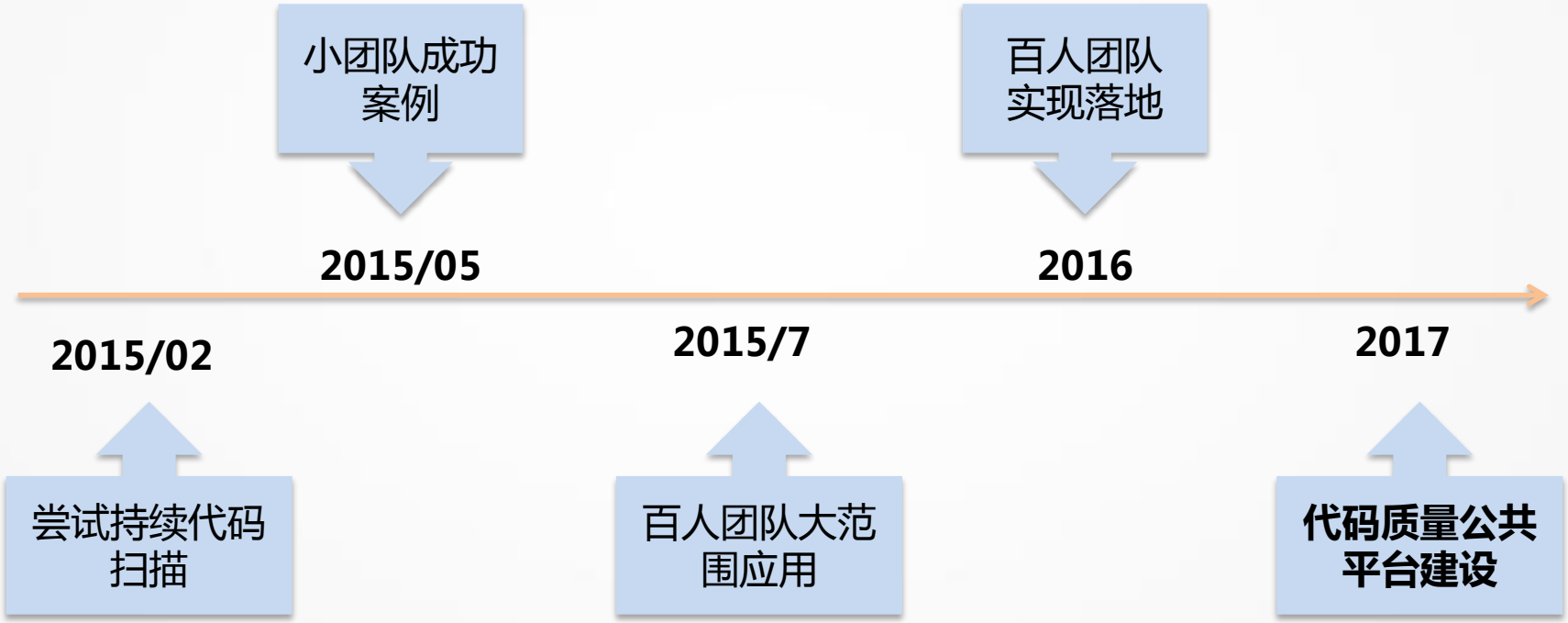
TiD 2017
质量竞争力大会

代码质量平台建设实践

京东　熊志男

软件研发
SOFTWARE DEVELOPMENT

时间线


小团队成功案例
百人团队实现落地
2015/05
2016
2015/02
2015/7
2017
尝试持续代码扫描
百人团队大范围应用
代码质量公共平台建设

开始之前：公司统一的持续集成平台

技术&实现

- 统一部署和统一入口；
- 基于SonarQube和Jenkins实现；
- 多Master，执行速度快；
- 集成其他插件（Findbugs、PMD、Checkstyle等）

