视左项的价值。

敏捷的由来：把大的完整的软件分成一个个小的功能（story），每个功能用较短的周期进行开发，并短周期经常性地交付可工作的软件。

敏捷开发工具/流派

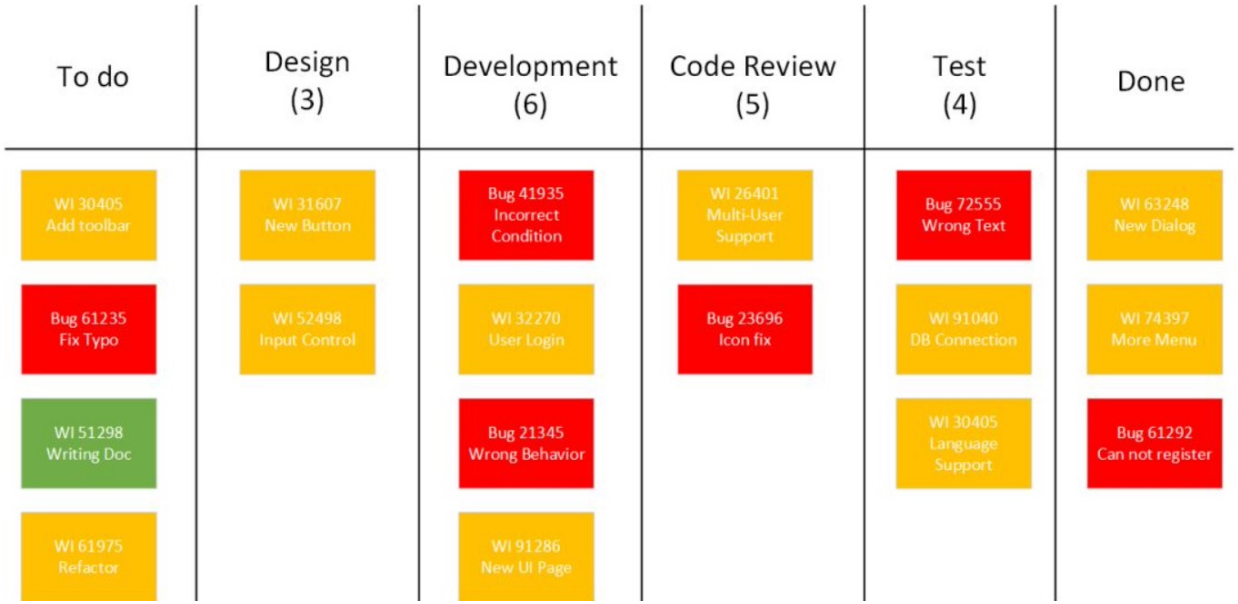
1、看板（KanBan）

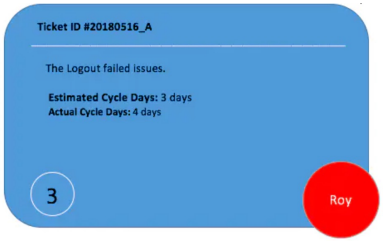
2、Scrum

3、极限编程（XP）

看板

用不同颜色的小纸片代表不同的任务(WI, Work Item)，小纸片上注明任务的各种属性。用一块大的白板(或黑板),在上面画出不同列，每一列代表任务的一个阶段，把小纸片放在相应的列下。





