codes

智 能 管 理 研 发 · 数 据 驱 动 效 能

一站式敏捷测试解决方案

四川无限智达科技有限公司

2023年05月

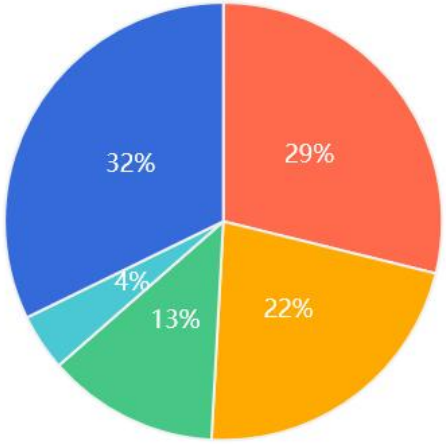
contents
目录

敏捷测试的兴起与现状

敏捷在行业中的应用状况

codes

超八成企业已在不同程度上实践敏捷开发
同比增长近3成



32.243%

部分团队已经使用一段时间，
目前处于优化阶段

28.823%

一半以上团队的敏捷开发实践在整个
组织内处于比较高的水平

22.042%

所有团队已熟练掌握敏捷开发，
可根据情况调整、改善和创新实践。

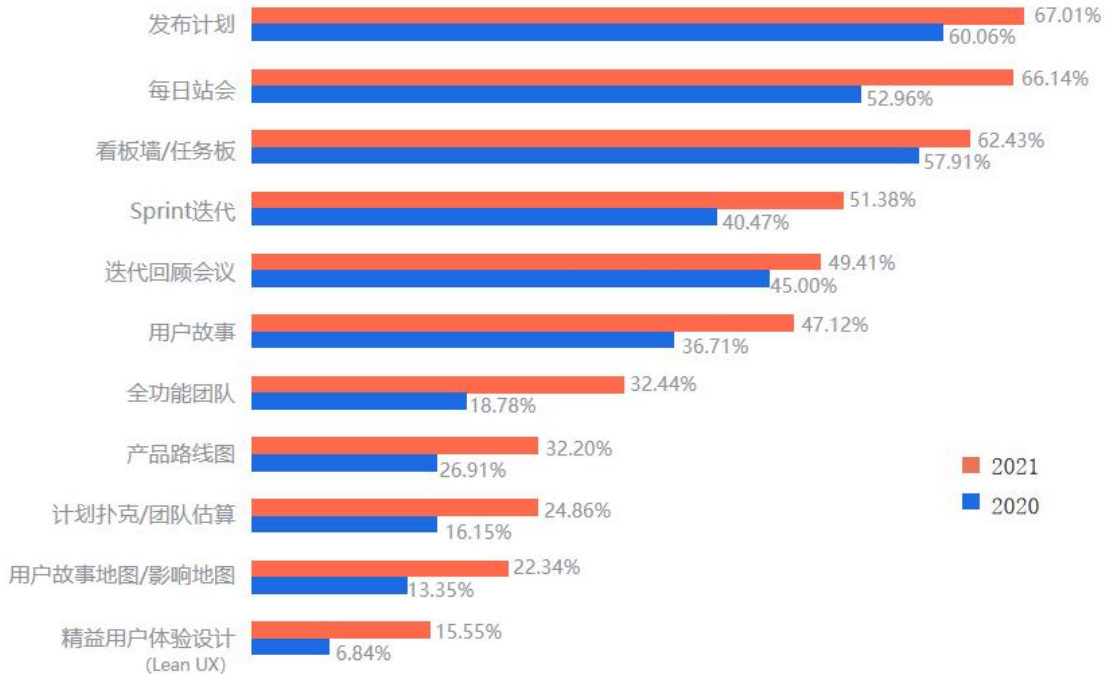
12.691%

尚未使用敏捷开发

4.2%

少数团队刚刚作为试点引入敏捷，
处于起步阶段

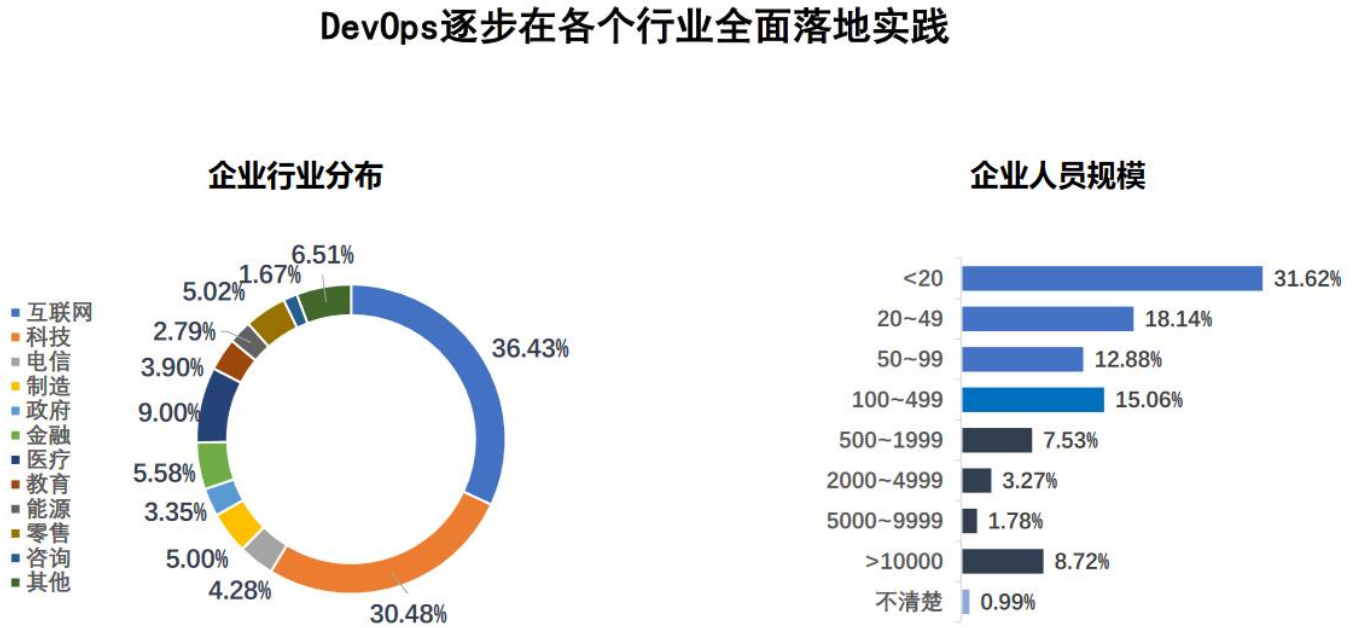
发布计划、看板、每日站会、Sprint迭代是使用最广泛的敏捷管理实践



调查发现，32.24% 企业在部分团队使用敏捷开发一段时间；28.82% 的企业已有一半以上团队的敏捷开发实践在整个组织内处于比较高的水平；22.04% 的企业在所在团队已经熟练掌握敏捷开发。

据调查显示，2021年占比分别为 67.01%、66.14%、62.43% 和 51.38%；而在2020年的使用比例分别为 60.06%、52.96%、57.91% 和 40.7%。普及程序明显提高

国内软件企业软件工程的现状及趋势

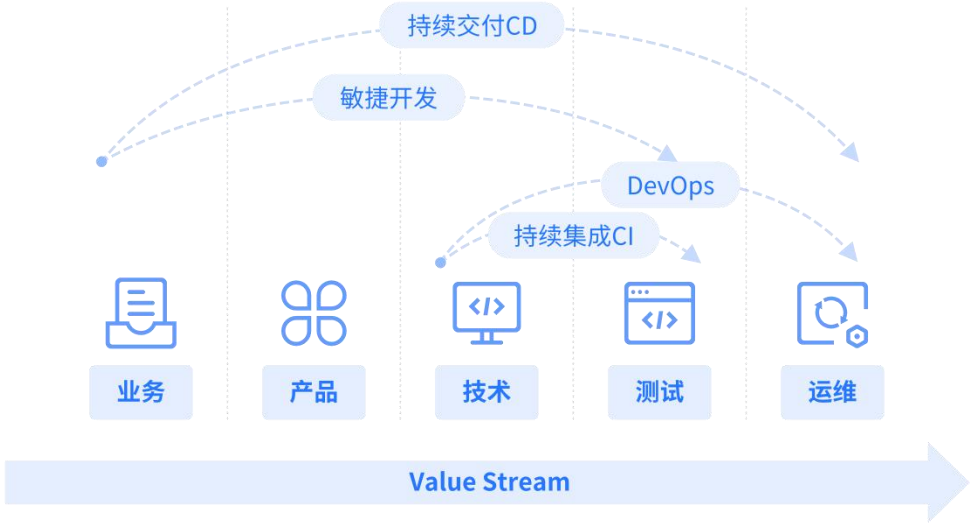


持续集成、持续部署、单元测试、自动构建和持续发布是企业应用最为广泛的敏捷工程实践。调查结果显示，企业多采用持续集成、持续部署、单元测试、自动构建和持续发布五种敏捷工程实践，占比分别为74.17%、64.80%、64.64%、62.06%、56.84%，均超过半数。

