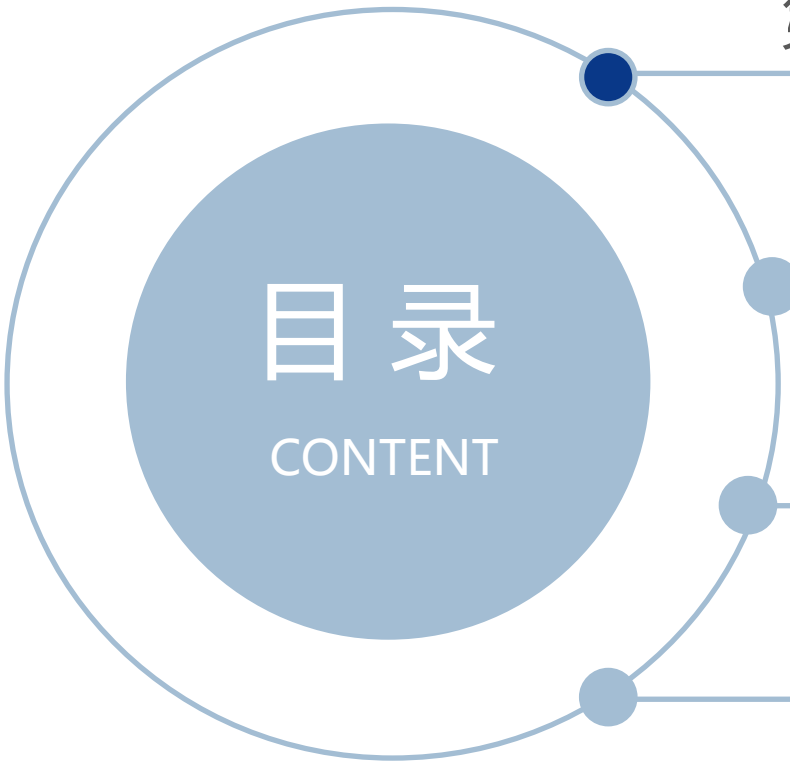
TiD2016
质量竞争力大会
软件研发顶级盛会

下一代软件研发
SOFTWARE DEVELOPMENT
更高 更新 更深

# 软件质量保证与量化管理

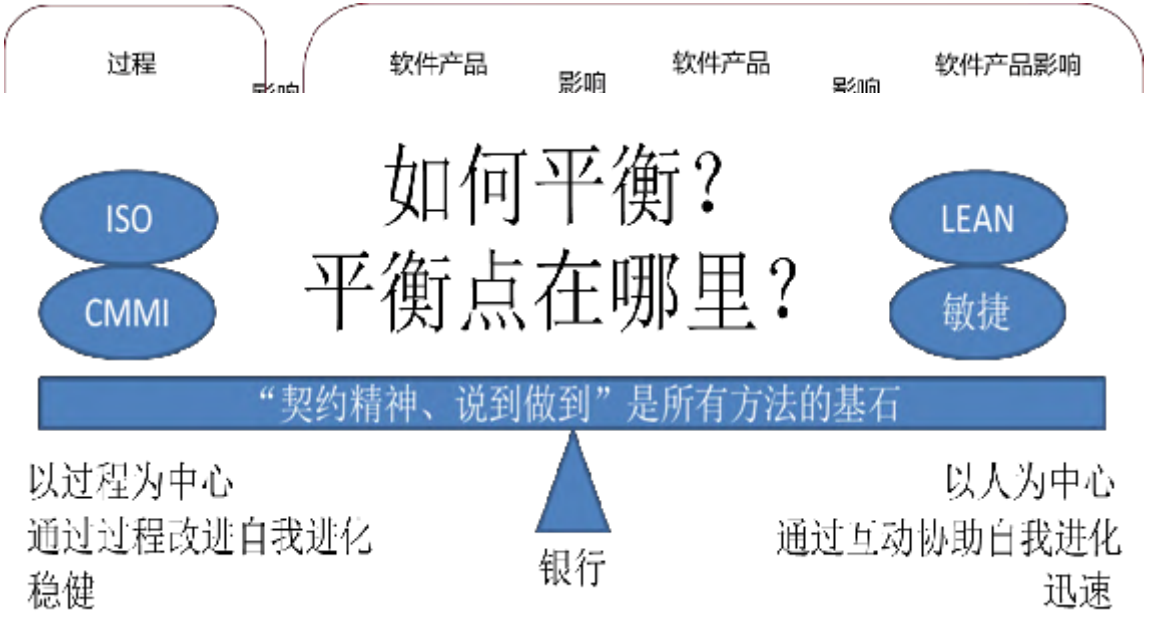
## 中国银行软件中心敏捷项目持续集成实践

中国银行软件中心 付大亮 2016.3



- 银行与互联网企业
  - 银行与互联网企业相同之处：都面对市场竞争；都需要提升用户体验，增强用户粘性
  - 银行与互联网企业环境差异
    互联网企业：
    小步快跑，低成本试错
    紧跟变化，用户粘性好
    银行：
    涉及资金，试错风险巨大
    批次管理，一年几次变更
- 敏捷转型
  - 面对相同的市场竞争，不同的质量要求。银行怎么做敏捷转型？

