研发效能-数字化研发过程

讲师简介

唐洪山
运维部负责人

- ✓ 曾在qunar、京东科技从事运维、安全、质量、研发效能提升相关工作
- ✓ 作为国内最早一批的DevOps/DevSecOps研究人员
- ✓ 同时还是多个国内外技术峰会的顾问专家或出品人

目录

1 为什么做研发过程数字化

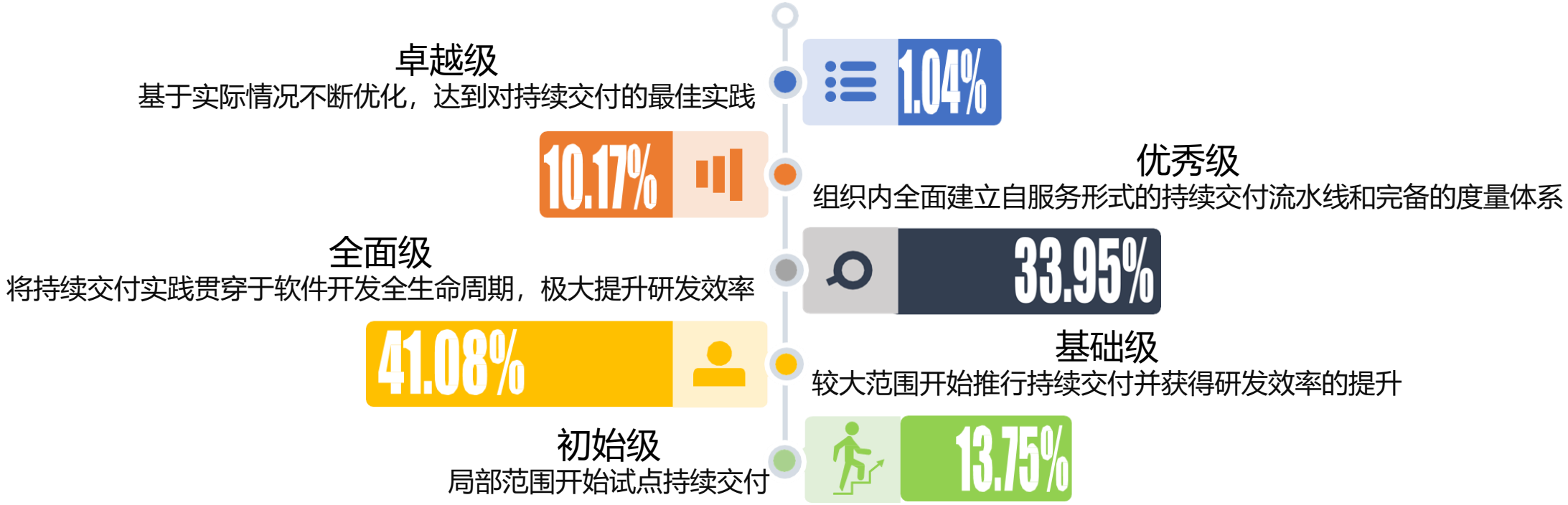
2 可信的、精细的研发过程管控

3 切实有效的度量和改进

4 未来规划

行业痛点-研发效能成熟度现状

超8成企业采用持续交付实践并获得研发效率的提升，但5成多企业处于基础级和全面级



行业痛点-如何实现快速价值交付

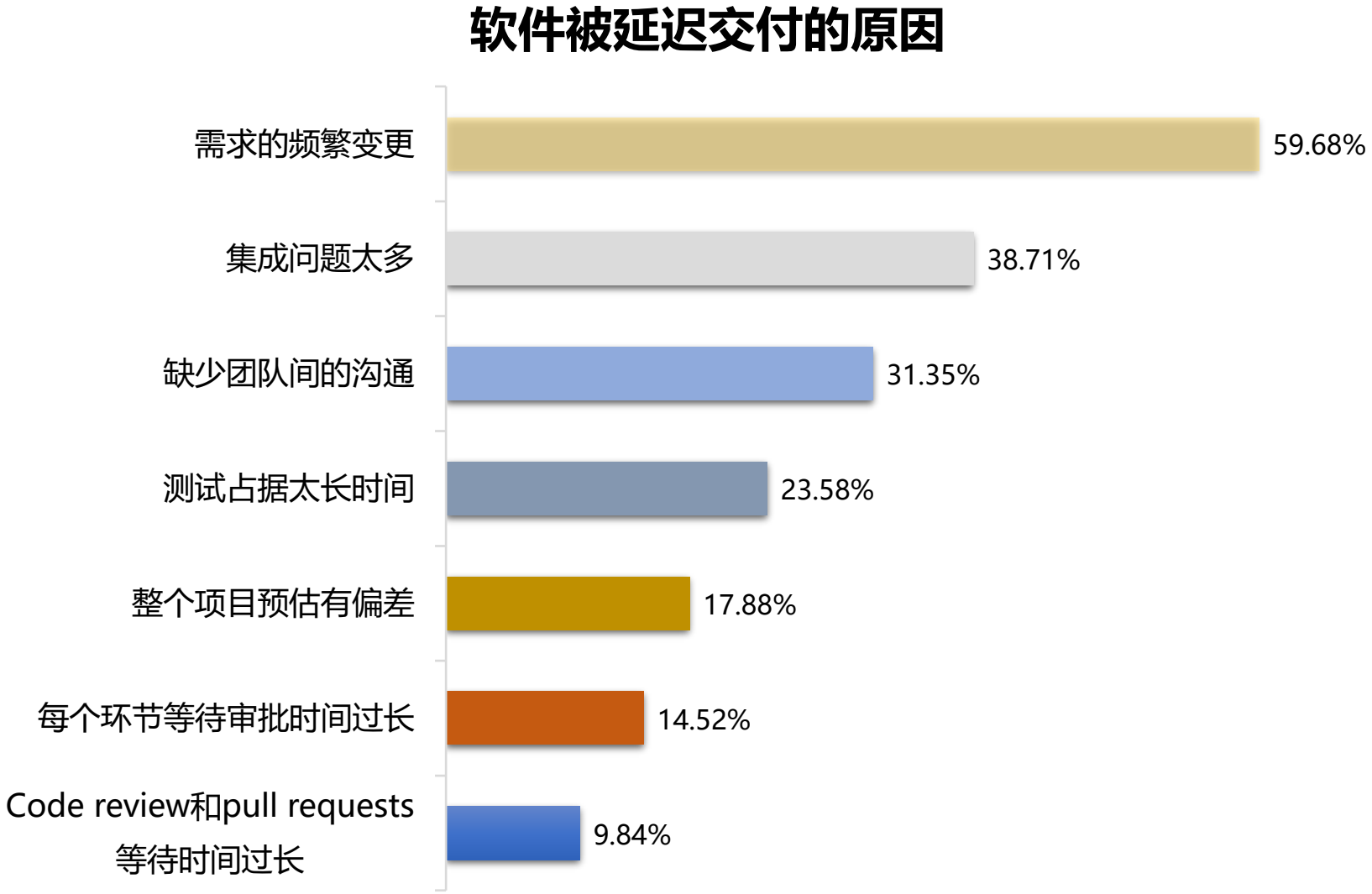
msup