有6成的企业实践了DevOps，且4成的企业还处在基础级。

敏捷测试是敏捷开发的必然需求

持续交付能力是企业核心竞争力

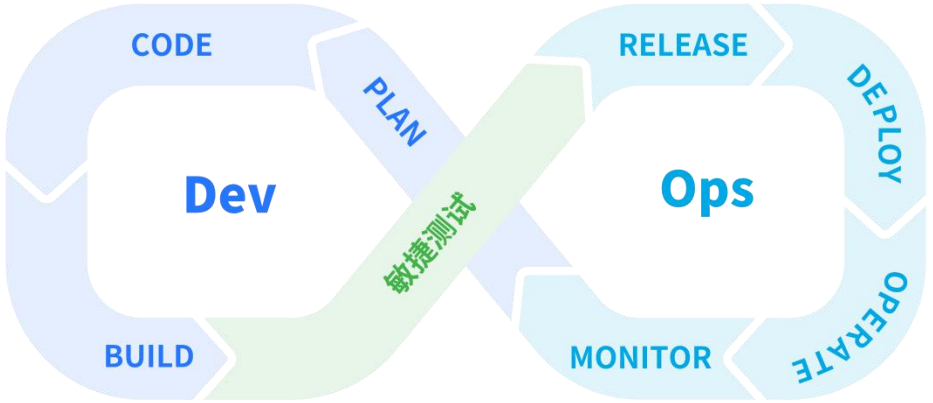


保证交付足够快

催生

促进

保证业务足够稳

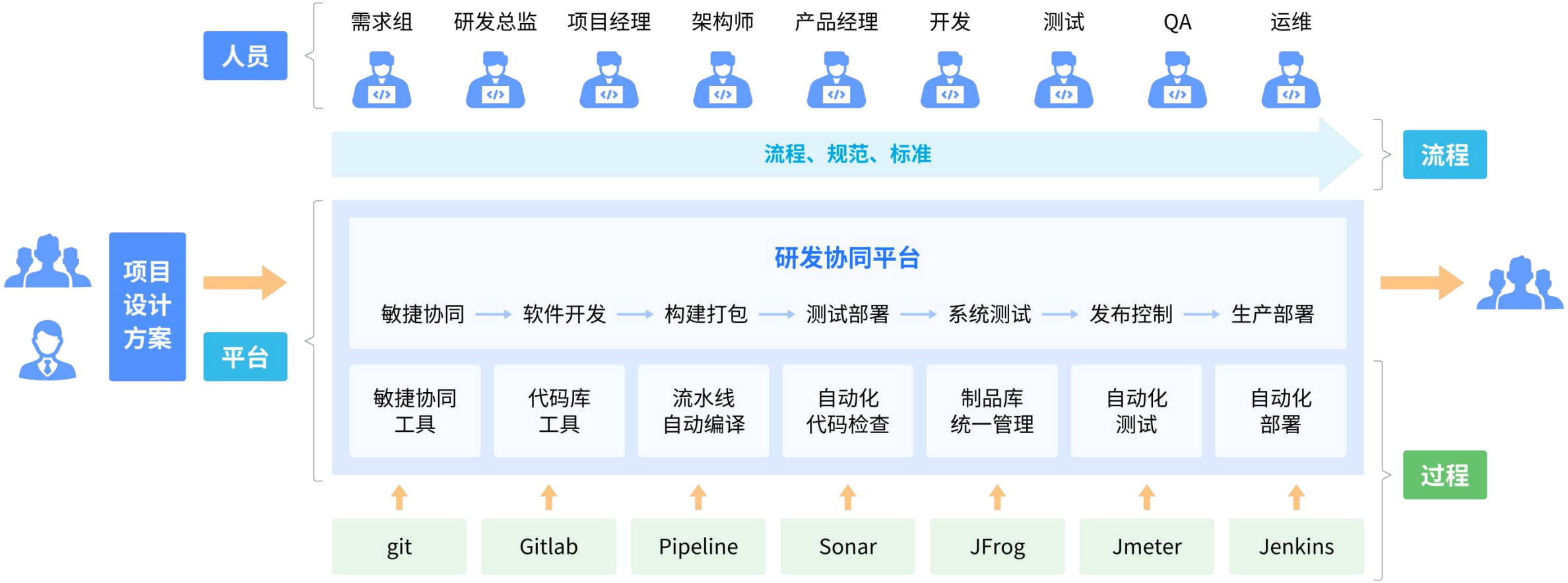


持续交付可以降低发布风险，提高可靠性，使软件能够根据用户反馈、市场变化和企业战略变更不断进行调整。

敏捷测试是持续、快速、有效的测试过程，作为软件交付流水线的重要一环，敏捷测试是企业改进加快交付的关键途径。

DevOps工程模型

codes



持续交付工具链常见问题

- 工具集成度低
- 学习成本高
- 部署难度大
- 数据不集中
- 管理复杂
- 人效不好评估
- 管理可观测性差

为什么测试环节会成为敏捷开发快速交付中的瓶颈

codes

敏捷测试平台是提高生产能效最高效方法之一

codes

交互能力提升

交付质量提升

交付速率提升

需求响应
速度提升

效能分析
持续改进

管理可视化

聚焦业务价值

减少损耗

工具固化流程

质量内建文化

测试如何敏捷起来

敏捷测试与传统测试有什么不同

codes

- 测试计划
- 测试准备
- 测试执行
- 测试报告
- 发布上线

过程都一样，关键在于测试的组织管理形式不同。

敏捷测试测试4要素及一基础

迭代

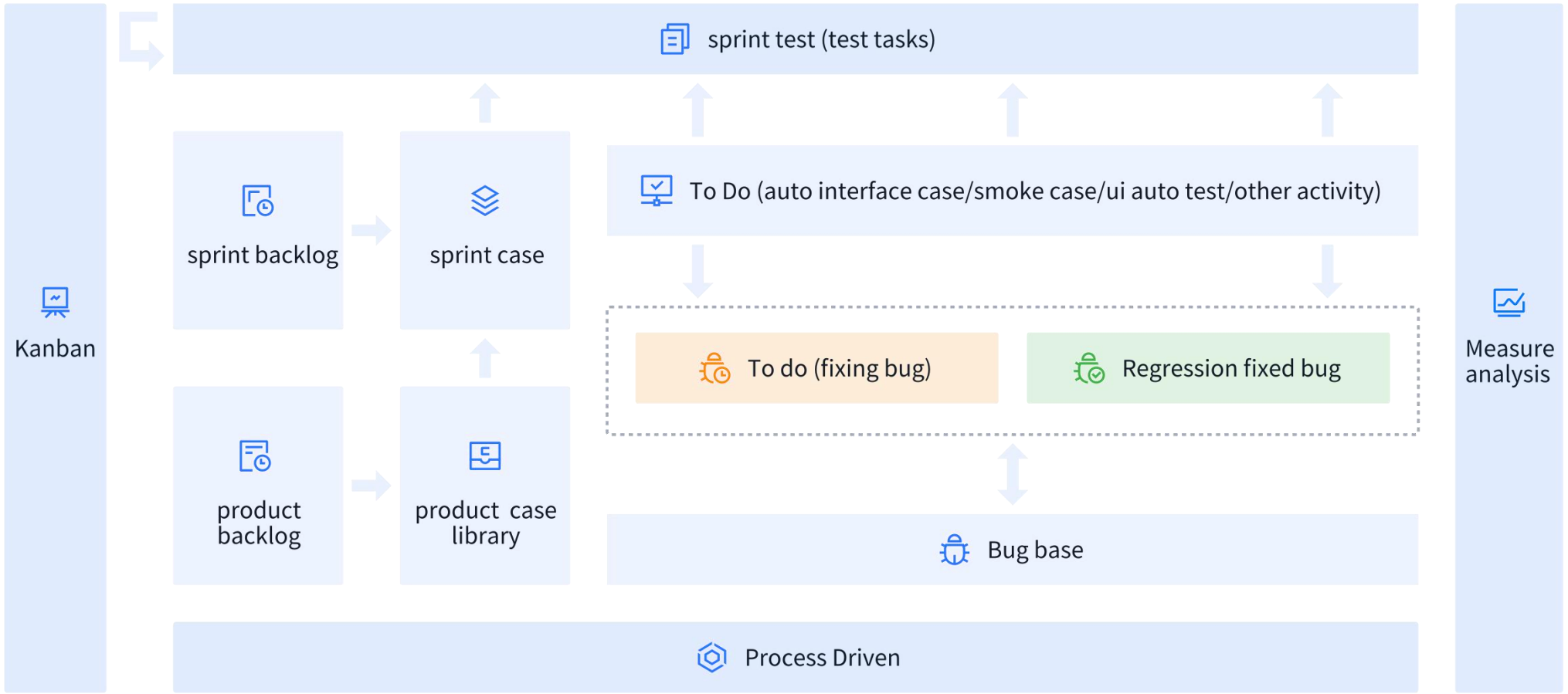
看板

自动化

度量

测试左右移

敏测试测逻辑模型


sprint test (test tasks)
Kanban
sprint backlog
sprint case
To Do (auto interface case/smoke case/ui auto test/other activity)
To do (fixing bug)
Regression fixed bug
Measure analysis
product backlog
product case library
Bug base
Process Driven

敏捷测试测整体目标

敏捷测试测基础(流程规范)建设

Bug 生命周期管理、龄期控制
Bug流程及Bug状态流转机制
Bug 填写规范及字典定义等

测试组织过程

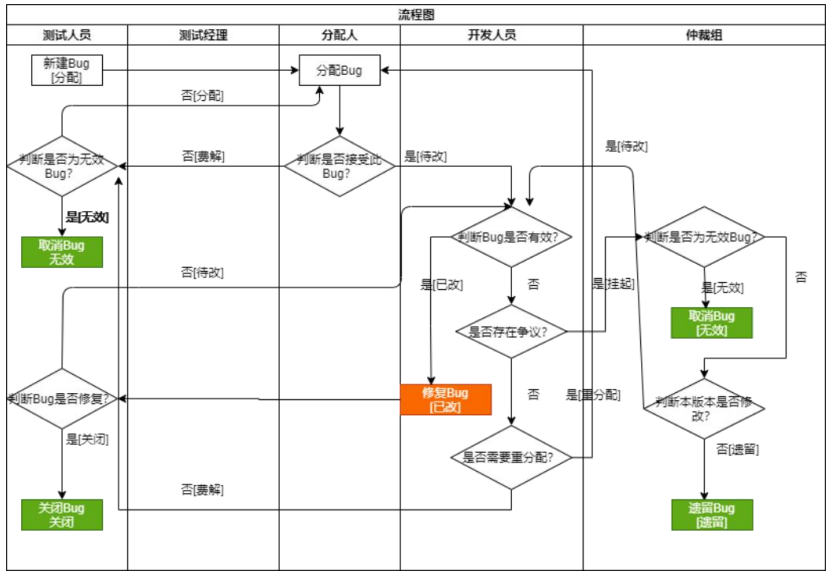
测试执行流程
用例便携规范，字典定义

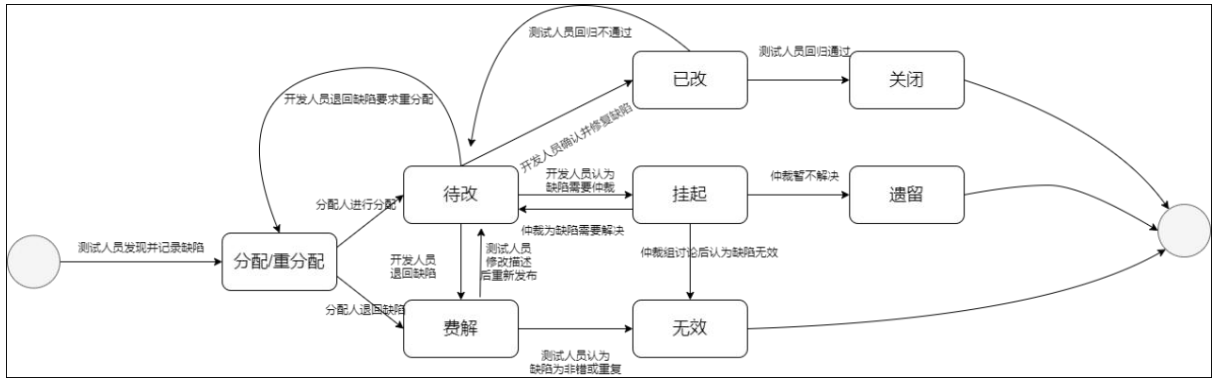
发版流程

测试报告及产物定义

建立质量管理体系，及项目管理体系

敏捷测试测基础(流程规范)建设





敏捷测试测团队建设

梯队：对测试人员进行分组，提拔一批小组长，识别并制订核心成员培养计划；新员工培训（职能与简介、业务范围，工作机制、年度规划，班级工作交叉）

质量意识：向全员宣教并培训整个质量管理体系，组长必须重点深刻领悟

技术：制定技术发展路线，为团队技术定基调

建立学习型团队 除了工具，团队技术和文化建设要跟上

敏捷测试测过程管理建设

迭代

定义迭代方式,定义迭代要内容，每个迭代任务分配和监督机制

过程监控

测试执行管理及监控，评审流程，环境管理流程,线上问题响应机制

汇报

对PMO 质量汇报机制，形式及内容

敏捷测试测平台建设

codes

迭代

- 没有管理工具辅助，需要比较长的周期才能培养团队氛围素质，导入敏捷测试管理平台可以无人驱动，工具辅助快速健全规范和落地质量体系的建立

策略

- 明确定义敏捷测试管理平台诉求
- 选型对比报告
- POC并对标质量管理体系
- 客观评估POC结果
- 以点带面、培训，再推广

敏捷测试测平台建设

01

具备的能力

- ✓ 以产品和版本迭代的维度追踪需求生产到落地的全过程;
- ✓ 版本测试计划的执行过程追踪;
- ✓ 以版本的维度、设计、编写、评审、维护和管理测试用例;
- ✓ 缺陷的追踪和管理
- ✓ 版本质量和测试过程的数据统计和分析。

02

安全&稳定性

- ✓ 系统没有阻塞性的缺陷、无数据量激增导致的性能缺陷，可长期运行;
- ✓ 系统内核心数据可进行数据备分，可导出进行保存;
- ✓ 系统有可靠的网络与数据安全机制，无明显的安全漏洞，难于通过后台接口或网页前端击穿系统;
- ✓ 系统无明显的性能缺陷，核心功能至少支持500人团队使用无性能问题，包括用例、缺陷、过程管理、数据分析等操作

03

易用性

- ✓ UI美观、友好，前端交互逻辑符合当下主流同类系统的设计规范;
- ✓ 功能表达清晰、准确，入手门槛低，简单培训即可使用;
- ✓ 核心数据，如测试用例、缺陷提供导入/导出，便于日常备份和数据迁移;
- ✓ 部署及日常维护简单。

04

二次开发&系统基础能力

- ✓ 至少支持开放源码或webhook两种方式中的一种，便于个性化需求的定制开发;
- ✓ 如开放源码，项目的技术栈需符合公司当前的技术栈，前端采用vue或jq,后端采用java web的目前主流的技术，spring boot、mybatis等;
- ✓ 如提供Webhook,至少需提供用户认证对接方式，核心数据提数接口;

测试过程管理

基于测试部日常工作流程的安全、稳定、易用、可定制的测试过程管理系统

Codes加速敏捷测试落地

codes的使命

加速融合研发、测试、运维一体化进程

- 常态下,刀耕火种的Test环节给自动化的Dev与Ops 踩下了刹车。
- Codes 以技术最薄弱，最不被重视的测试为发力点，通过落地敏捷测试打通了研发与运维中间的枢钮润滑环节。
- 解决了Test 在DevOps快速迭代中的木桶效应，促进了研发、测试、运维一体化融合进程。

