DevData Talks | 思码逸

软件研发效能度量：
认知，标准和改进

任晶磊 思码逸 CEO

在 DevData Talks
开放务实地聊聊研发效能

分享合辑

研效交流群

每月直播 干货满满

行业 KOL + 企业研发效能负责人 + 资深效能顾问
从效能建设路径到不同场景应用，分享前沿观点与先进实践

开放探讨 共同成长

直播 Q&A + 交流社群
直播嘉宾面对面答疑解惑，交流群随时随地探讨效能话题

关于思码逸

作为DevData Talks 的发起方，思码逸专注为企业研发团队提供效能数据分析，呈现效率、质量、人才发展等多视角数据洞察，辅助团队决策与工程改进。

思码逸已服务 腾讯、美团、滴滴出行、中国平安、泰康保险、戴尔EMC等众多行业标杆客户。

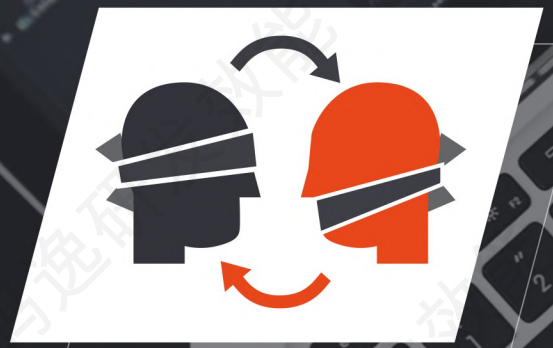
内容提纲
Content outline

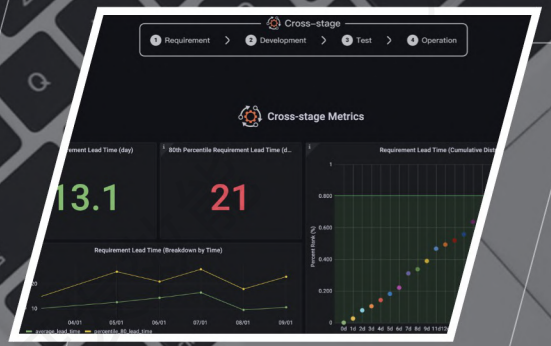
1.研发效能度量认知

2.研发效能指标体系

3.研发效能改进方法

4.研发效能数据平台





研发效能度量认知

Software is eating the world
软件正在吞噬世界

—— Marc Andreessen

研发效能度量认知

What gets measured gets managed
度量才能管理

—— Peter Drucker 彼得德鲁克

By Jeff McNeill, CC BY-SA 2.0

研发效能度量认知

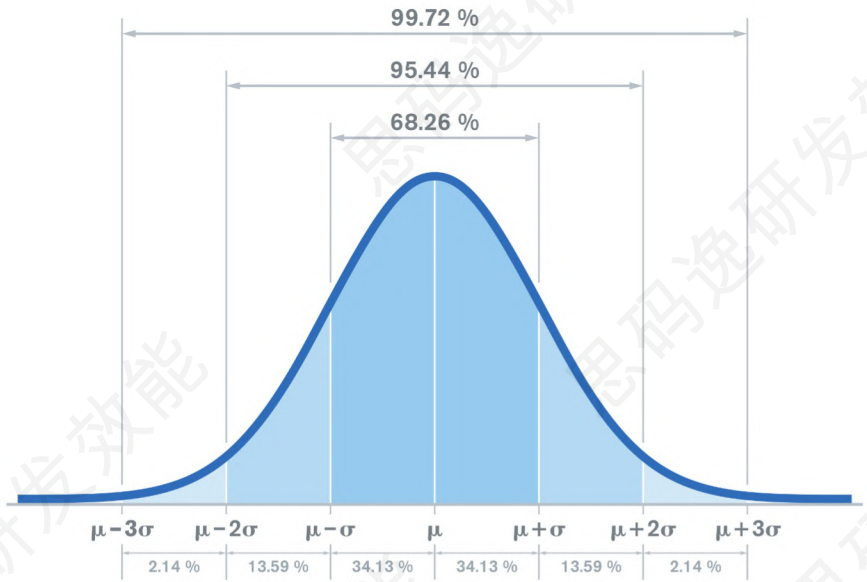
研发效能

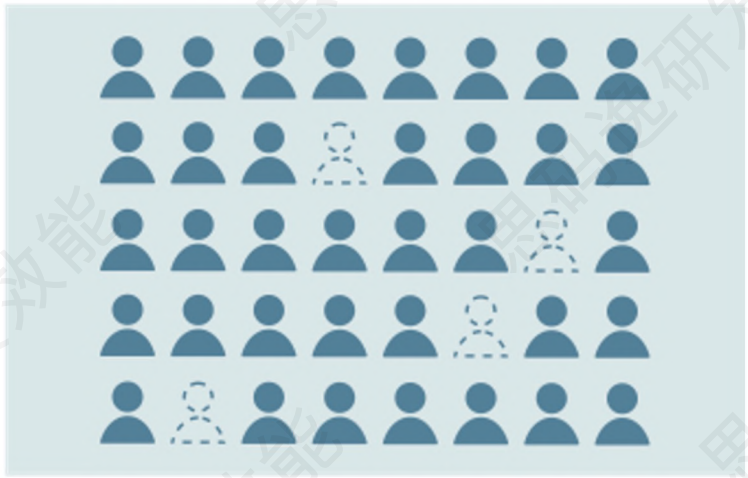
度量?

研发效能度量认知

度量分两种

- 物理度量：追求极致精确。
- 统计度量：接受反例，需要系统思维。





研发效能度量认知

制衡机制

- 代码行数 vs. 代码当量
- 需求吞吐量 + 需求粒度

系统思维　例如：千行代码缺陷率 = 代码质量

- 线上缺陷率
- 缺陷停留时长
- 代码复用度
- 代码评审轮数
- ……

研发效能指标体系

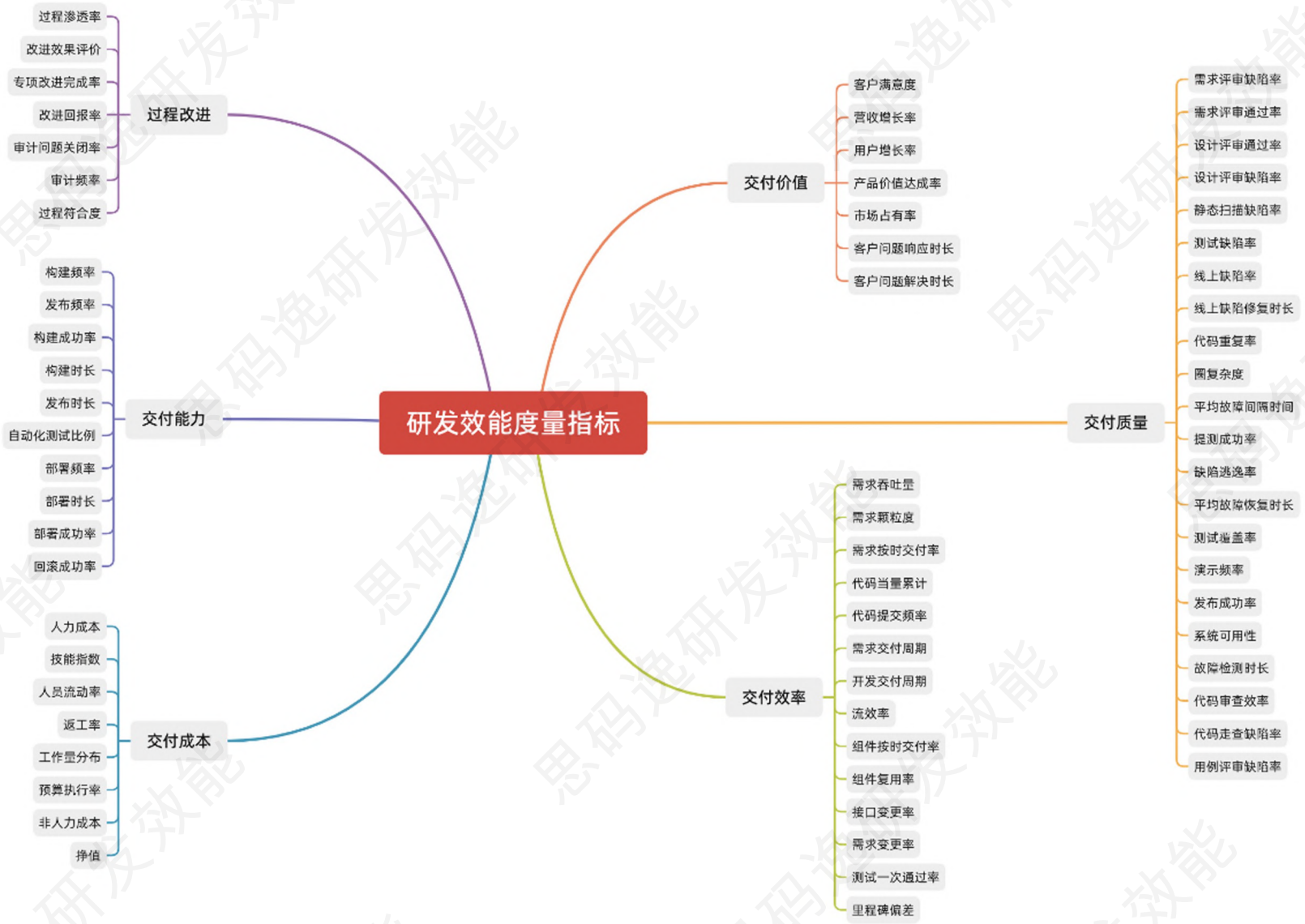
日前，由中关村智联软件服务业质量创新联盟、中国软件协会过程改进分会发起的**《软件研发效能度量规范》**团体标准完成立项，有望成为该领域国内首创、国际领先的相关标准。