背景：企业需要快速创新，但往往缺少精确的用户洞察，需求模糊或多变，业务人员期望快速试错、验证、反馈，而软件研发各环节普遍的痛点问题，影响了将有价值的产品快速交付到最终用户手中

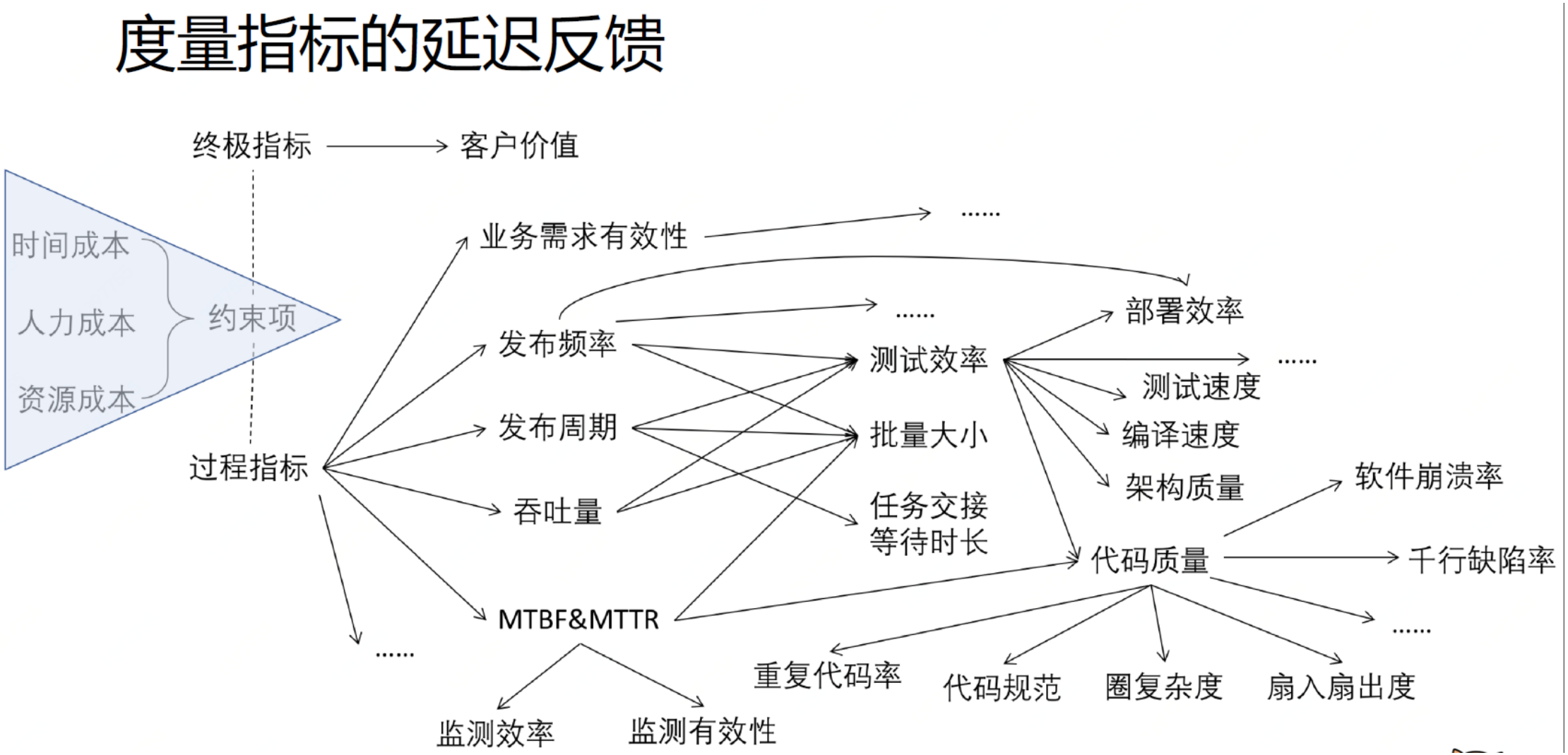
需求	开发	测试	发布	运维
需求交付周期长 需求流转不透明	需求变更频繁 团队并行开发，版本分支管理容易混乱 编码规范不统一，代码质量得不到保证 频繁手工构建、部署，耗时长 接口联调依赖上下游环境和数据构造	环境配置不统一，多环境管理困难 用例、缺陷与需求未关联，不易追溯 手工测试耗时长，回归量大 开发人员提测质量无法快速验证 测试人员技术水平参差不齐，质量无法保证	手工打包发布，人为操作风险高 串行发布上线，业务暂停时间长 发布失败无法快速回滚，业务中断风险高 生产与预发、测试环境配置不一致，导致发布失败	生产环境复杂，手工运维压力大 缺少实时监控和告警，无法及时发现风险