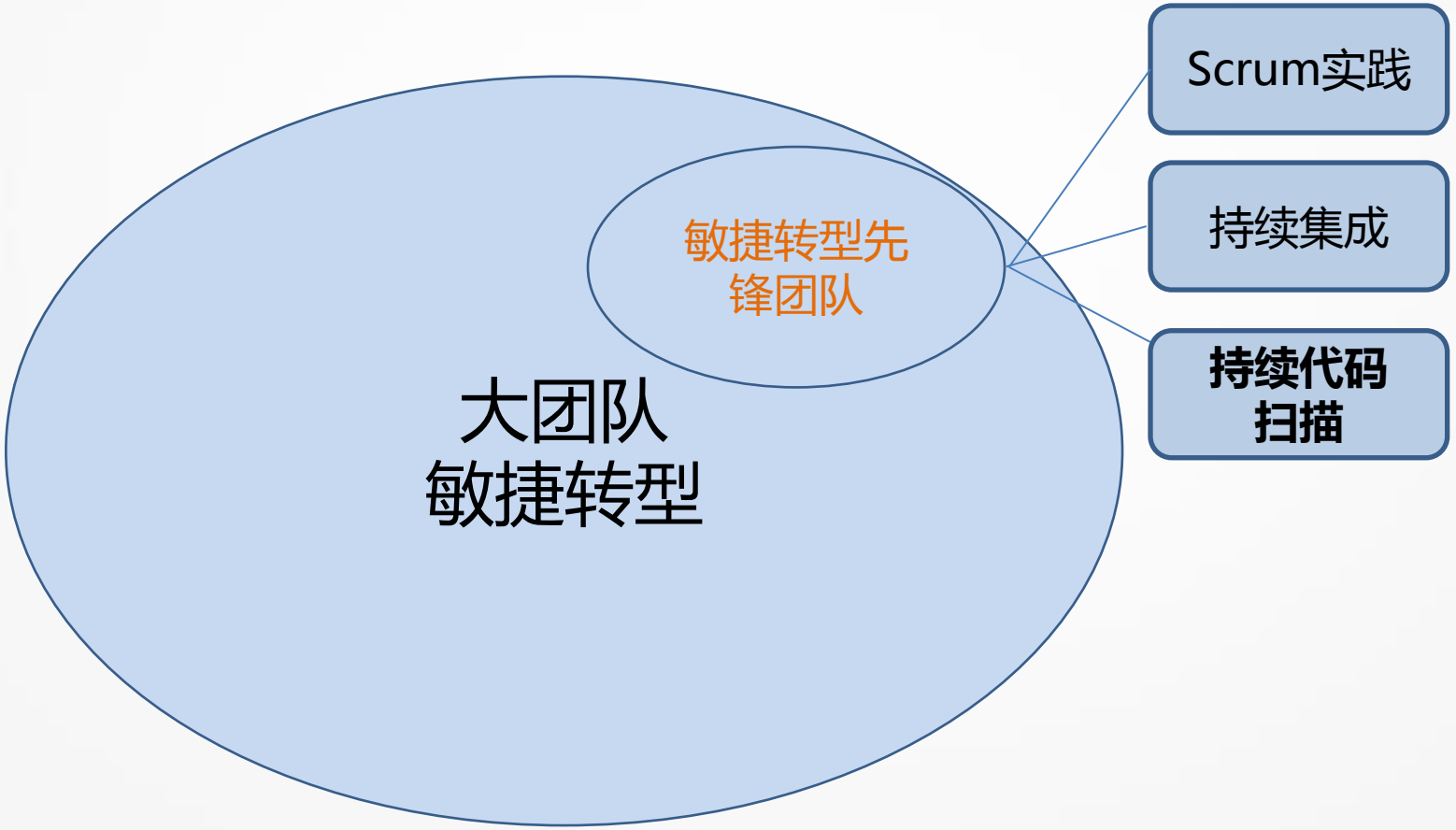
流程&标准

- 固定周期代码扫描；
- 上线前代码质量分析；
- 全量规则代码扫描；
- 基于SonarQube的Total Quality 质量体系（多维度）
- 代码质量周报

团队&文化

- 代码质量靠部分成员自觉；
- **团队代码质量数据在研发体系垫底**

敏捷先锋团队：首先尝试持续代码扫描



持续集成流水线的节点：持续代码扫描

技术&实现

- 原生的开源框架：SonarQube和Jenkins；
- 单Master，单执行机，满足小团队需求；
- 及时反馈机制：Jenkins的Monitor view
- 集成其他插件（Findbugs）
- 以手工配置为主

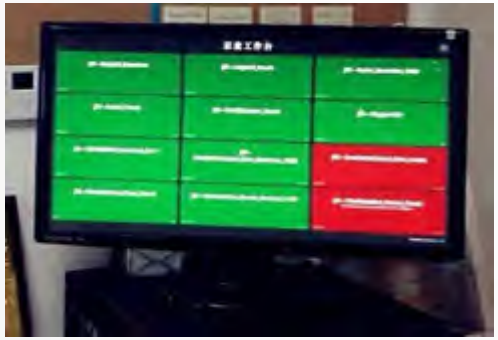
流程&标准

- 团队单分支开发模式；
- 提交触发代码扫描；
- **基于80/20原则的TOP20代码规则集；**
- 关注单个质量维度：代码违规
- 设置代码违规数量阈值