450+公司在使用codes

codes

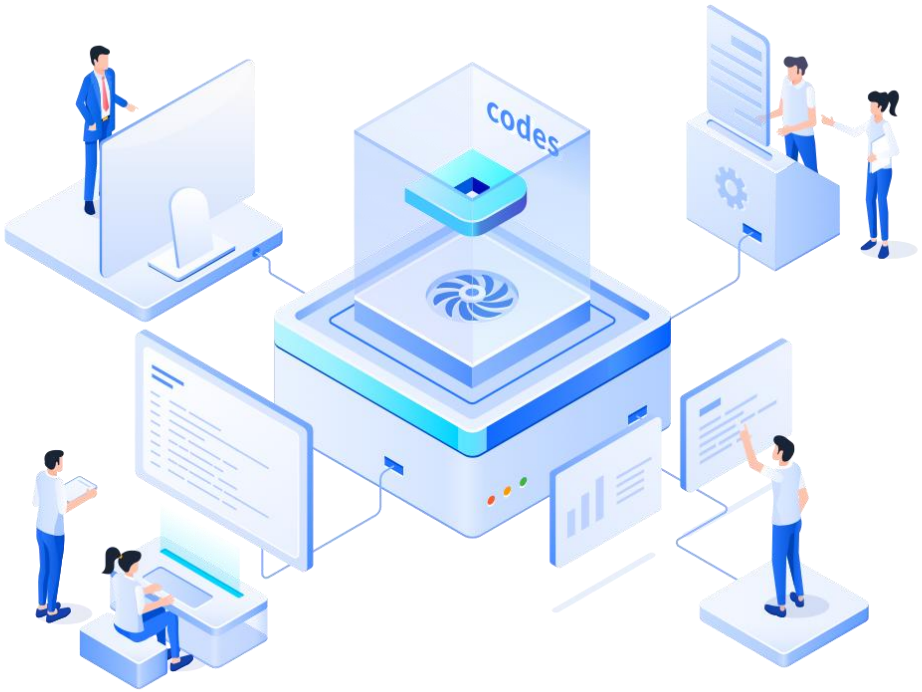
已有450+企业正在使用：Codes社区版&Codes商业版

codes平台一站式

codes

以测试为基础左右移从而覆盖研发管理的全生命周期，提升研发效能

codes协作一站式



技术团队一体化融合，构建研发管理流水线

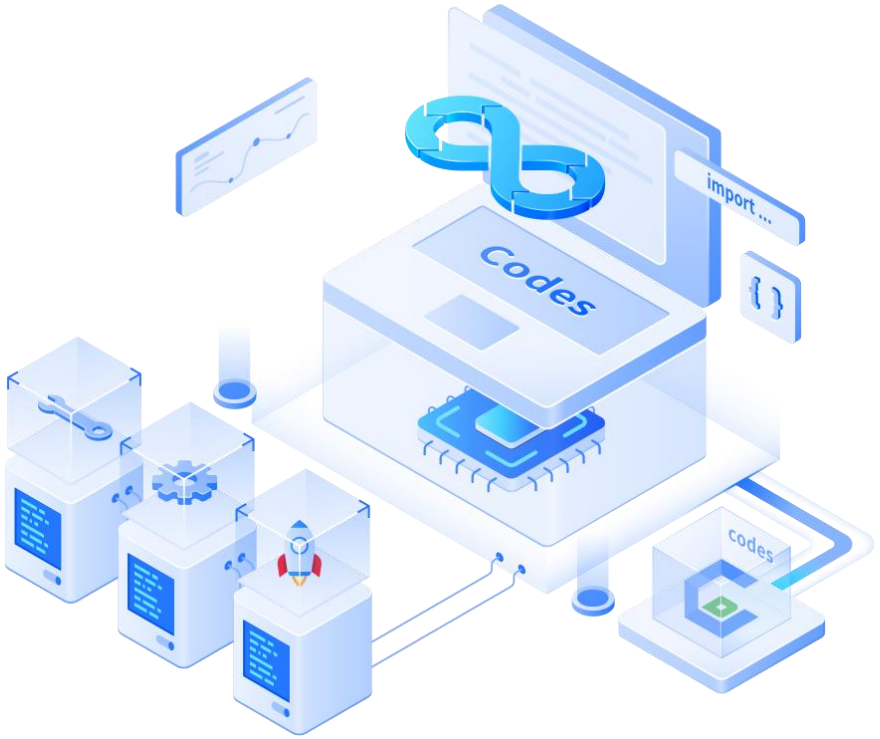
- 以最佳实践助力研发管理人员，打破研发工作黑盒，确保研发流程全程可观测及可控。Codes 以技术最薄弱，最不被重视的测试为发力点，通过落地。
- 围绕需求拉通所有研发活，确保干系人信息对齐。
- 加速产品、研发、测试、运维一体化融合，杜绝“割裂”管理，提升整体研发效能。

codes数据一站式

数据驱动研发效能升级

- 通过一站式研发管理平台，精准沉淀研发过程全场景数据。
- 以数据为管理抓手，实现研发效能可度量，可数字化。
- 协助技术团队研发效能升级。

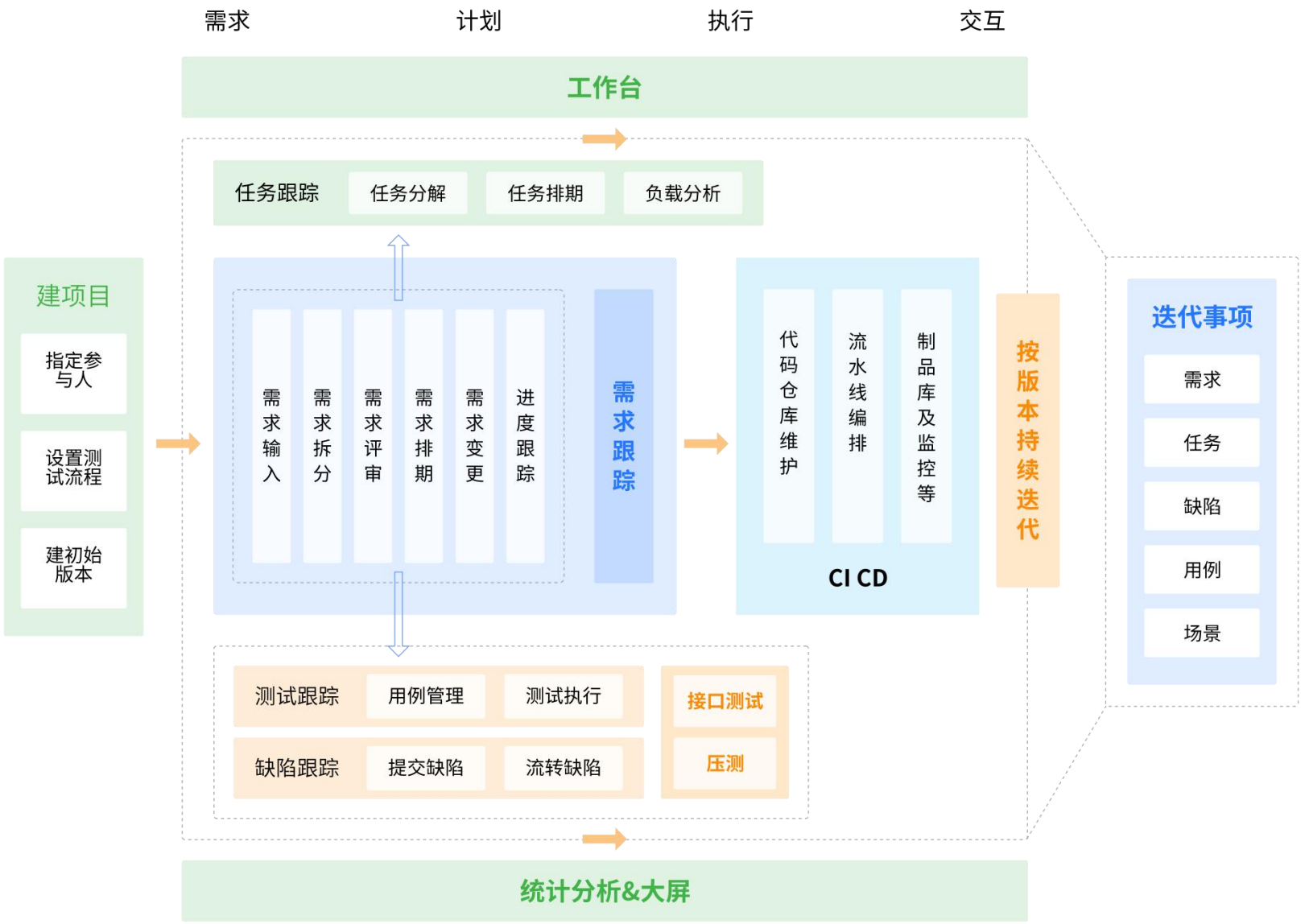
codes高效、简洁、轻量、开箱即用的研发效能平台



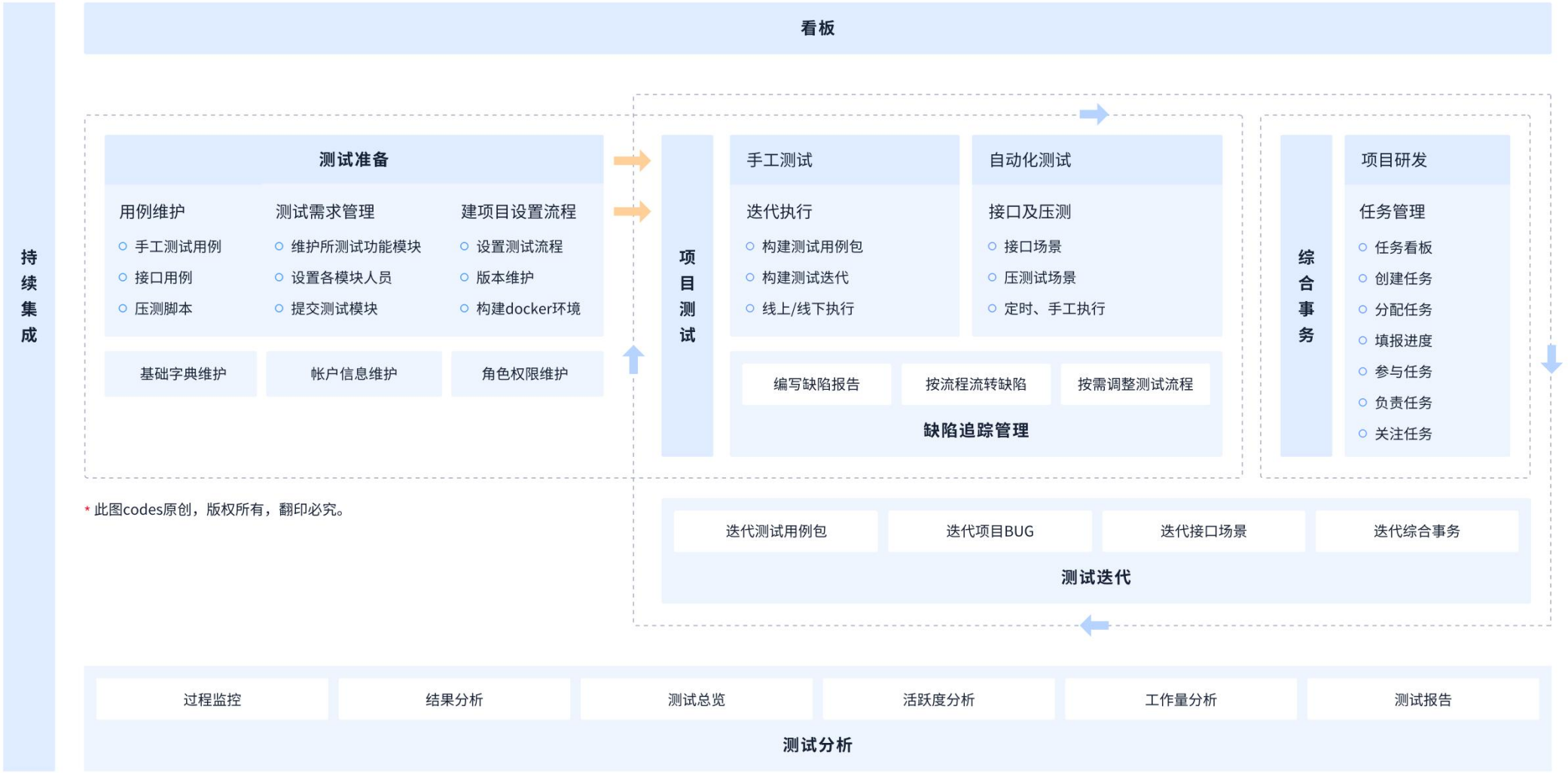
0探索，快速落地，开箱即用的devTestOps解决方案

- 本地化部署，数据安全有保障，我的数据我做主。
- 0代码拖拽式ci cd流水线编排，录制及0 代码拖拽式接口测试，0代码压测，轻运维。
- 开箱即用，直接从最佳实践弹射起步，赋能企业每个员工，让员工聚焦于业务价值。