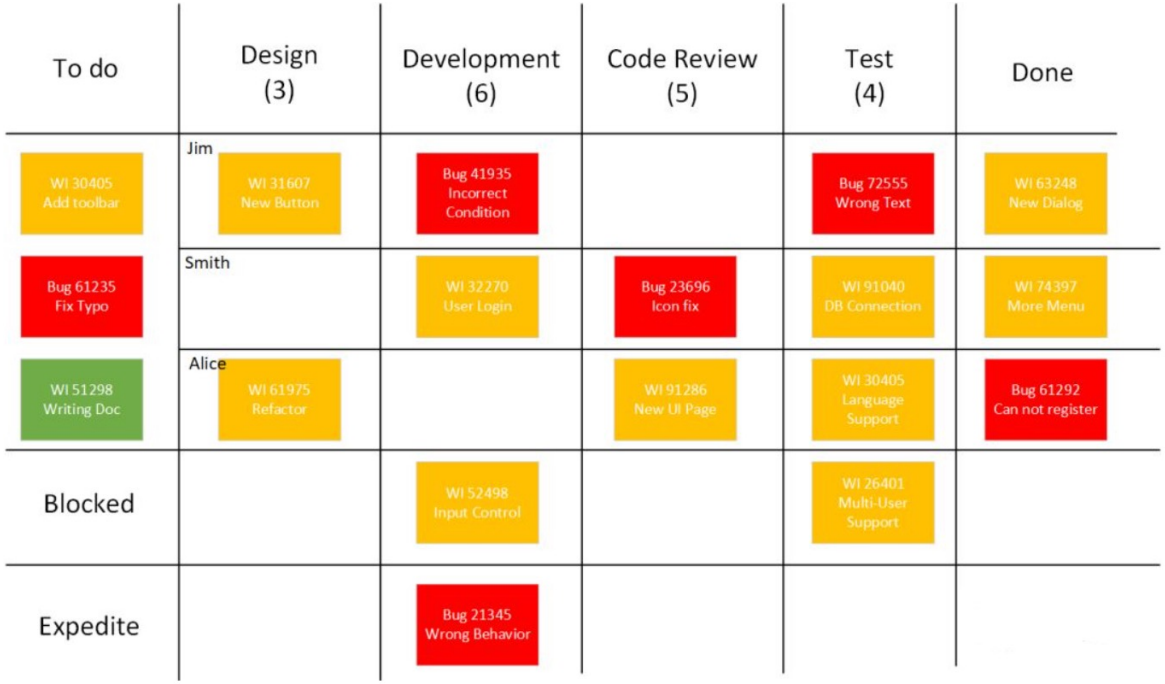
看板

其他概念

Cycle Time：一张代表任务的小纸片在白板上，从左到右，经历所有流程所花费的时间（越小越好）。

Throughput：在单位时间内，在白板上经历所有流程，完成任务的小纸片的数量，可以根据每个任务的大小加权计算（越高越好）。

Swimlane（泳道）：从另一个维度增强看板的功能。



看板

最佳实践

1、将看板摆在项目组中最显眼的位置。
2、为团队每日站会提供讨论依据。
3、可以方便的和其它敏捷开发方法做集成。

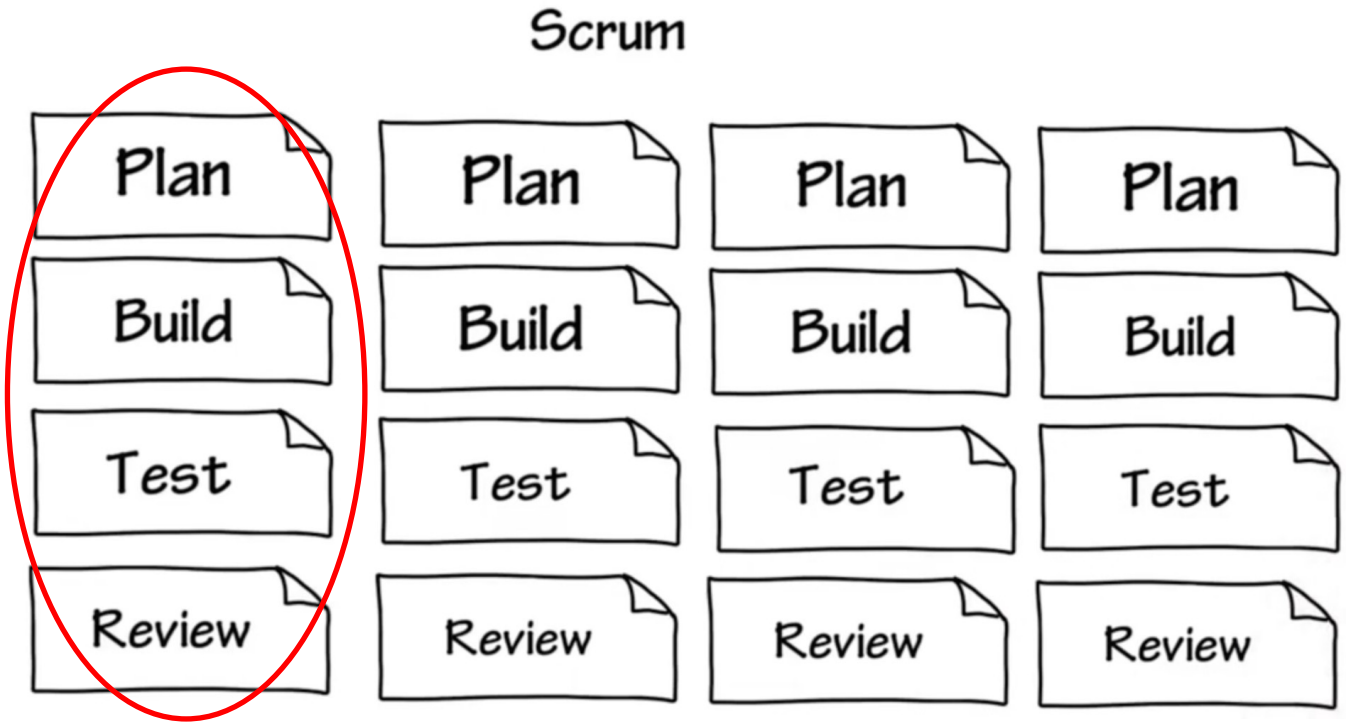
Scrum

最常用的敏捷开发方法之一。
核心思想：小步快跑，逐渐迭代。

Scrum

最常用的敏捷开发方法之一。
核心思想：小步快跑，逐渐迭代。

Sprint：Scrum把软件开发的过程从时间上分为不同的Sprint（冲刺），每个Sprint的时间可以是1到4周，通常是用2周，一个Sprint包含一个Sprint周期中需要完成的用户功能（user story）。