团队协作沟通成本高　项目过程缺少数据度量

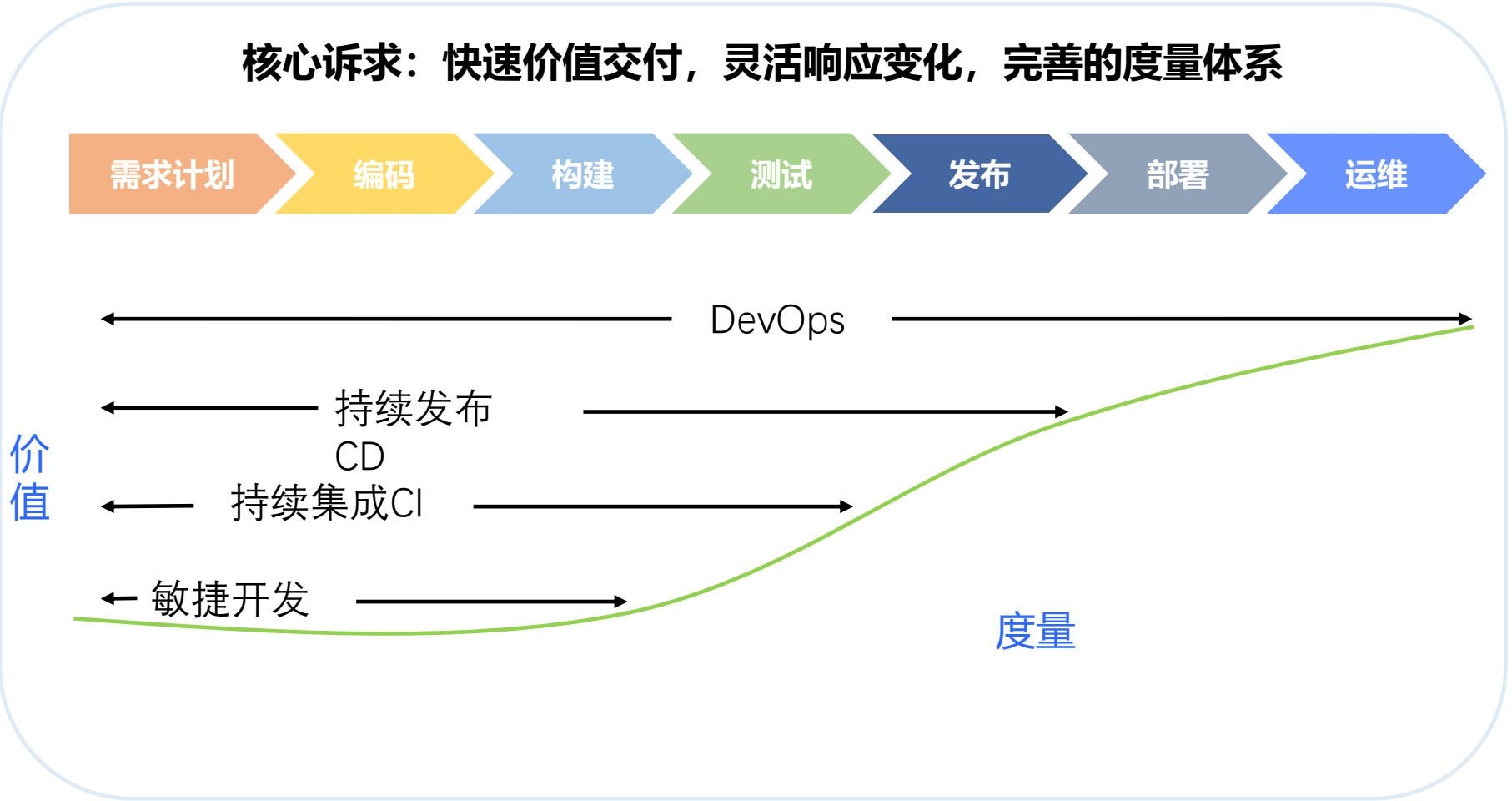
行业痛点-影响需求快速交付的因素



行业痛点-度量指标的延迟反馈



核心诉求-解决端到端的价值交付



目录

1 为什么做研发过程数字化

2 **可信的、精细的研发过程管控**

3 切实有效的度量和改进

4 未来规划

精细化研发过程-平台能力

msup

统一门户

代码托管中心 | 项目管理 | 持续集成与交付 | 后台管控中心 | 统计中心

中央仓库 | 持续集成 | 持续交付

代码托管

- Git + SVN
- 自助权限控制
- 固化仓库状态
- 分布式可扩展

能力支撑

- 接口控制
- 实时同步
- 代码评审
- 分支策略

项目管理

- 需求、迭代
- 任务拆分
- 测试用例、计划
- Bug管理

研发过程

- 变更管理
- 提测流程
- 多卡点配置
- 安全质量规则

发布交付

- 变更管理
- 发布流程配置
- 物料交付
- 基线控制

统计反馈

- 能效统计
- 代码统计
- 横纵向对比
- 质量分析

精细化研发过程-流程控制

msup

1 项目管理
项目立项、需求调研及确认

产品定位 → 调研报告

1 项目立项

2 需求分析会

需求 → 组织会议 → 项目排期

组织会议 → 里程碑

2 敏捷迭代
使用敏捷思想完成快速迭代

开发 → 任务 → 敏捷看板

3 需求拆解

测试 → 测试用例 → 测试计划

3 测试准备

4 产品迭代

3 持续集成
将迭代过程持续化、流水化

迭代

5 代码托管

权限认证 → 代码库

代码库 → Merge Request

代码库 → 代码评审 → 评审报告

6 代码扫描

代码扫描 → Sonar Coverity Fortify → 缺陷报告

7 持续构建

自动构建 → 依赖树采集 黑名单检查 冲突类检查

分支策略 单元测试 安全扫描 → 部署包 容器镜像

服务依赖 → 环境生成

8 测试过程

自动化测试 接口测试 压力测试 → 功能测试 → 测试报告

4 持续部署
通过自动化部署的手段持续交付

9 环境管理

配置中心 服务器 IP池

上线审批

服务器 容器调度

Nexus Yum源

10 持续部署

5 价值反馈
多维度度量反馈

辅助决策

产品演进

13 信息反馈

12 数据统计

效率统计 质量统计 人效分析

研发过程 服务监控 机器监控

