敏捷工程实践

目录

敏捷工程实践技术

- 用户故事
- 结对编程
- TDD（测试驱动开发）
- 持续集成
- CodeReview
- 发布规则

敏捷工程实践：用户故事（user story）

什么是用户故事

- 用户故事是一个用来确认用户和用户需求的简短描述。
- 典型的描述句式为：作为一个XXX客用，我需要XXX功能，能够带来XXX好处。

- 用户故事是站在用户角度描述需求的一种方式；
- 每个用户故事须有对应的验收测试用例；
- 用户故事是分层分级的，在使用过程中逐步分解细化；

敏捷工程实践：用户故事（user story）

用户故事的关键要点

- I – Independent，可独立交付给客户
- N – Negotiable，便于与客户交流
- V - Valuable ，对客户有价值
- E - Estimable ，能估计出工作量
- S - Small ，分解到最底层的用户故事粒度尽量小，至少在一个迭代中能完成
- T - Testable，可测试

用户故事的好处

- 站在用户视角便于和客户交流，准确描述客户需求；
- 用户故事可独立交付单元、规模小，适于迭代开发，以获得用户快速反馈；
- 用户故事强调编写验收测试用例作为验收标准，能促使需求分析人员准确把握需求，牵引开发人员避免过度设计。

用户故事便于团队站在用户角度分解细化需求并制定验收标准

用户故事（user story）

故事样例

初始需求：1.作为网络规划人员，我想要配置一个媒体网关，因为想要增加网络容量和服务

初次分解：1.1作为网络规划人员，我想把媒体网关参数上传到管理系统

1.2作为网络规划人员，我想从管理系统下载媒体网关参数

再次分解：1.2.1作为网络规划人员，我想用文件方式从管理系统下载媒体网关参数

用例：用户在管理系统上选择以文件方式下载媒体网关参数，执行成功后，检查文件是否正确下载到本地且内容正确

1.2.2作为网络规划人员，我想用MML结构方式从管理系统下载媒体网关的参数

用例:.....

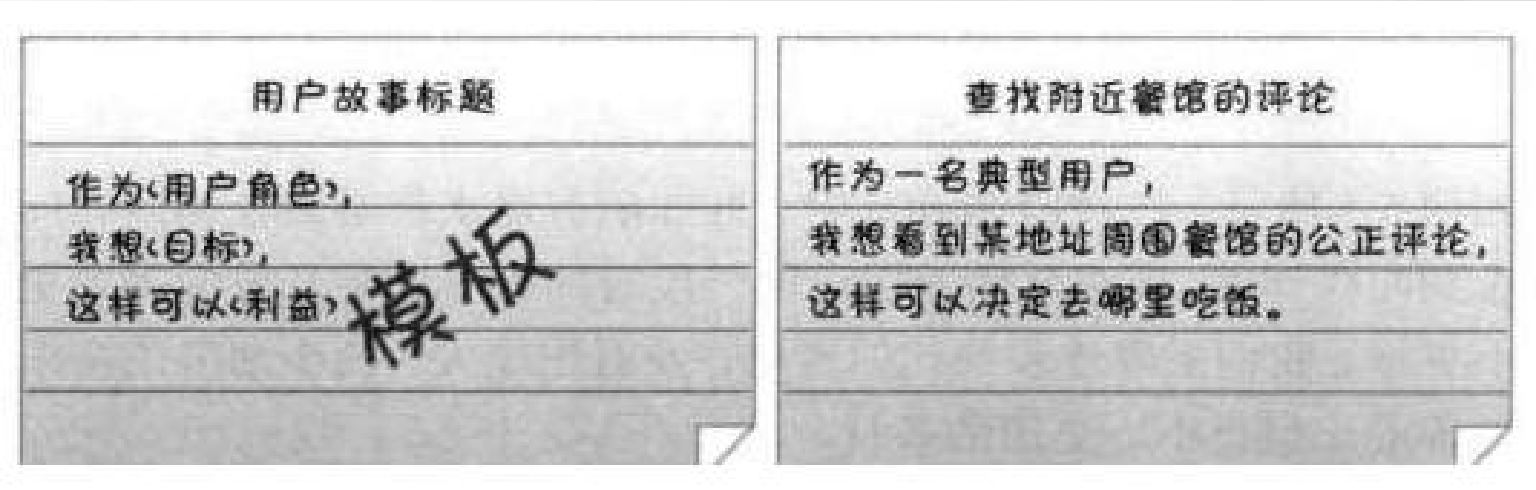
用户故事便于团队站在用户角度分解细化需求并制定验收标准

用户故事特征

- 体现客户（用户）价值，轻量级的点位符
- 遵循3C原则
  - 卡片（Card）：作为XX，我希望XXX，这样可以XXX
  - 对话（Conversation）：不描述到细节，由团队通过持续对话细化，激发大家的深度理解
  - 确认（Confirmation）：有明确的验收标准