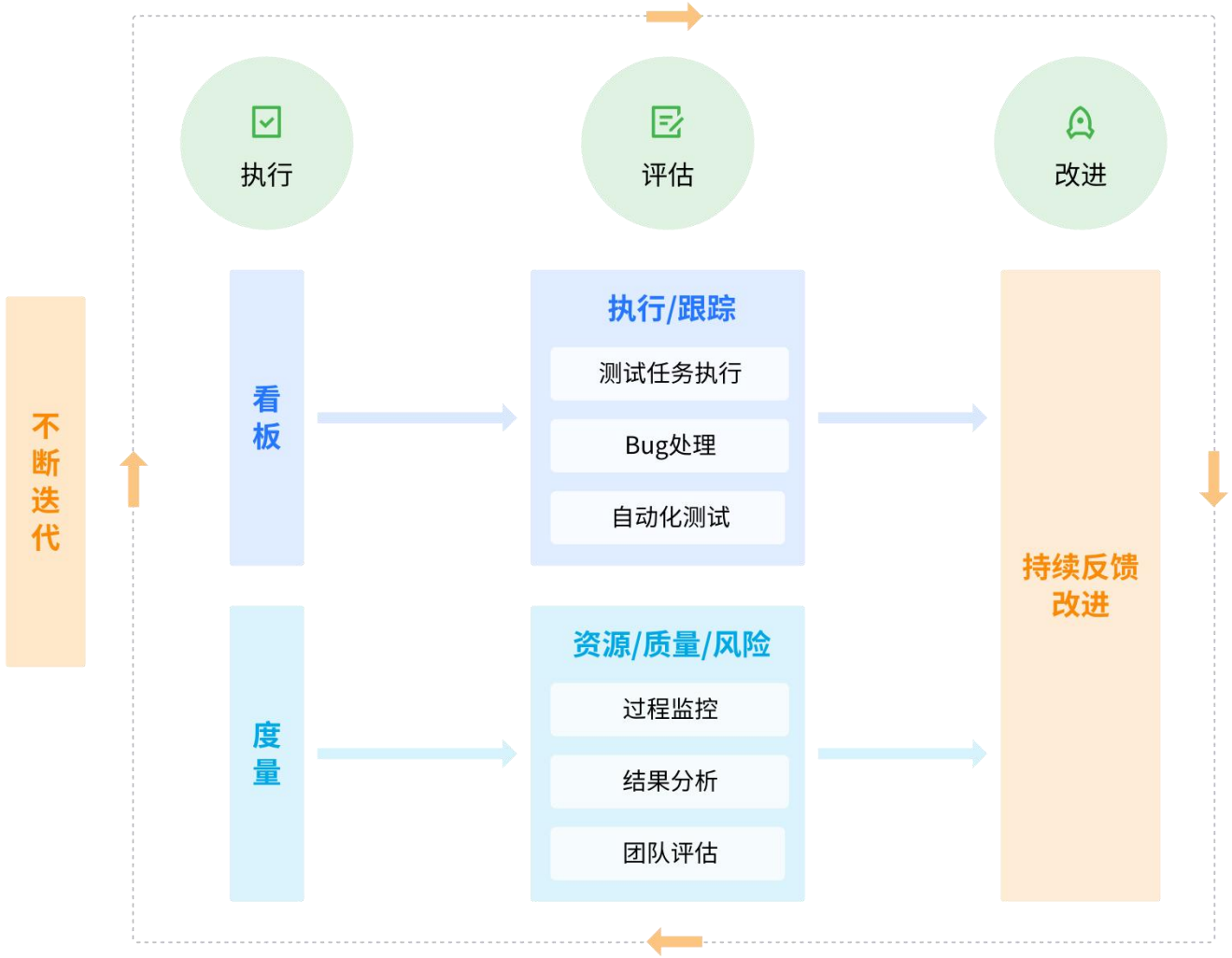
codes一站式敏捷测试解决方案架构



codes测试过程全程跟踪



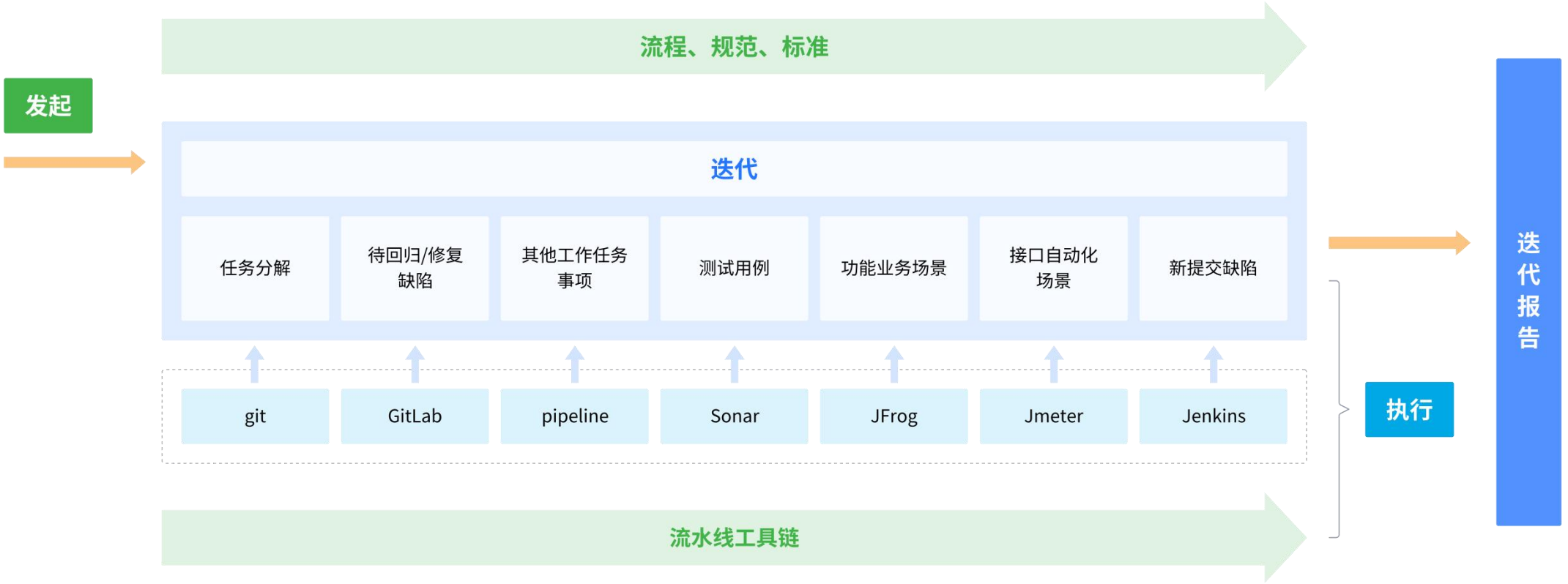
codes测试过程持续改进



各阶段研发活动公开透明

- 迭代、看板、度量、自动化以井然有序，融会贯通的方式融合到测试协同中。
- 让流程管理，敏捷测试以“润物细无声”的方式落地。
- 流程驱动测试，流程推动缺陷流转，不同的流程对应不同的状态演化，反应不同管控目的，并可实时按需调整。
- 离线处理用例再同步到线上，以适应快速变更时用例同步。

codes测试过程持续改进



tips

传统的测试计划只定义了工作项，且只是用例相关，最关键，只有工作项，没有分工，不便于团队协作，也不便于跟踪各自的进度。在同一个迭代下分配工作即有全局又有个体的分工及任务跟踪。

测试工作量化

Codes冲刺 | 需求 | 待修复|回归缺陷 | 任务 | 测试用例 | 功能业务场景 | 接口场景 | 新提交缺陷 | 场景内用例 | 人员分工

参与人员	需求数	任务数	测试用例数	功能场景数	接口场景数	缺陷数
刘一手	4	3	5	2	1	533
刘一招	9	0	0	0	0	209
黄一手	5	0	7	3	0	189
周工	9	2	0	2	0	91
陈一招	1	0	0	0	0	97
李工	2	0	0	0	0	8
侯工	1	0	0	0	0	7
陈sir	1	0	0	0	0	6
邓哥	3	0	0	0	0	2

总用例数	通过数	未通过数	阻塞数	不适用数	未测试数	冒烟通过率	进度
34	14	5	3	0	12	27.8%	64.7%

执行进度

分配人	总用例数	通过数	未通过数	阻塞数	不适用数	未测试数	进度
刘一手	5	3	2	0	0	0	100%
黄一手	7	3	2	0	0	0	71.42%

总览 | 需求 | 任务 | 业务场景 | 用例明细 | 用例进度 | 接口场景 | 接口 | 缺陷

需求摘要	提交	完成	排期	已上线	延期	拒绝	
	39	14	39	0	21	1	
任务摘要	总任务数	分配	暂停	进行中	完成	终止	
	5	4	0	0	1	0	
业务场景摘要	总业务场景数	完成数	延期数				
	19	1	0				
用例摘要	总用例数	通过数	未通过数	未测试数	阻塞数	不适用数	
	34	14	5	12	3	0	
业务场景摘要	接口场景总数	完成数	延期数				
	1	0	1				

在同一个迭代下分配工作即有全局又有个体的分工及任务跟踪，而不仅是事项分不清个体任务。

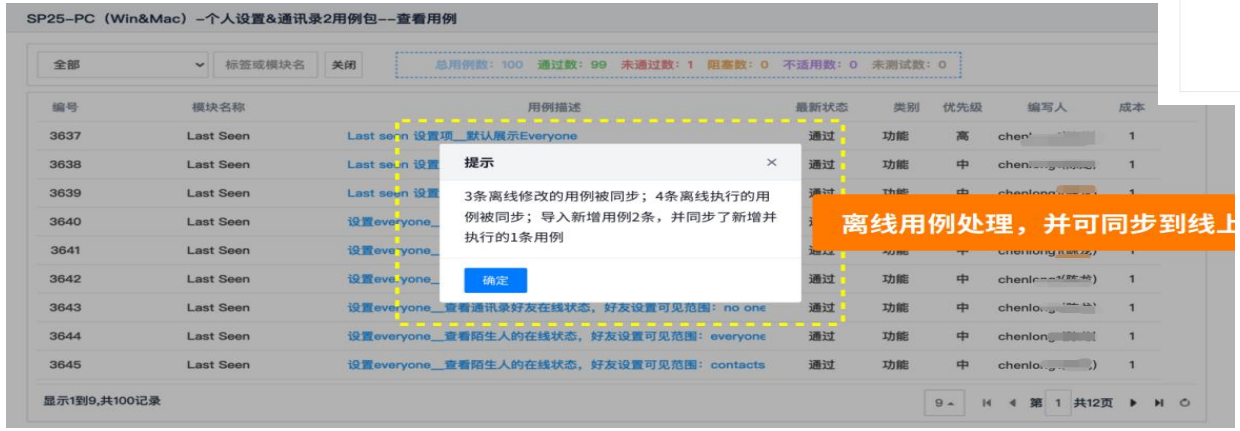
测试工作量化

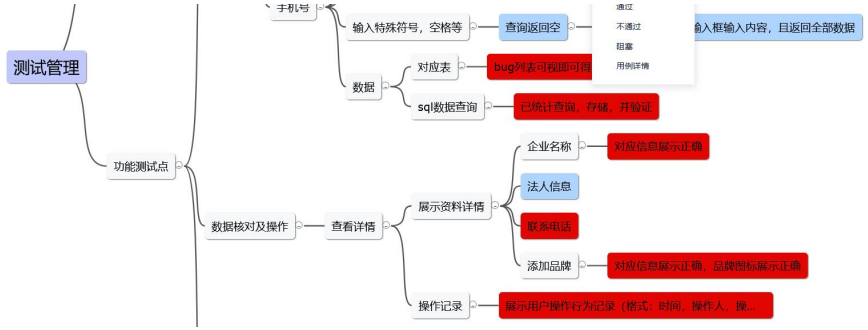

总览
需求
缺陷
用例
接口
工作量
吞吐量
CI/CD
统计时间：2022年10月18日 至今
本周
本月
本周用例
320 (新增) 118 (修改)
本周bug
97 (新增) 46 (修改)
本周接口
30 (新增) 13 (修改)
本周构建
16 (构建) 11 (部署)
本周事故
5 次
最活跃项目TOP5
01、工行淄博和烟台验证环境
活跃值: 128
02、安卓和路由规则验证测试
活跃值: 116
03、iOS拆分三验证测试
活跃值: 97
04、安鹏irr计算逻辑变更测试环境测试包
活跃值: 86
05、兴邦二手车路由测试环境
活跃值: 77
本周bug曲线
用例曲线
bug总数
bug提交
bug修改
120
80
80
60
40
20
0
总数: 62
提交: 50
修改: 53
2021.10.08
2021.10.09
2021.10.10
2021.10.11
2021.10.12
2021.10.13
2021.10.14
最不活跃项目TOP5
01、工行淄博和烟台验证环境
活跃值: 9
02、安卓和路由规则验证测试
活跃值: 15
03、iOS拆分三验证测试
活跃值: 22
04、安鹏irr计算逻辑变更测试环境测试包
活跃值: 28
05、兴邦二手车路由测试环境
活跃值: 32
本周bug来源
1.06%
17.94%
21.64%
59.36%
测试人员 150
研发人员 26
客户 17
线上bug 3
本周bug龄期
本周bug引入阶段
100
80
60
40
20
0
零天
一天
二天
三天
四天
五天
六天
七天以上
本周用例执行结果
2.54%
16.78%
100
总数
80.68%
通过 72
失败 23
阻塞 5