11 数据沉淀

精细化研发过程-质量关卡

msup

代码评审

- 代码在线评审
- 多种分支
- 代码评审提测卡点

代码扫描

- 安全漏洞扫描
- Sonar/Coverity
- 代码扫描提测卡点

规则检查

- 分支策略检查
- Maven安全规则检查

测试

- Junit单元测试
- 功能测试代码覆盖率
- 单测覆盖率提测卡点

构建过程

- 分支策略检查
- Maven安全规则检查

提测准入

- 代码扫描缺陷数
- 代码评审
- 单测覆盖率

发布准入

- 功能测试代码覆盖率
- 测试过程产生Bug清零
- 上线审批

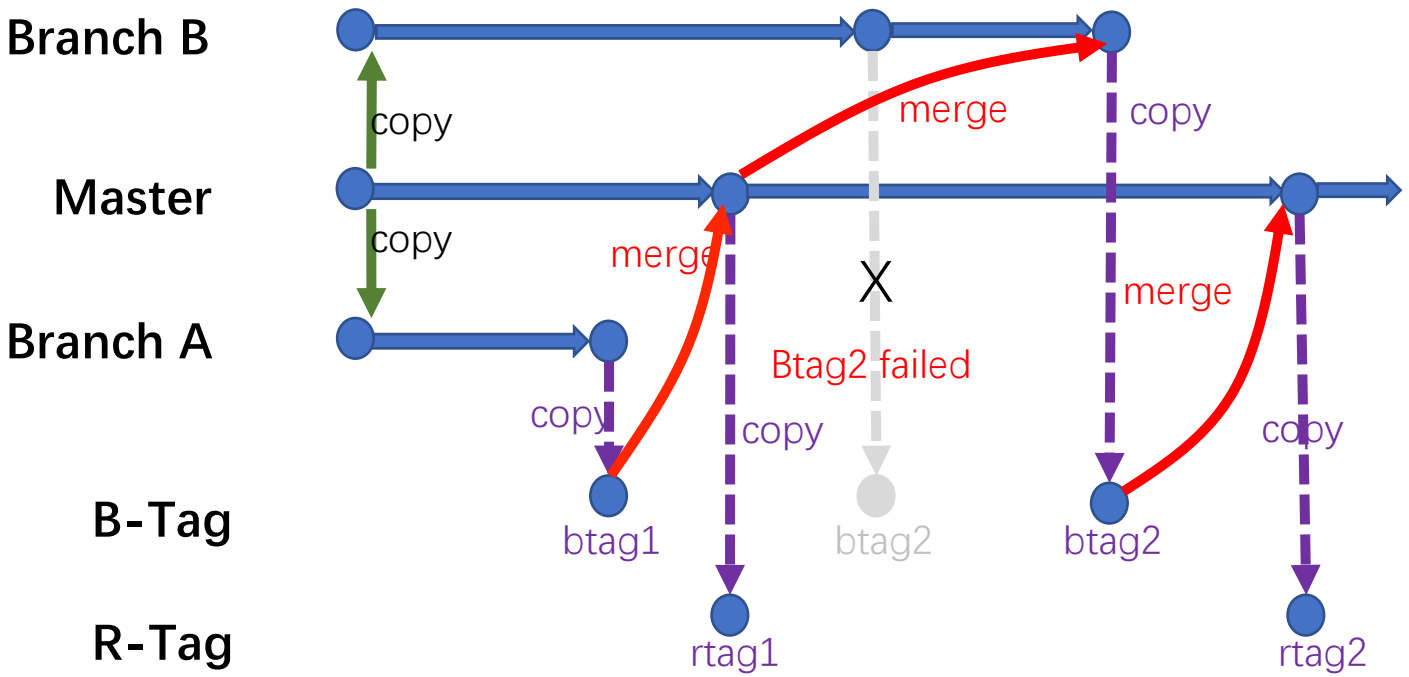
精细化研发过程-安全审计

msup

安全策略左移，将安全对应融入全生命周期

精细化管控-分支策略

ToC模式下的分支策略



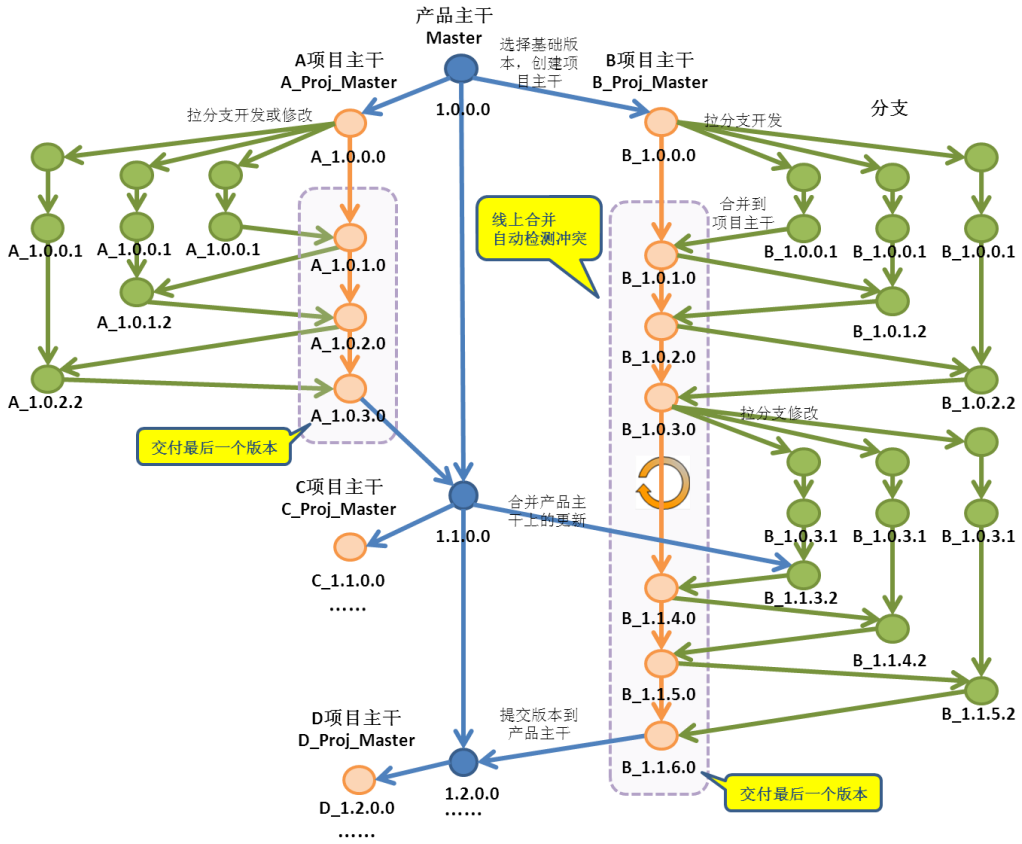
分支策略特点：

- Master分支开发同学只能只读，平台可写。
- Master分支代码与线上保持一致。
- 功能分支来源Master分支，开发同学可写。
- 支持并行开发。

精细化管控-分支策略

msup®

ToB模式下的分支策略



产品+项目双主干开发模式：

产品主干应用于共性需求的迭代更新，项目主干可支撑项目中产品的定制化需求的开发。

主要解决两方面问题：

- 一是分支策略不规范，产品迭代不稳定；
- 二是重复创建工程，项目成果难以积累，沉淀产品。

精细化管控-安全规则

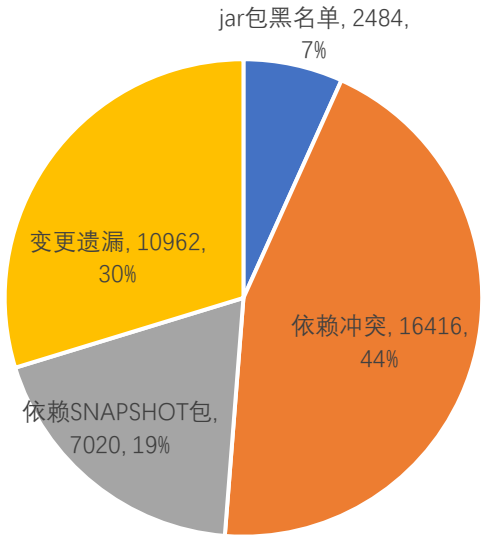
msup

可用的maven编译规范 暂时不可用的maven编译规范

jar包黑名单 禁止依赖的jar包	**groupId规范** groupId规范	**version规范** 项目本身的version规范	**插件白名单** 必须依赖的插件
禁用节点 pom里禁用的节点	**packaging类型** 支持的packaging类型	**子模块packaging类型** 检查子模块的packaging类型	**强制修改版本号** 强制修改pom的version号
依赖冲突检查 检查jar包依赖冲突	**重复类检查** 检查重复类并给出警告	**插件黑名单** 禁止依赖的插件	**资源文件白名单** 可以打进组件的资源文件白名单