卡片（Card）

- 用户故事描述的传统形式是手工书写的用户故事卡，卡片上应该只有几句话来捕获需求的精髓或目的。
- 通常的格式：作为一个<角色>，我想要<功能>，以便于　<商业价值>



会话（Conversation）

- 会话指的是卡片上所记录的用户故事是可以进行讨论和细化的，它包括利益相关人（客户/用户）、产品负责人及开发团队之间进行更细化地讨论用户故事的可行性。用户故事经过会话确认后，才能正式进入开发阶段（用户故事实现）。
- 敏捷开发的流程完整体现了用户故事（需求）的流转过程。

用户故事的流转过程（以Scrum为例）

- 1.产品负责人负责整理user story，形成product backlog。 —— **用户故事整理**
- 2.发布计划会议：product owner负责讲解user story，对其进行估算和排序，会议的产出就是制定出这一期迭代要完成的story列表，sprint backlog。 —— **用户故事确认**
- 3.迭代计划会议：项目团队对每一个story进行任务分解，分解的标准是完成该story的所有任务，终每个任务都有明确的负责人，并完成工时的初估计。 —— **用户故事分解**
- 4.每日立会：每天scrum master召集站立会议，团队成员回答昨天做了什么今天计划做什么，有什么问题。 —— **用户故事实现**
- 5.演示会议：迭代结束之后，召开演示会议，相关人员都受邀参加，团队负责向大家展示本次迭代取得的成果。期间大家的反馈记录下来，由po整理，形成新的story。 —— **用户故事的二次整理**

确认（Confirmation）

- 用户故事确认可以理解为对用户故事是否达到验收标准的检测。用户故事需要一系列的验收测试用以保证故事功能的完成及软件按照预期运行。同时要保证这个用户故事最后实现是可以带来商业价值的。
- 用户故事的确认由测试人员完成。测试人员在测试版本所关联的用例列表里执行用例，完成测试，然后生成测试报告。
- 测试报告是对用户故事实现程度的最直接体现。如果一个用例执行失败，可以直接由这个测试用例创建一个Bug，由开发人员进行二次开发和修复，直到测试通过。

用户故事大小级别

- 史诗故事（1-2个月）
- 特性故事（1-2周）
- 冲刺故事（1-2天）
- 任务（几小时）：可分工执行

用户故事INVEST标准

- 独立性（Independent）：故事之间松耦合，具有独立性，不应该依赖于其他的用户故事。
- 可协商（Negotiable）：开始仅用于做占位符，逐步细化。
- 有价值（Valuable）：用户故事对于最终的用户是有价值的，因此应该站在用户的角度去编写。
- 可估算（Estimatable）：设计、开发、测试团队可估算工作量和成本。（不可估算的原因：太大需要分解，或信息不全需要进一步探索）
- 短小（Small）：故事应该尽量的短小（如两周冲刺，故事一般是2天以内的）
- 可测试（Test）：有相应测试验收标准。

用户故事约束（验收条件）

- 作为用户故事的约束体现（Card背面）

上传文件

作为一个维基用户，
我想上传一个文件
到维基，这样我可
以和同事们分享。

满意条件

验证txt和doc 文件
验证jpg,gif和png文件
验证小于等于1GB的mp4文件
验证无数字版权管理限制的文件

非功能性需求如何表达？

国际化

作为用户，我想要可以支持英语、一种罗曼语与一门复杂语言的接口，这样可以有很高统计概率应对需要支持的所有30种语言。

网络浏览器支持

系统必须支持IE8、IE9、FireFox6、FireFox7、Safari5和Chrome15

知识获取性故事如何表达？

- 一般偏重后台实现，例如：原型、概念验证、学习、探针等
- 必须向PO说明价值，并且不宜出现太多

迁移到新版Oracle
作为开发人员，我想迁移系统到最新版Oracle数据库管理系统上去，这样可以避免在即将退役的Oracle版本上运营。

自动构建
作为开发人员，我想让构建在我提交代码时自动运行，这样可以避免在引入回归错误的时候就发现它。

如何收集用户故事

- 用户故事写作研讨会，产生首批用户故事
  - 参与者：PO、SM、Team、内部干系人、用户
  - 方法：头脑风暴、人物角色、思维导图、故事地图
  - 时间：几小时到几天
- 故事地图
  - 史诗故事按时间流横向排开
  - 纵向按优先级排序