- ISO 9126质量模型
- 敏捷与CMMI冲突吗？




传统质量管理：管理过程，保证过程执行且过程执行合规

传统质量保证：审核过程交付物存在且合规

# 目 录

CONTENT

- 检验、集成、可视、反馈

零件自检


反馈

反馈

反馈

反馈

集成

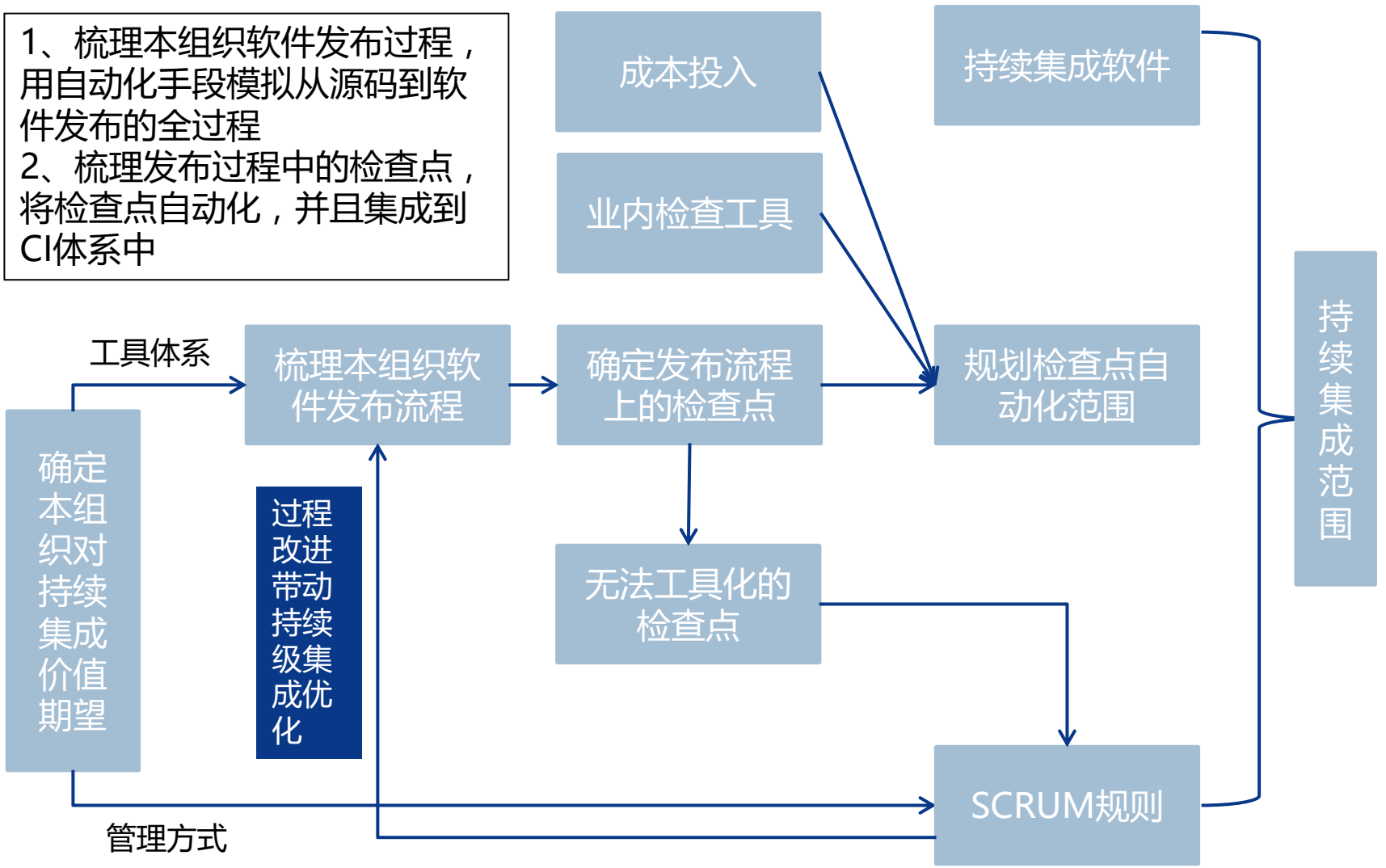
灰度升级

可视

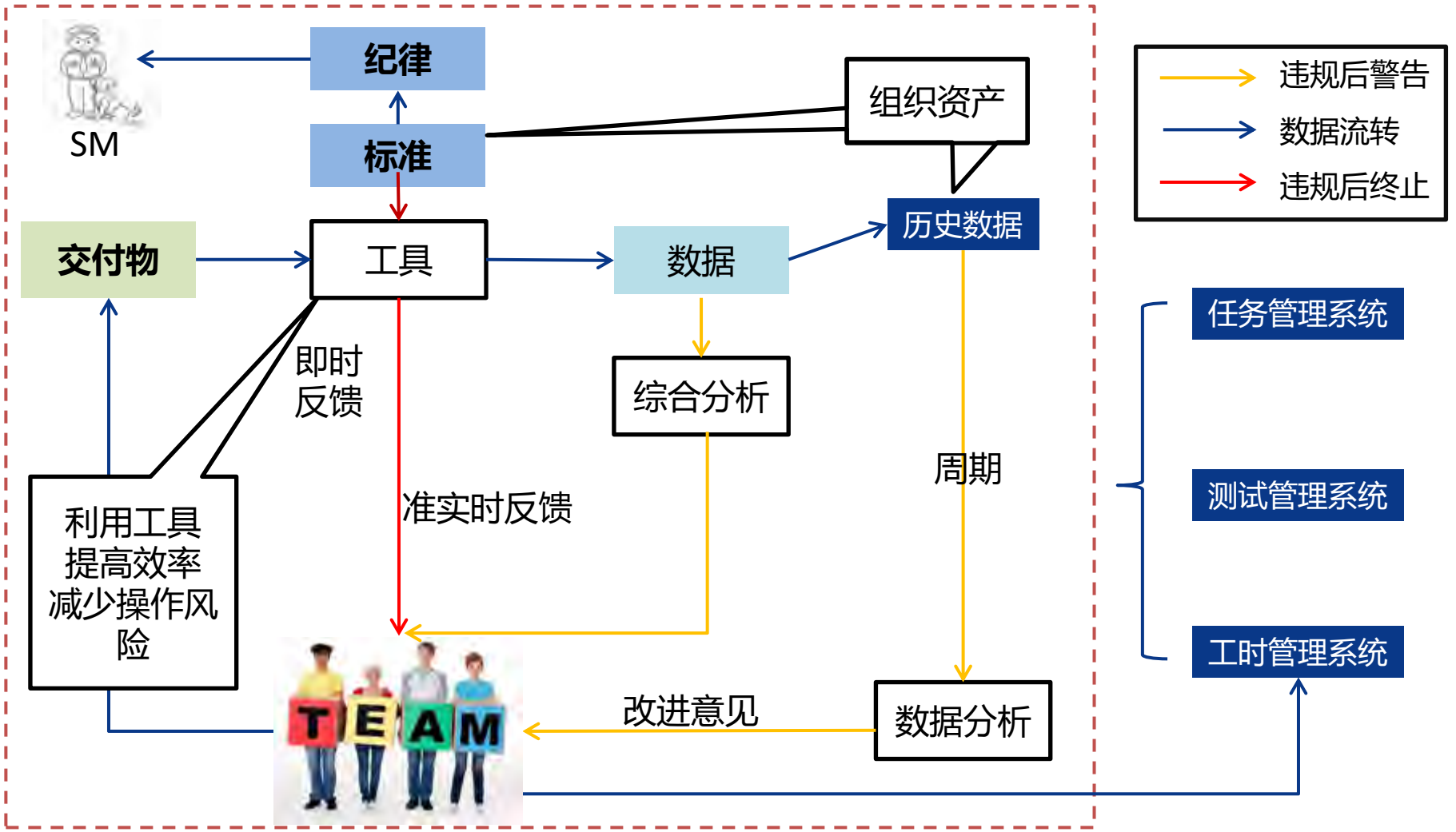
- 持续集成提高软件可见性，缩短反馈周期。
- 验证组件对接情况
- 运用工具减少错误
- 运用工具提升效率
- 运用工具量化软件属性