从产品需求开始之初即可编写测试用例、组织测试计划

创新的用例执行模式，方便快捷

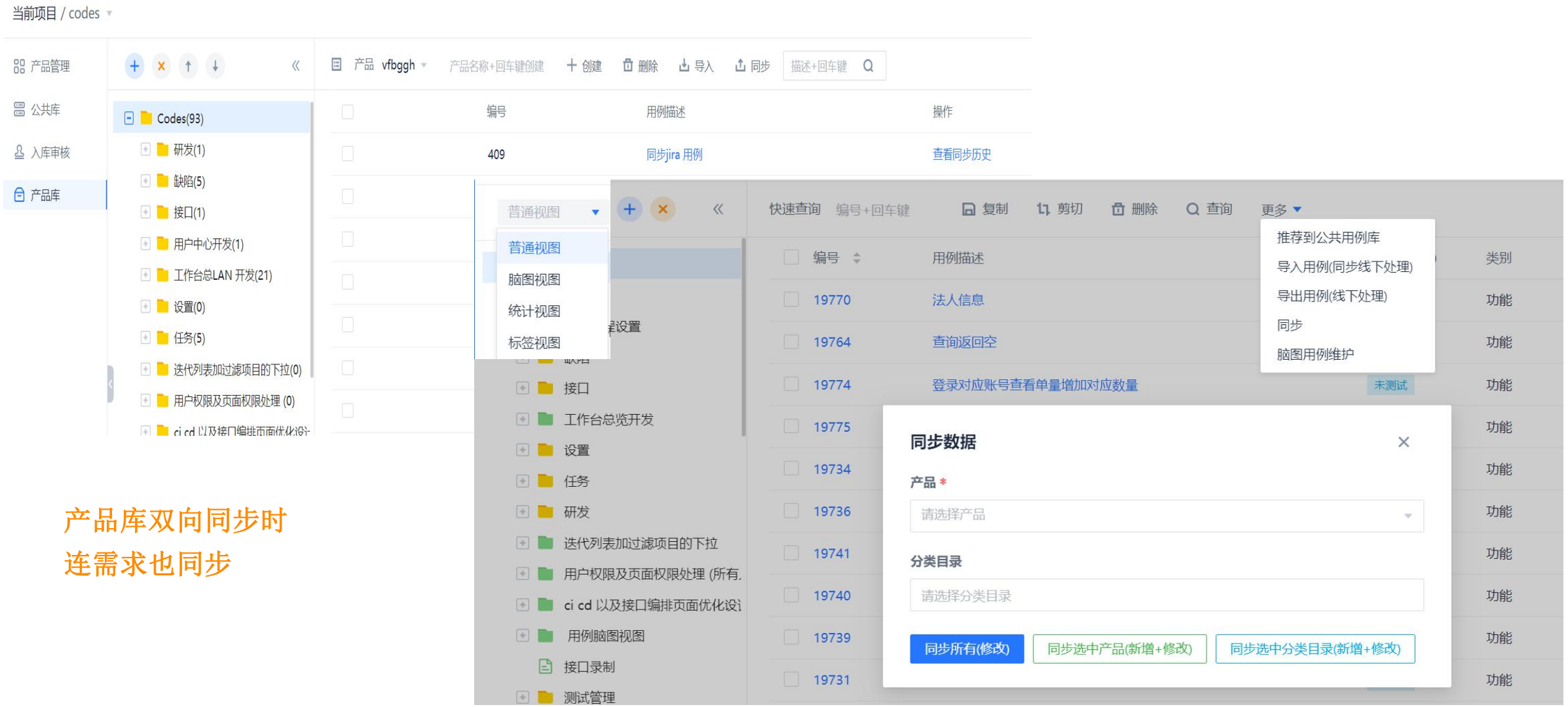
codes





以产品和例库和公共用例库实现用例重用，多种用例视图

codes

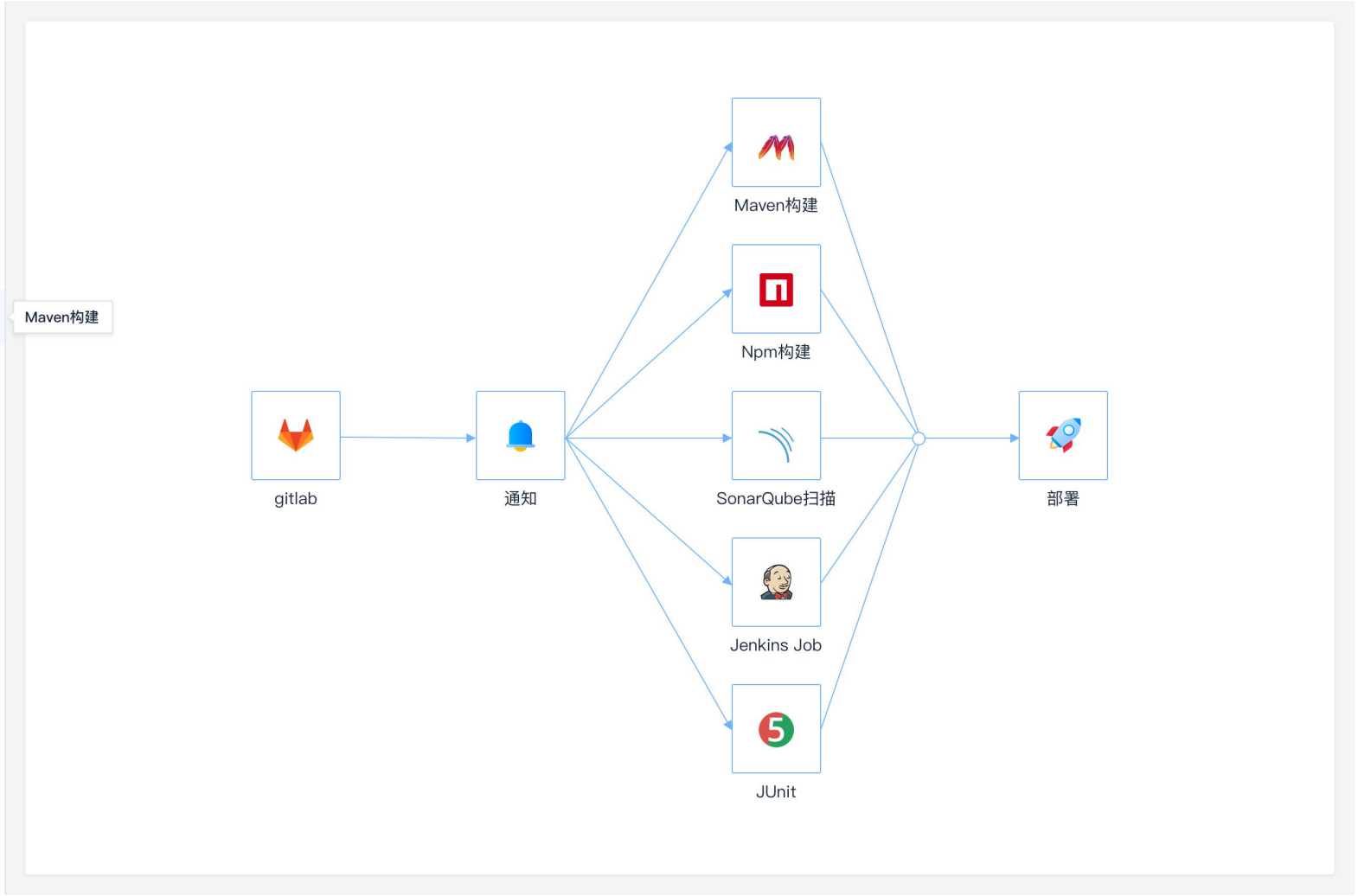


产品库双向同步时
连需求也同步

全局看板：保持透明度又统筹全局

以方向的方式通过定制查询来设置个人的看板，一板透视全局。

测试左右移的现状与问题：依赖于运维和研发，等靠要是常态



各阶段研代代码测试活动左、右移

- 产品需求开始之初即可编写测试用例、组织测试计划。
- 零代码拖拽式接口测试、以及接口场景编排。
- 自动推导接口拓补及调用链，便于快速完成缺陷重现、追踪。
- 零代码拖拽式CI CD流水线编排。
- “润物细无声”的方式实现代码扫描，代码管理与质量流程无缝对接，不增加测试人员认知负荷。
- “轻”运维，帮助测试人员零基础右移，打通测试与运维的屏障，提升测试效率。

接口测试的现状与问题：接口测试多样化

codes

01. 工具型接口测试

优势：

简单，易用，普及度广。

缺点：

规范难以制定，依赖个人的测试习惯。

场景：

灵活，适合个人。

02. 框架型接口测试

优势：

灵活度高，功能多。

缺点：

维护与编写成本高，更加依赖测试人员个人能力。

场景：

灵活的业务场景与测试方式。

03. 引入平台型接口测试

优势：

适合团队，方便度量和集成。

缺点：

闭源商业化平台价格较贵，开源平台的稳定性欠佳和功能较少。

场景：

适合团队和集成 DevOps 等。

04. 自研型接口测试

优势：

需求满足度高，自主可控。

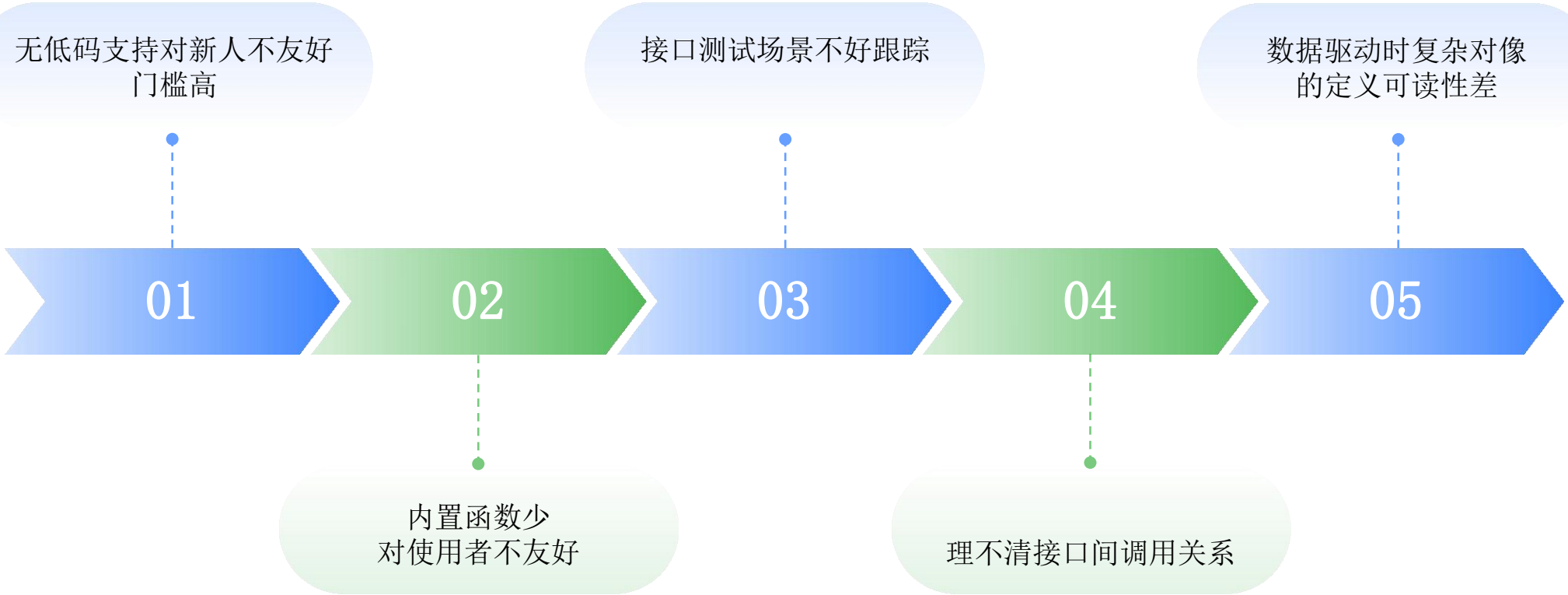
缺点：

投入产出比低，研发人力成本级别高。

场景：

适合团队和集成 DevOps。

接口测试常见问题


无低码支持对新人不友好
门槛高
接口测试场景不好跟踪
数据驱动时复杂对像
的定义可读性差
01
02
03
04
05
内置函数少
对使用者不友好
理不清接口间调用关系