敏捷工程实践：结对编程

什么是结对编程

- 两位程序员在一台电脑前工作，一个负责敲入代码，而另外一个实时检视每一行敲入的代码；
- 操作键盘和鼠标的程序员被称为“驾驶员”，负责实时评审和协助的程序员被称为“领航员”；
- 领航员检视的同时还必须负责考虑下一步的工作方向，比如可能出现的问题以及改进等。

结对编程提高代码质量和工作效率

结对编程的关键要点

- 程序员应经常性地在“驾驶员”和“领航员”间切换，保持成员间平等协商和相互理解，避免出现一个角色支配另一个角色的现象；
- 开始新Story开发的时候即可变换搭档，以增进知识传播；
- 培养团队成员积极、协作的心态能够增进结对编程效果；
- 实施初期需要精心辅导，帮助团队成员克服个性冲突和习惯差异。

结对编程的好处

- 有助于提升代码设计质量；
- 研究表明结对生产率比两个单人总和低 15%，但缺陷数少 15%，考虑修改缺陷工作量和时间都比初始编程大几倍，所以结对编程总体效率更高；
- 结对编程能够大幅促进团队能力提升和知识传播。

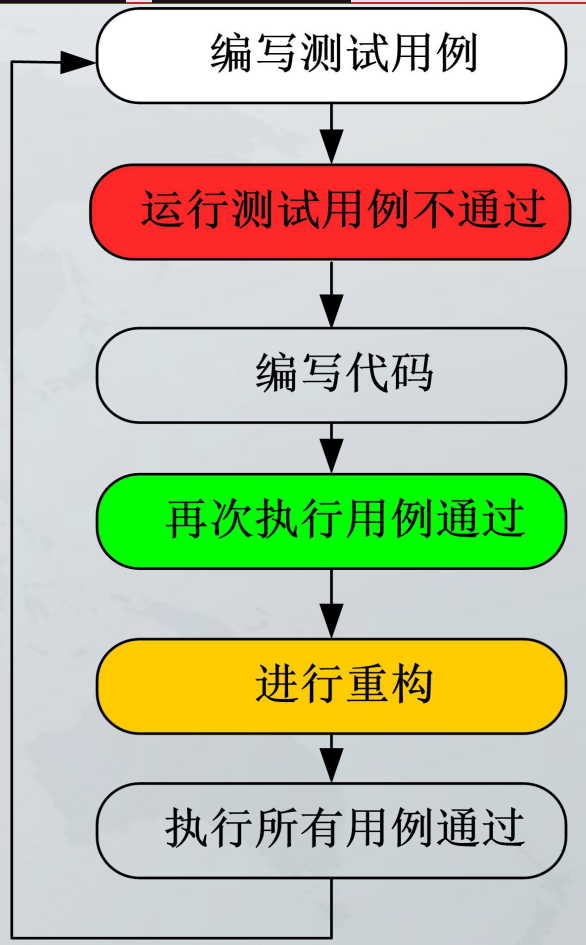
敏捷工程实践：测试驱动开发（TDD）

什么是测试驱动开发

- TDD以测试作为编程的中心，它要求在编写任何代码之前，首先编写定义代码功能的测试用例，编写的代码要通过用例，并不断进行重构优化；
- TDD要求测试可以完全自动化运行。

测试驱动开发的好处

- 和代码同步增长的自动化测试用例，能为代码构筑安全网，保证代码重构的质量；
- TDD有助于开发人员优化代码设计，提高代码可测试性。



测试驱动开发的关键要点

- 测试代码和源代码一样都需要简洁，可读性好；
- 测试用例的设计要保证完备，覆盖被测单元的所有功能；
- 每个测试用例尽量保持独立，减少依赖，提高用例的可维护性；
- 当功能单元较大时，为降低难度，可分解为多个更小的功能单元，并逐一用 TDD 实现。