研发效能指标体系

软件研发效能度量框架

效能 = 认知 + 改进

研发效能指标体系

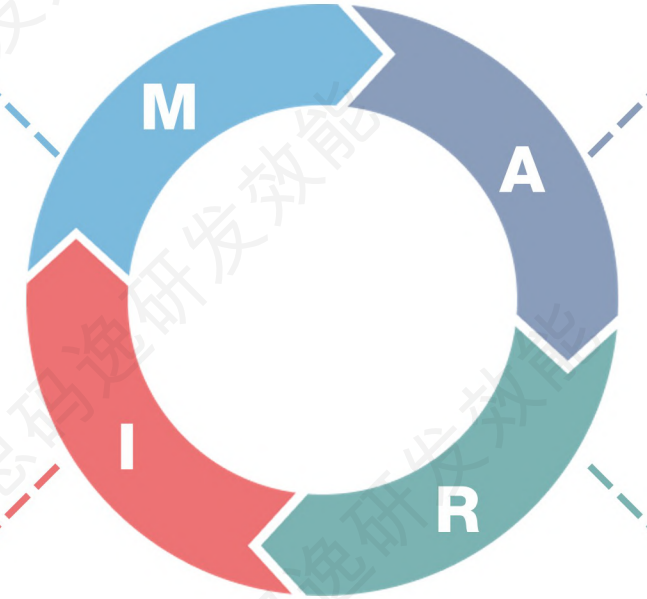

研发效能度量指标
过程改进
过程渗透率
改进效果评价
专项改进完成率
改进回报率
审计问题关闭率
审计频率
过程符合度
交付能力
构建频率
发布频率
构建成功率
构建时长
发布时长
自动化测试比例
部署频率
部署时长
部署成功率
回滚成功率
交付成本
人力成本
技能指数
人员流动率
返工率
工作量分布
预算执行率
非人力成本
挣值
交付价值
客户满意度
营收增长率
用户增长率
产品价值达成率
市场占有率
客户问题响应时长
客户问题解决时长
交付效率
需求吞吐量
需求颗粒度
需求按时交付率
代码当量累计
代码提交频率
需求交付周期
开发交付周期
流效率
组件按时交付率
组件复用率
接口变更率
需求变更率
测试一次通过率
里程碑偏差
交付质量
需求评审缺陷率
需求评审通过率
设计评审通过率
设计评审缺陷率
静态扫描缺陷率
测试缺陷率
线上缺陷率
线上缺陷修复时长
代码重复率
圈复杂度
平均故障间隔时间
提测成功率
缺陷逃逸率
平均故障恢复时长
测试覆盖率
演示频率
发布成功率
系统可用性
故障检测时长
代码审查效率
代码走查缺陷率
用例评审缺陷率

研发效能改进方法

Measure：度量 / Analyze：分析 / Review：回顾 / Improve：改进

度量
Measure
- 需求驱动度量
- 关注度量收益
- 寻找北极星指标

分析
Analyze
- 不是简单罗列数据
- 纵横比较看变化

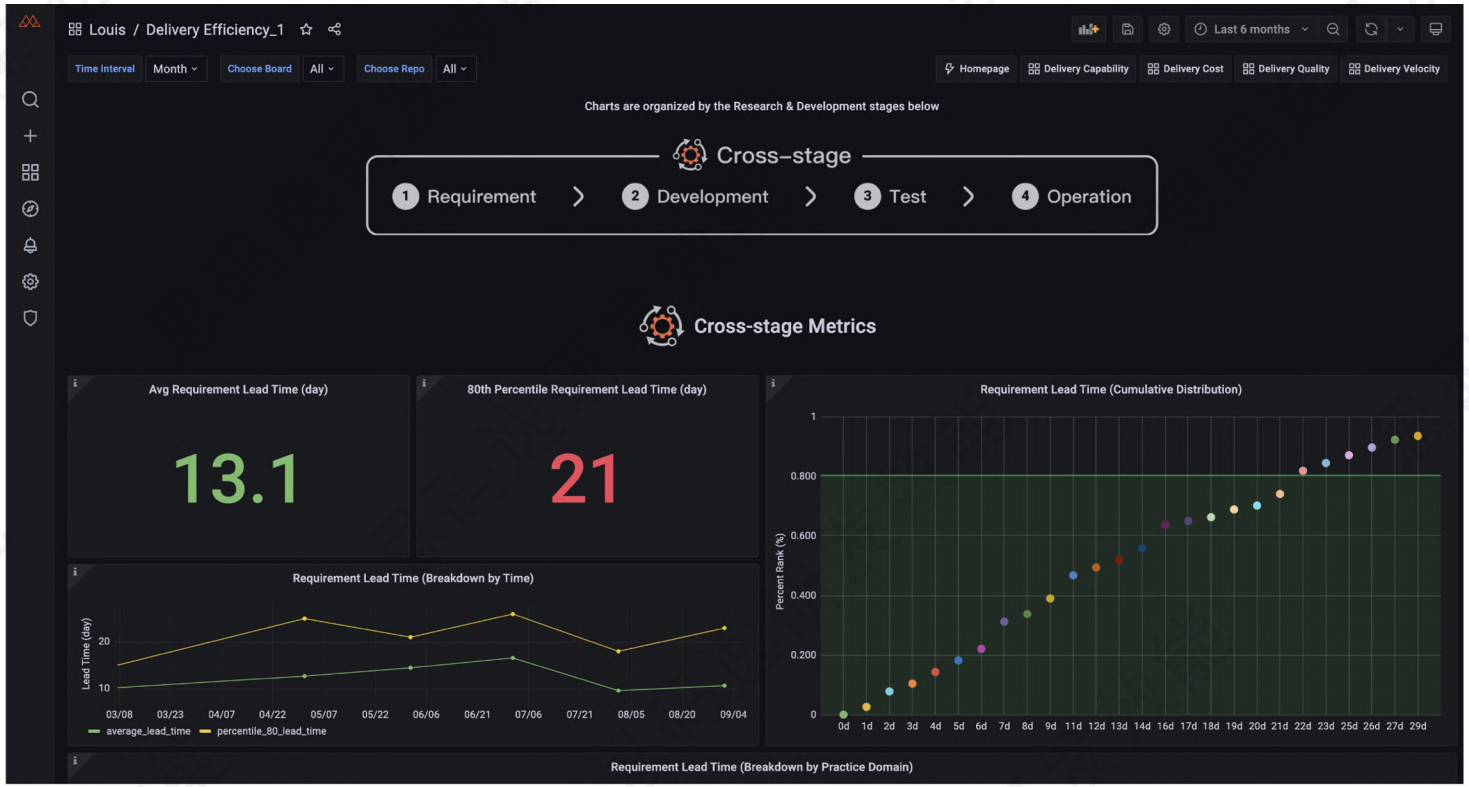


改进
Improve
- 措施具体可落地
- 小步提升有目标

回顾
Review
- 问题分析挖根因
- 过程复盘重改进

研发效能数据平台

开源Dev Lake: **https://github.com/merico-dev/lake**



思码逸

$ 深度代码分析

Thanks

面向程序代码库的大数据分析
挖掘公司最有价值的智力资产
来自微软、加州大学伯克利分校、斯坦福大学、GitHub的世界一流智囊团为您服务

官方网址：**www.merico.cn**　　思码逸客服：**400-8637-426**

思码逸

$ 深度代码分析

研发效能度量实践方法

MARI

关钦杰 思码逸咨询总监

内容提纲
Content outline

1.什么是MARI方法

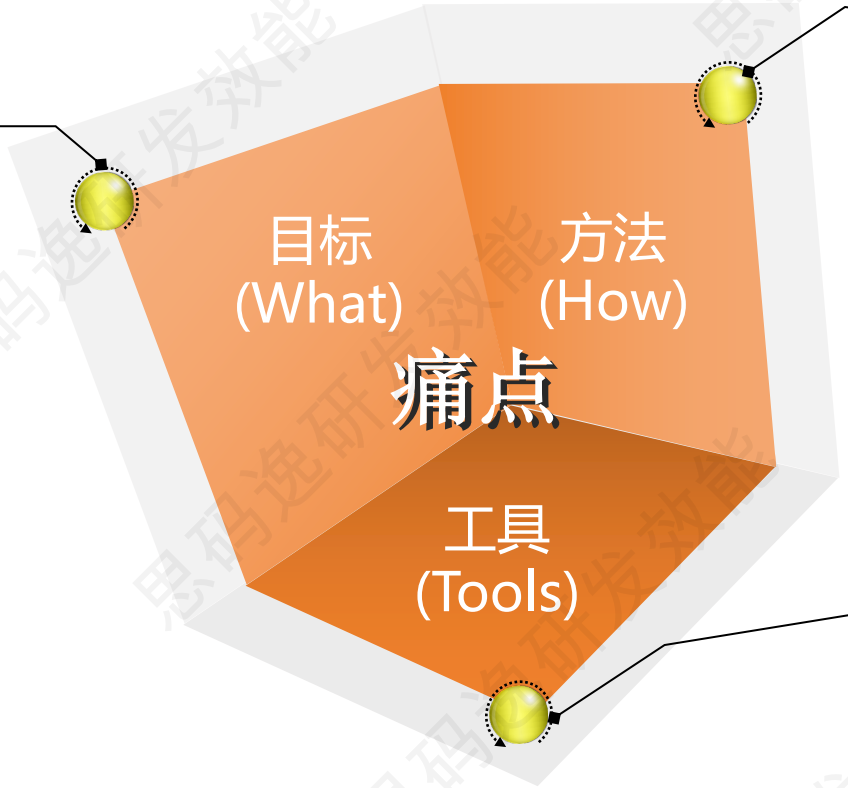
2.MARI方法的步骤

3.MARI应用实例

研发效能度量的典型痛点

思码逸
$深度代码分析

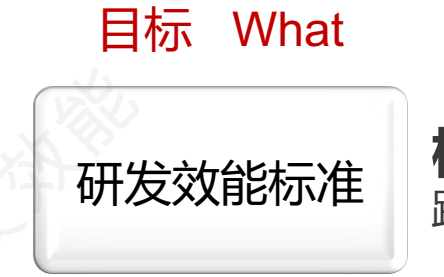
- 不知该使用哪些指标，照葫芦画瓢引入“**明星指标**”。
- 采集大量数据，却不知如何服务于**商业目标**。