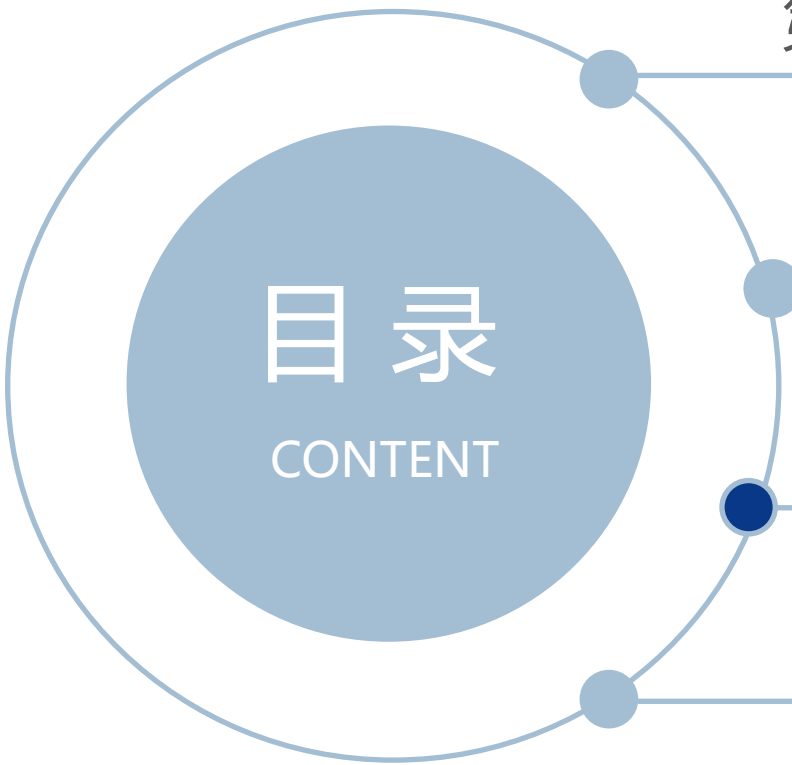
- 确定持续集成范围方法

1、梳理本组织软件发布过程，用自动化手段模拟从源码到软件发布的全过程
2、梳理发布过程中的检查点，将检查点自动化，并且集成到CI体系中



- 标准化、工具化、自动化、可视化

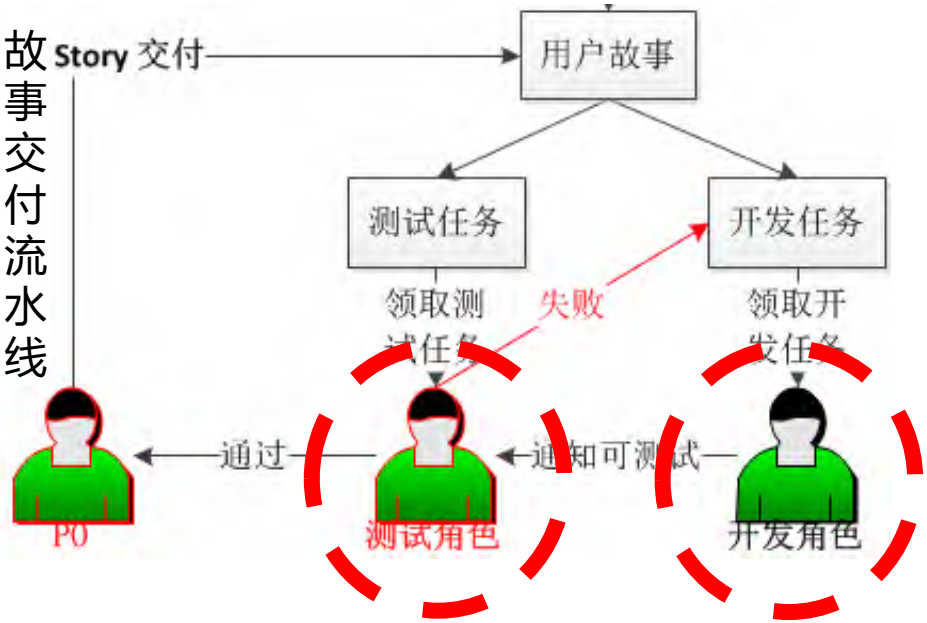




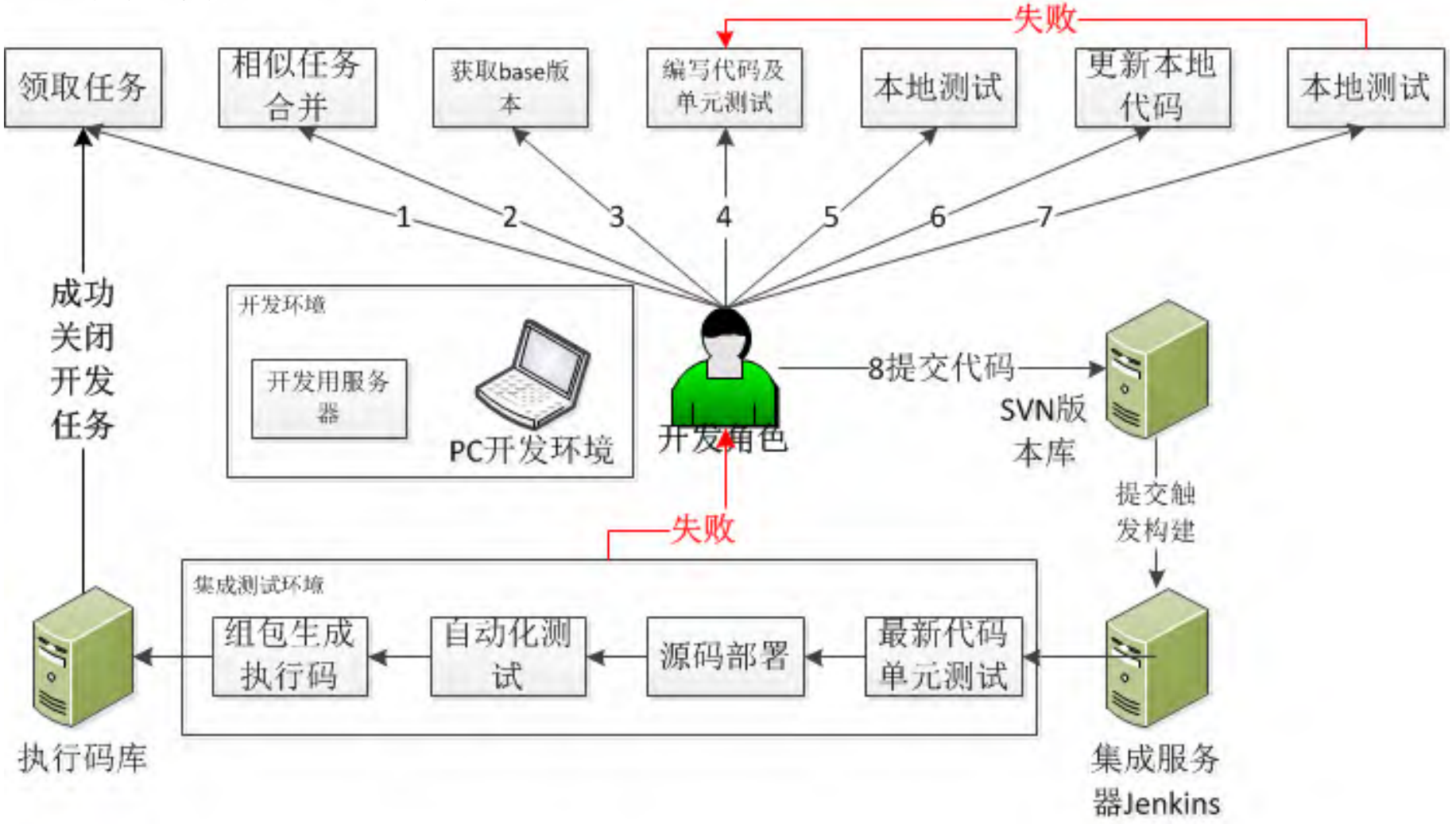
- 需求流水线


全流程敏捷
过渡期开发敏捷
需求获取
方案制定
生产任务排期
Scrum迭代
投产


项目组迭代流水线
24 hours
Product backlog
Sprint backlog
2-4 weeks
Deliverable
SPRINT

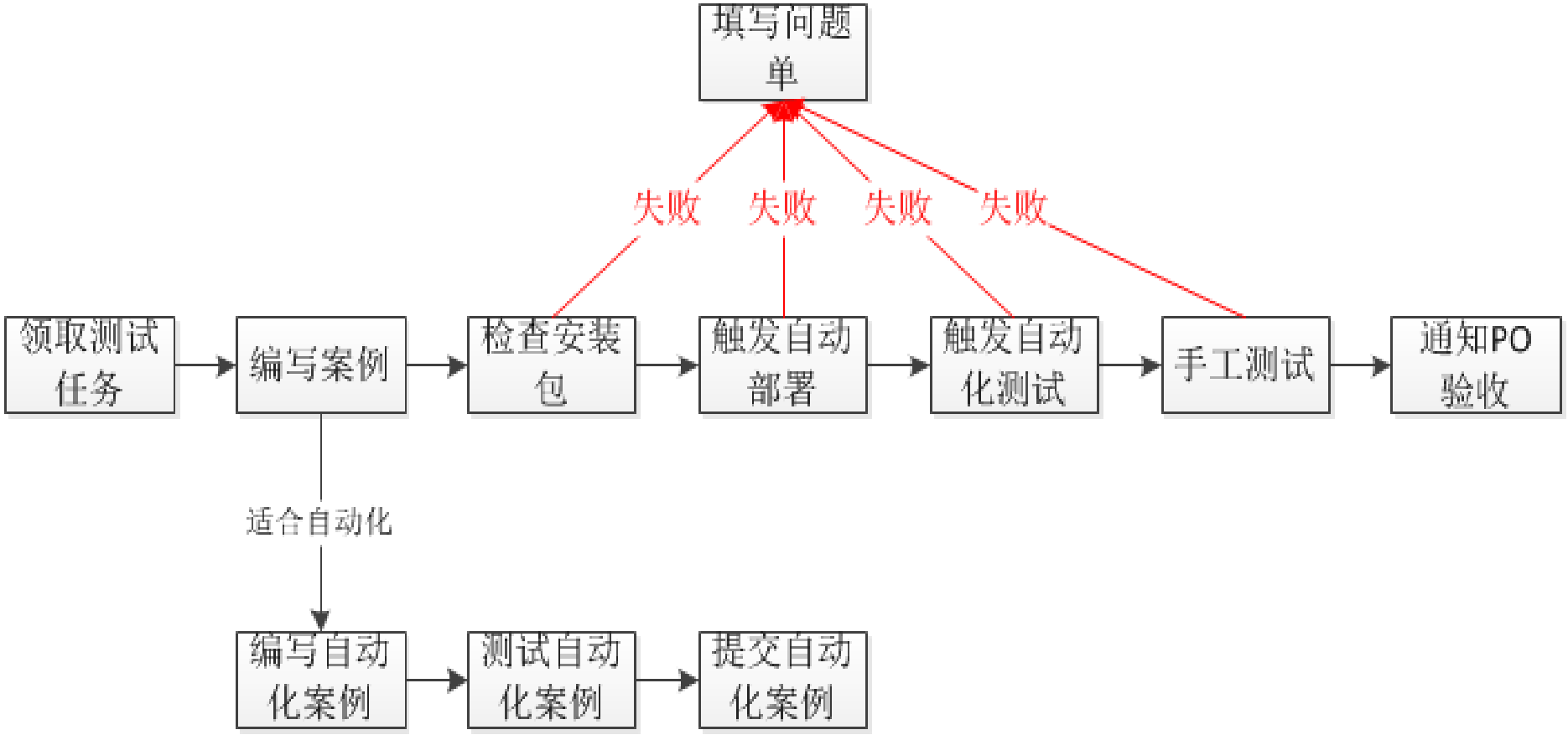

故事交付流水线
Story 交付
用户故事
测试任务
开发任务
领取测试任务
失败
领取开发任务
通过
通知可测试
PO
测试角色
开发角色

- 开发流水线说明
  - 开发流水线从领取任务开始，到交付的程序通过持续集成所有检查结束。