测试驱动开发保证代码整洁可用（Clean code that works）

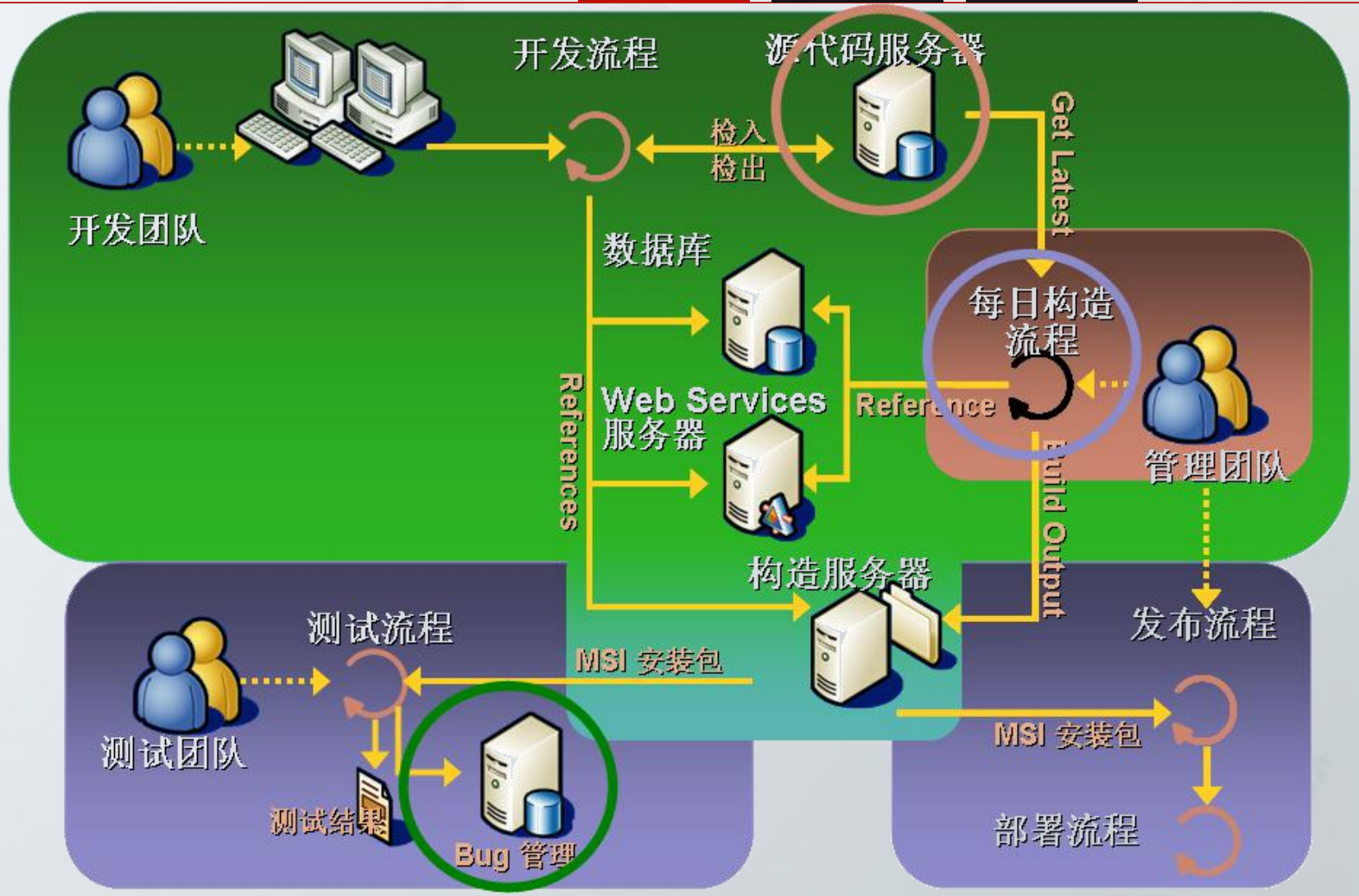
敏捷工程实践：持续集成(CI)

什么是持续集成

- 持续集成（CI）是一项软件开发实践，其中团队的成员经常集成他们的工作，通常每人每天至少集成一次，每次集成通过自动化构建完成。

持续集成的好处

- 大幅缩短反馈周期，实时反映产品真实质量状态；
- 缺陷在引入的当天就被发现并解决,降低缺陷修改成本；
- 将集成工作分散在平时，通过每天生成可部署的软件；避免产品最终集成时爆发大量问题。