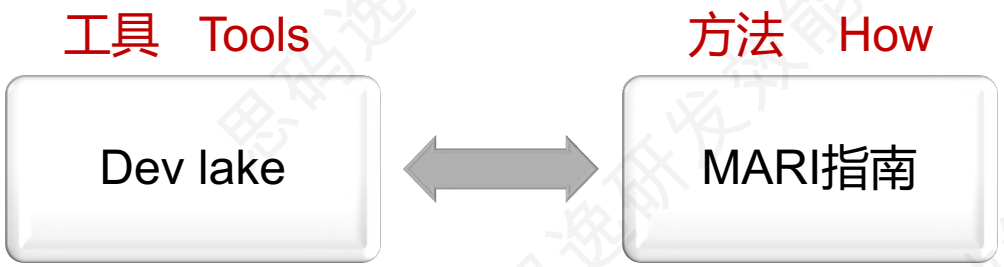
- 针对一系列的效能分析，缺少建立**闭环**的有效方法。
- 永远紧盯“短板”，过度改进，缺少平衡**投入产出**的合理依据。

- 大量数据**手工录入**，占用人力成本。
- 不同工具间数据孤岛没有被打通，无法自动生成**指标视图**。

MARI背景：研发效能标准体系



框架层：覆盖五大认知域、六大实践域、数十个过程指标。



工具层：自动集成JIRA、Gitlab、Merico企业版的开源数据可视化图形面板。

实践层：基于MARI方法的指标度量、分析、回顾、改进闭环实施指南。

MARI方法：度量改进的闭环

Measure：度量　**A**nalyze：分析　**R**eview：回顾　**I**mprove：改进

M：度量

需求驱动度量

关注度量收益

寻找北极星指标

A：分析

不是简单罗列数据

纵横比较看变化

I：改进

措施具体可落地

小步提升有目标

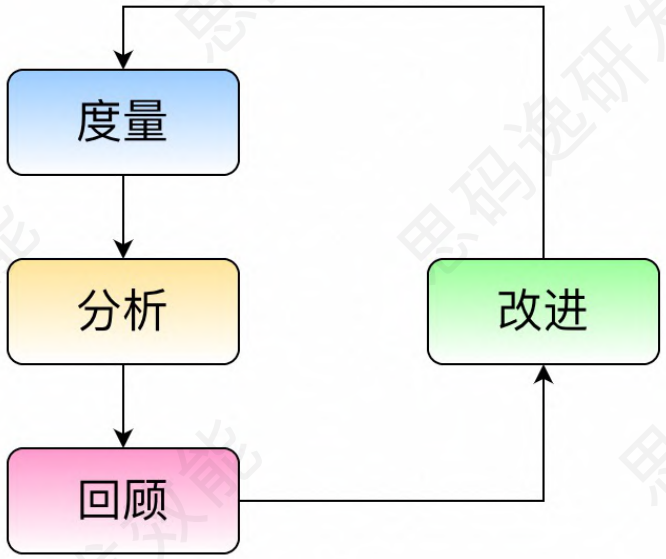
R：回顾

问题分析挖根因

过程复盘重改进

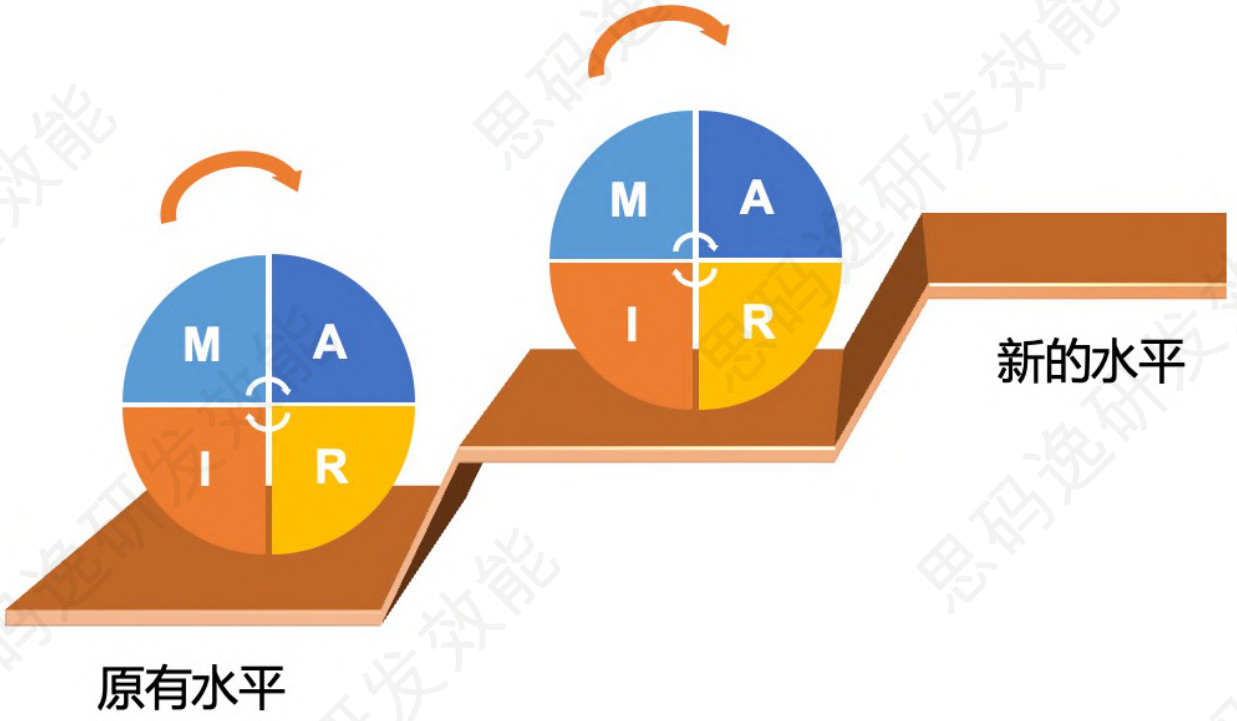
MARI的优势

层层推进，追问根因



MARI数据驱动环

构建闭环，持续改进



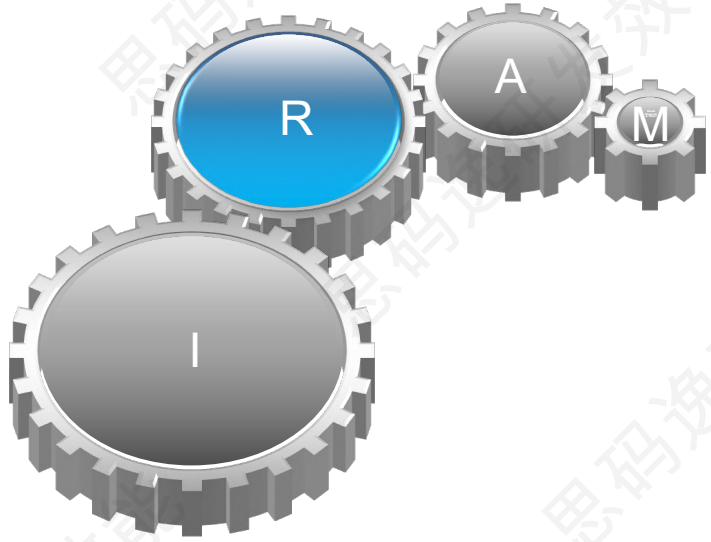
MARI四步骤

1. 无论任何改进活动，首先需结合组织及团队实际需求，面向改进目标通过量化数据对过程及目标进行刻画，并统一数据及指标的采集方法，即建立**度量**。

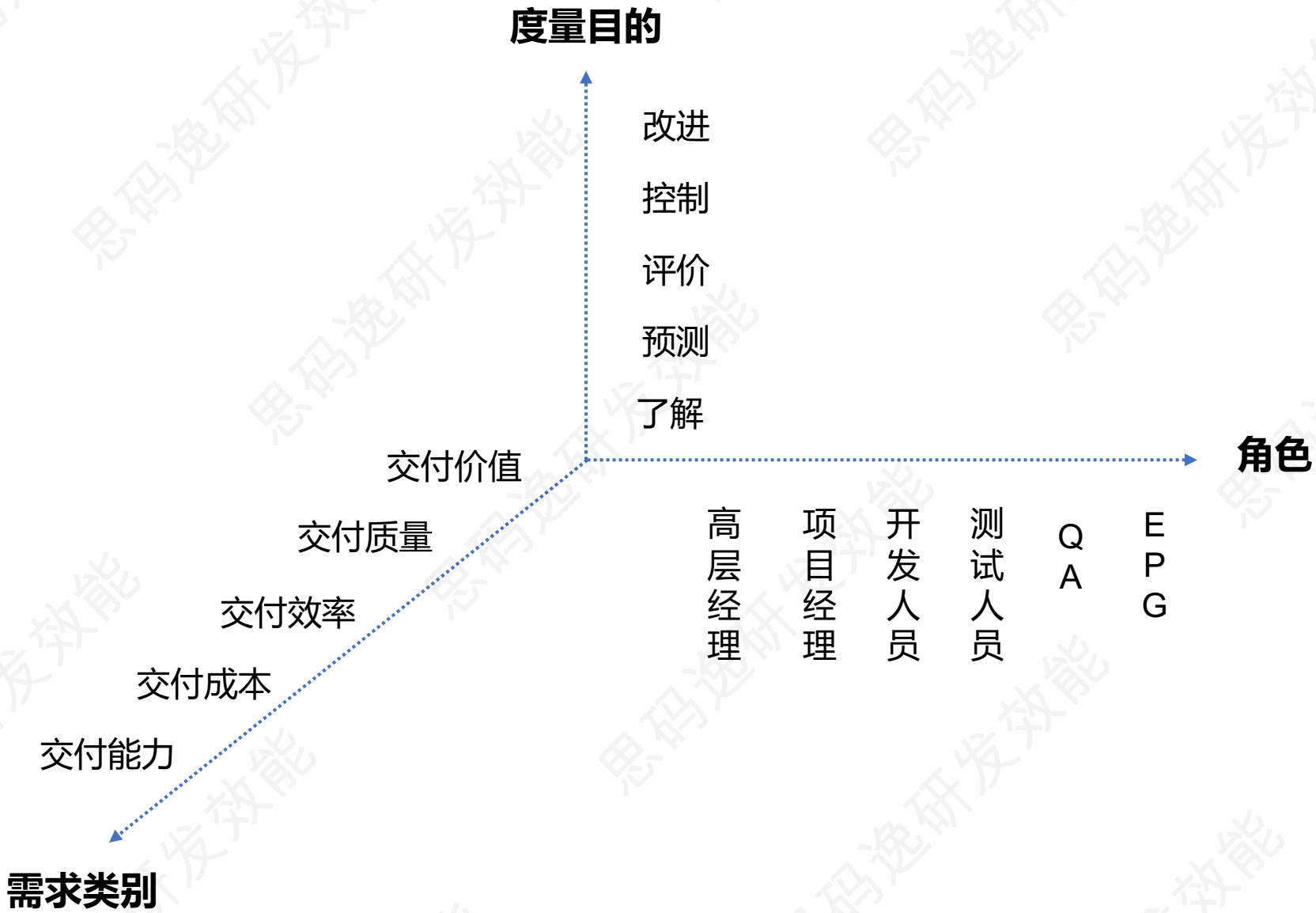
2. 有了量化指标，运用统计分析方法，对数据的趋势、分布、关联等信息进行**分析**，得到对现状的量化理解。