  - 开发人员进行本地可执行测试



## 测试流水线说明

- 在开发任务完成后，测试角色领取已完成故事的测试任务
- 先手工执行测试任务，如果发现该测试任务符合自动化条件，编写自动化测试案例。

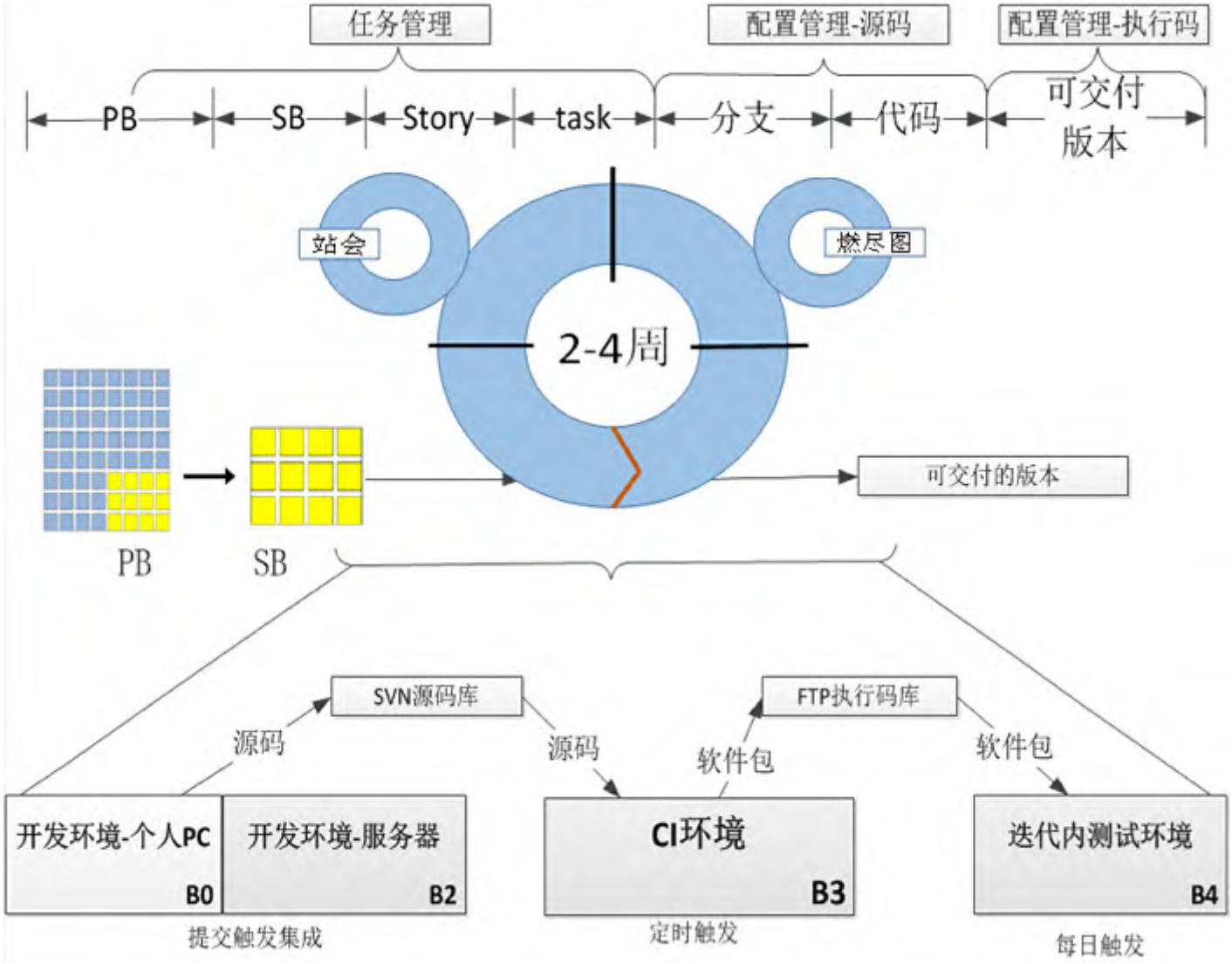


- 任务、团队、环境

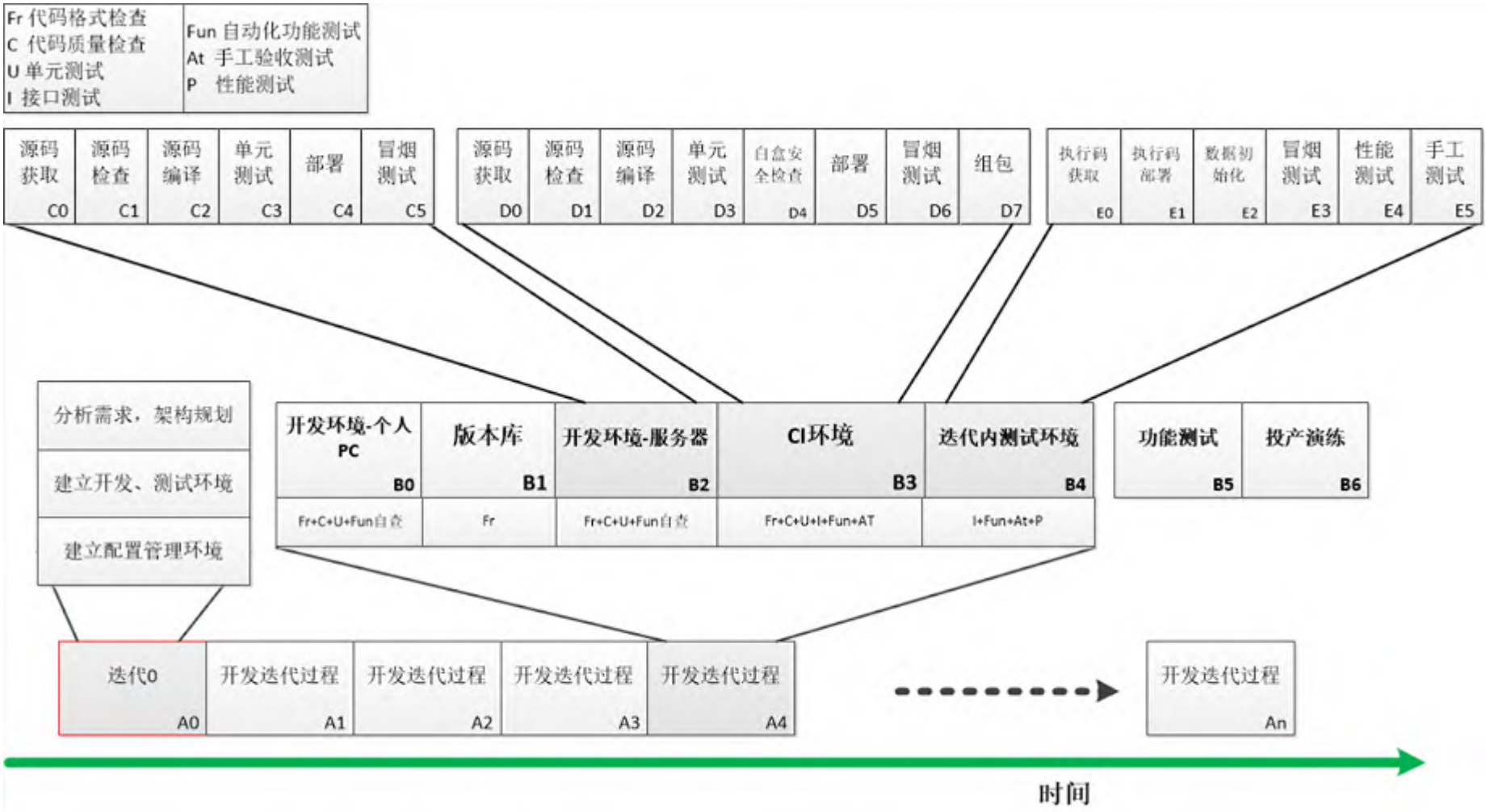
- 右图上部任务流程
- 右图中间是SCRUM运行流程
- 右图下部是TEAM与团队的交互。

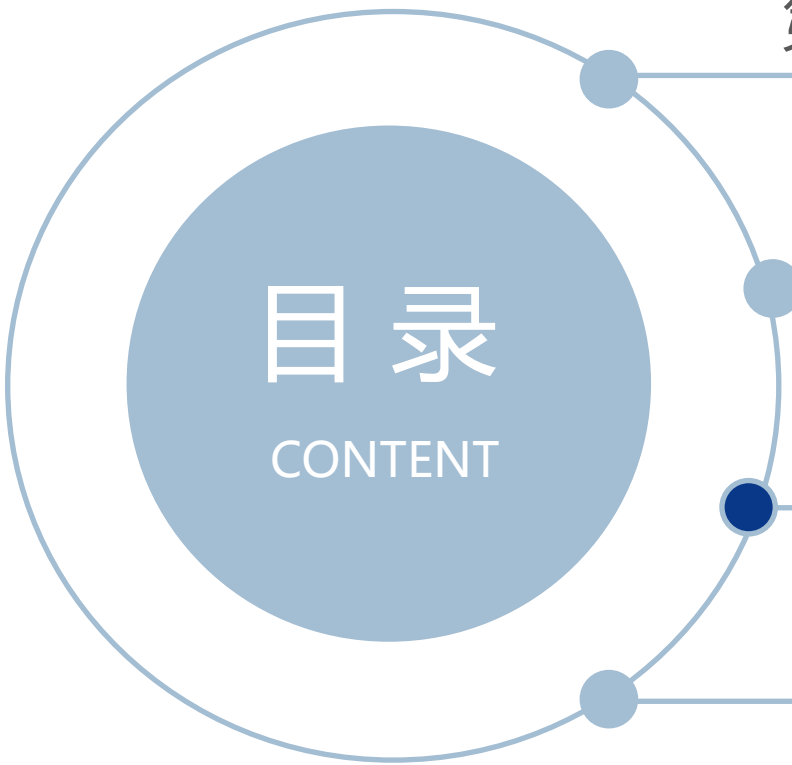
- 规划三类环境

- 开发环境 ，包括个人PC和配合测试用的服务器
- CI环境，将所有人提交的代码集成，并运行所有检查。
- 迭代测试环境，模拟生产环境，通过执行码部署，部署后首先执行自动化测试，然后供测试人员测试



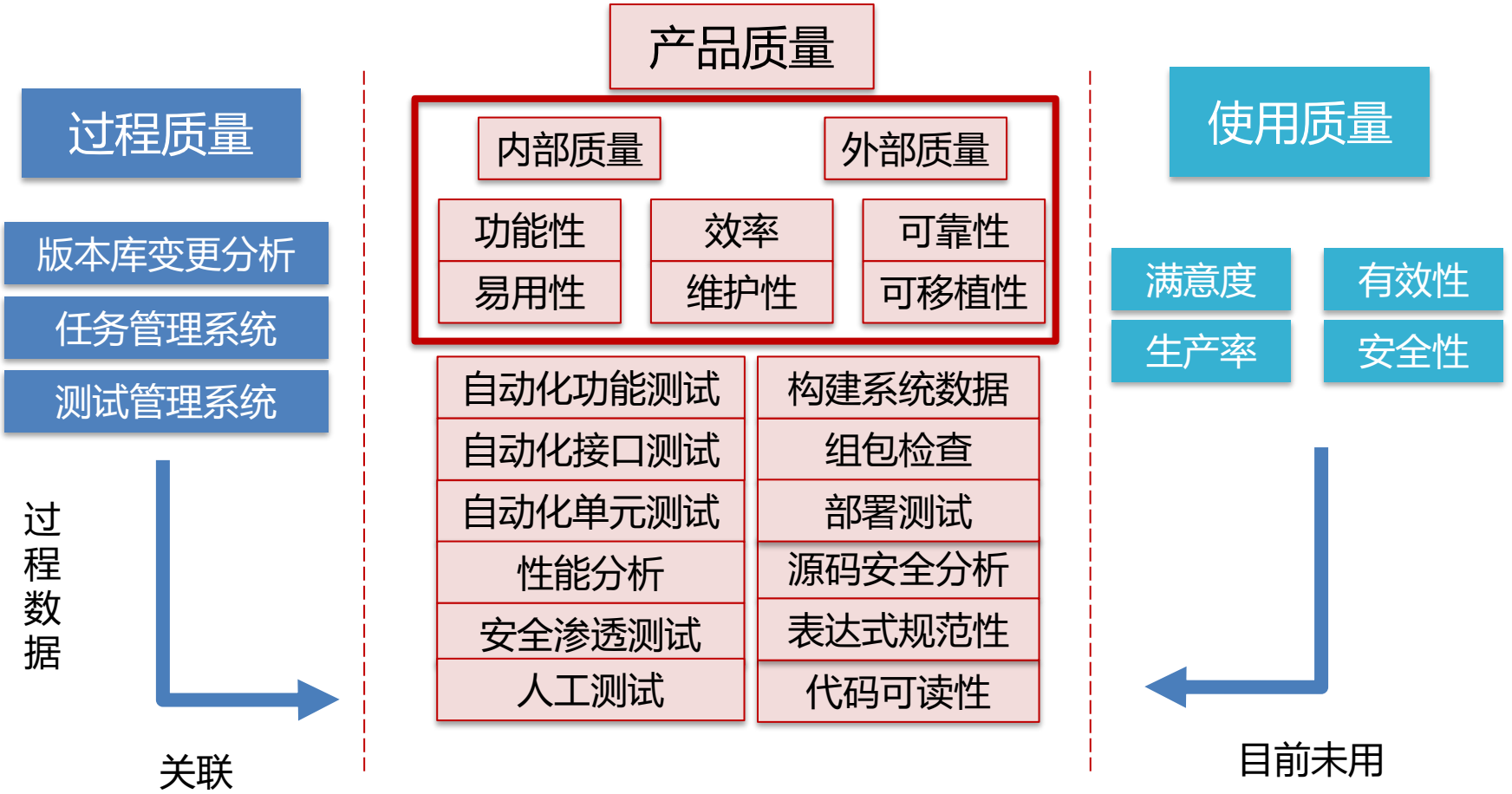
- 环境与反馈
  - 不同的环境承担不同类型的检查
  - 不同环境参与敏捷过程不同周期的反馈环