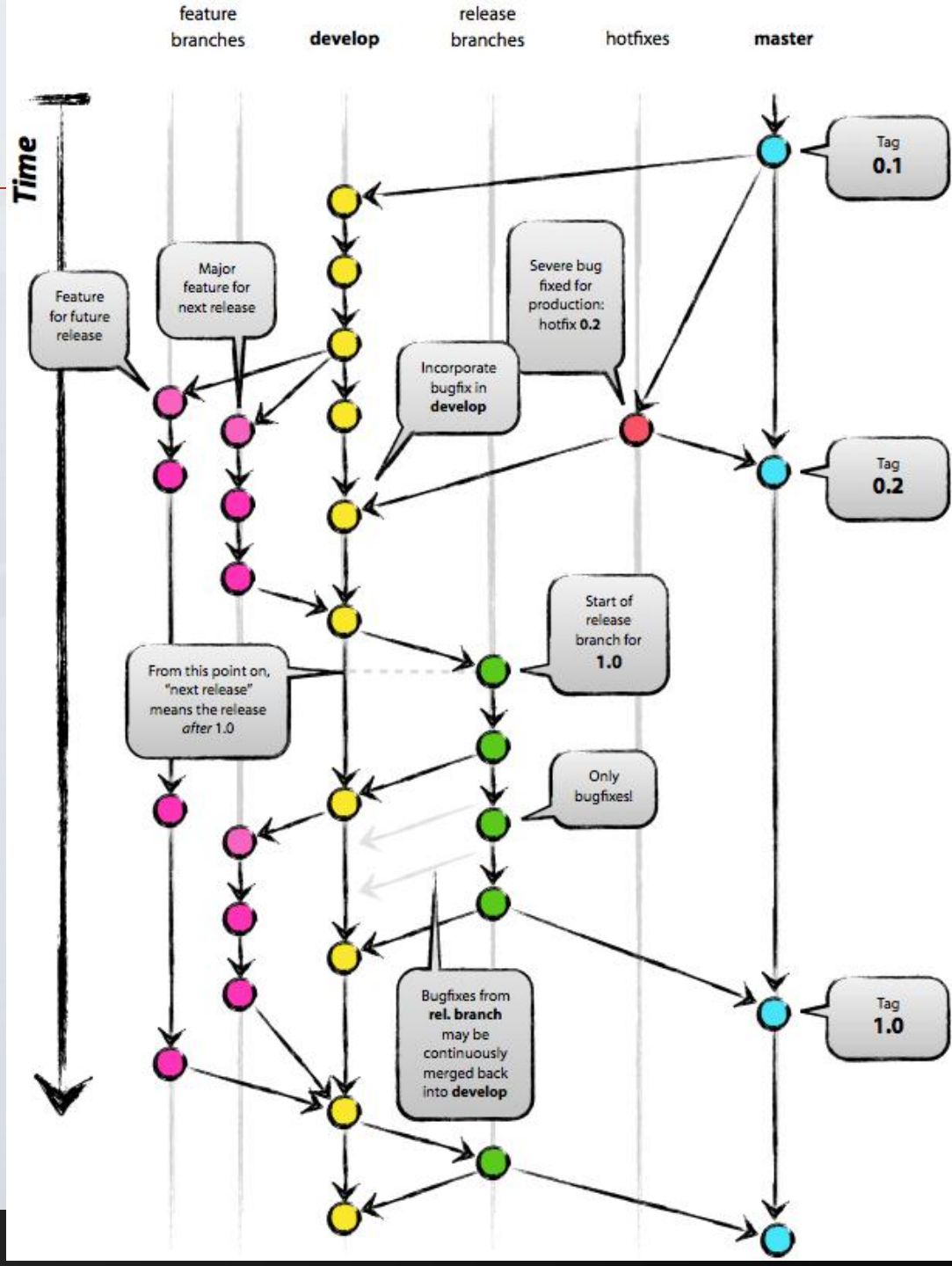


持续集成提供产品质量的快速反馈，保证随时拥有可工作的软件

敏捷工程实践：持续集成(CI)

持续集成的关键要点

- 持续集成强调 “快速”和“反馈” ，要求完成一次系统集成的时间尽量短，并提供完备且有效的反馈信息；
- 自动化测试用例的完备性和有效性是持续集成质量保障；
- 修复失败的构建是团队最高优先级的任务；
- 开发人员须先在本地构建成功，才可提交代码到配置库 ；
- 持续集成的状态必须实时可视化显示给所有人；
- 大系统持续集成需分层分级，建立各层次统一的测试策略。