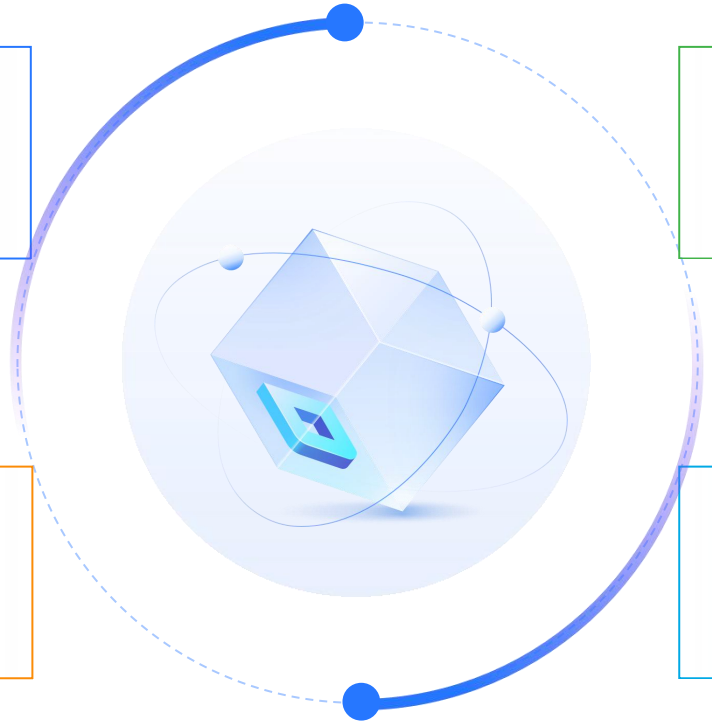
Codes接口测试主要解决什么问题

01

接口测试低代码化，组件解耦

02

接口测试调用链跟踪



03

接口依赖关系可视化，自动推导执行顺序

04

快速便捷的接口健壮性测试

低代码测试活动左、右移

接口用例 → 接口场景 → 接口mock → 接口编排

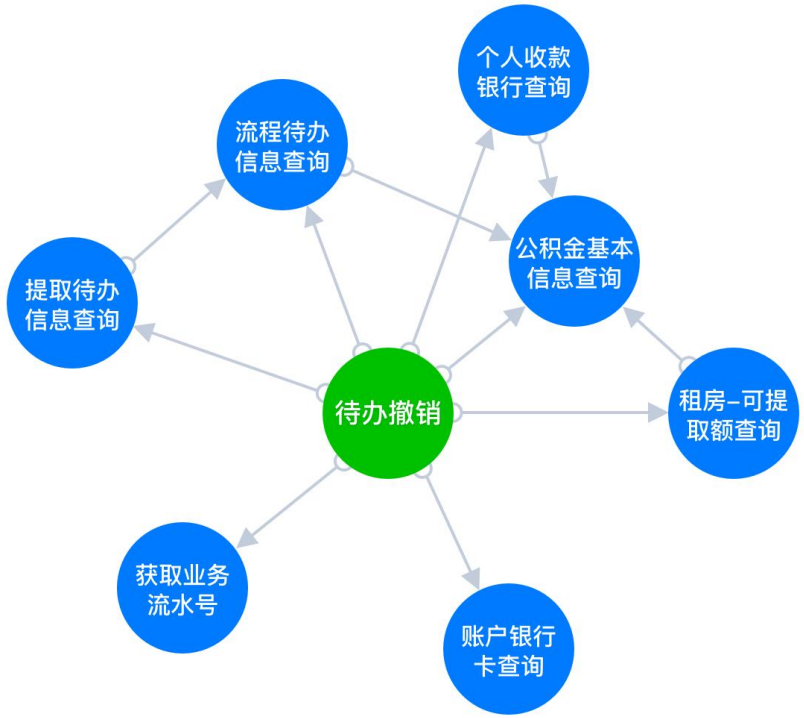
低代码测试活动左、右移


提取参数
响应数据
根节点
total
rows
rows_0
packageId
taskId
packageName
executor
execEnvironment
remark
createTime
updateTime
createrId
userId
user_id
第1 拖拽式提取，取个变量名
保存
重置
取消
请求参数
请求头
断言
提取参数
全局参数
拖拽式提取参数
编码提取参数
$.rows[0].userTestCasePkgs[0].userId:user_id
第2 自动生成提取表达式
请求参数
请求头
断言
提取参数
全局参数
+ 新增
编辑器模式
参数名称
类型
默认值
dto.user.loginName
string
testId
dto.user.password
string
第3 其他接口下拉选引用
可选择也可输入，标签之间用空格隔开
fdf_变量
dfg_变量
user_id_变量
authVar_变量
access_token_变量
啊啊_变量
time_变量
执行时间：15ms
{
"total": 288,
"rows": [
{
"packageId": "ff8080817a3d24ea017a3d8cecd30019",
"taskId": "ff8080817a0d94ee017a2dc4aeb73ccf",

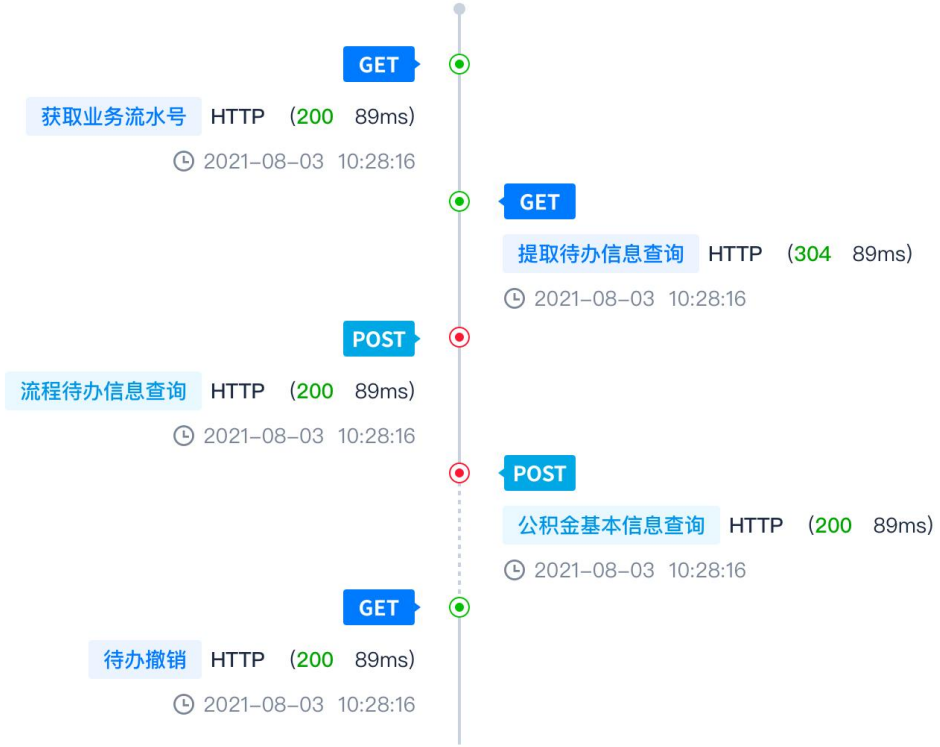
低代码测试活动左、右移

codes

待办撤销接口依赖拓扑图

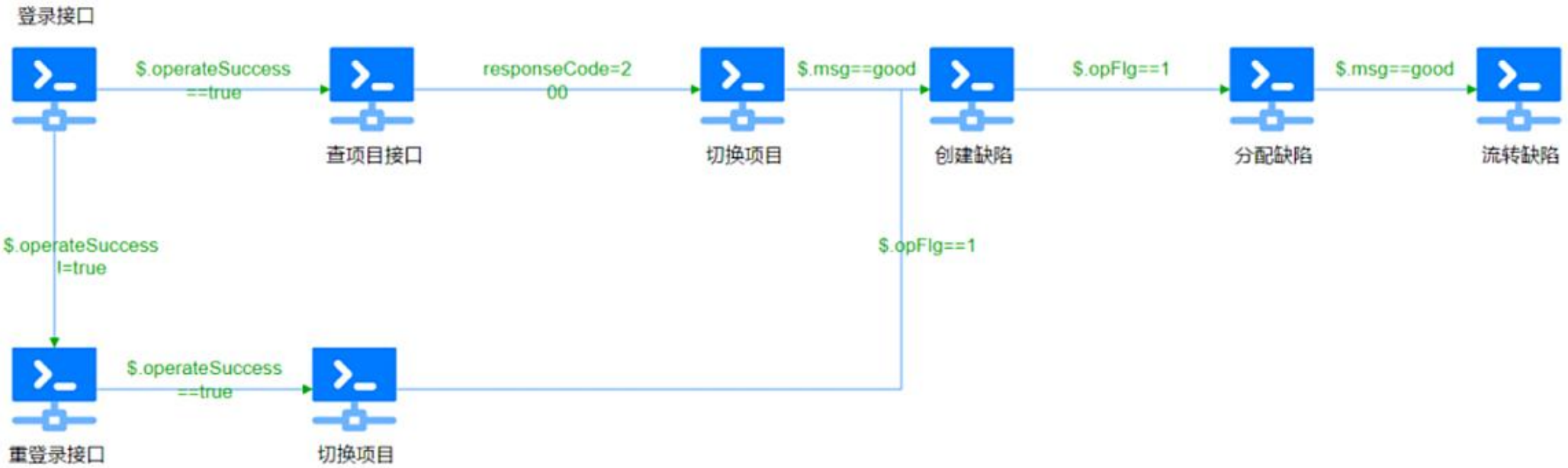


待办撤销业务场景调用链



自动推导接口依赖拓补关系图，让接口关系不再是黑匣子，便捷的接口调用链。

低代码测试活动左、右移


登录接口
$.operateSuccess
==true
查项目接口
responseCode=2
00
切换项目
$.msg==good
创建缺陷
$.opFlg==1
分配缺陷
$.msg==good
流转缺陷
$.operateSuccess
!=true
$.opFlg==1
$.operateSuccess
==true
重登录接口
切换项目

Codes的压测理念

零代码生成压测脚本

零代码配置分布式压测集群

零代码发起压测

点点点完成分布式压测

点点完成分布式节点管理，

让分布式压测环境分分钟准备好；

几步设置好压测试场景和压测参数，

分分钟跑完压测试场景；

重用接口用例为压测场景复用，

不再从0起步，助您弹射"起飞"。

低代码压测让实习生也可搞定压测

codes

当前项目 / codes

快速查询 关键字+回车键

+ 创建压测节点

节点名称	ip地址	jmeter端口	状态	节点类型	操作
华为云ECS	[redacted]	1099	禁用	单机	修改 | 安装 | 启动 | 删除
集群test			启用	集群	修改 | 节点维护 | 删除
stressTest	[redacted]	1099	禁用	单机	修改 | 停止 | 删除

生成压测文件

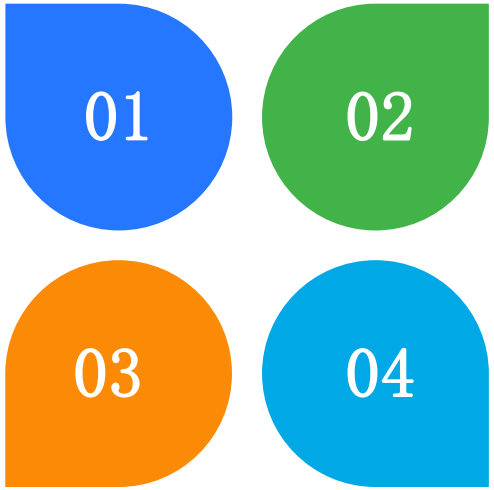
快速查询 关键字+回车键 压测文件名称 + 生成

接口名称	接口描述	描述	编写人
ew UP 3	fff	fff	
压测节点列表mock查询3	压测节点列表查询接口2	压测节点列表查询接口2	
测试接口断言	接口断言	接口断言	
压测节点列表mock查询2	压测节点列表查询接口2	压测节点列表查询接口2	
aaa	aaaa	aaaa	
ew	fff	fff	
压测节点列表查询1	压测节点列表查询接口1	压测节点列表查询接口1	

查看报告

请求...	请求...	出错...	出错...	平均...	发现人	响应...	最小...	最大...	90% L...	95% L...	99% L...	吞吐量	接收...	发送...
Total	2	0	0 %	83.5		83.5	44	123	123	123	123	11.90...	2.00	2.28
first	1	0	0 %	123		123	123	123	123	123	123	8.130...	1.37	1.56
second	1	0	0 %	44		44	44	44	44	44	44	22.72...	3.82	4.37

