

# 源于社区 服务社区

 中国DevOps社区峰会 2023 · 广州

## 零售行业数字化转型BizDevOps建设实践

陈顺生 无限极（中国）有限公司 测试与效能经理



# 陈顺生

无限极 测试与效能经理

擅长测试管理、敏捷与DevOps

- 公司JIRA、云效平台、Coding产品管理员
- 公司研发效能推广者



# 目录

1 背景和挑战

2 破局思路

3 实践和分享

4 收益和思考





## 背景和挑战

- 你眼中的交付团队？
- 最好的时代，最坏的时代？
- 数字化转型对软件交付的挑战



# 你眼中的交付团队？



## 面向业务

我的需求提交好久了，  
什么时候可以上线？  
——不敢承诺



## 面向产品

大家都说需求好急，  
排序怎么做合理？  
——没有规划



## 面向研发

大家都很忙，但是  
其他部门还说工作量不饱和？  
——没有证据

# 最好的时代，最坏的时代？

数据分析：描述性分析

各种工具的推出和使用

单一功能线上化 -> 数据孤岛

业务数字化：线下 -> 线上

信息化

DS/ES/LSS/LIMS/QMS...

数据分析：诊断性分析

按场景形成数据闭环，用户旅程

云+大数据：人与人，人与物

数字业务化：数据闭环，驱动运营

数字化

CDP/MA/DAM/CMS/标签/大数据/算法  
触点/极系列/商城小程序/企微/微信

数据分析：预测性+处方性分析

千人千面，千人千程，推荐任务

实现业务转型和创新

数智业务化：AI/算法驱动运营

智能化

千人千面/GPT/洞察/精准/驱动





# 数字化转型对软件交付的挑战

## 快速交付和迭代

过去按月发布的版本，现在基本都是按周按天计算

## 跨平台、复杂

从传统的管理系统延展到多触点，移动应用、云服务、SaaS、海量大数据

## 技术更新快

CDP/MA/DAM/大模型/NLP/标签/算法/  
大数据/实时计算

## 组织文化和人员技能

瀑布/敏捷/持续学习/创新/协作





## 破局思路

- 交付困境的三板斧
- 打造数字化时代的高绩效组织
- 敏捷迭代也要统一的流程





# 破局密码-三板斧

组织文化  
(思想破局、组织形态破局)

流程  
(破局)

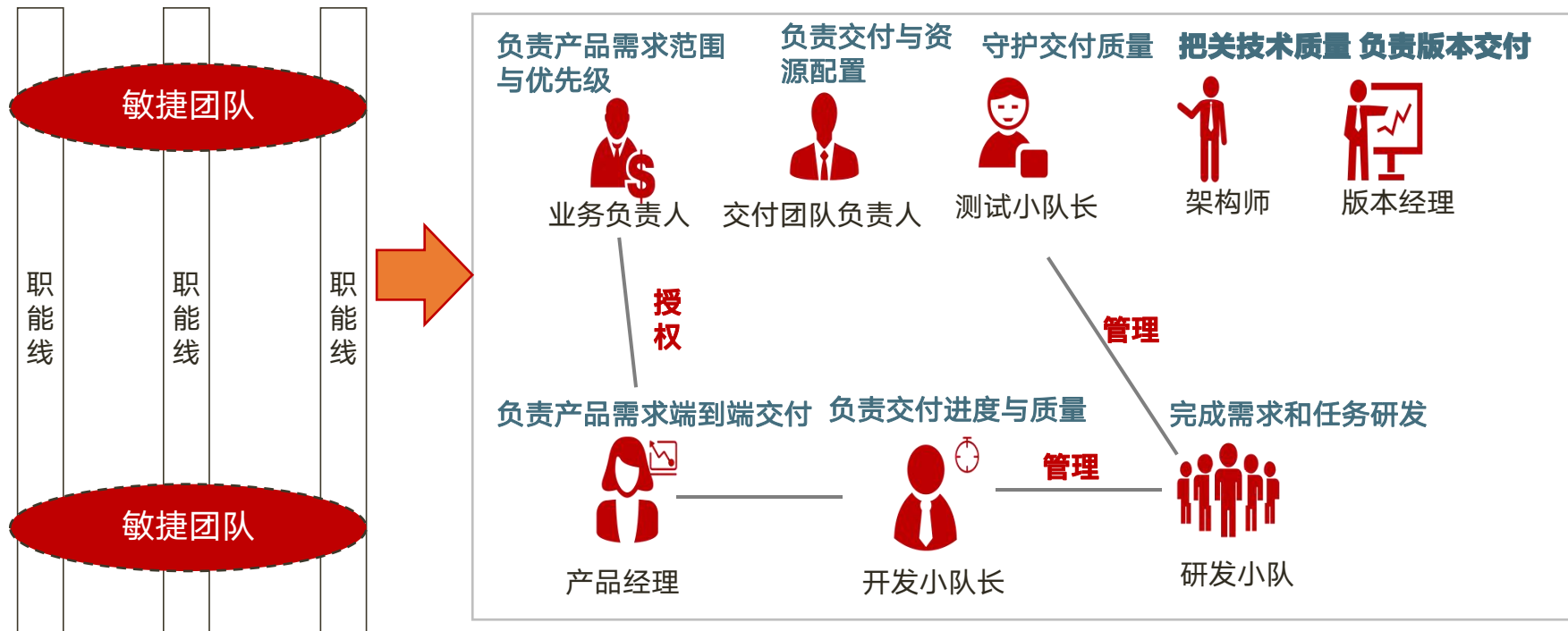
工具  
(赋能)