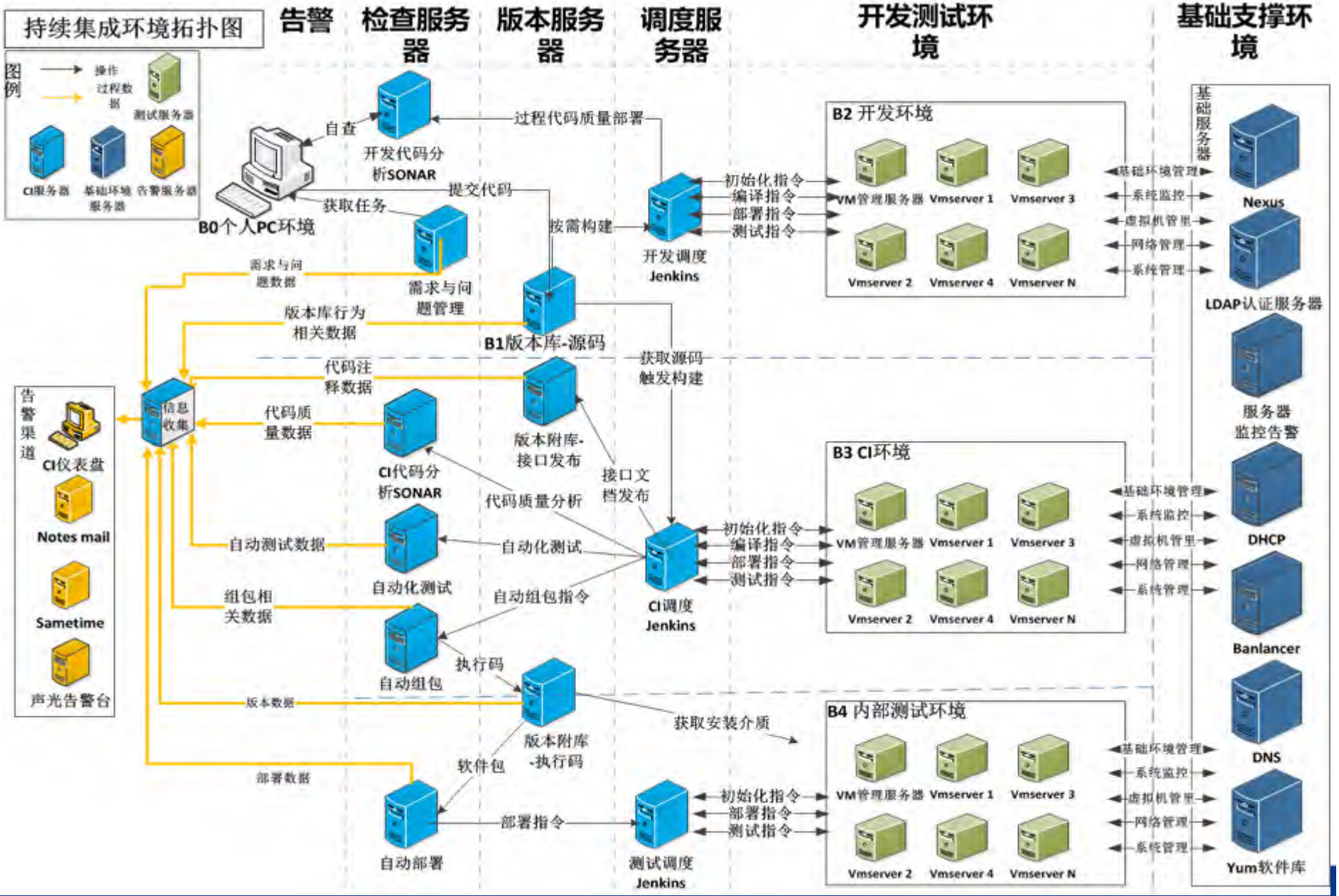


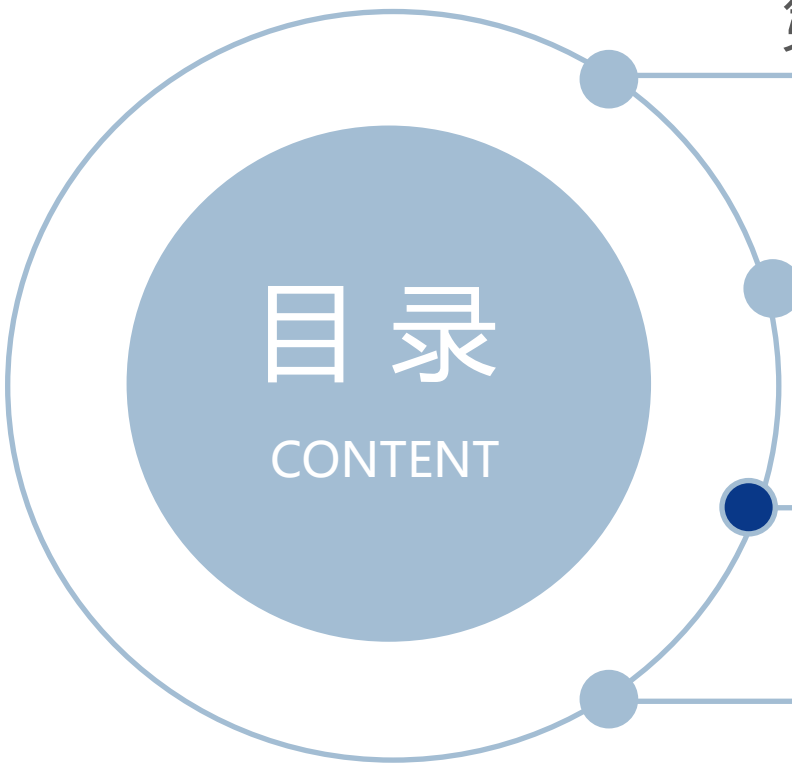


- 度量

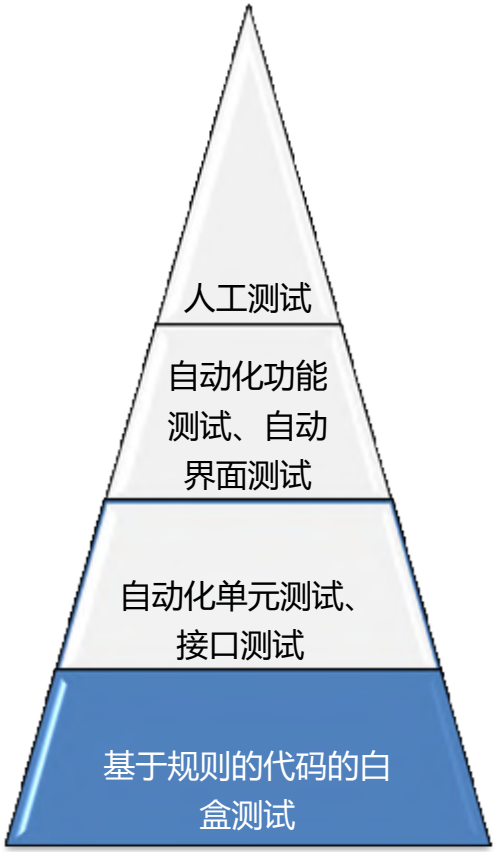
谁，因为什么原因（需求），在什么时间修改了那些内容（版本库变更），
用了多少时间（任务管理），修改效果如何（持续集成检查结果）



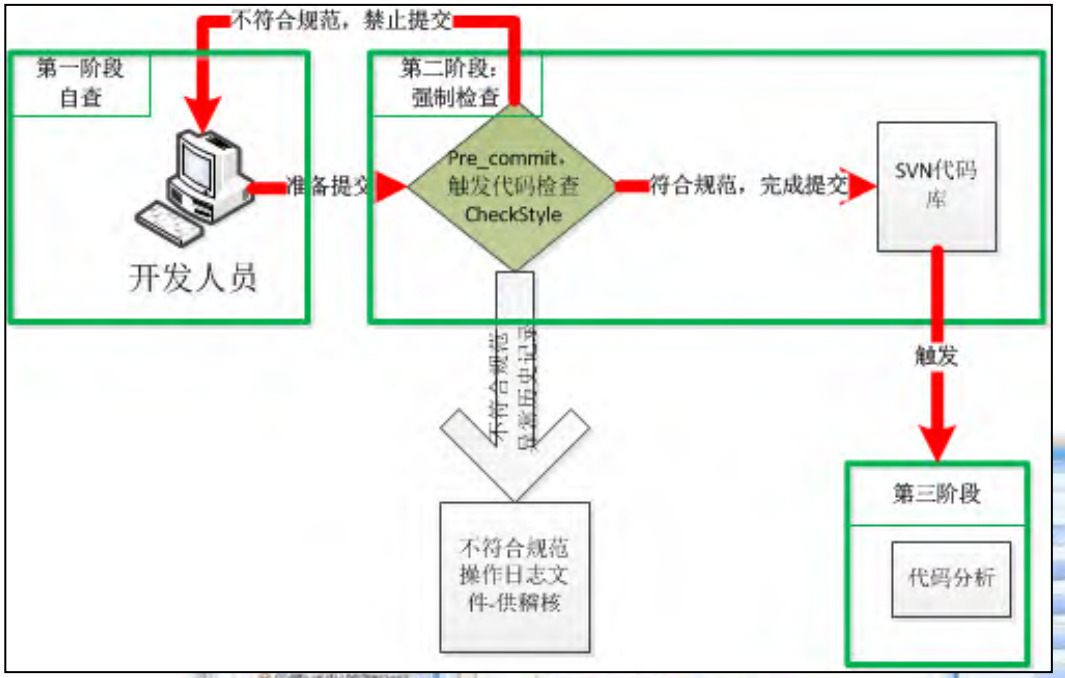

持续集成环境拓扑图
告警
检查服务器
版本服务器
调度服务器
开发测试环境
基础支撑环境
图例
操作
过程数据
测试服务器
CI服务器
基础环境服务器
告警服务器
自查
开发代码分析SONAR
过程代码质量部署
提交代码
获取任务
B0个人PC环境
需求与问题数据
需求与问题管理
按需构建
开发调度Jenkins
初始化指令
编译指令
部署指令
测试指令
B2 开发环境
VM管理服务器
Vmserver 1
Vmserver 3
Vmserver 2
Vmserver 4
Vmserver N
基础环境管理
系统监控
虚拟机管里
网络管理
系统管理
基础服务器
Nexus
LDAP认证服务器
版本库行为相关数据
B1版本库-源码
获取源码触发构建
代码注释数据
代码质量数据
信息收集
CI代码分析SONAR
版本附库-接口发布
接口文档发布
代码质量分析
服务器监控告警
DHCP
Banlancer
DNS
Yum软件库
告警渠道
CI仪表盘
Notes mail
Sametime
声光告警台
自动测试数据
自动化测试
自动化测试
CI调度Jenkins
B3 CI环境
组包相关数据
自动组包指令
自动组包
执行码
版本数据
获取安装介质
版本附库-执行码
软件包
部署数据
自动部署
部署指令
测试调度Jenkins
B4 内部测试环境