必须存在的节点 pom里必须存在的节点	**检查maven的release版本** 检查maven仓库的release版...	**jar包白名单** 必须依赖的jar包	

精细化管控-安全规则

多维度预防，实现由问责制向预防制转化，重复故障0出现



- 创新主导并实现JAR包黑名单拦截功能，减少了**2484次**故障的发生
- JAR包版本冲突检查控制到**16416次**，避免线上发生故障
- 变更遗漏监控拦截到**10962次**，避免因变更导致的线上问题重复出现

Maven依赖查询系统开发，为后续故障分析快速、高效、精准定位问题范围提供了强有力的工具

精细化管控-提测准入

校验**代码扫描**结果；如果扫描缺陷数大于配置的对应数据，无法进行提测

扫描人	扫描时间	版本号(1657)	Coverity扫描缺陷			Sonar扫描缺陷			是否通过标准	报告
			高级	中级	低级	高级	中级	低级		
[redacted]	2019-07-17 18:14:24	1657	5 10↓	20 20↓	104 10↑	9 10↓	47 20↑	58 10↑	不通过	

精细化管控-提测准入

应用提测时，校验是否通过**代码评审**；如果未通过，则不允许提测。

精细化管控-提测准入

校验代码**单元测试**结果；如果**单元测试覆盖率**小于配置的对应数据，无法进行提测

单元测试
覆盖率结果

目录

1 为什么做研发过程数字化

2 可信的、精细的研发过程管控

3 **切实有效的度量和改进**

4 未来规划

研发过程质量度量

度量类型	研发过程	度量指标	度量方法	度量目标
研发质量度量	代码	代码扫描健康度	统计时间段内代码扫描健康趋势	度量代码质量
		有效代码率	上线周期内有效代码率行/总提交代码行	度量代码质量
		各阶段提交代码率	开发/测试阶段提交代码量	度量提测代码质量
	提测	提测次数	单次上线提测次数	度量提测质量
	构建	构建失败率	构建失败数/构建总数	度量构建质量
	BUG	Bug存量趋势（新增、未解决、已关闭、已解决）	累计各个阶段的bug按时间进行趋势分析	度量bug是否收敛
		Bug Reopen率	Bug Reopen次数/总缺陷数	度量研发的解决质量
	测试用例	测试用例失败率	用例失败数/用例总数	度量交付质量
	CI/自动化	单元测试覆盖率/失败率	单测完成情况	度量开发过程中开发有完善的资产自测
	发布	发布失败率	发布失败数/发布总数	度量发布质量
		发布回滚率	发布回滚数/发布总数	度量发布质量