传统缺陷管理的现状



缺陷流转流程不合理，
不可按项目及团队进行调整

缺陷状态太少难以反应实况
流转和标准处理流程无关

缺陷流转用时信息难以收集

缺陷时效性评估难的问题
缺陷分析不充分

传统缺陷管理的现状

新人需要长时间磨合才能写出合格的BUG

BUG缺少关键信息，增加沟通成本

没有各环节流转用时信息

BUG流转没有有效的流转机制

BUG修改和代码提交无关联

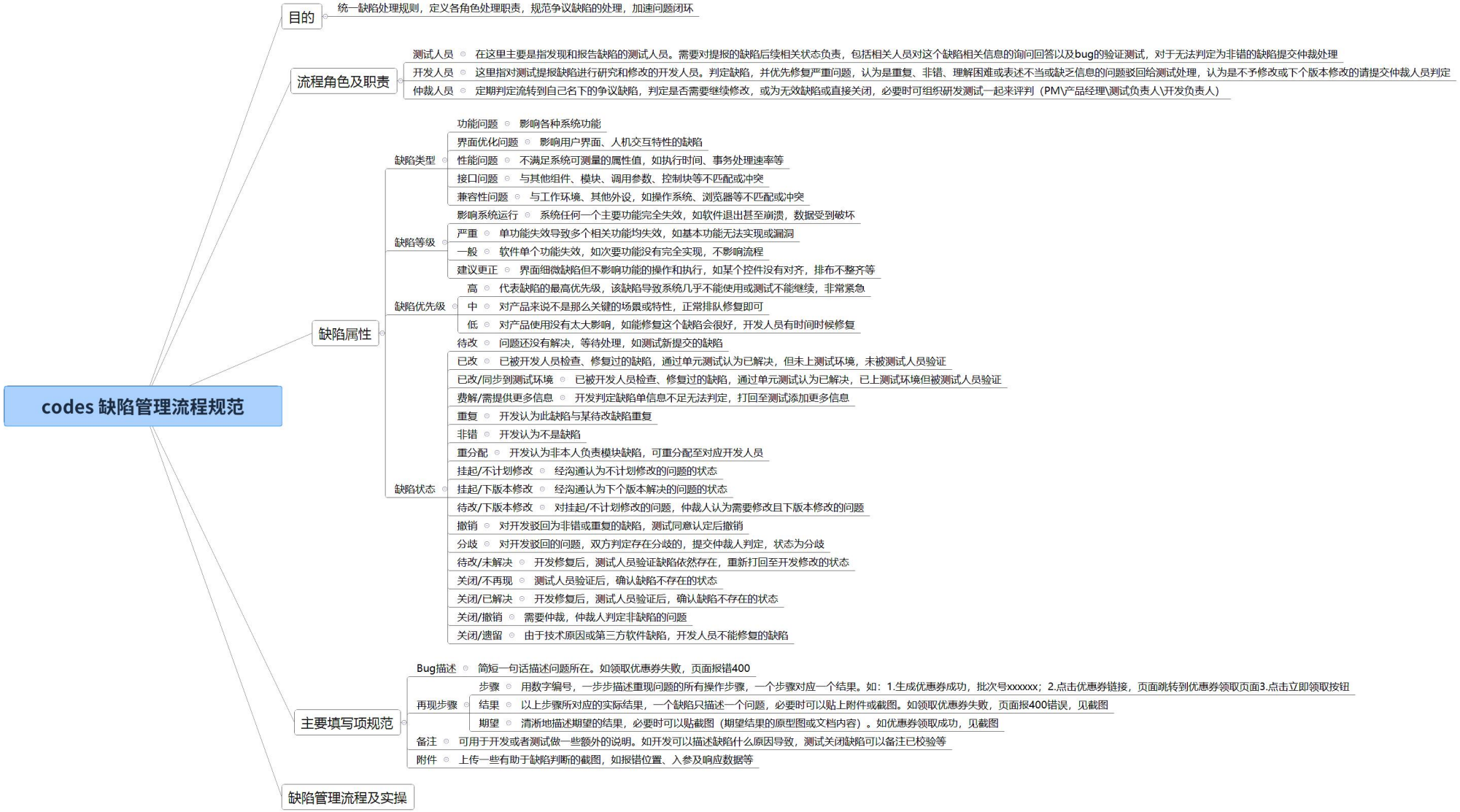
BUG状态不能精准反应BUG实际情况

Codes流程驱动缺陷管理

流程驱动测试，告别一刀切

“因地制宜”，可按需实时调整测试流程，
以反应不同管控目的；
不同流程对应不同的bug状态 ，
更能反应项目实况，
并根据流程推动bug状态的演化。

Codes流程驱动缺陷管理



Codes流程驱动缺陷管理

缺陷处理-1260 | 基本信息 | 流转历史及用时 | 附件信息 | commit历史 | 意见交流记录

分配用时：0　开发用时：120.64　仲裁用时：0　测试确认用时：308.26　整体用时：449.97　重开次数：2

- 刘一手 2023-02-21 21:23:26
  - 状态为: 待改
  - 新提交问题
- 陈一招 2023-02-25 21:06:08
  - (修改在版本:)把状态从:待改修改为:已改
- 刘一手 2023-03-08 13:41:45
  - (校验在版本:1.1)把状态从:已改修改为:待改/未解决
  - 39上 不知是没更新 还 是没改对
- 陈一招 2023-03-09 14:37:38
  - (修改在版本:1.1)把状态从:待改/未解决修改为:已改
  - 39上 不知是没更新 还 是没改对
- 刘一手 2023-03-11 18:17:56

缺陷时效统计分析　统计指标 所有　需求范围 请选择　时效维度 BUG等级　起始时间 2022-11-03 00:00:00　结束时间 2023-03-11 23:59:59

缺陷时效统计分析结果

指标	缺陷达标数	缺陷总数	达标率
开发组缺陷修复时效达标率	308	1021	30.16%
测试组缺陷复测时效达标率	334	1009	33.10%
缺陷仲裁时效达标率	6	7	85.71%
缺陷分配时效达标率	0	0	0%
缺陷修复时效达标率	88	1020	8.62%

缺陷耗时	缺陷平均耗时
105010	102.95

Codes流程驱动缺陷管理

缺陷和需求关联，随时可查看需求详情

缺陷还可以用例关联

通过webhook 还有和gitlab自动关联commit

需求以及任务也可以和gitlab自动关联commit

工具链平台常见问题

codes

01 研发过程是一个黑盒，风险滞后、各种问题只能靠问和最终结果判断

02 人员能力参差不齐、工程实践需要学习和适配

03 业务-产品-研发的协作摩擦多、需求/架构/代码的知识传递效率低、失真度高

04 自动化的工具很多、但能高效解决工程问题的少，工具带来很大认知负担

05 研发平台/工具的学习、使用成本高、为了使用而使用

06 研发流程没有标准化、日常工作中的协作界面/人数/角色多

工作台:综合看板，一切工作可以工作台办理，3分钟入门

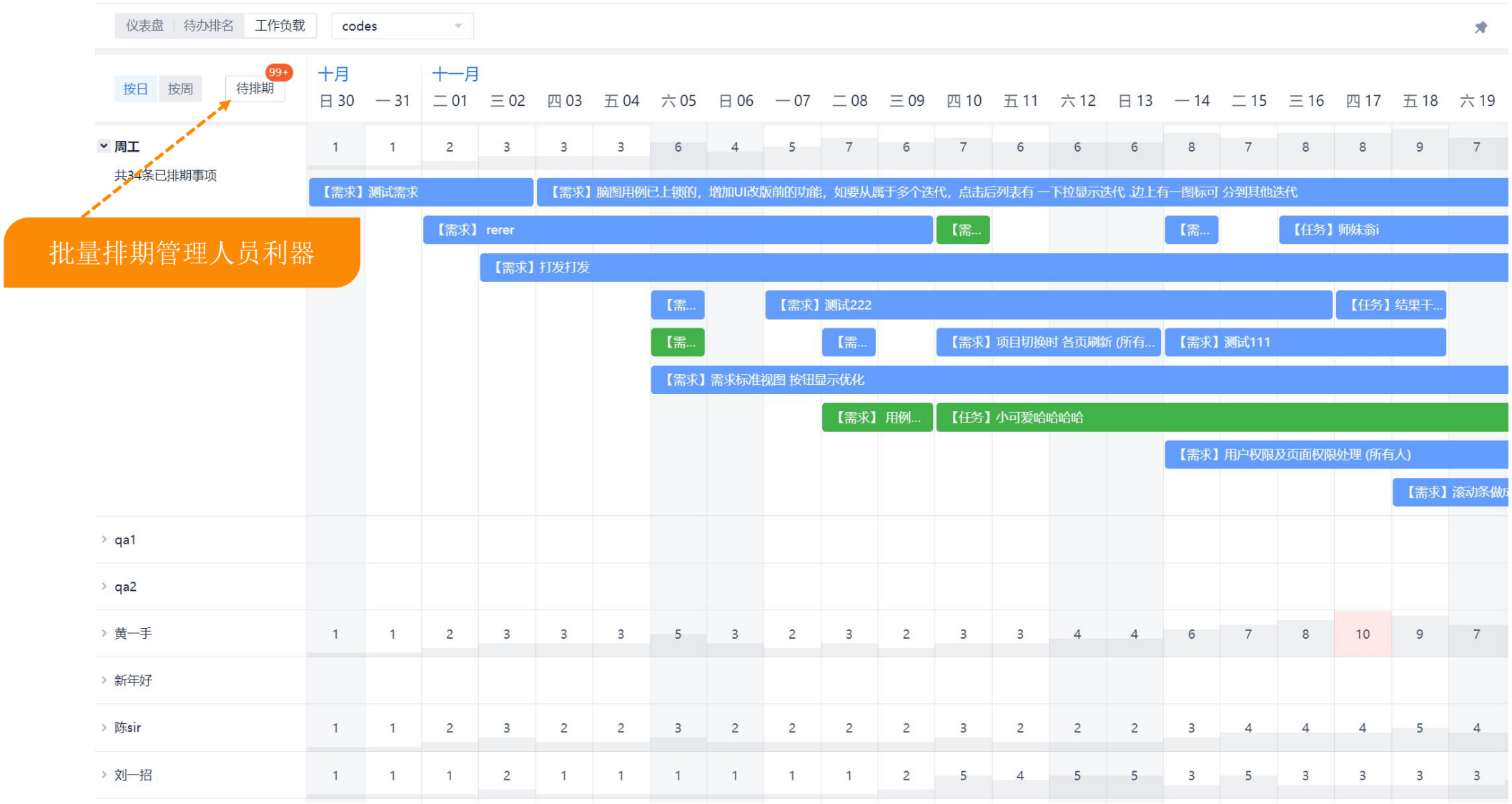
codes

可分别满足不同角色需要并可按需定制；能满足管理和执行人员的不同需要。

工作台:综合看板，一切工作可以工作台办理，3分钟入门

codes

工作台:综合看板，一切工作可以工作台办理，3分钟入门



工作台:综合看板，一切工作可以工作台办理，3分钟入门

codes

总览 我的事项 全局事项 我创建的 迭代|计划 看板 统计分析 动态

待批 6 | 需求 26 | 任务 42 | 缺陷 16 | 用例 27 | 请输入缺陷名称+回车键 | codes

编号	缺陷描述	状态	流
1468	邀请邮件中点接受邀请，自动跳到登录页时 应自动把帐户填上	已改	2
1455	更多查询条件时，选择了进度 填进度的框不显示 移光标上去才显示 这很不友好	已改	2
1464	邀请用户时，免费的只要超过30人就不能再邀请	已改	2
1449	迭代下人员分工 加一列 显示各人员的总进度	待改	2
1441	BUG 设置为已改或是已改同步到测试环境时，显示工时信息并可以填，同时也增...	待改	2
1437	工作台--看板--如选了具体项目，右边有一个加号的图标 指上云下拉浮出来 需求...	待改	2
1434	字典中初始化时增加一个待处理缺陷的集合状态（实现时和刘工讨论）	待改	2
1424	迭代中统计用例覆盖率	待改	2
1421	拖拉编排场景支持数据驱动	待改	2
1418	脑图用例加自动保存（实现是和刘工讨论，有可能出现加载出载导致内容被覆盖）	待改	2
1416	自动场景编排，也应支持设置图框起来的设计（优先级低实现时再找刘工）	待改	2
1252	测试--场景：在需求模块中已经删除的需求，在场景中不能显示被删除的需求	挂起/下版本修改	2
1195	缺陷编辑页面：附件信息删除，建议修改成：删除附件后，在基本信息中点击确定...	挂起/下版本修改	2