3. 基于分析结果，对产生“果”（结果）的“因”（影响因子），进行**回顾**，挖掘对结果产生影响的根本原因，定位关键问题。

4. 针对关键问题，建立可落地的**改进**措施，通过调整“因”（影响因子），最终影响“果”（目标）的达成，并进入下一轮度量验证。



Measure：度量需求的“三维一体”



Measure：高层经理的度量需求案例

	了解	预测	评价	控制	改进
交付价值	公司平均客户满意度 某个项目是否发生了客户投诉		客户满意度是否有提升		提高客户满意度
交付质量		某个项目上线后的质量水平	公司平均质量水平与业内的标杆对比		提升质量
交付效率			公司平均生产效率与业内的标杆企业对比		提高效率
交付成本	公司年度可提供的工作量 某个项目是否超出了预算	公司年度累计支出 公司全年所有项目需要的工作量		公司总体年度支出不能超预算	降低成本
交付能力	公司平均人员流动率		公司人员比例是否合适	公司人员数量增长不能太快	

Measure：需求前置时间案例

思码逸
$深度代码分析

➢ 获取需求

	了解	控制	评价	改进	预测
高层经理	需求平均交付周期是多少天？		与历史及行业比较，需求交付周期是否合理	消除需求交付周期中的冗余等待	组织需求吞吐量。
产品经理	团队的需求吞吐能力与吞入需求量是否匹配？	可定位到交付慢的Epic。			
项目经理	80% 的需求在多少天内可交付？	可定位到交付慢的阶段。		合理缩短需求交付时长	

➢ 梳理指标

- **需求前置时间平均值 Lead time (day)**

  了解 Lead time 均值。

- **需求前置时间（百分位分布） Requirement Lead Time Distribution**

  观察需求交付时长的分布，了解 80% 的需求在多少天内交付。

- **需求前置时间（按 Natural Time 分解）**

  观察需求前置时间是否随着时间越来越短。

- **需求前置时间（按 Epic 分解）**

  定位交付慢的 Epic。

- **不同阶段需求停留时长（按 阶段 分解）**

  定位交付慢的阶段。

- **需求数量（按 计划及实际 度量）**

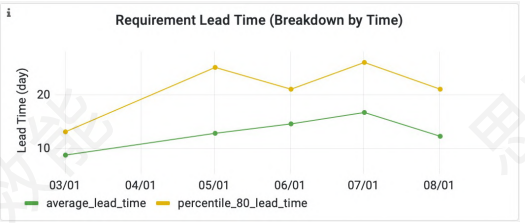
  了解需求吞吐量。

效能度量的5原则

1. 度量不是免费的，尽可能**自动化**
2. 选择**适当的**，而非“标准的”指标，若发现指标没用，果断舍弃。
3. 务必了解指标意图和针对的**商业目标**。
4. 设计“**北极星指标**”，指标数量越多，边际收益递减。
5. **变化趋势**的价值高于指标绝对值。