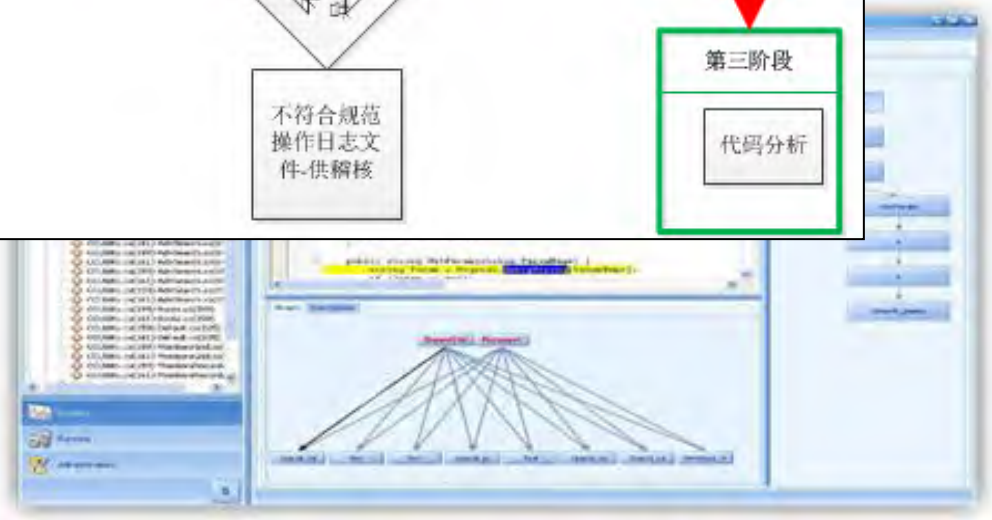


- 基于规则的静态检查




- 工具示例



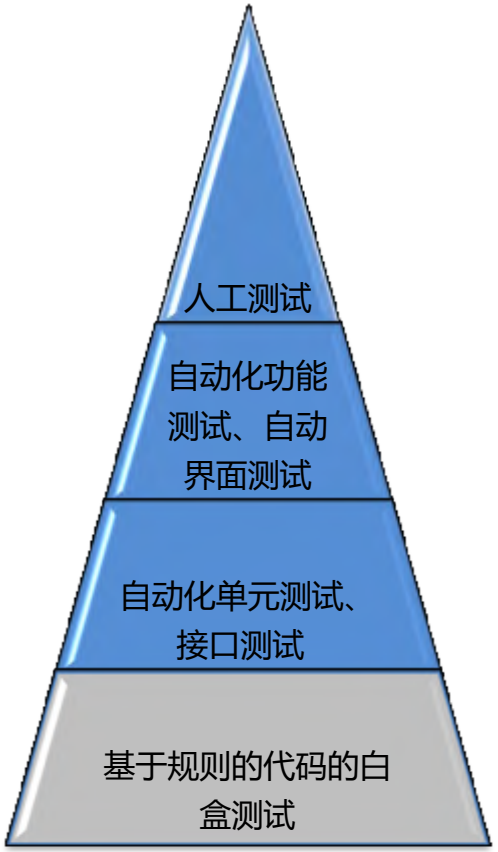
# 基于规则的代码规范检查示例

```
private String n1 = null;
private int ct = 0;
private List cd = null;

public String whatAreYouDoing(final St
    boolean ir = false;
    n1 = gcn;
    for (int i = 0; i < cd .size(); i+
        if (gcn.equals(cd.get(i))) {
            ir = true;
        }
    }
    if (ir == true) {
        return get0(gcn);
    }
    else
    {
        if (n1.equals("tom")) {
            ct = 1;
        }
        else if (gcn.equals("jerry"))
            ct = 2;
        }
        else if (n1.equals("bill")) {
            ct = 3;
        }

        return get1(ct);
    }
```

```
private String nameOfCustomer = null;
private int customerType = 0;
private List customerDatabase = null;

public String orderTheDinner(String givenCustomerName) {

    nameOfCustomer = givenCustomerName;

    String customerOrder = null;
    if (isCustomerAlreadyRegistered(givenCustomerName)) {
        customerOrder = getOrderFromHistory(givenCustomerName);
    }
    else
    {
        customerType = getTypeOfCustomer(givenCustomerName);
        customerOrder = suggestOrderForCustomer(customerType);
    }

    return customerOrder;
}


private boolean isCustomerAlreadyRegistered(String givenCustomerName) {
    for (int i = 0; i < customerDatabase .size(); i++) {
        if (givenCustomerName.equals(customerDatabase.get(i))) {
            return true;
        }
    }

    return false;
}

private String suggestOrderForCustomer(int customerType2) {
```

- **产品持续发展**
  - 通过标准、规范有效落实，提高产品质量与可维护性。
  - 规范的开发能够减缓人员流动对产品质量的影响
- **留住愿付出的人**
  - 以量化为基础，展现不同TEAM在技术上、经验上、工作量上的优势。
  - 通过量化方法及相关工具配合减少管理工作对程序员的干扰，让程序员安心去写程序。
- **提升质量**
  - 通过工具自动落实标准规范。
  - 通过量化，每一点提升都是看得见的，能够激发团队改进。

- 基于案例的测试




- 分层策略

开发环境执行　　手工测试环境执行

| 分层 / 检查项 | 即时检查 | | 定时检查 | | 手工检查 | |
|---|---|---|---|---|---|---|
| | 提交前 | 提交后 | 小时 | 天 | | |
| 代码格式检查 | ★ | ★ | ★ | ★ | X | X |
| 表达式检查 | ★ | ★ | ★ | ★ | X | X |
| 代码安全检查 | | | ★ | ★ | X | X |
| 单元测试 | | ★ | ★ | ★ | X | X |
| 接口测试 | | | ○ | ○ | X | X |
| 界面测试 | | | | ★ | ★ | |
| 组包测试 | | | | ★ | | |
| 部署测试 | | | | ★ | ★ | |
| 安全黑盒 | | | | ★ | ★ | |
| 性能测试 | | | | ★ | ★ | |
| | | | | | | |

版本库执行　　CI环境执行

工具的选择：
功能精深专一；易于集成，避免数据孤岛。最好开源或开发接口，易于二次开发。

| 检查项 | 软件名称 | 说明 |
| --- | --- | --- |
| 代码格式检查 | CheckStyle | 1、通过Subversion插件调用<br>2、通过sonar调用 |
| 表达式检查 | Findbugs | 1、通过eclipse调用<br>2、通过sonar调用 |
| 代码安全检查 | CheckMarx | 1、Jenkins调用 |
| 单元测试 | Junit/Jasmine | Maven调用 |
| 接口测试 | 暂时未开展 | 无 |
| 界面测试 | RobotFrameWork | Jenkins调用 |
| 组包测试 | 自研组包工具 | Jenkins调用 |
| 部署测试 | 自研自动部署系统 | 自有调度 |
| 安全黑盒 | AppScan | 暂时未集成 |
| 性能监控工具 | Dynatrace | 正在集成 |

## 持续部署意义

- 使软件功能持续可见
- 集成软件与环境
- 验证发布流程
- 降低投产风险

## 工具选择考虑

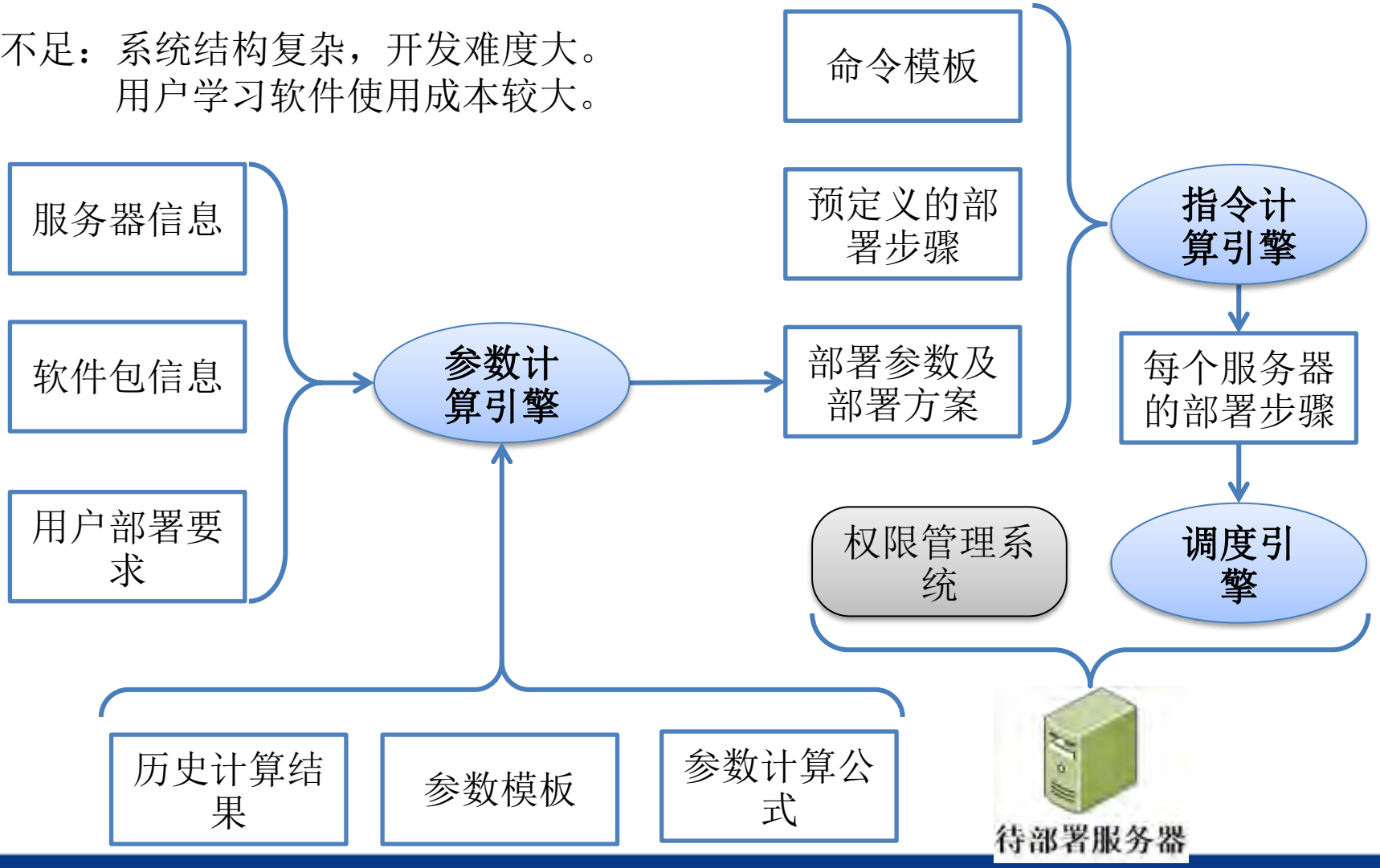
- 单机软件还是分布式软件
- 固定环境还是可变环境
- 软件发布流程
- 服务器数量与部署频率
- 是否形成IT环境配置库