团队&文化

- 研发Leader重视代码质量；
- 团队成员积极性高
- 敏捷文化形成

效果明显：小团队影响力提升



横向推广：百人团队持续代码扫描

技术&实现

- 整合开源框架：POPCI系统；
- 单Master，多执行机，执行机与应用绑定；
- 两种反馈机制：实时反馈和周报数据反馈；
- 集成其他插件（Findbugs，CPD）；
- 通过脚本实现配置和批量导出数据；

流程&标准

- 各团队开发模式不统一；
- 主干提交触发代码扫描；
- **基于80/20原则的TOP9代码规则集；**
- 基于SonarQube的质量体系：SQALE（多维度）
- 设置代码违规数量基线阈值（不新增）

团队&文化

- 各研发Leader积极推动减少代码违规数据；
- 各团队成员配合要求修改违规
- 各团队文化各不相同

阶段成果：代码质量数据提升

代码质量数据研发体系排名第一；

研发人员逐步建立代码质量红线意识；

遇到问题？

数据排名第一了，新的目标？

基础框架不完善，维护任务量井喷；

流程标准不灵活，各团队无法定制；

易用性较差，团队成员学习成本高；

持续改进

从研发痛点入手

- 与架构师沟通
- 了解团队痛点
- 个人违规数据汇总

流程/标准定制化

- Merge Request流程
- 代码规则集裁剪

优化基础框架

- 执行机资源池
- 并行多流水线

简化运维工作

- 自动化运维脚本
- 引入开源类库

持续改进：目标的转变

提升排名

追求绝对数字

消灭TOP9违规

易用性差

向上负责

服务研发

提升相对质量

过滤低级错误

提高易用性

体现研发个人代码质量

向下负责

持续改进：流程的转变

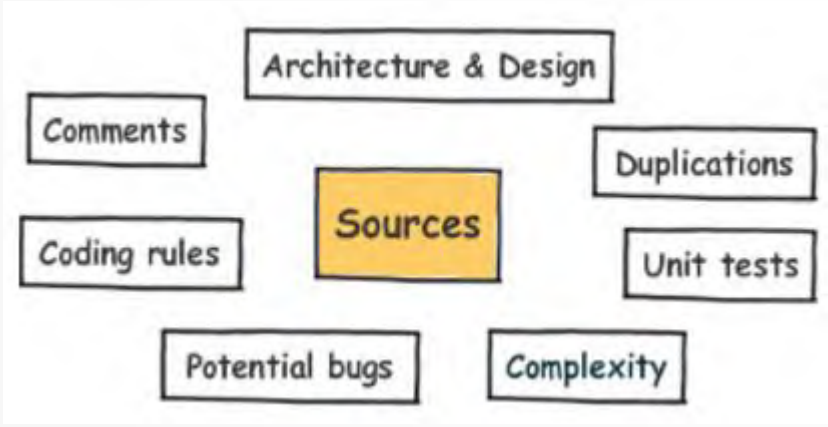
由统一规则集 转变为 统一规则集和定制规则集相结合；

由分散复杂的质量维度 转变为 简明集中的质量维度；

由单个串行流水线 转变为 多个并行流水线；

由单一的代码扫描机制 转变为 全流程的代码扫描机制；

持续改进：质量维度聚合



可发布性：

可靠性：

安全性：

可维护性：

流程改进：全流程代码扫描

本地代码扫描

- 编辑器插件代码扫描；（团队规则集）
- GIT Pre-commit 代码扫描（零容忍规则集）

分支代码扫描

- 分支变更代码扫描；（团队规则集）
- GIT Merge request代码扫描：（团队规则集）

主干代码扫描

- 主干变更代码扫描；（POP基础规则集）

框架改进

小团队时期

单服务器&单执行机

大团队推行

多服务器（主/备）

执行机资源池
（独占资源池、共享资源池）

外部系统对接
（获取批量系统数据）

扫描提速
（Maven任务多线程执行）

拆分系统
（独立编译中心）

简化运维

手动配置

手动配置代码扫描任务

手动导出代码质量数据

脚本化&系统化

自动配置代码扫描任务
（Python Jenkins Client）

自动导出代码质量数据
（SonarQube api）

对接gitlab系统
（Python gitlab Client）

提供一键接入功能及接口

百人团队落地

由面向用户的前端系统	转变为	主要提供服务的后台中心系统
由要求不新增违规	转变为	清理遗留代码违规
由TOP9规则集为主	转变为	POP基础规则集（50）为主
由以代码质量为主	转变为	代码质量、安全质量、可维护性多维度
由测试推动为主	转变为	研发主动要求接入代码扫描