Analyze：需求前置时间分析案例

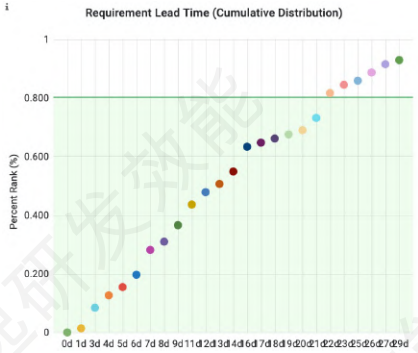
需求前置时间



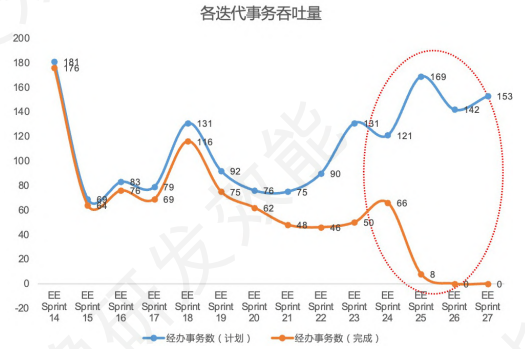
趋势分析

周（迭代）/月/季度趋势

同比环比分析



下钻分析



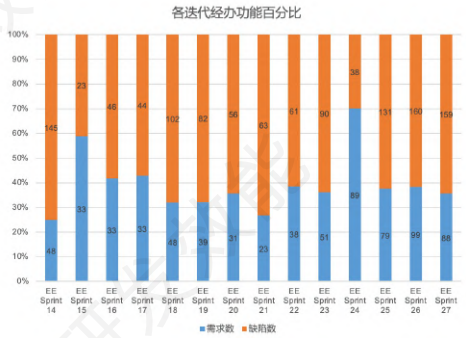
关联分析

需求数量、需求吞吐量

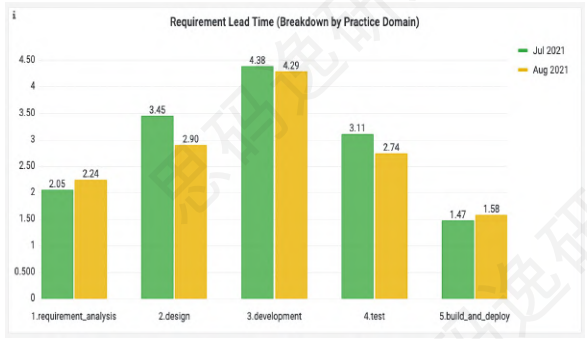
线上故障密度

代码质量（技术债）

按项目下钻



按阶段下钻

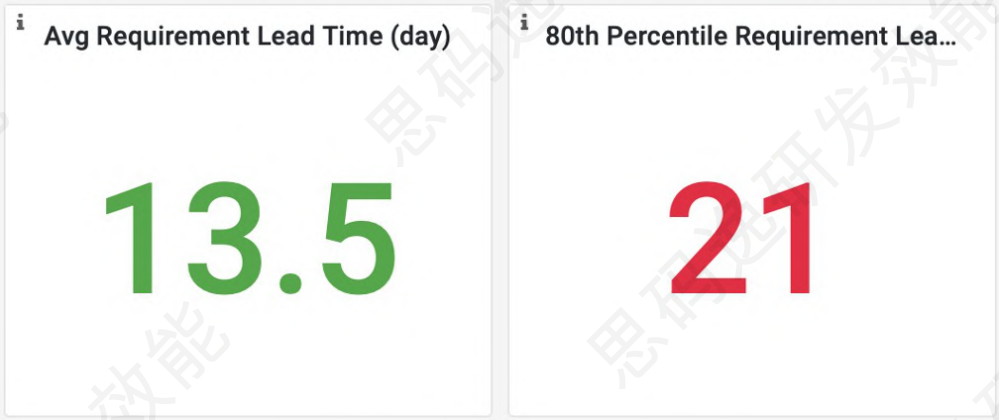


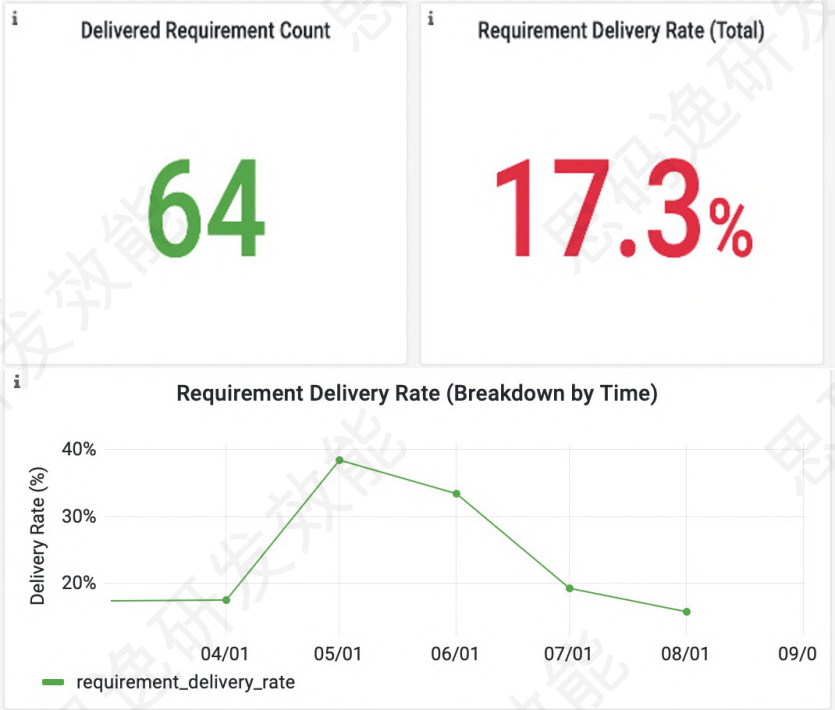
按团队下钻

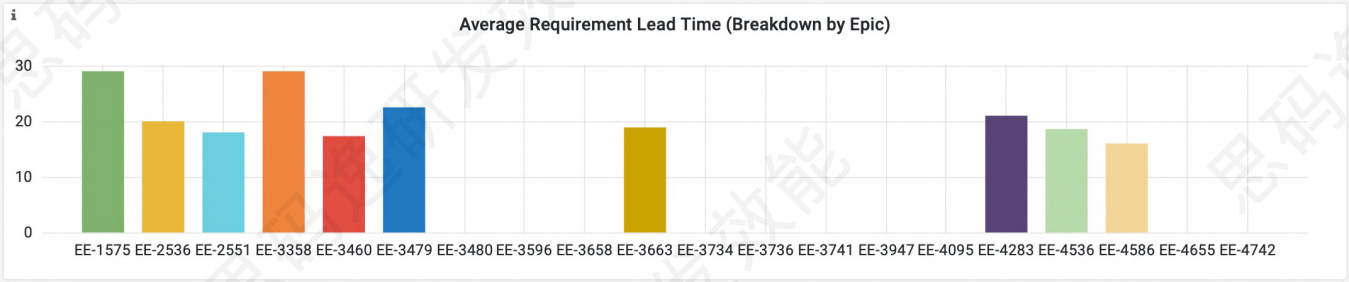
Analyze：需求前置时间分析案例

思码逸
$深度代码分析

- 80%需求前置时间达到21天
- 需求交付率仅为17%，较历史下降47%



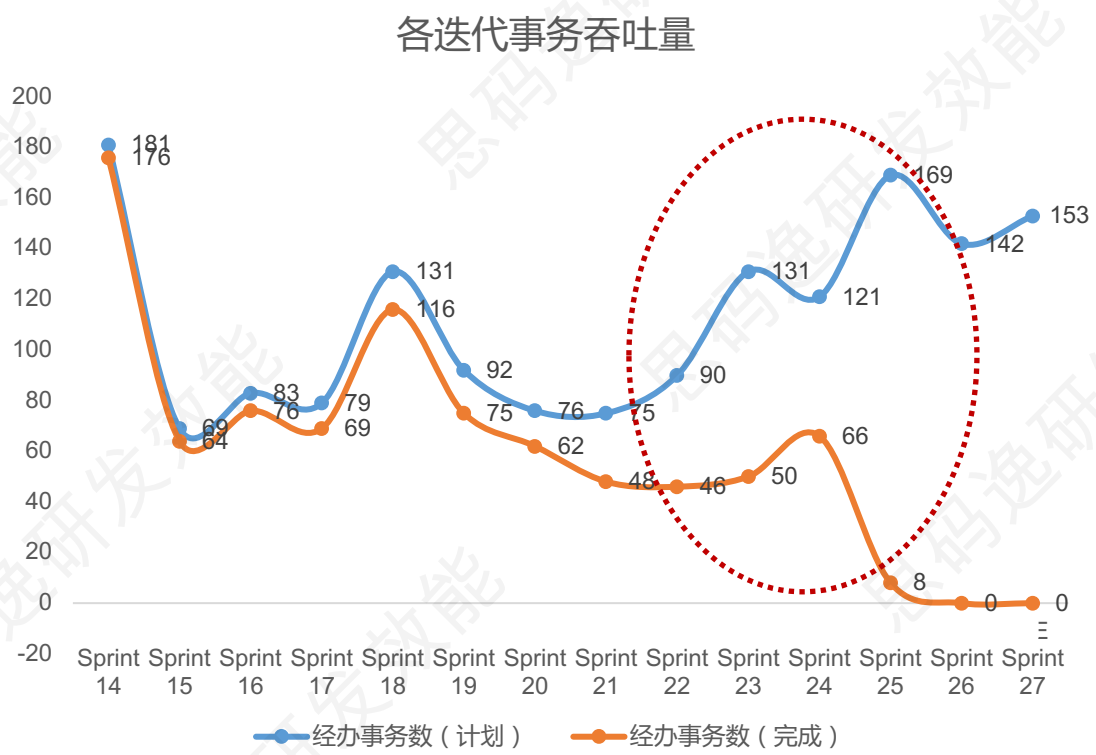




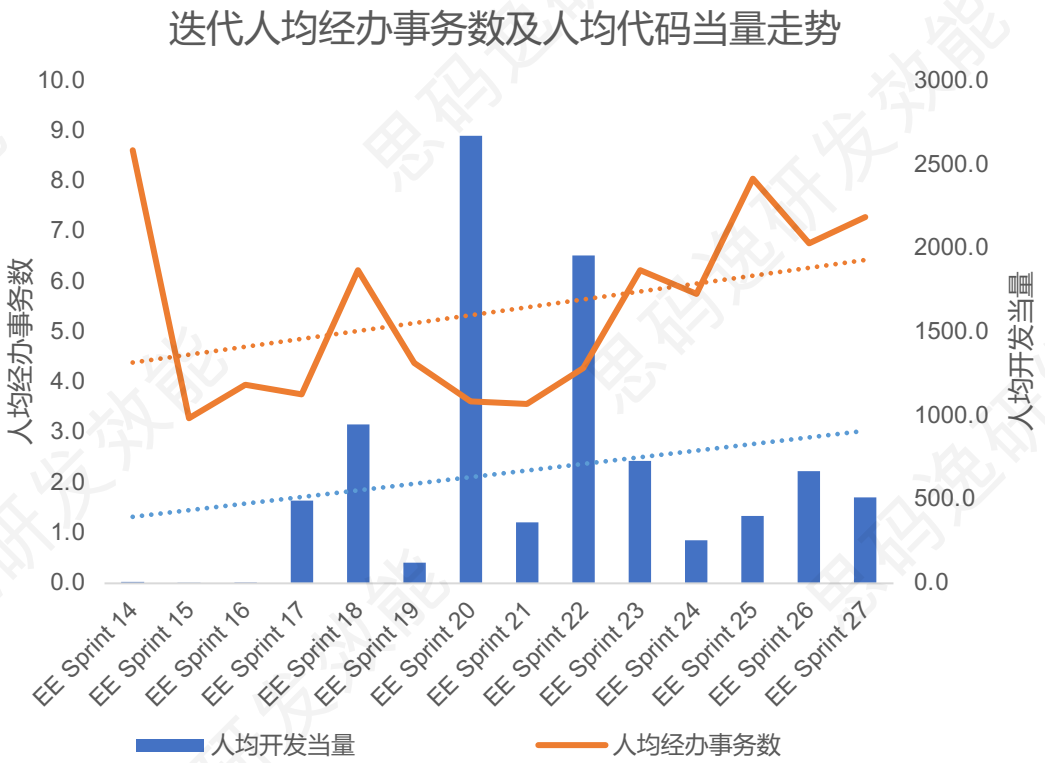
Analyze：需求前置时间分析案例

思码逸
$深度代码分析

- 需求吞吐能力连续4个迭代周期下降