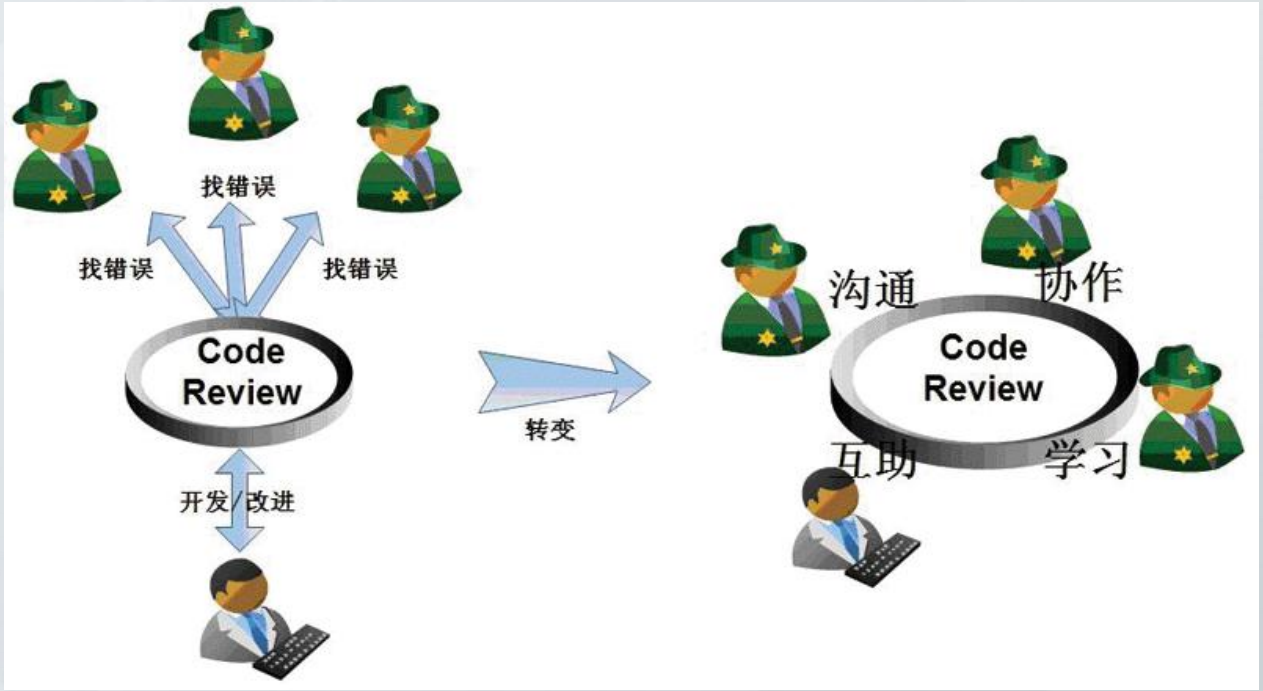


敏捷工程实践：Code Review

目的：持续的提高开发团队的工作质量

好处：

- 代码复查者(reviewer)能从他们的角度来发现问题并且提出更好的解决方案。
- 确保至少团队的另一个其他成员熟悉你的代码，通过给新员工看有经验的开发者的代码能够某种程度上提高他们的水平。
- 公开reviewer和被复查者的想法和经验能够促进团队间的知识的分享。
- 能够鼓励开发者将工作进行的更彻底，因为他们知道代码将被其他的人阅读。



在项目开发的过程中，25%的时间应该花费在code review上

敏捷工程实践：产品发布规则

每次迭代发布，移动互联网项目，自主性研发产品.....

多次迭代发布，传统项目、大型项目

目录

训练

- 和身边的同学组成4-5人项目小组，确定一个PO
- 每个小组构想一款面向在校大学生用户群体的产品
  - 定义至少10条以上产品特性的PB列表，每项特性以用户故事的格式表达
  - 对所有PB中的特性定义优先级别和估算工时（人天数）
  - 以2周为迭代周期，定义第一个迭代周期的迭代PB，并将每项特性分解为任务，估算各项任务的工时（人时数），并分工到团队成员上
  - 模拟2-3天的例会进度，画出和跟踪燃尽图的变化
  - 以界面原型为工作成果，在评审会上演示工作成果
  - 召开回顾会，探讨流程中的可改进点（集体发言，并最终归类为：避免犯的错误，可以做的更好的经验，继续发扬的好做法）

总结

1. 用户故事
2. 结对编程
3. 测试驱动开发
4. 持续集成
5. CodeReview

THANKS