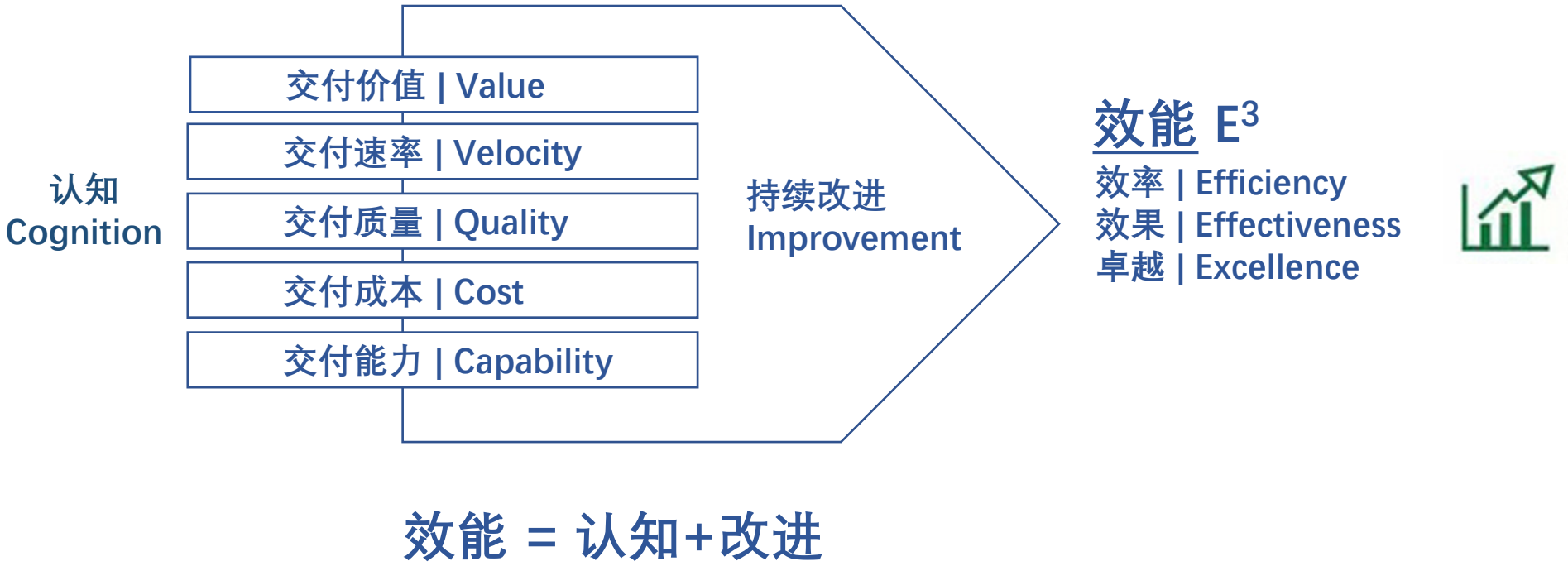
透明研发效率和质量数据、捕捉效能盲点，促进研发过程改进

研发过程效率度量

度量类型	研发过程	度量指标	度量方法	度量目标
研发效率度量	BUG	响应时长	Bug从新建到下一个状态持续时间	度量开发的相应速度
		解决时长	Bug从新建到解决的持续时间	度量开发的解决速度
		验证时长	Bug从解决到关闭的时间	度量测试的验证速度
	需求	响应时长	需求从新建到第一个状态的改变时间	度量产品经理和开发的相应速度
		完成时长	需求从新建到完成的持续时间	度量需求的解决效率
	任务	相应时长	任务第一次发生活动的时长	度量任务的相应效率
		完成时长	任务从新建到完成的持续时间	度量任务的解决效率
	构建	构建持续时长	构建完成时间-构建开始时间	度量构建效率
	部署	部署持续时长	部署完成时间-部署开始时间	度量部署效率
	发布	发布持续时长	发布完成确认时间-上线审批完成时间	度量发布效率