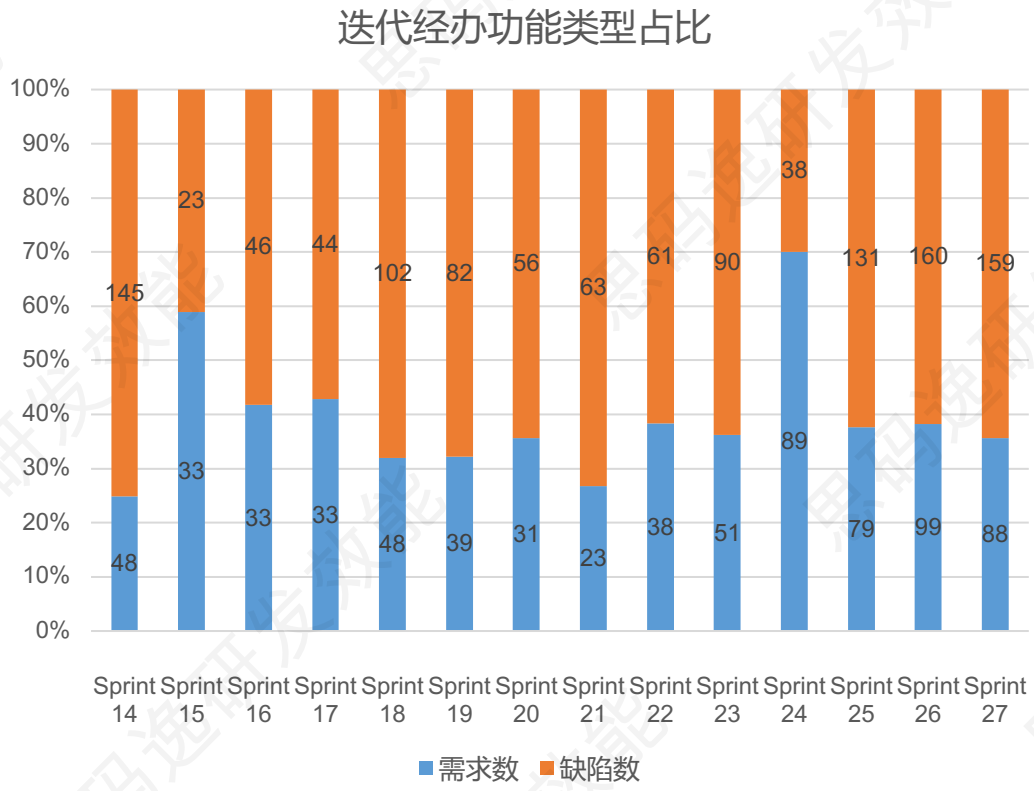
- 通过代码当量校准，需求拆分粒度存在较大差异



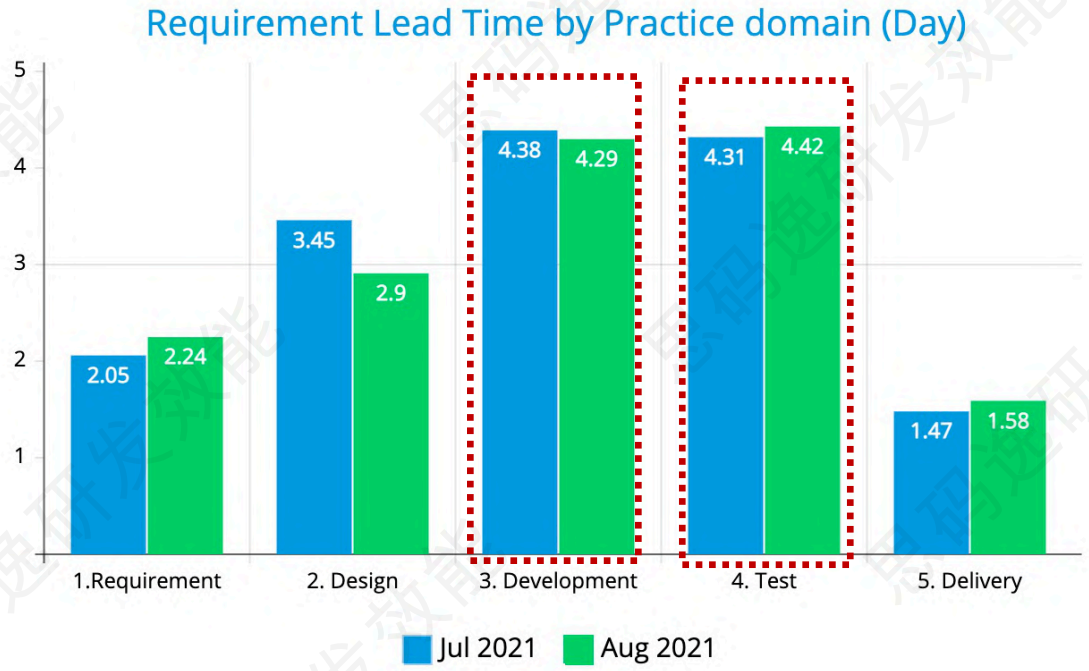
Analyze：需求前置时间分析案例

思码逸
$深度代码分析

- 迭代价值交付比，缺陷高于需求21%



- 需求在开发/测试状态停留时间最长



开脑洞：数据的片面性

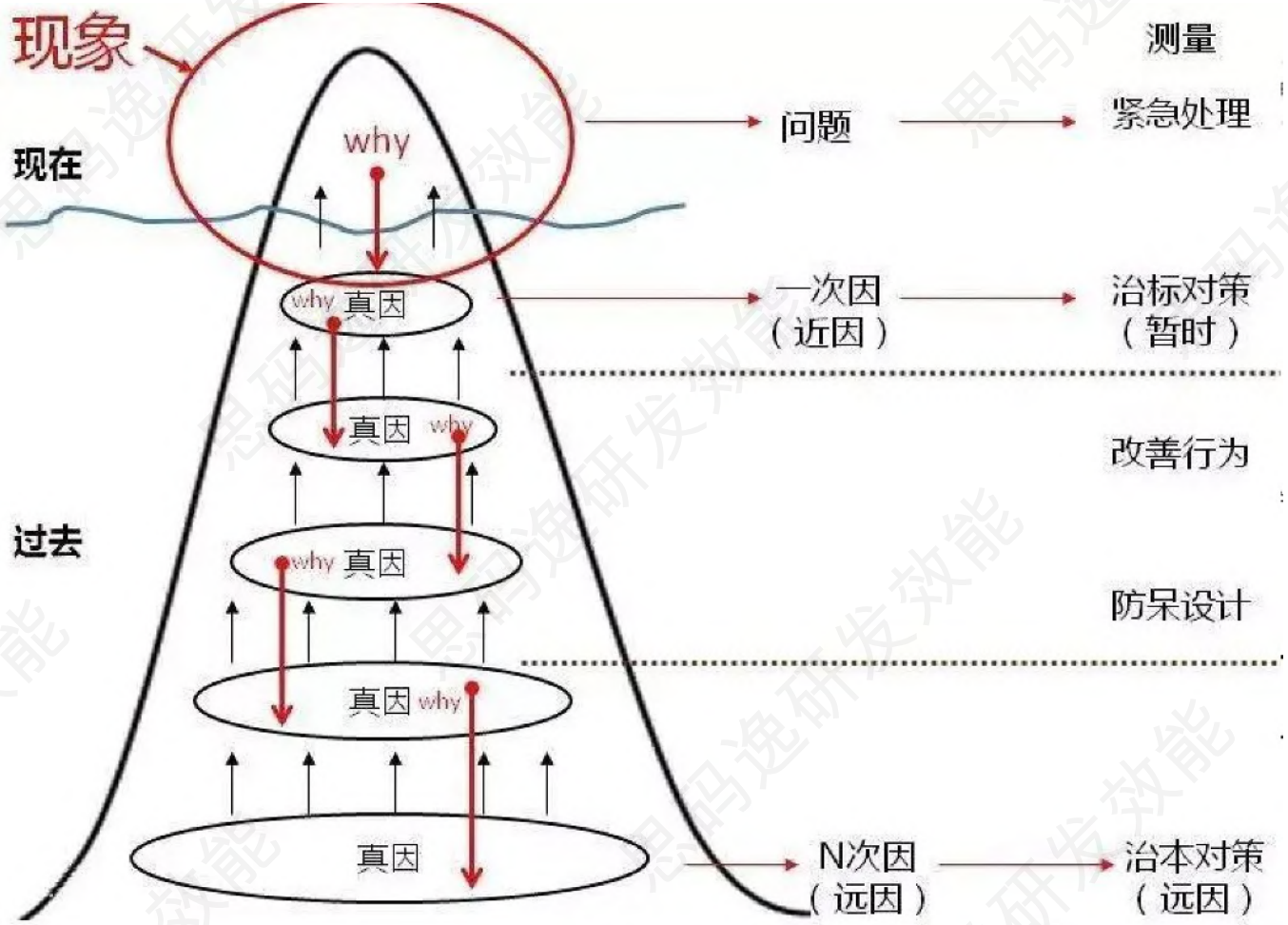
People who can’ t understand numbers are useless. The Gemba where numbers are not visible is also bad. However, people who only look at the numbers are the worst of all.

——Taiichi Ohno

那些不懂数字的人是糟糕的，

而那些只看数字的人是最最糟糕的。

Review：回顾的目的是挖出根本原因



Review：回顾的实施要点

提问的3个层面：

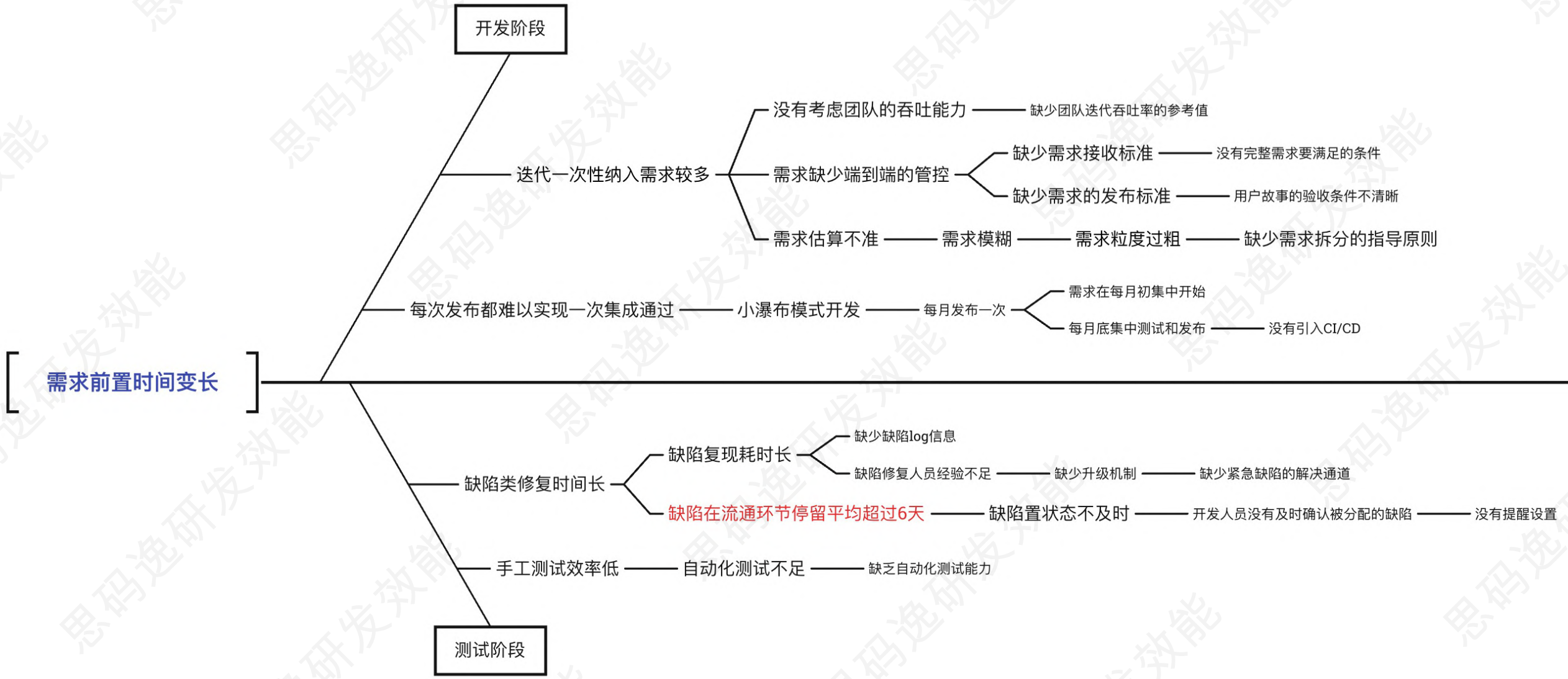
1. 为什么会发生？从“制造”的角度。
2. 为什么没有发现？从“检验”的角度。
3. 为什么没有从系统上预防事故？从“体系”或“流程”的角度。