架构  
(升级)

数据驱动

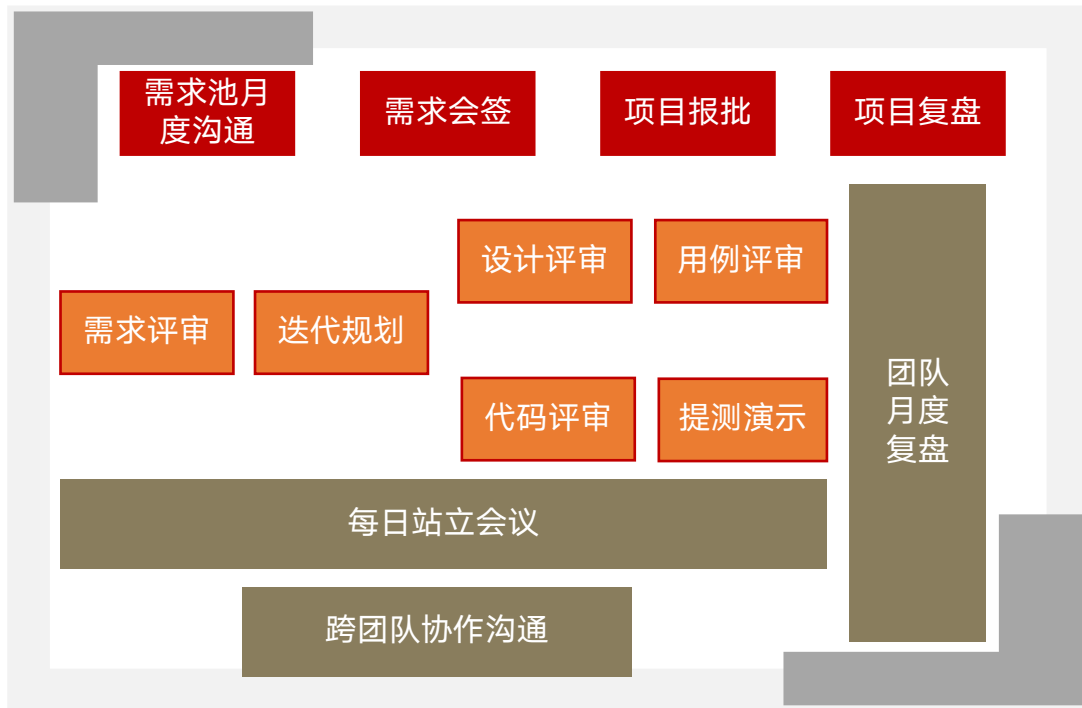


# 打造数字化时代的高绩效研发组织



随着“岗位”的边界消失，技能将替代岗位，成为员工和工作的连接点

- 流程管理级别 (M1、M2、PM)
- 团队级别 (scrum)
- 需求级别
- 共享团队日历

[illegible]