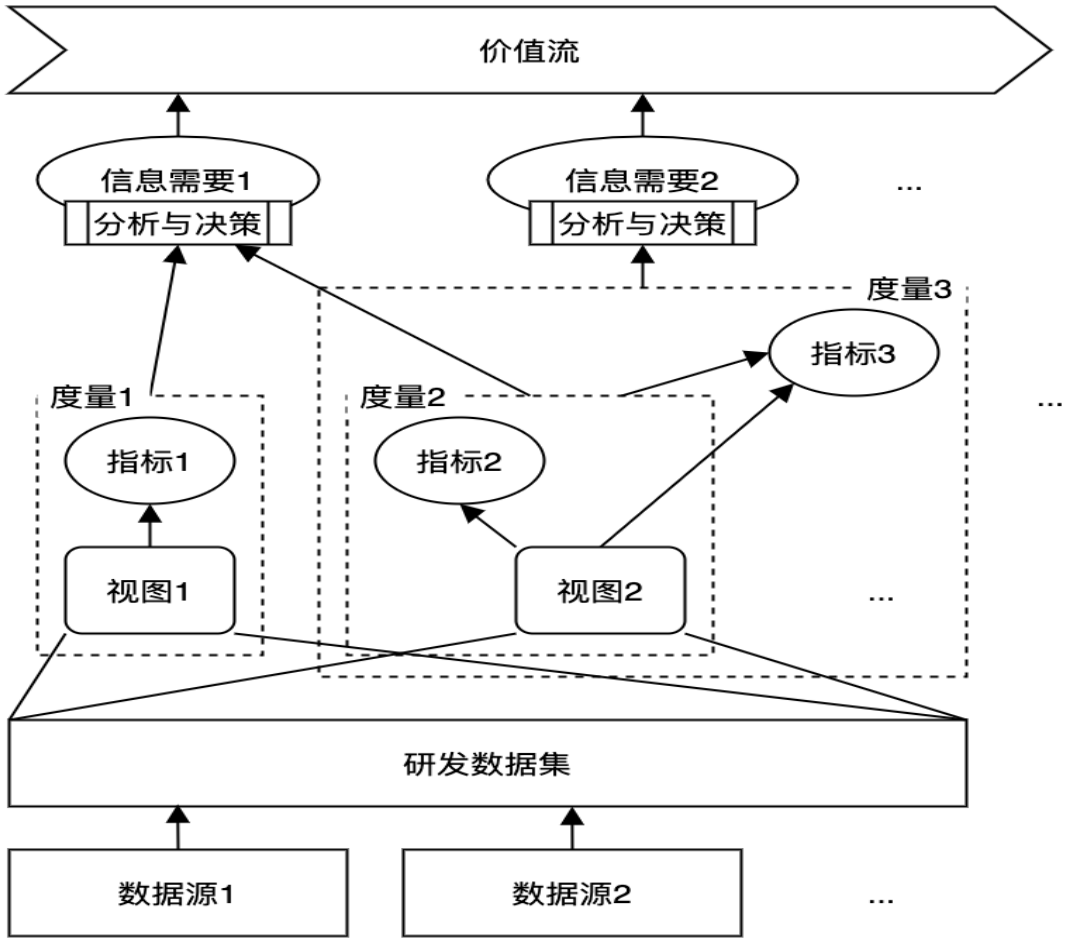
透明研发效率和量数据、捕捉效能盲点，促进研发过程改进

研发过程度量体系



效能 = 认知+改进

研发过程度量指标模型



1. 研发效能指标是为解决研发实践问题、获得研发效能认知、产生研发效能价值而设计的一种可量化的概念和相应的计算方法。

2. 研发效能度量是将指标应用于特定研发数据集合从而计算出数值结果的过程，称为度量。

度量 = 指标 + 视图

对于研发数据集合D，指标的度量m(D)为

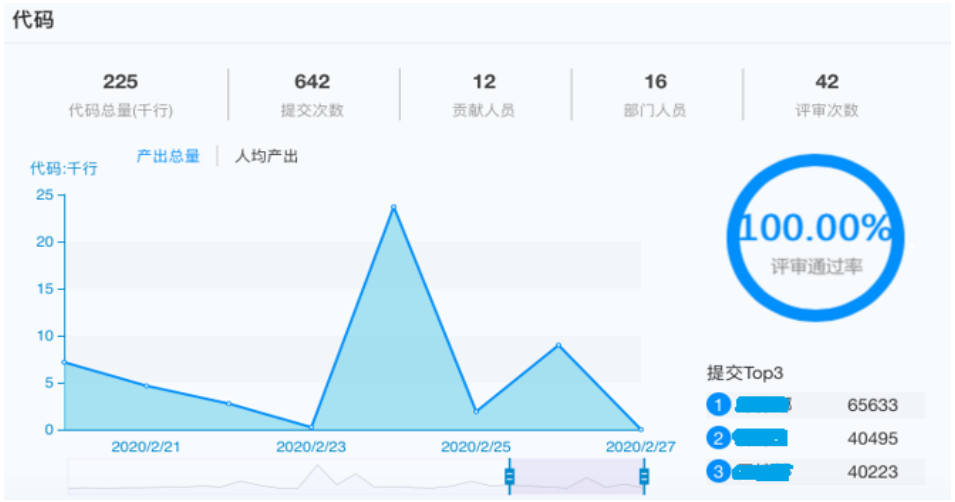
m(D) = f(D，M(D))

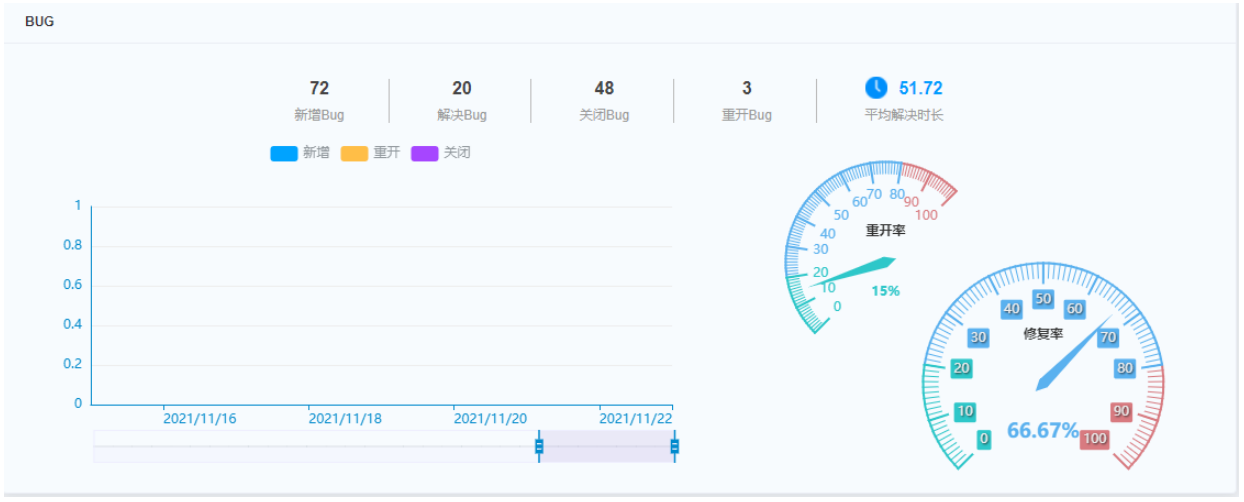
其中，M为其依赖的其他度量的集合，f为指标的计算方法。基础指标的M为空集。

3. 为了对价值流产生认知，用户提出不同的信息需要，从而实施度量，并对度量值进行分析与决策。根据度量指标的定义，一个指标可能依赖于其他指标，并在具体的数据视图上计算出度量值。这些视图是研发数据集的一部分，而研发数据是从各类相关工具、系统或调研中收集汇总。