反向逻辑：

根因分析完成后，一定要从最后一个“为什么”反向的追溯到“问题的现象”，确认反向逻辑在理论上也是正确的。

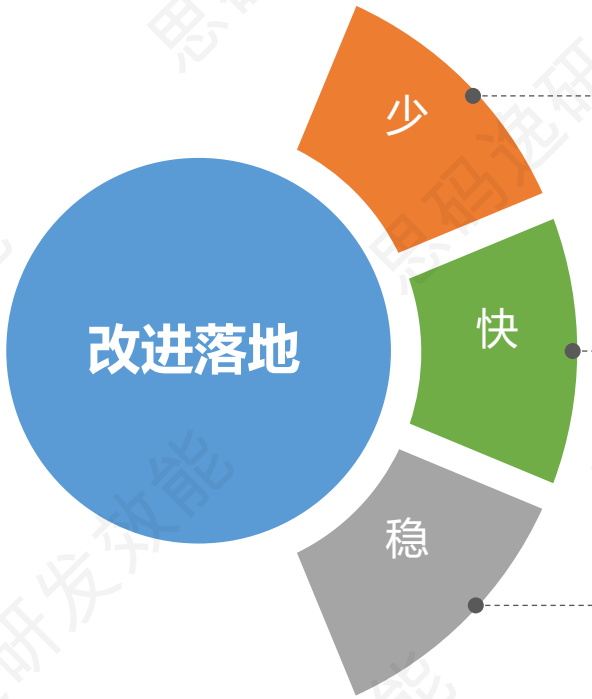
Review：需求前置时间回顾案例（根因分析）

思码逸
$深度代码分析



Improve：改进落地的三驾马车

有没有时间短、见效快、成本低的改进方法？



【指导原则】

要治标，更要治本。

要聚焦，不要分散。

【指导原则】

小步改进，及时见效。

简化流程，高效产出。

【指导原则】

评估影响，措施具体。

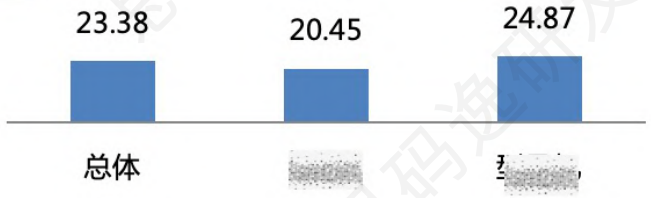
效果评估，数据说话。

Improve：需求前置时间改进案例（聚焦）

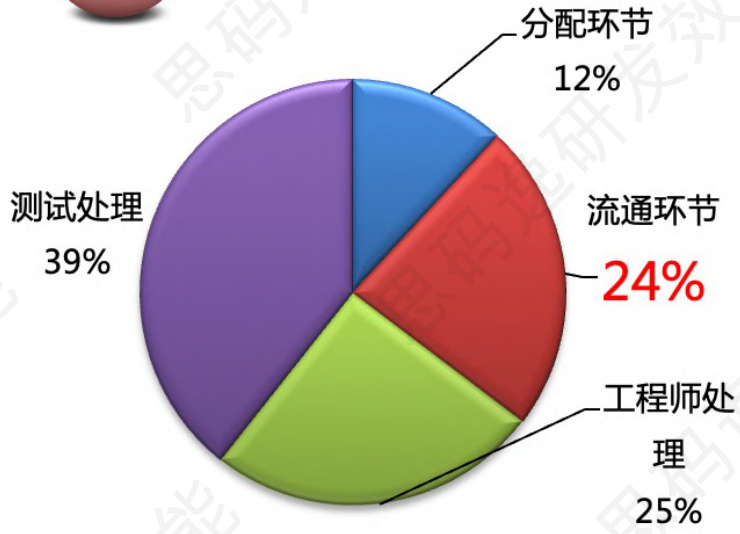
思码逸
深度代码分析

改进前分析：软件缺陷处理时长

01 缺陷平均处理时长



02 缺陷处理各环节平均时长

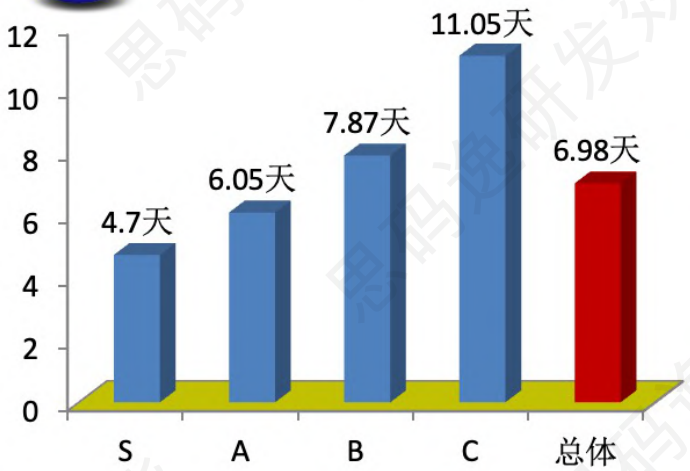


注：
分配环节（submit-Assign)
流转环节（Assign-modify)
工程师处理（Assign-Resolve-Accept/Fail)
测试处理（Accept-Close).

改进目标：

通过合理缩短缺陷相关环节时长，缩短缺陷处理的整体时长10%+，有效提升缺陷解决效率，进而缩短产品上市周期

03 不同等级缺陷流通环节平均时长



Improve：需求前置时间改进案例（跟踪）

思码逸
$深度代码分析

	#	改进专题	状态	效果评价标准	跟踪频率	计算公式	数据来源	采集人	验证人	指标跟踪人	下次跟踪时间
交付效率	6	缺陷处理时长	已落地	改进后：Rxx 4.5的缺陷解决时长较上半年均值数据下降54%约为4.66天，NXx较上半年型号机均值下降57%，约为6.98天，效率提升幅度明显。	1次/半年	PR平均处理时长	JIRA BI系统	[illegible]			2021.6
	7	HCCB问题处理时长	已落地	改进后：经过对HCCB处理环节时长分析截止2017年4月，上述数据分别将为：5.87天、3.79天和20.1天，HCCB问题处理时长得到缩短。rxx处理时间偏长问题目前已通过JIRA仪表盘进行日跟踪，流转状态24小时内处理。	1次/半年	HCCB问题处理时长	JIRA BI系统	[illegible]			2021.6

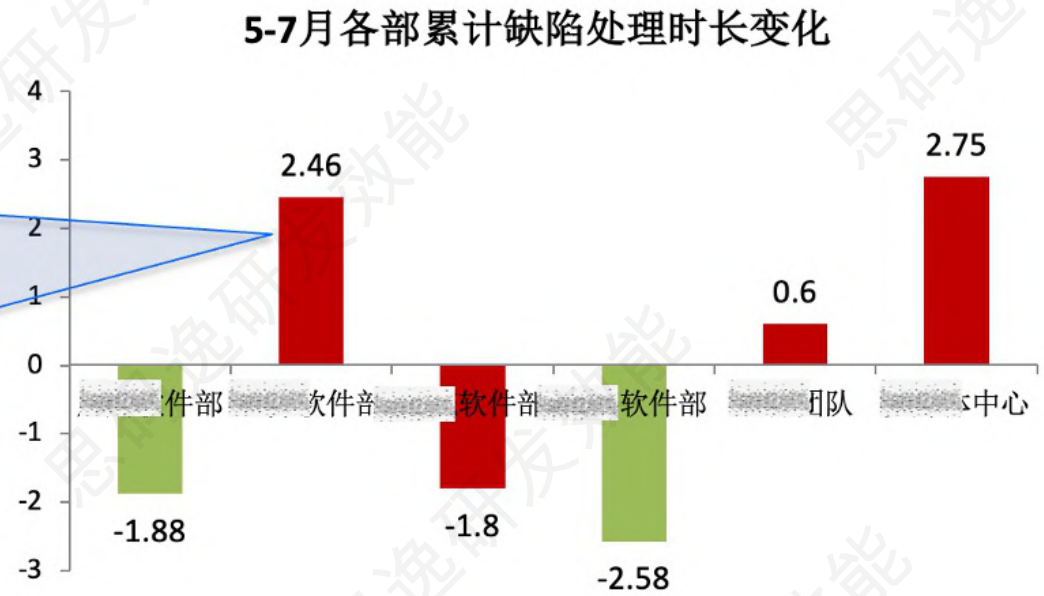
Improve：需求前置时间改进案例（验证）

数据跟踪：各部累计缺陷处理时长变化

部门	5月	7月	处理时长变化
[illegible]软件部	13.61	11.73	-1.88
[illegible]软件部	13.16	15.62	2.46
[illegible]软件部	10.09	8.29	-1.8
[illegible]软件部	9.67	7.09	-2.58
[illegible]团队	4.92	5.52	0.6
[illegible]本中心	10.4	13.15	2.75

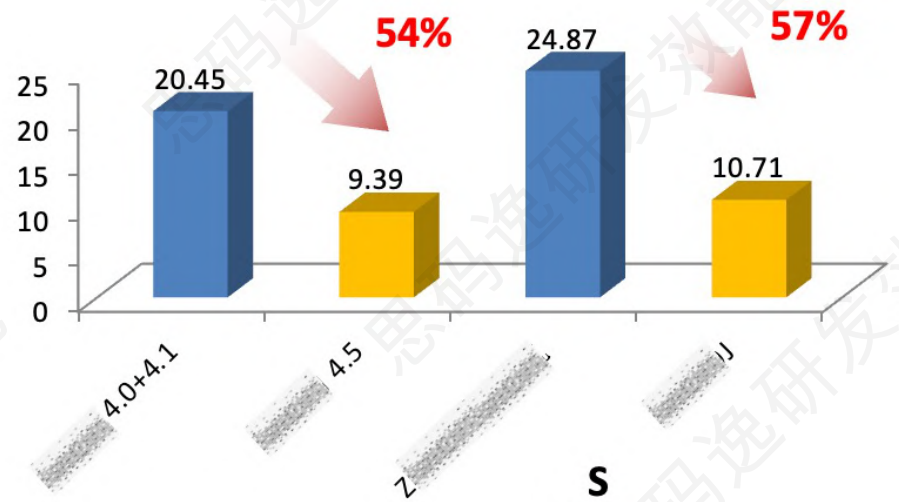
影响数据比较因素的说明：

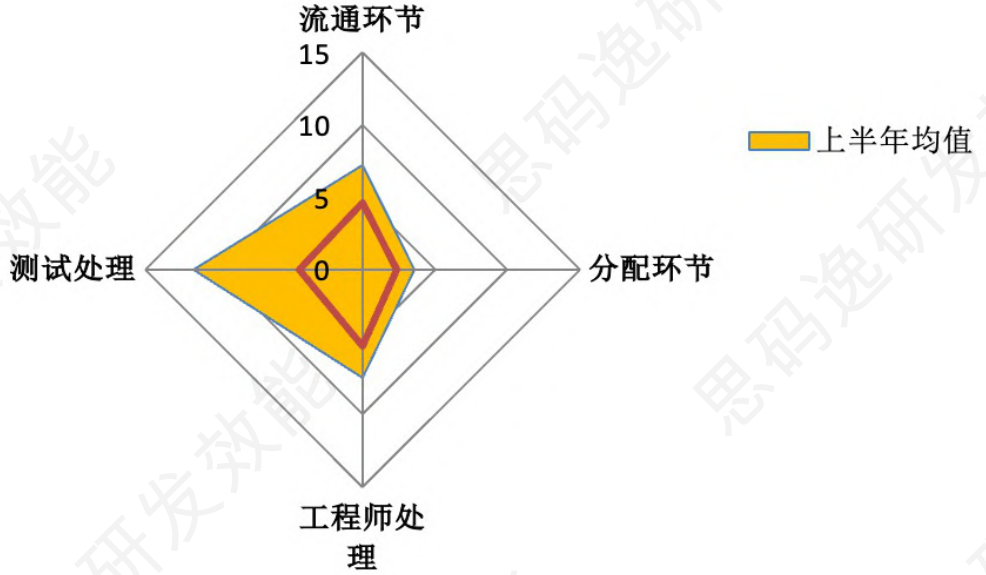
1. 约[illegible]数据产生了影响
2. 名[illegible]始的