效果：研发团队

内建质量提升

- 过滤低级违规
- 清理遗留违规
- 清理冗余代码

交付效率提升

- 减少返工
- 早发现、早修改

代码质量意识提高

- 主动扫描代码
- 不断提出改进需求

影响力提升

- 优秀团队展示平台
- 对外输出实践

效果：测试团队

尽早介入项目

- 尽早发现风险
- 利用工具协助提升判别代码质量能力

高研发测试比

- 过滤低级错误
- 抽出时间开发测试效率工具

全年无线上事故

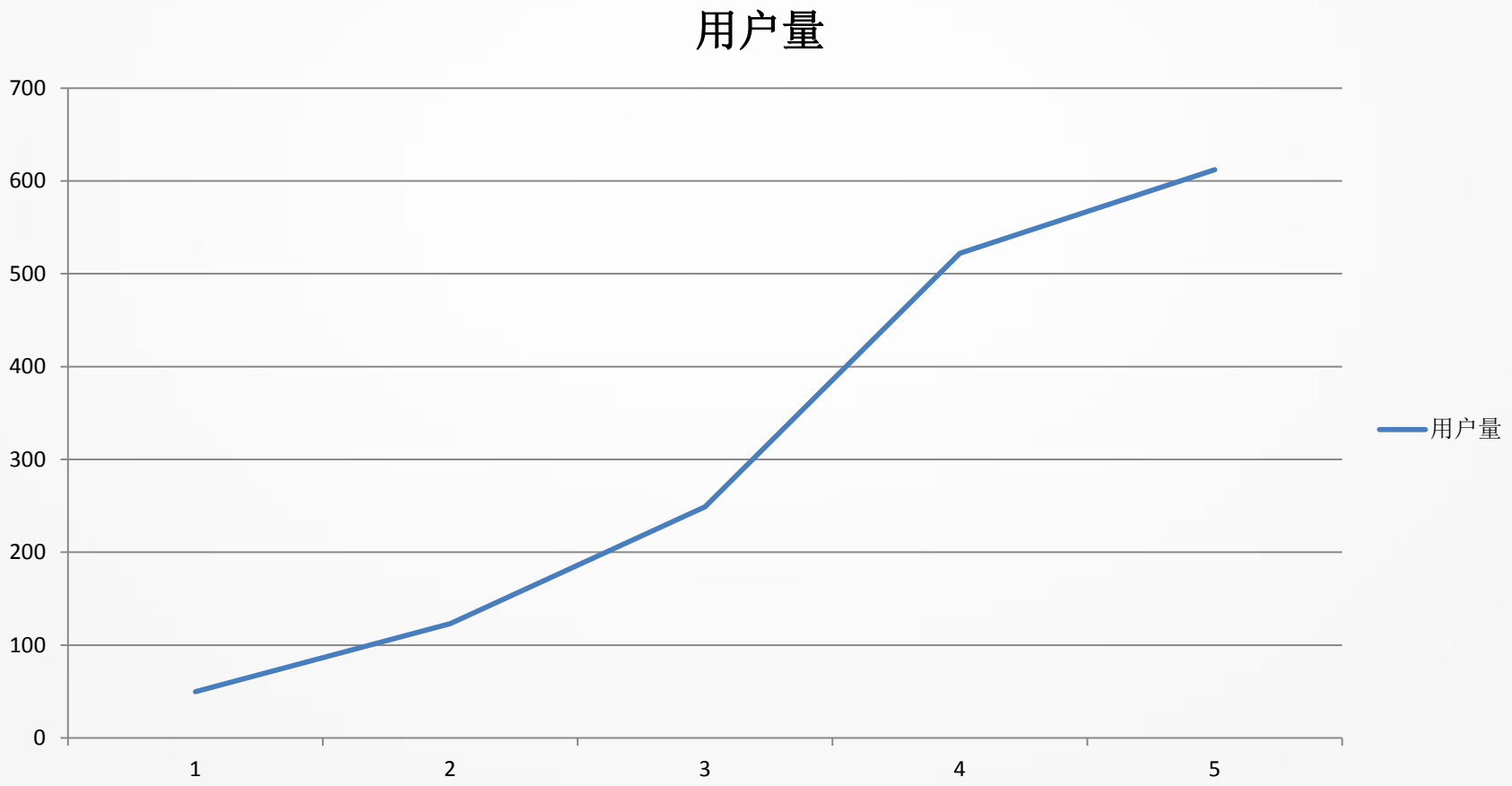
- 通过关注代码提升测试覆盖率
- 更加关注业务逻辑问题

影响力提升

- 完整解决方案输出
- 单个工具输出

新的挑战：代码质量公共平台

用户数据



思考？

代码质量 重要吗？

软件测试 重要吗？

感 谢