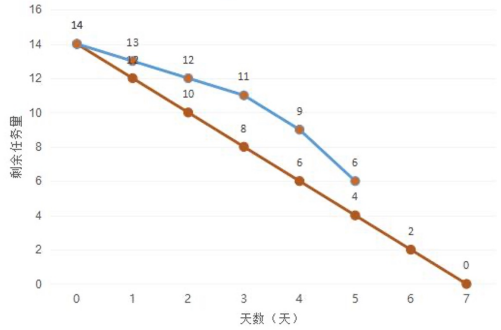


Scrum核心概念

三种角色：
Product Owner
Team
Scrum Master

三种工具：
User Story
Backlog
Burndown Chart

四种会议：
Sprint计划会
每日站会
Sprint评审会
Sprint回顾会



大牛对Scrum的介绍：https://www.bilibili.com/video/BV1rW411K7vD

Scrum注意事项

角色人选：
Product Owner
Scrum Master

会议时间：
Sprint计划会：2小时
每日站会：15分钟
Sprint评审会：1小时
Sprint回顾会：2小时

技术还债：定期重构
国内特殊环境：
- 站会内容
- PO和SM合一
- 越级管理

极限编程

另一个常用的敏捷开发方法之一。
核心思想：将目前公认的对编程有益的理念进一步强调并发挥到极致。

极限编程

另一个常用的敏捷开发方法之一。
核心思想：将目前公认的对编程有益的理念进一步强调并发挥到极致。

举例：

软件测试是重要的	→	先写测试用例再写代码（TDD）
代码审查是重要的	→	结对编程
用户反馈是重要的	→	要求客户直接参与到代码生产中

极限编程

极限编程的12条具体实践：

极限编程

极限编程的12条具体实践：

精细反馈：结对编程、计划博弈、测试驱动开发、现场客户

极限编程

极限编程的12条具体实践：

精细反馈：结对编程、计划博弈、测试驱动开发、现场客户

持续过程：持续集成、代码重构、小型发布

极限编程

极限编程的12条具体实践：

精细反馈：结对编程、计划博弈、测试驱动开发、现场客户

持续过程：持续集成、代码重构、小型发布

共享认知：编码标准、系统命名隐喻、代码集体共有、简单设计

极限编程

极限编程的12条具体实践：

精细反馈：结对编程、计划博弈、测试驱动开发、现场客户

持续过程：持续集成、代码重构、小型发布

共享认知：编码标准、系统命名隐喻、代码集体共有、简单设计

程序员福利：可持续步调

极限编程

极限编程进一步讨论：

1、结对编程（结对方式、优点、缺点）

2、结对编程和代码集体共有

3、简单设计与代码重构（技术还债）

4、可持续步调

选择合适的敏捷方式

并不需要完全拘泥于某一具体敏捷流派，适合自己团队的才是最好的。

Thoughtworks：50%Scrum+40%XP+10%KanBan

敏捷开发在实践中的阻碍

1、敏捷开发就是快速开发，进一步榨干程序员的价值。
2、团队敏捷，但文档，一个都不能少。
3、结对编程就是浪费资源。
4、甲乙方关系的不对等性。

软件项目管理

软件项目管理

软件项目管理的特殊性：
1、软件是纯知识产品，其开发进度和质量很难估计和度量，生产效率也难以预测和保证。
2、软件系统的复杂性也导致了开发过程中各种风险的难以预见和控制。

> 例：Windows这样的操作系统有1500万行以上的代码，同时有数千个程序员在进行开发，项目经理都有上百个。这样庞大的系统如果没有很好的管理，其软件质量是难以想象的。

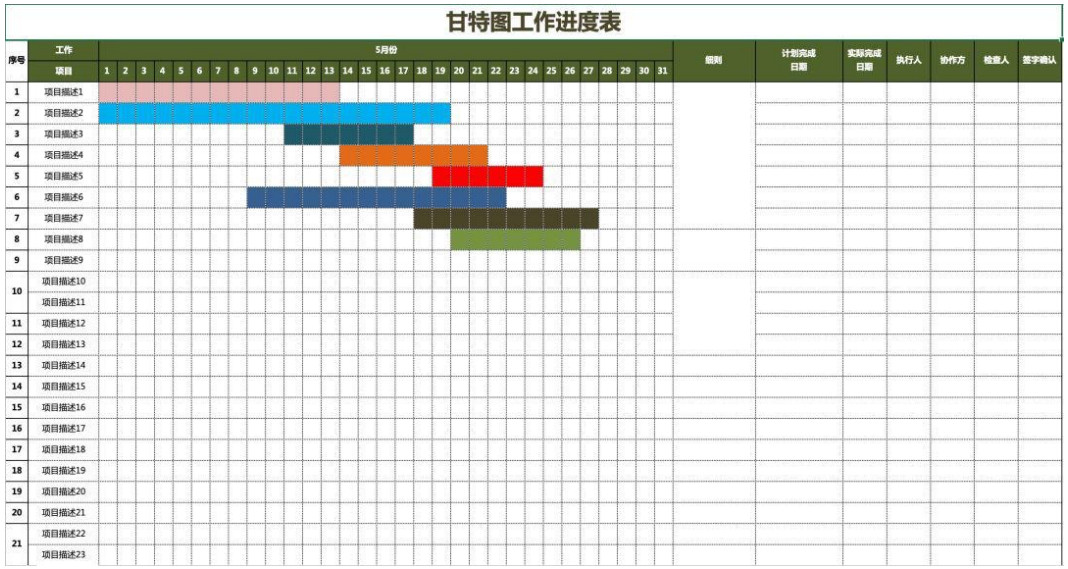
软件项目管理的内容主要包括：人员的组织与管理，软件度量，软件项目计划，软件任务跟踪，软件配置管理等。