## 软件持续部署实践

| | | | | |
|---|---|---|---|---|
| 环境 | 开发环境 | 集成环境 | | 测试环境 |
| 时机 | 随时部署 | 定时部署 | | 按需部署 |
| 介质 | 源码 | 源码 | 源码 | 执行码 |
| 工具 | 中间件自带工具，开源工具，自己写的脚本等。 | 中间件自带工具，开源工具，统一脚本等。 | 中间件自带工具，开源工具，统一脚本等。 | 自主开发的面向分布式应用、无代理自动部署。 |

优点：适应不同软件在不同的环境下部署，复用率高。
准确记录部署信息，及应用各种配置

不足：系统结构复杂，开发难度大。
用户学习软件使用成本较大。



- 反馈意义

目标： 配合敏捷管理 构建不同周期的反馈环 ，满足不同角色信息需要

数据来源：持续集成体系中的各种工具的运行数据，各种交付物属性数据。

- 反馈方式

| | | | | | |
|---|---|---|---|---|---|
| | 版本库 PreCommit钩子 |  |  |  |  |
| 通知范围 | 个人 | 团队 | 个人 | 个人<br>团队 | 个人<br>团队<br>相关人 |
| 通知时机 | 即时事前 | 即时事后 | 准实时 | 通知 | 事后通知 |

# 持续集成——仪表盘

单元测试覆盖率、成功率

功能测试成功率

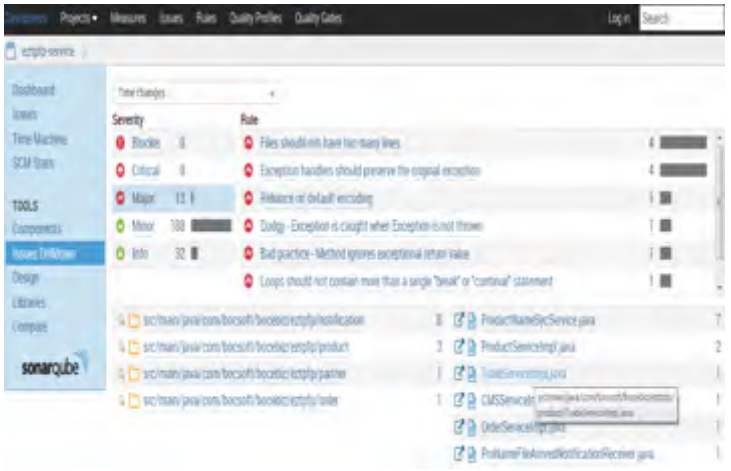
代码构建状态

代码复杂度、重复度

代码问题和技术债务

分类：阻断、严重、主要、次要
趋势：日-周-月
定位：程序级






- 综合展示工程过程数据、产品质量数据，方便组织级实施量化、精细化管理。
- 数据更新频率在0.2-2小时之间。组织能够及时感知项目运行状况。减少试错成本。
- 所有数据都按照时间序列保存，为组织自我优化提供数据基础。
- 数据透明化，便于实施绩效激励，便于统一思想

# 仪表盘采集的数据

| | 参数名称 | 参数来源 | 来源类型 |
|---|---|---|---|
| 版本库 | 变更号 | 版本库日志 | 原始参数 |
| | 变更人 | 版本库日志 | 原始参数 |
| | 变更时间 | 版本库日志 | 原始参数 |
| | 变更类型 | 版本库日志 | 原始参数 |
| | 变更对象名 | 版本库日志 | 原始参数 |
| | 变更内容 | 版本库日志 | 原始参数 |
| | 所属的分支名 | 版本库日志 | 原始参数 |
| | 对应的需求编号 | 提交注释 | 原始参数 |
| | 业务版本号 | 提交注释 | 扩展参数 |
| 代码安全检查 | 规则名称 | 检查日志 | |
| | 规则类型 | 检查日志 | 原始参数 |
| | 规则等级 | 检查日志 | 原始参数 |
| | 语言类型 | 检查日志 | 原始参数 |
| | 违规文件名称 | 检查日志 | 原始参数 |
| | 违规时间 | 检查日志 | 原始参数 |
| | 违规人 | 检查日志 | 原始参数 |
| 代码格式检查 | 规则名称 | 检查日志 | 原始参数 |
| | 规则类型 | 检查日志 | 原始参数 |
| | 违规文件名称 | 检查日志 | 原始参数 |
| | 违规时间 | 检查日志 | 原始参数 |
| | 违规人 | 检查日志 | 原始参数 |
| 表达式检查 | 规则名称 | 检查日志 | |
| | 规则类型 | 检查日志 | 原始参数 |
| | 规则等级 | 检查日志 | 原始参数 |
| | 语言类型 | 检查日志 | 原始参数 |
| | 违规文件名称 | 检查日志 | 原始参数 |
| | 违规时间 | 检查日志 | 原始参数 |
| | 违规人 | 检查日志 | 原始参数 |

| | 参数名称 | 参数来源 | 来源类型 |
|---|---|---|---|
| 构建服务器 | Job名称 | CI服务器 | 原始参数 |
| | 产品名称 | CI服务器 | 原始参数 |
| | 分支名称 | CI服务器 | 原始参数 |
| | 构建内容 | CI服务器 | 原始参数 |
| | 执行时间 | CI服务器 | 原始参数 |
| | 执行结果 | CI服务器 | 原始参数 |
| | 耗时 | CI服务器 | 原始参数 |
| 代码分析与单元测试 | 代码复杂度 | 计算 | 扩展参数 |
| | 代码重复度 | 计算 | 扩展参数 |
| | 单元测试覆盖率 | 计算 | 扩展参数 |
| | 单元测试成功率 | 计算 | 扩展参数 |
| | 自动化单元测试脚本数量 | 计算 | 扩展参数 |
| 组包测试 | 组包时间 | 组包日志 | 原始参数 |
| | 组包结果 | 组包日志 | 原始参数 |
| | 组包稳定性 | 计算 | 扩展参数 |
| 部署测试 | 部署时间 | 部署日志 | 原始参数 |
| | 部署结果 | 部署日志 | 原始参数 |
| | 冒烟测试结果 | 部署日志 | 原始参数 |
| | 部署测试稳定性 | | |
| 界面测试 | 自动化功能测试案例数量 | 计算 | 扩展参数 |
| | 自动化功能测试比率 | 计算 | 扩展参数 |
| | 自动化功能测试成功率 | 计算 | 扩展参数 |
| | 自动化功能测试稳定性 | 计算 | 扩展参数 |

## 环境纪律

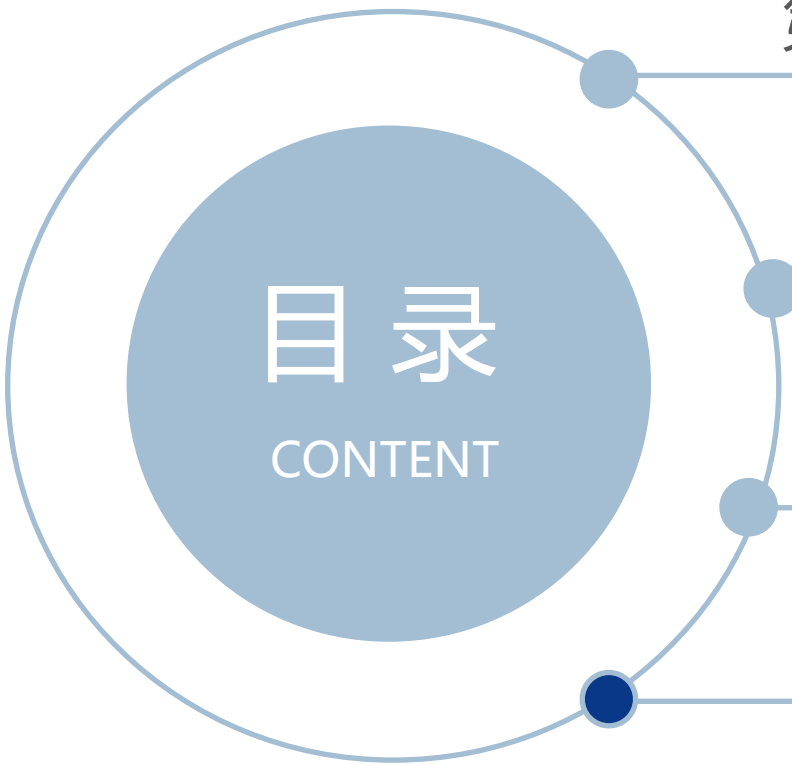
- 进入功能测试阶段后分支代码冻结
- 不做多余功能
- 每天至少提交一次代码
- 关注 CI，失败不下班（整个开发团队）
- 代码提交严格遵守七步提交法