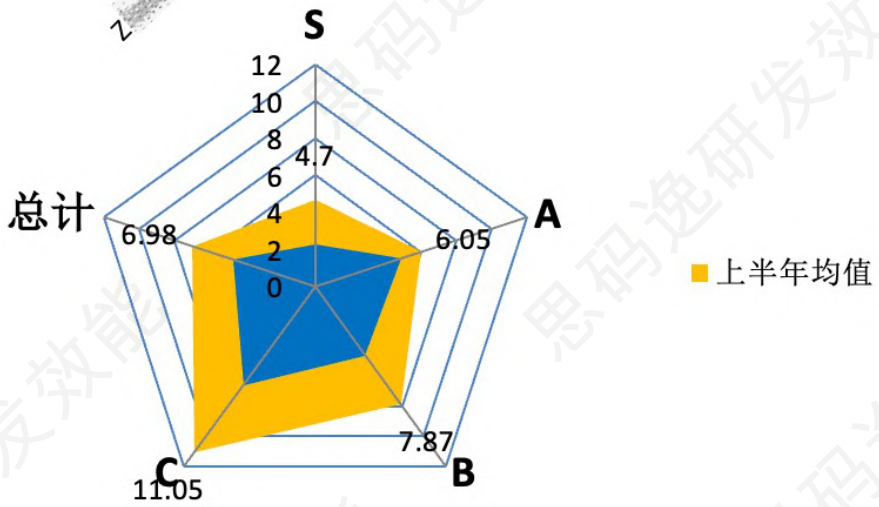


Improve：需求前置时间改进案例（评估）

改进效果，用数据说话





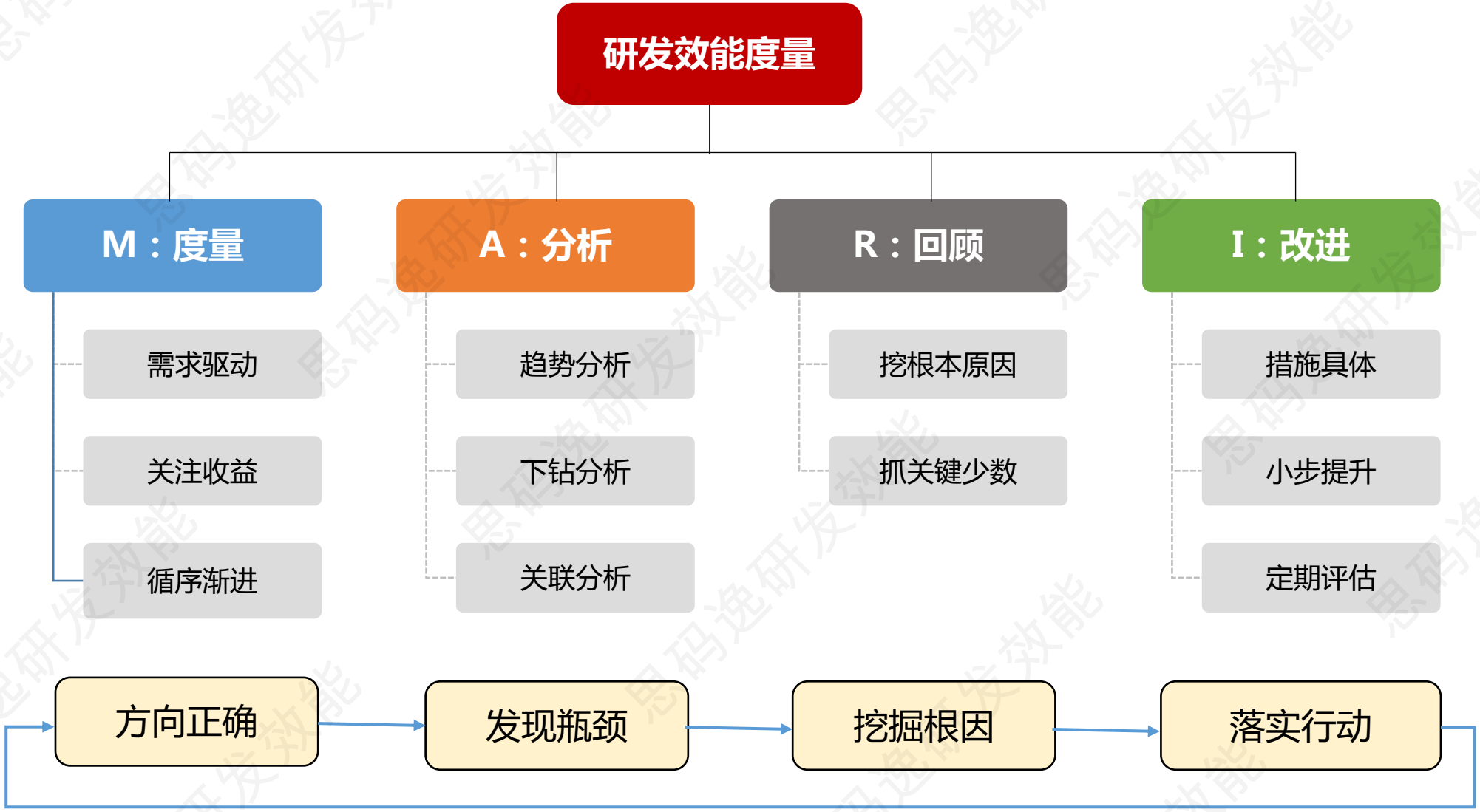


nxx和roo项目，跟上半年均值相比，整体缺陷平均处理时长、各环节处理时长、各等级故障平均时长均有明显缩短。

Improve：改进落地5项注意

1. 切忌眉毛胡子一把抓，重点突破。
2. 先定个小目标，循序渐进。
3. 影响大的措施，先试点、再推广。
4. 尽可能工具化，减少重复劳动。
5. 用效果说话，黑猫白猫、能抓老鼠就是好猫。

MARI方法要点总结



MARI卡片

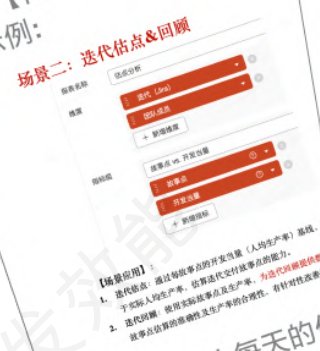
思码逸
$深度代码分析

MARI卡片

思码逸
$深度代码分析

度量（Measure）

1.统计项目维度周期内的故事
2.统计项目维度周期内的【代
量】。
3.计算 每故事点的代码行数
于【代码行数/代码当量】除
示例：



4. 统计每人每天的代码
5. 统计【每个需求的
位数/80%分位）。
6. 统计【每个需求的
位数/80%分位），
（或者用户可配置）

分析（Analyze

1. 一个良好的Story
INVEST原则，根
行分析。
- Independent:
- Negotiable:
- Valuable: 有
- Estimable:
- Small: 短小
- Testable:

2. 对比 每故事
与 每人每天
大小。

3. 分析80%的
间，比如
个迭代（

4. 分析80%
间，比如
一个迭代

回顾（Review）

**针对以上对Story的INVEST
调研是否满足以下6个方面**

1. 独立的：这个功能是否
的，低耦合；

2. 可商讨：这个功能是
可以讨论更具体的

3. 有价值：这个功能
或验证，不是幻想

4. 可估计：每个Sto
量，是否估算了

5. 合适的小：是否
费一两天可以
行数/代码当量
码行数/代码当

6. 可测试：这个
依赖其他的

改进（Improve）

**针对回顾结果，对阻塞环节，从规
为等方面给出针对性改进措施，明
施、验证周期及责任人。针对改进
MARI（度量、分析、回顾、验证**

1. 增加Story的独立性

2. 增加Story的细节

3. 合理拆分需求，
一个迭代（周）
法完成就尝试
付的Story才是
Story对于当前

4. 为Story估算

5. 尽量保证每
量 接近 每
量，让一
完。

6. 保证Story
可测试。

指标名称	**需求粒度**
指标类型	**基础指标**
指标定义	**需求的颗粒度，例如：故事点数/时长**
指标价值	**需求分解颗粒度，足够的细致，才有足够的质量，才能保证客户满意度。而对需求粒度的把控意义在于，评估“项目管理”能力，以及“达成商业目的”的变通能力。**
指标来源	**Jira**
实践域	**需求、开发**
认知域	**交付质量**
衍生指标	**无**
度量范围	**项目、周期（周/月/迭代/版本）**
度量频率	**按需**

思码逸

$ 深度代码分析

Thanks

面向程序代码库的大数据分析
挖掘公司最有价值的智力资产
来自微软、加州大学伯克利分校、斯坦福大学、GitHub的世界一流智囊团为您服务

官方网址：**www.merico.cn**　　思码逸客服：**400-8637-426**

在 DevData Talks
开放务实地聊聊研发效能

分享合辑

研效交流群

每月直播 干货满满

行业 KOL + 企业研发效能负责人 + 资深效能顾问
从效能建设路径到不同场景应用，分享前沿观点与先进实践

开放探讨 共同成长

直播 Q&A + 交流社群
直播嘉宾面对面答疑解惑，交流群随时随地探讨效能话题

关于思码逸

作为DevData Talks 的发起方，思码逸专注为企业研发团队提供效能数据分析，呈现效率、质量、人才发展等多视角数据洞察，辅助团队决策与工程改进。

思码逸已服务 腾讯、美团、滴滴出行、中国平安、泰康保险、戴尔EMC等众多行业标杆客户。