强大的工具往往可以帮助我们事半功倍：
1、常用项目管理工具：看板、甘特图、燃尽图
2、常用项目管理软件：Project、Teambition、PingCode

制定和追踪项目计划

甘特图（Gantt chart）：通过条状图来显示项目任务和其他相关要素的内在关系随时间进展的情况，以提出者Henry Laurence Gantt的名字命名。

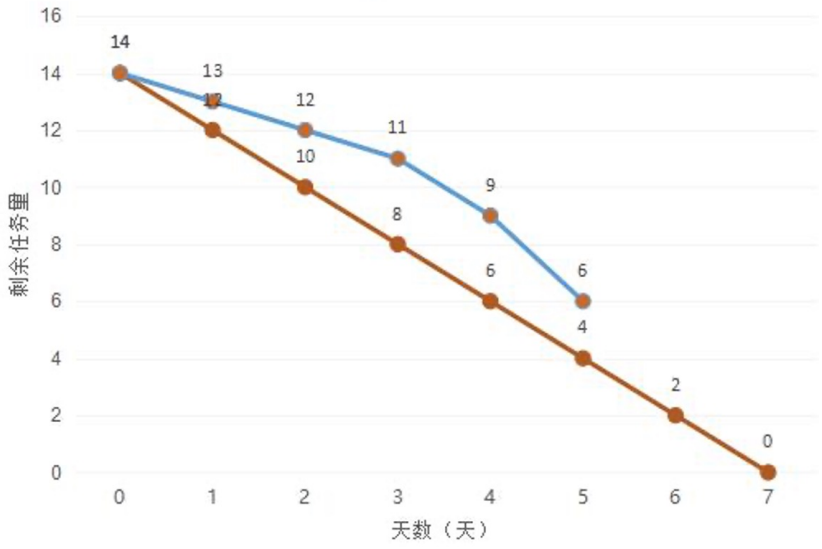
任务：是指软件开发中的一个具体工作。
资源：是指完成任务所需的人员、设备和原材料等。
工期：是完成某项任务所需活动工作时间的总长度，通常是从任务开始日期到完成日期的工作时间量。
里程碑：是一个工期为零，用于标识关键项目节点的重要日程事项。



制定和追踪项目计划

燃尽图（burn down chart）：是用于表示剩余工作量的工作图表，横轴表示时间，纵轴表示剩余工作量，可以直观的预测工作的完成进度，常用于敏捷开发当中。

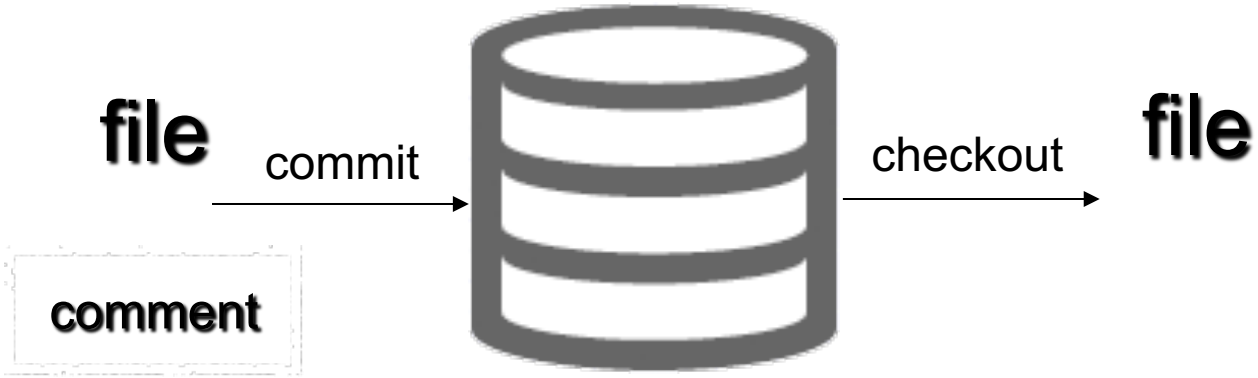
若实际工作线高于理想工作线时，则意味着剩余的工作量比原先预测的多，并且项目落后于计划。
若实际工作线低于理想工作线，则意味着剩余的工作量比原先预测的少，并且项目提前完成。