研发过程度量指标（展示示例）

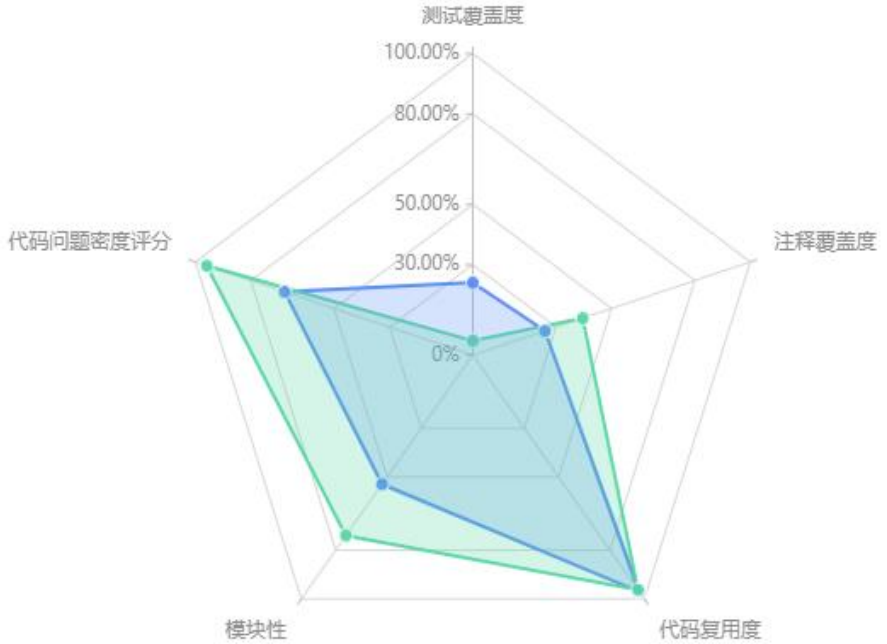
msup®





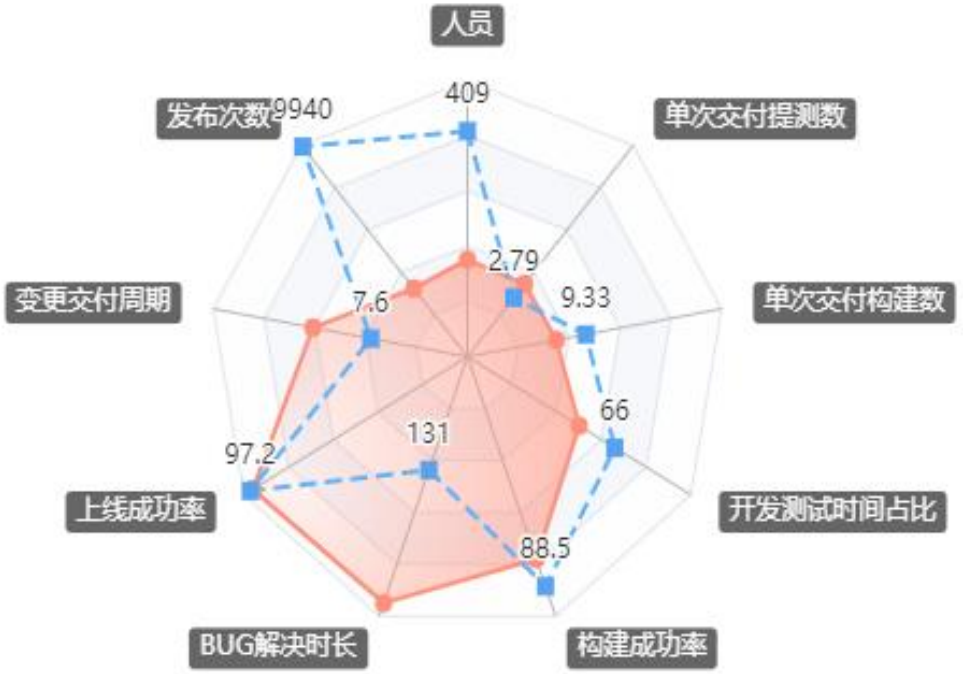
研发过程度量指标（展示示例）

图17-5　组织胜任力评估模型（过程维度）

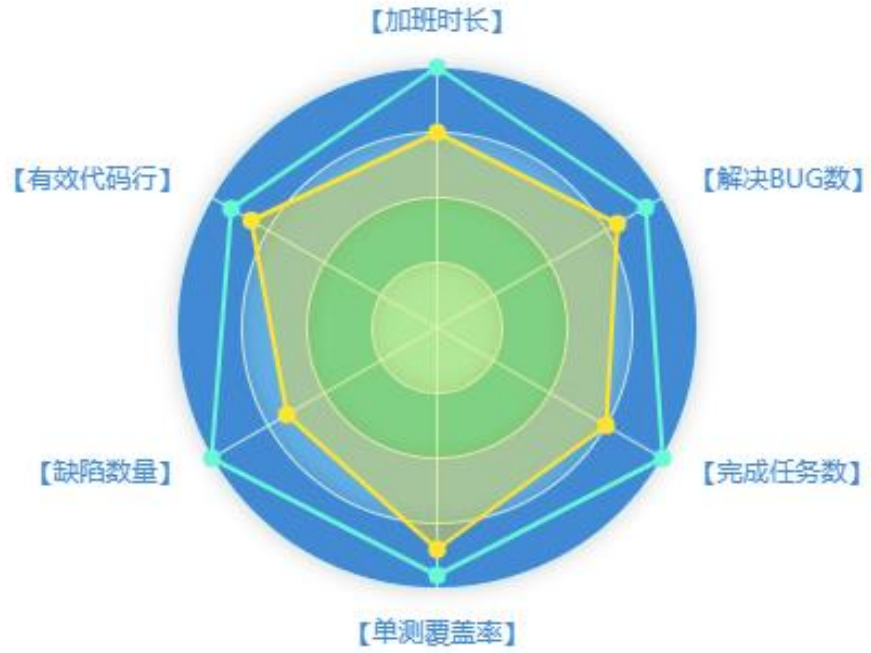


研发过程度量指标（展示示例）

部门效能数据分析对比



个人效能数据分析对比



研发过程度量指标（展示示例）

msup

研发过程

2021-10-22 10:06:21 创建变更

变更人员 [redacted]

变更名称 1.0.0_[redacted]

对应代码库地址 https:/[redacted]/git/devops-data-server.git

代码库分支 1.0.0[redacted]

2021-10-31 00:44:33 提测申请

提测人员 [redacted]

提测Btag [redacted]20211031004433

2021-11-02 16:14:23 测试通过

测试人员 [redacted]

测试报告 点击跳转

2021-11-02 16:21:00 上线申请

申请人员 [redacted]

上线包 [redacted]20211102T144516907.zip

MD5: [redacted]b79c

预发包 ------

2021-11-02 16:23:28 审批通过

审批人员 [redacted]

2021-11-02 16:23:44 上线完成

操作人员 [redacted]

目录

1 为什么做研发过程数字化

2 可信的、精细的研发过程管控

3 切实有效的度量和改进

4 **未来规划**

BizDevOps探索

关注msup公众号

获取更多AI落地实践

麦思博(msup)有限公司是一家面向技术型企业的培训咨询机构，携手2000余位中外客座导师，服务于技术团队的能力提升、软件工程效能和产品创新迭代，超过3000余家企业续约学习，是科技领域占有率第1的客座导师品牌，msup以整合全球领先经验实践为己任，为中国产业快速发展提供智库。