# 团队复盘会议资料

## 复盘会的目标和过程

### 会议目标

- 从整体上了解团队的研究效能情况。及时发现研究过程的问题。通过问题分析和深入分析，找出问题的根源和改进点，从而驱动团队的持续改进。

### 参会人员

- 必选：产品PM、设计PM、开发PM、测试PM、开发人员、测试人员。
- 可选：关心项目问题的管理层或业务人员。
- 主持人：由产品、开发、测试每月轮流组织，对团队的效能提升负责，负责团队迭代复盘会的准备和主持引导，并赋能团队成员主持复盘会。

### 参会时间

- 每月月底周四下午15:00-17:00（迭代发布第二天），时间尽量控制在2小时内。

### 参会地址

- 线上/线下，地址由主持人提前一周发出。

### 会议输入

- 团队当前的研究效能现状与效能目标的对照；
- 本次迭代燃尽图和工作项完成情况；
- 上一次复盘会的改进项列表；
- 相关物料准备（云共享文档）。

### 会议过程

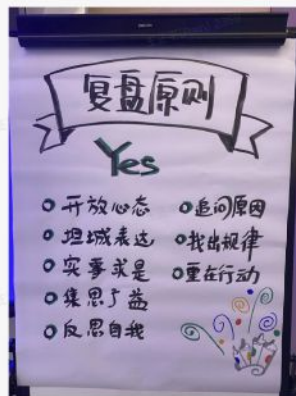
- 明确复盘会的原则和初衷；
- 确认上一次复盘会的改进项和完成情况；
- 讲解团队当前的效能情况，并明确与目标的差距；
- 讲解本次迭代的燃尽图和工作项完成情况；
- 参会人员真实反馈现状（花、苞、刺）；
- 分析数据和反馈，找出根源，明确改进方案；
- 落实行动，确定改进方案的负责人和期望完成时间。