版本控制与代码协作

版本控制

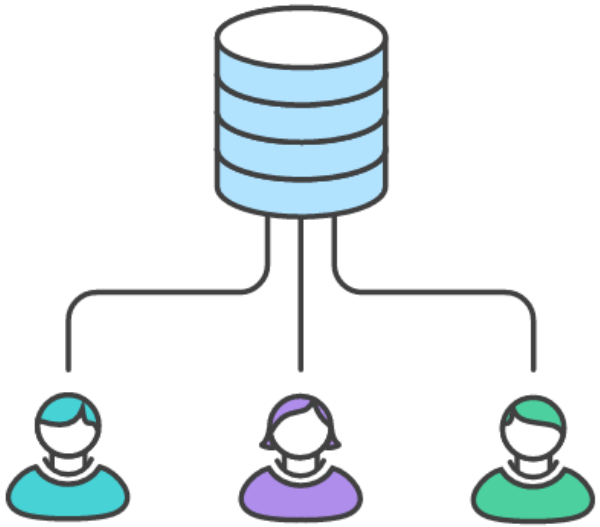
版本控制软件也被称为代码时光机。有了版本库，代码任何更改都会保存起来，并且我们可以对每一次更改提交添加评论（comment），说明更改的内容或目的。



版本库

版本控制

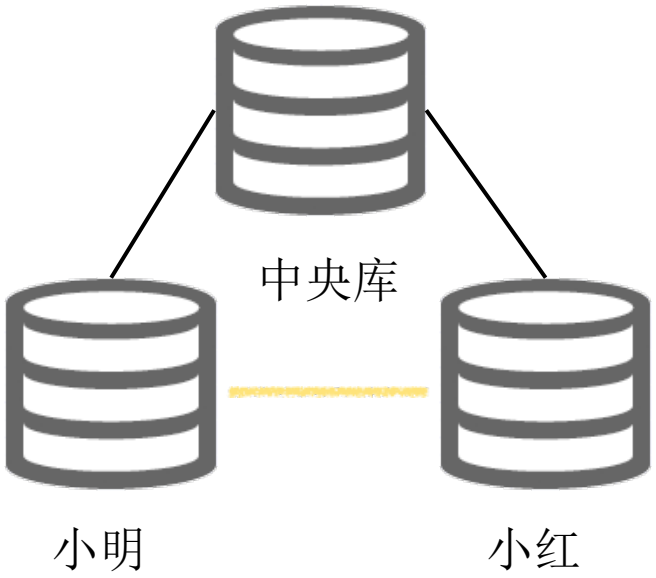
版本控制系统分类：



集中式：如SVN、CVS

做什么事都需要与中央库进行交互，包括查看日志。

缺点：**单点失效**

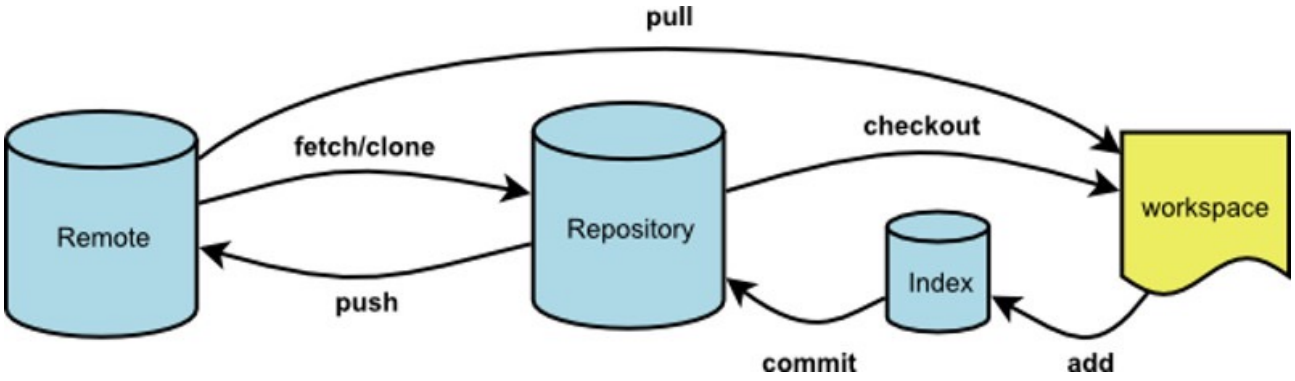


分布式：Git

只需要第一次与中央库进行交互（克隆版本库），之后只需要单机提交到本地版本库。在合适的时候，再与“中央库”进行合并。

版本控制软件——git

git init
git remote add origin git@github.com:帐号名/仓库名.git
git clone git@github.com:帐号名/仓库名.git
git add
git commit -m
git push origin master
git pull origin master
git fetch origin
git checkout
git rm
git branch
git status
git log
……