总览 我的事项 全局事项 我创建的 迭代 统计分析 动态 数字大屏

审批 8 | 需求 42 | 任务 40 | 缺陷 576 | 用例 34 | 接口 0 | 请输入缺陷名称+回车键 | codes | + 创建缺陷

编号	缺陷描述	状态	流转于	时长(h)	发现人	待处理人	报告日期
1325	需求详情中 工作量后后 H 不应换行	重分配	2023-03-11 18:49:06	21.61	刘一手	黄一手	2023-03-11
914	统计分析滚动条	关闭/不再现	2023-03-11 18:38:45	-	刘一手		2023-02-01
424	排期完成后，如有时间，vgantt 中点人员上事项的详情时，右这gantt 显示...	撤销	2023-03-11 18:36:25	-	刘一手		2022-12-03
911	报表不出下拉滚动条	关闭/不再现	2023-03-11 18:35:55	-	刘一手		2023-02-01
329	进了工作台 再进 测试总览 就没滚动条了	挂起/下版本修改	2023-03-11 18:33:05	21.88	刘一手	刘一手	2022-11-26
1183	在自动升级的类中加一个SQL 数据是写死的，插入一个菜单 在设置下的集成...	关闭/已解决	2023-03-11 18:27:46	-	刘一手		2023-02-18
1255	样式应和函数那那一样	关闭/已解决	2023-03-11 18:27:20	-	刘一手		2023-02-21
1263	接口帮助 数据驱动TAB 加步骤内的内容	关闭/已解决	2023-03-11 18:27:11	-	刘一手		2023-02-22
1299	任务模板中 这几下拉 无数据换为 在字典中 维护	关闭/已解决	2023-03-11 18:25:17	-	刘一手		2023-03-07

共有 576 条数据 < 1 2 3 4 5 ... 64 > 9 条/页

工作台:综合看板，一切工作可以工作台办理，3分钟入门

总览 我的事项 全局事项 我创建的 迭代|计划 看板 统计分析 动态

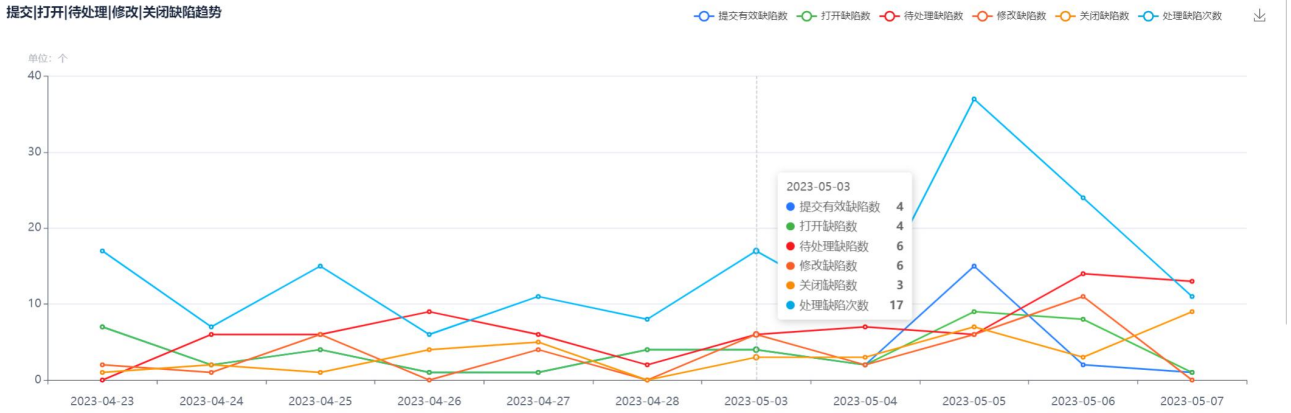
待批 0 | 需求 23 | 任务 7 | 缺陷 44 | 用例 5 | 请输入缺陷名称+回车键 | demo | 所有待处理人

编号	缺陷描述	状态	流转于	时长(h)
802	错别字(字典中是正常的)	待改	2023-05-07 20:24:58	15.41
796	工作台--统计分析---测试--开发工作量分析，只要龄期不断从天和周之前切换或更...	关闭/已解决	2023-05-07 13:34:49	-
792	测试下的场景模板check box 要去掉	关闭/已解决	2023-05-07 13:34:26	-
795	图1完成的需求是8个 但图2需求脑图中统计的是4个 ,图2的统计不对	关闭/已解决	2023-05-07 13:34:17	-
794	引导页 画线那一句不通顺 要更改为：Codes以邮件方式邀请用户	关闭/已解决	2023-05-07 13:33:34	-
799	测试需求树左左的树高度不对 参考缺陷管理左边的树	关闭/已解决	2023-05-07 13:33:24	-
793	两个括号 全半角不同 应统一为英文	关闭/已解决	2023-05-07 13:31:35	-
797	如图光标指上迭代名称时，应悬停提示查看迭代报告	关闭/已解决	2023-05-07 13:31:28	-
791	看板如不是管理人员 ，人员下拉缺省要显示为本人	关闭/已解决	2023-05-07 13:31:19	-
800	指上去时应有悬停提示 按项目查询	关闭/已解决	2023-05-07 13:31:09	-
789	建议需求详情，各行文本间改为和用例一样	挂起/不计划修改	2023-05-06 20:08:49	39.68
788	建议任务的窗口也改为侧滑	挂起/不计划修改	2023-05-06 20:05:09	39.74
790	任务框起来的几个TAB 缺省负责人应是所有，不应是当前人员	关闭/已解决	2023-05-06 19:37:24	-

总览 我的事项 全局事项 我创建的 迭代|计划 看板 统计分析 动态 数字大屏

迭代 需求 测试 缺陷 接口 | demo

提交|打开|待处理|修改|关闭缺陷趋势 起始时间 2023-04-23 00:00:00 结束时间 2023-05-07 23:59:59



日期	提交有效缺陷数	打开缺陷数	待处理缺陷数	修改缺陷数	关闭缺陷数	处理缺陷次数
2023-04-23	7	7	-	2	1	17
2023-04-24	2	2	6	1	2	7

Codes不止是测试

常见研发管理困局


仅以开发任务进行迭代
迭代
联动
测试关相活动和需求关系不直观，脱节
流程规范
研发活动及产出物和需求的生命周期脱节，需要手工去梳理费时费力
跟踪
协同
研发任务和需求弱关联，割裂协同难

常见研发管理困局

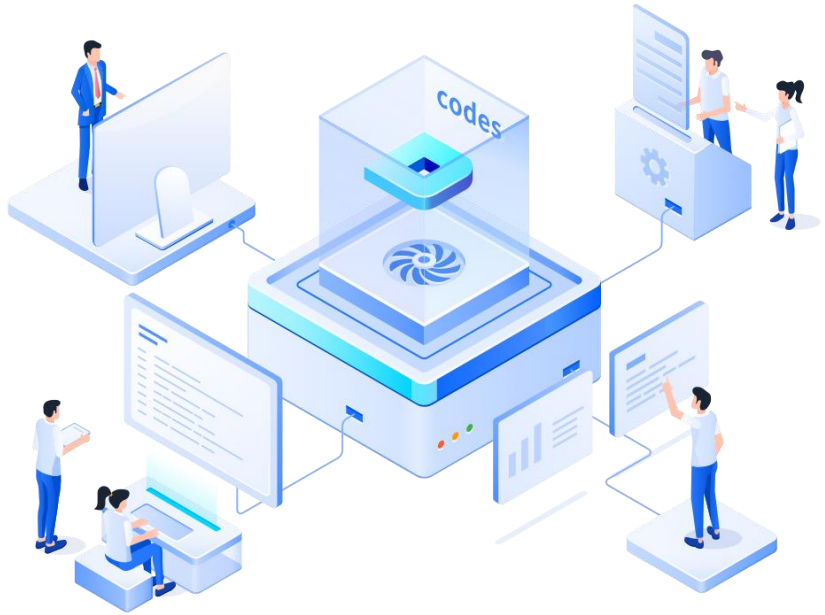
codes

管理规范不健全

没有质量管理体系及质量控制...

团队整体水平不强

人员梯队，技术能力，工程能力，敏捷基因...



发展空间小

新技术，培训体系，能力模型，知识传递靠口口相传...

基础设施不完整

软硬件，环境管理，平台工具...

工程文化意识弱

质量意识不强，测试管理薄弱...

Codes还能解决研发管理的一些问题

人员负荷可观测性差

进度瓶颈难于识别

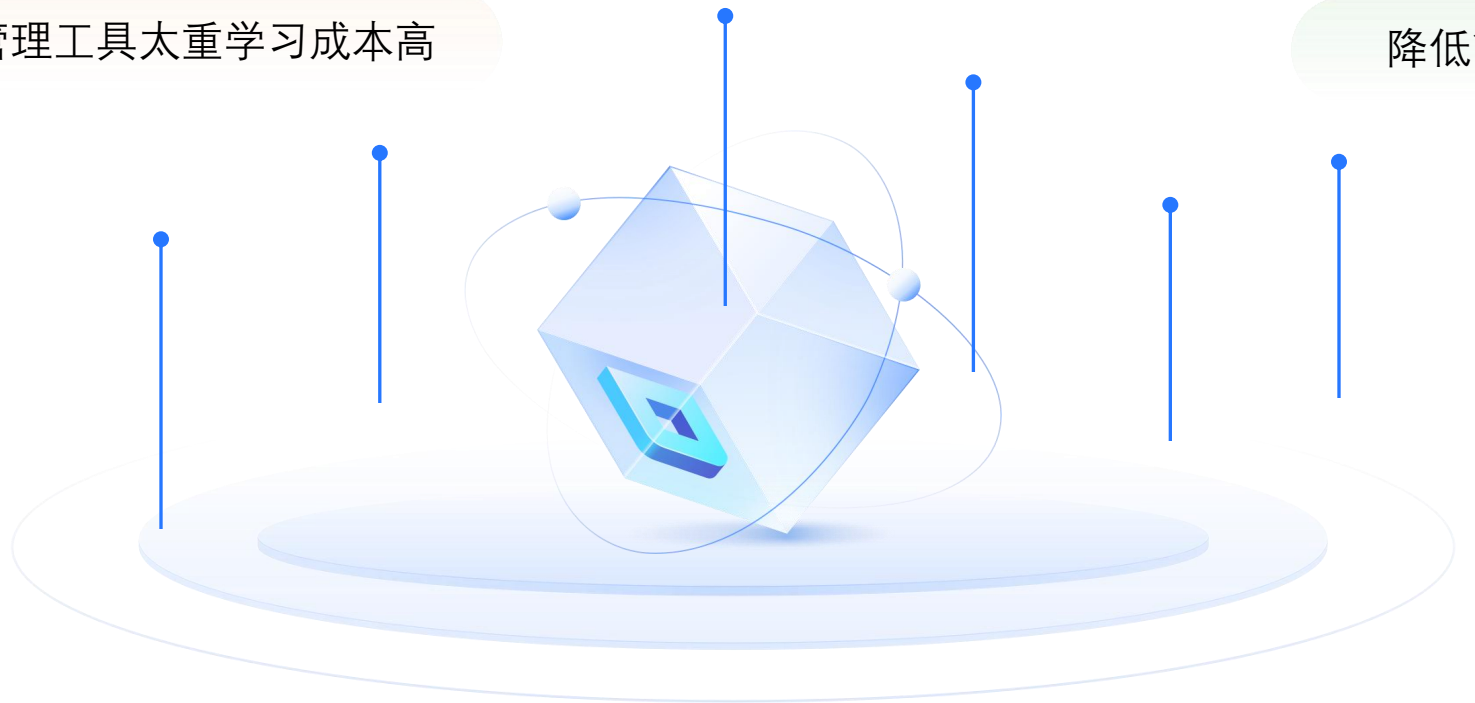
产出难以量化

任务排期麻烦

敏捷开发难以落地（流程、规范）

研发管理工具太重学习成本高

降低管理成本



常见需求管理问题

需求零乱，需求粒度不规范，一句话需求

需求变更随意，反复多，各方疲于奔命

需求和研发任务及测试弱关联，追溯困难

01

02

03

04

05

关键流程节点没把控好，相关评审流于形式

需求变更后同步不及时

Codes还能解决需求管理的这些问题

保证研发的源头可靠，减少后续反复带来的损失

项目成员间需求信息同步困难的问题

围绕需求拉通所有研发活动让需求可追溯

需求管理不规范

从需求可透视关联的一切研发活动

建立需求管理体系

Codes核心优势

以用户为中心， 降本增效

降低管理成本，优化管理过程，为用户提供快速devTestOps实施落地

打破研发黑盒，实现数智化管理

全场景数据精准可量化、为精细化运营高效产出提供管理抓手

简单易用，超低学习成本

开箱即用，唾手可得最佳实践，减少试错过程，成效快速获得

安全可靠，我的数据我作主

本地化部署，SaaS模式收费，快速成长型企业用得起用得好

匠心打磨，持续创新

无论是项目模式的融合，还是敏捷测试管理,以及接口测试、压力测试、CI CD等

我们都会持续进行业务创新，同时又践行简单实用的低代码理念

Codes初心

codes

为中、小型企业/团队效能提升做出自己的贡献

30人以下（含30人）免费使用、不限制功能、本地化部署

（注：30人以上仅按照多出部分人数计算价格）

https://icodes.work/

codes

智 能 管 理 研 发 · 数 据 驱 动 效 能

感谢您的聆听

thank you

www.icodes.work