## 关于组包出版本

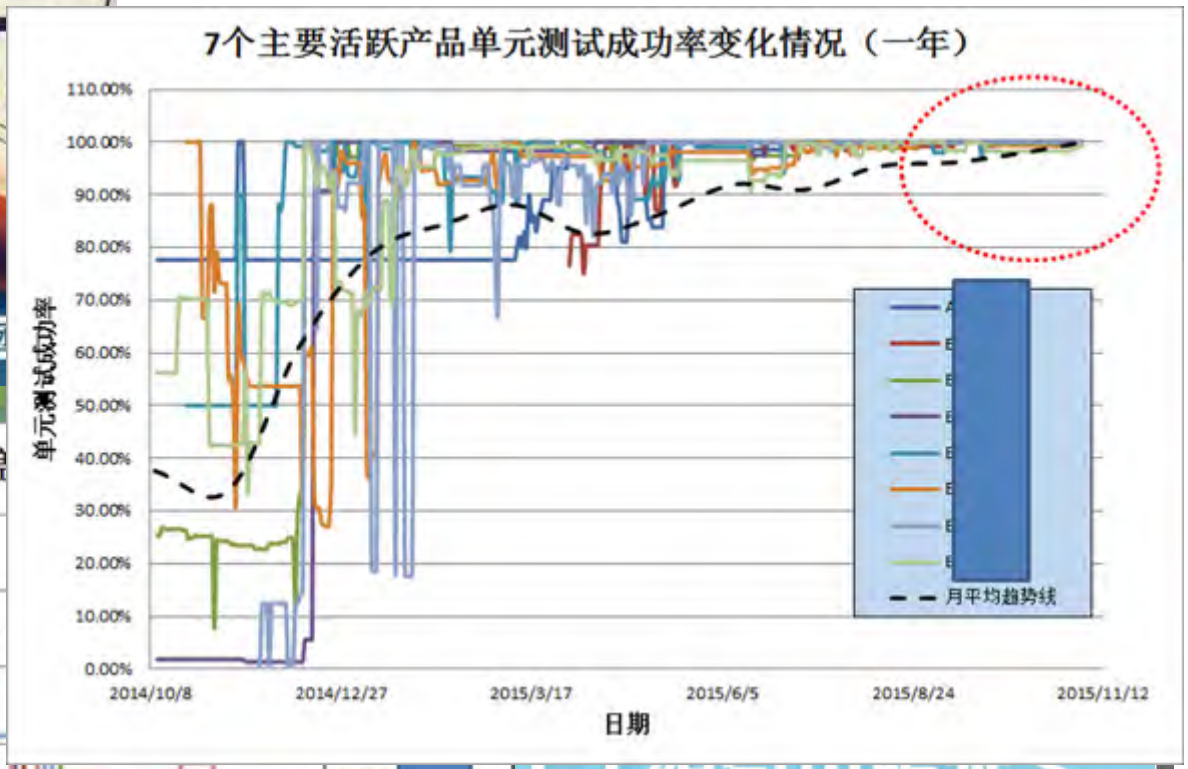
- 内部依赖与外部依赖隔离
- 外部依赖包，禁止换号不换包
- 出版本的包名中禁止出现日期和 SNAPSHOT
- 出版本之前要通过冒烟测试
- 发布的软件包务必在版本编译环境编译组包

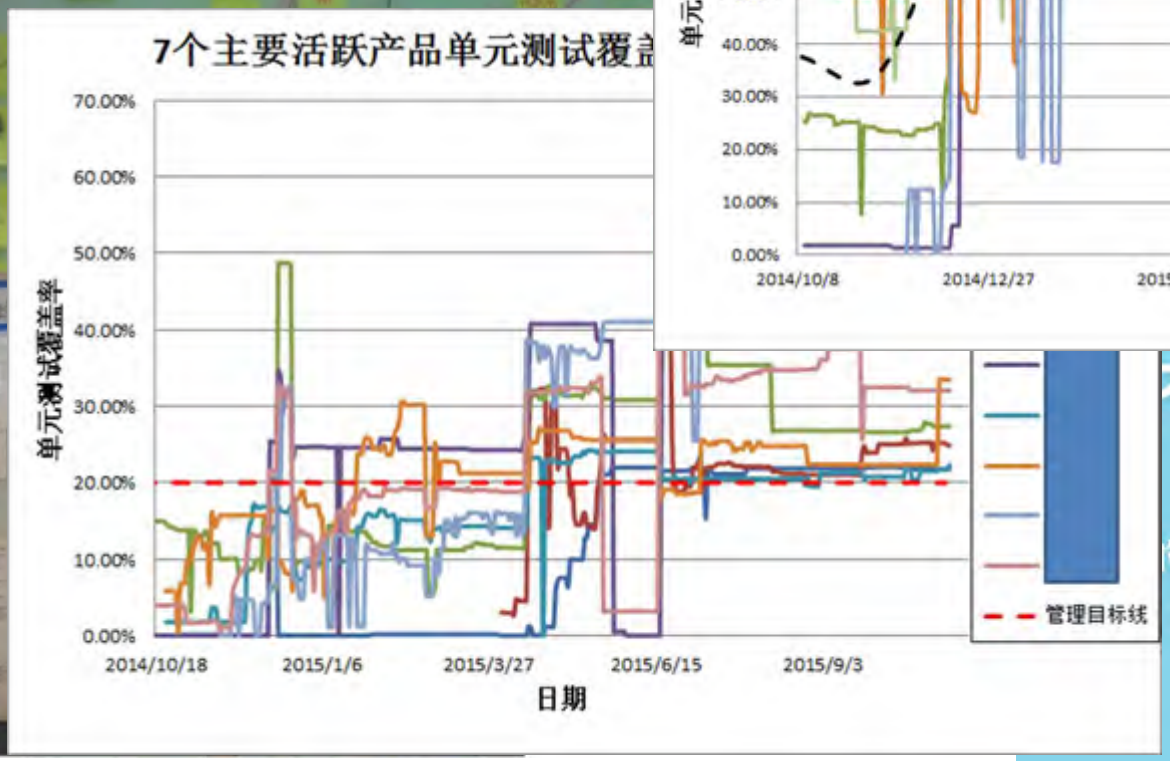
## PO 与 SM

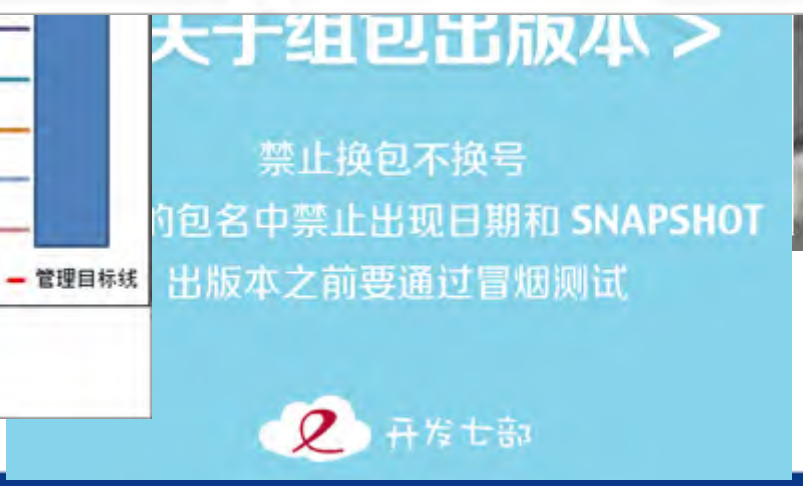
- PO 要对价值负责
- SM 要对流程负责，消除一切破坏流程的人和事
- PO 是团队迭代目标的唯一输入方
- SM 是团队敏捷流程的唯一权威
- PO 应保证未经明确的需求不进入迭代
- SM 应保证不符合 DOD 的交付物不出迭代
- 迭代内故事的实现顺序由团队自主，PO 和 SM 都不得干预
- 迭代过程中 PO 不得新增和修改故事，除非团队主动要求
- 交付功能测试后，所有需求变更和细化均冻结
- 特殊情况下，PO 可以中止本轮迭代




纪律！
不动 UAT 环境
UAT 后分支代码冻结
不夹带版本
每天至少提交一次代码
关注 CI，失败不下班（整个开发团


7个主要活跃产品单元测试成功率变化情况（一年）
单元测试成功率
110.00%
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
2014/10/8
2014/12/27
2015/3/17
2015/6/5
2015/8/24
2015/11/12
日期
月平均趋势线


7个主要活跃产品单元测试覆盖
单元测试覆盖率
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
2014/10/18
2015/1/6
2015/3/27
2015/6/15
2015/9/3
日期
管理目标线


关于组包出版本 >
禁止换包不换号
的包名中禁止出现日期和 SNAPSHOT
出版本之前要通过冒烟测试
开发七部

# 质量

持续集成——在不破坏SCRUM等敏捷实践的灵活性前提下，产生高质量的软件。

低成本试错，在金融企业可能是一个伪命题，或者说目前还没有找到合适的试错方案！

## 工具改进点

- 提高工具的自动化程度
- 集成更丰富的工具
- 提高CI工具的准确率，降低误报
- 扩大自动化的数据采集范围
- 对TEAM及CI工具产生的数据分析加工

## 管理改进点

- 强化敏捷质量意识培养
- 尝试规模化敏捷
- 敏捷质量保证方法改进
- Devops推广
- 优化产品质量量化管理

# 分享完毕

# 谢谢

fudaliang@bankofchina.com

微信号： fudl1999