版本控制基本概念

版本

每一次正确的代码提交，会为版本库生成一个版本，所有版本可以基于依赖关系构成一条代码演进的时间线（树）。

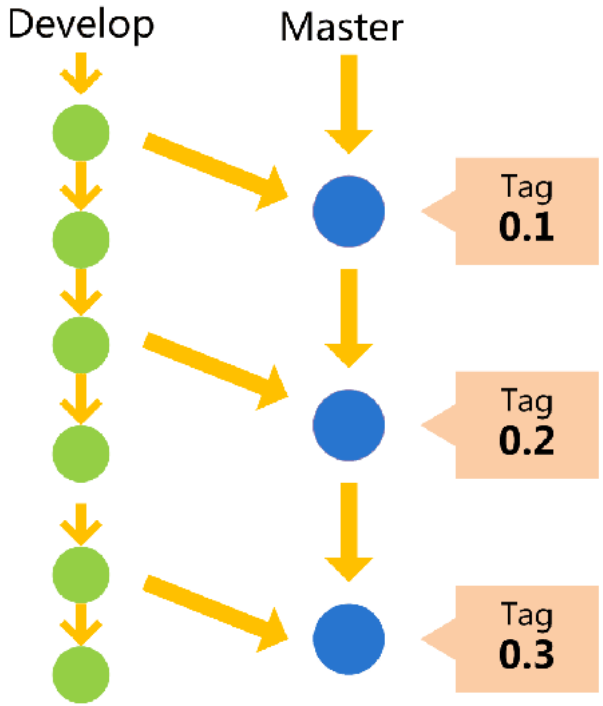
分支

使用分支意味着可以把你的工作从开发主线上分离开来，以免影响开发主线。分支可以根据不同的目的进行划分。

分支分类

1）以软件状态为导向

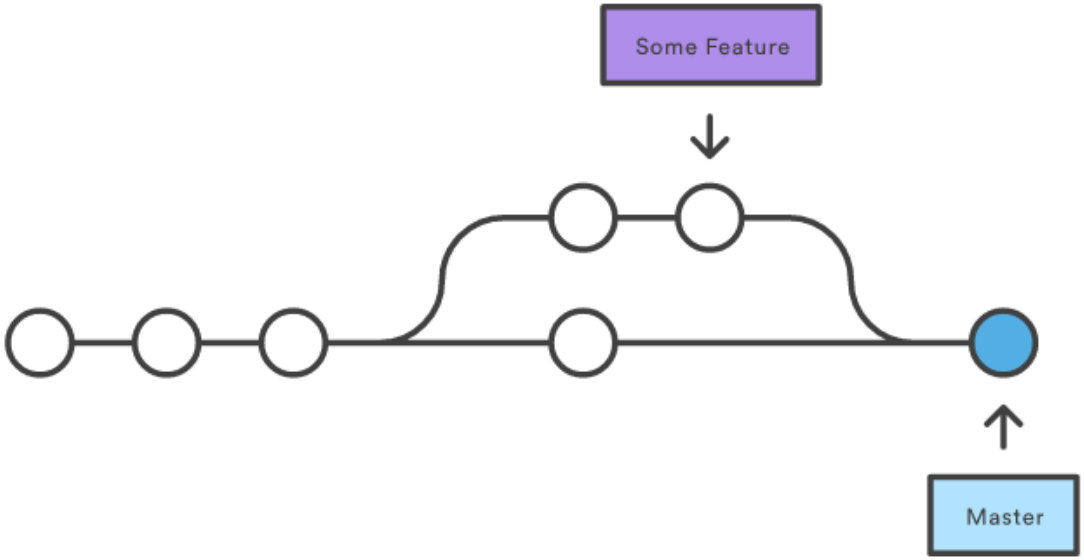
我们可以将分支设置为多个：如用于给用户提供稳定的软件版本master主分支；用于项目开发的develop分支；用于软件发布准备的release分支；用于系统维护、修复bug的hotfix分支等。



分支分类

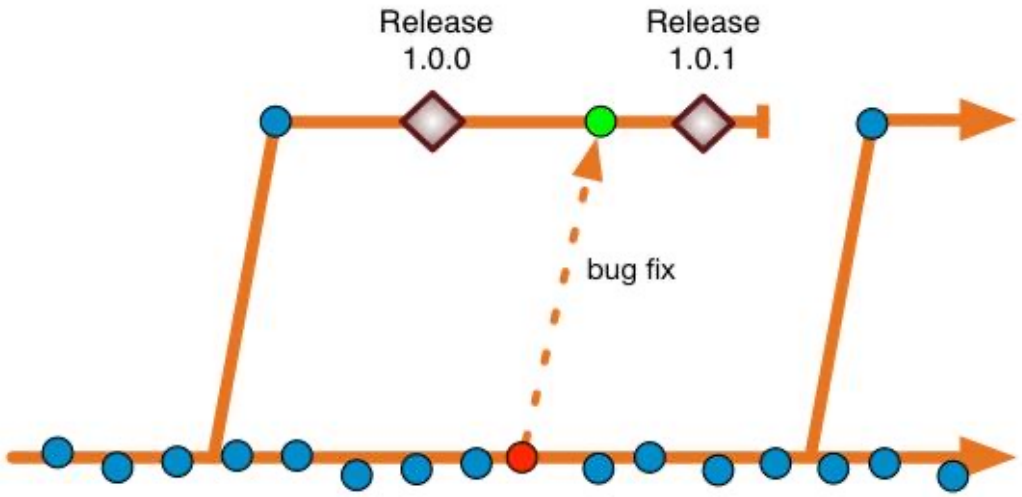
2）以功能开发为导向

基于功能增加来完成迭代的项目，产生多个分支的目的是为了单独开发某一功能。最终为了实现统一的整体，需要在功能开发完成之后进行分支合并（merge）。



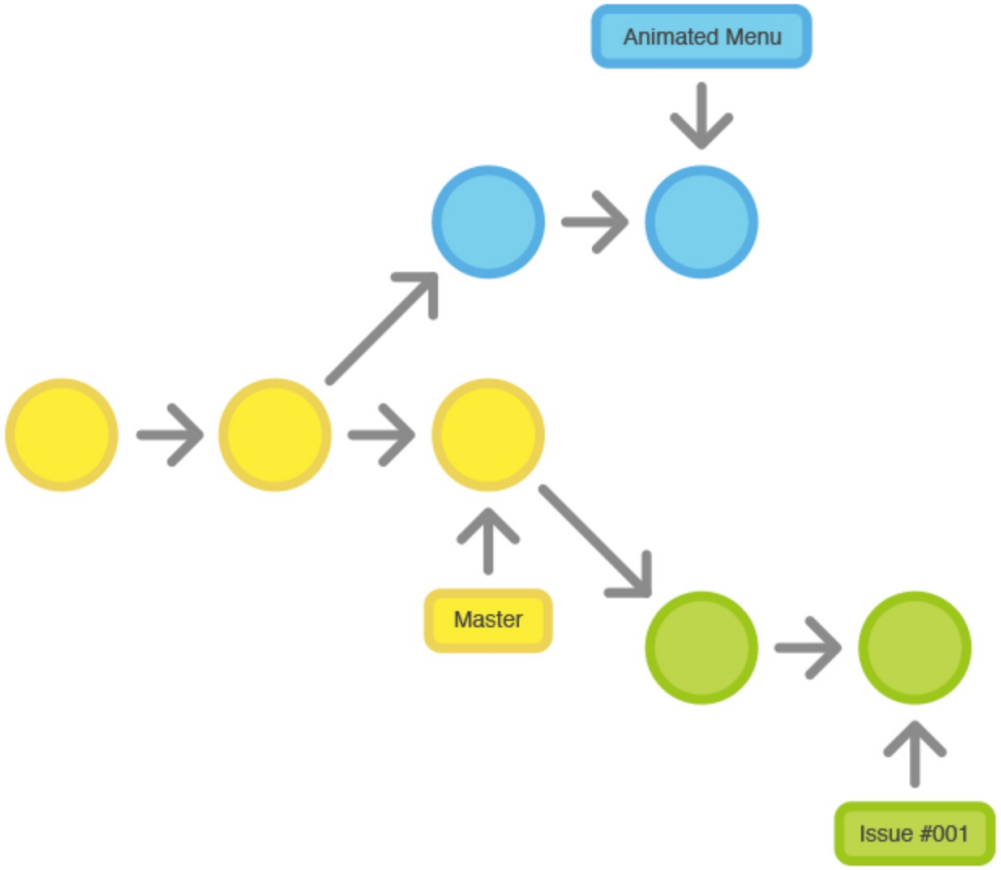
软件协作工作流

1、集中式工作流（trunck工作流）



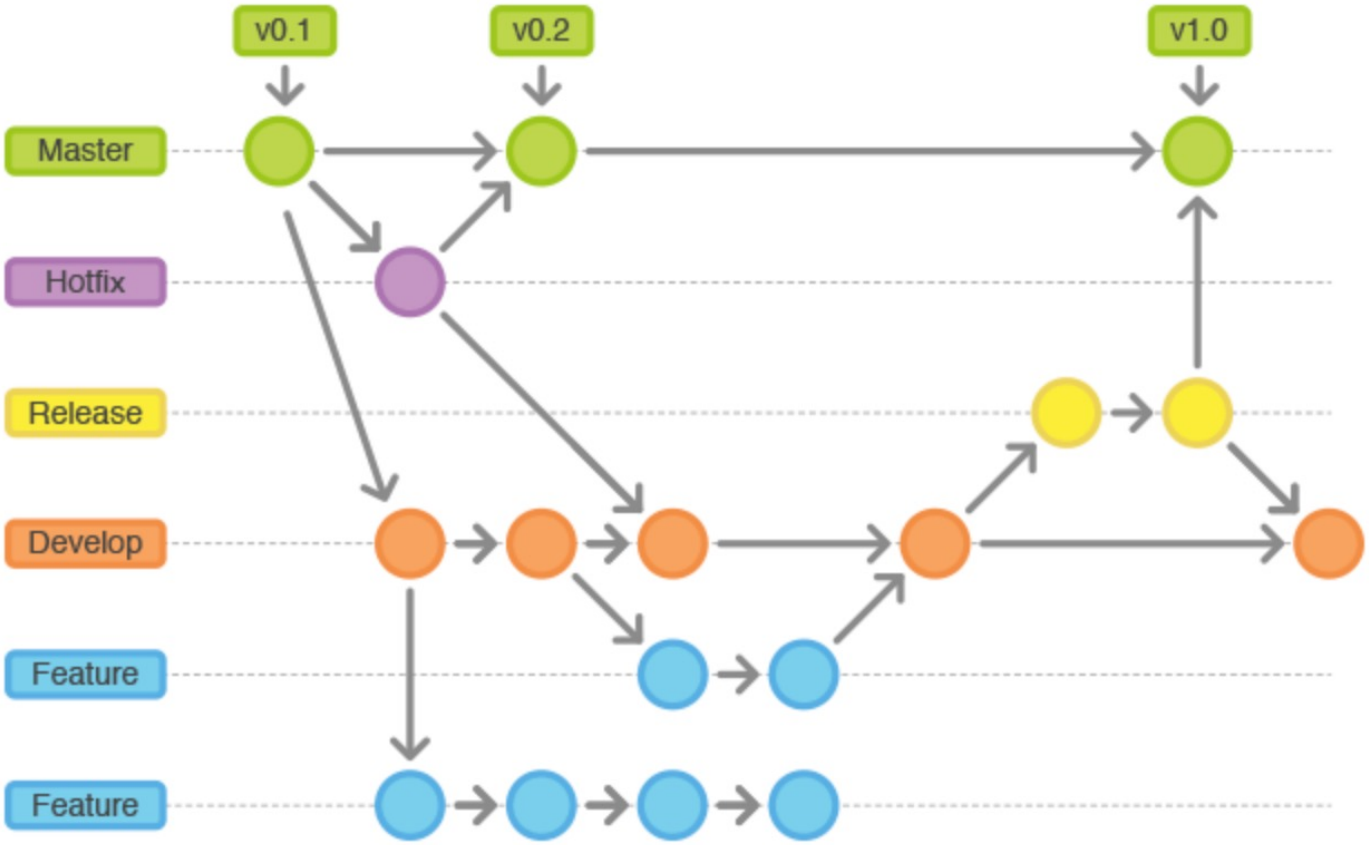
软件协作工作流

2、功能分支工作流



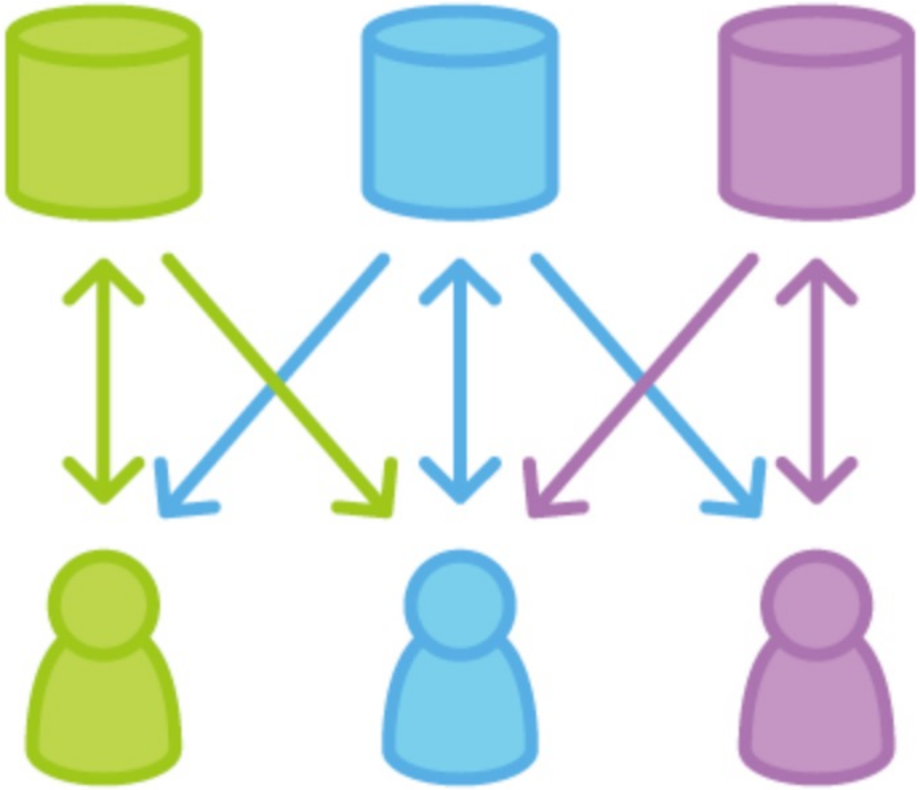
软件协作工作流

3、gitflow工作流



软件协作工作流

4、forking工作流



实验环节

完成本课程实验三：

- 实验内容见《实验指导书3》
- 要求了解Teambition或类似项目管理软件的用法
- 要求掌握版本管理软件git及远程仓库github/gitee的用法
- 不需要提交实验报告

项目要求：

- （鼓励）使用Teambition或类似软件进行项目协作管理
- 采用Github/gitee创建在线版本库并使用git进行项目开发
- 选定一个合适的软件协作工作流进行协作开发