### 会议输出

- 改进项列表，及对应的负责人和期望完成时间；
- 复盘会会议纪要。

## 明确复盘会的原则

YES



NO





# 好团队需要遵守的价值观

过生日文化，营造氛围

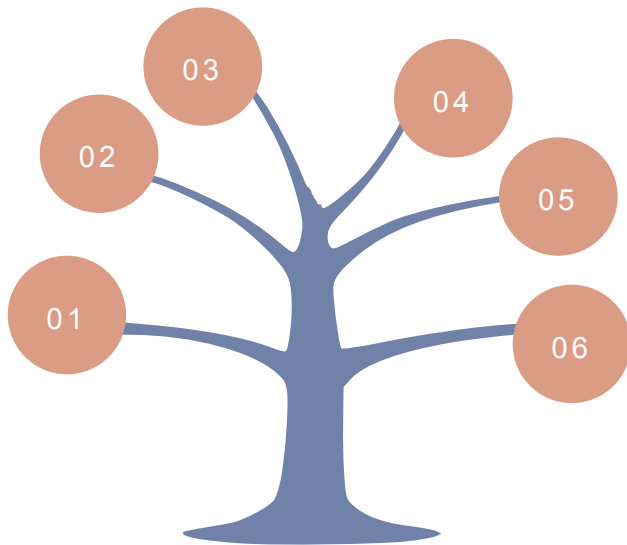
有奖有罚

透明公开，可视化工作

排期透明，进度透明，问题透明

信守承诺，协作提升

站立会议检讨到期日期



度量一切，持续改进

别啰嗦，看你的指标

经营意识,关注ROI

不谈钱，伤感情

知识分享，复制成长型组织

成为最好的自己-好东西要分享，  
培养主持：我做你看>你做我  
看>轮流负责

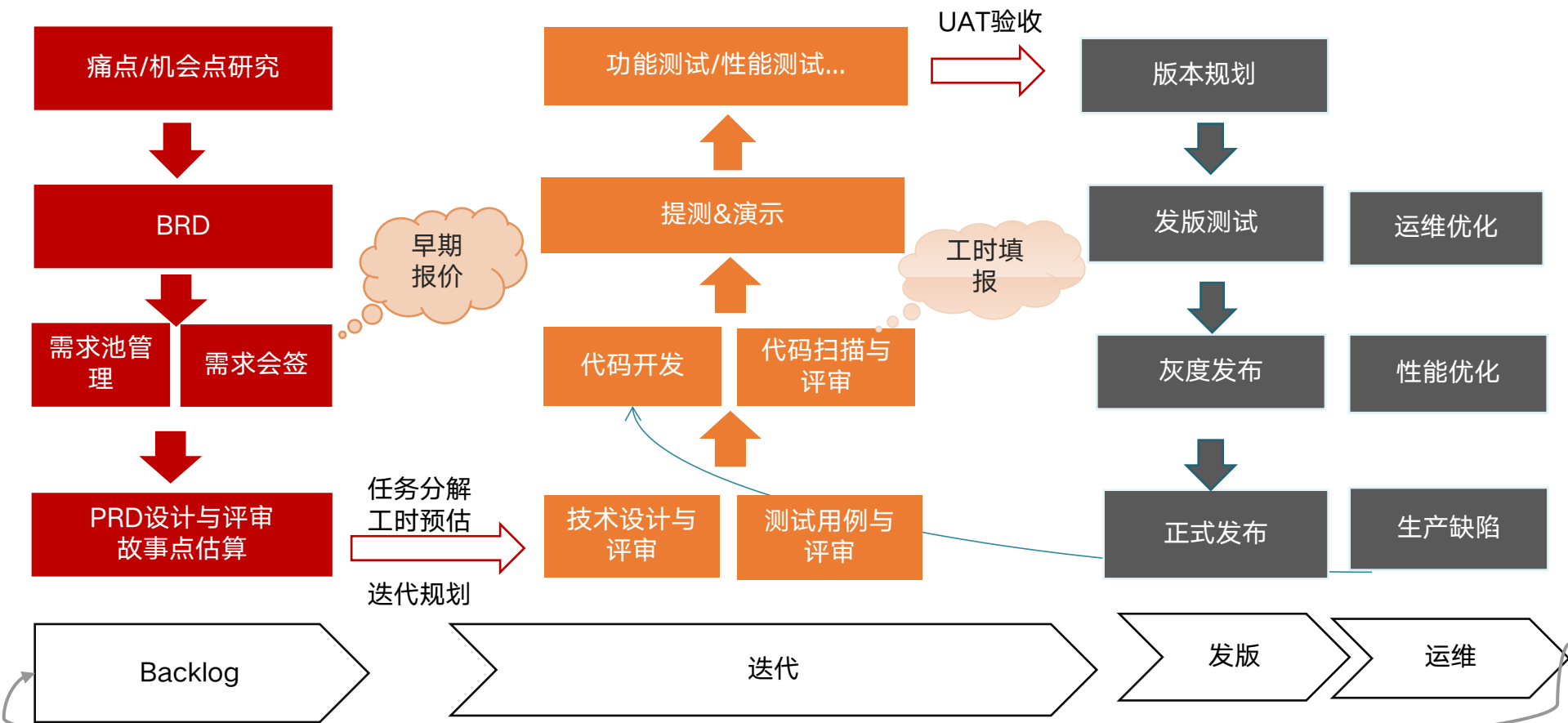


# 合适的人做合适的事

研发模式	简述	适用类型	优点	缺点
自有人力	员工	核心系统守护	熟悉业务、代码设计水平高，产出质量高	负载有限，容易成为瓶颈
ODC	在案外包人员，由员工PM带队	1、短平快的需求 2、产品迭代型	1、响应快速 2、可减轻员工具体实施工作 3、熟悉流程规范（正规军）	1、员工PM管理成本高 2、成本固定
供应商SOW	项目全包、半包	1、交付周期明确 2、交付范围清晰 3、独立完整性	1、灵活用工 2、具体实施过程介入少	1、采购周期不确定 2、规范性执行弱、较少考虑通用扩展性及运维 3、项目验收把控要严谨 4、核心人员变动影响大



# 敏捷迭代也需要统一的流程





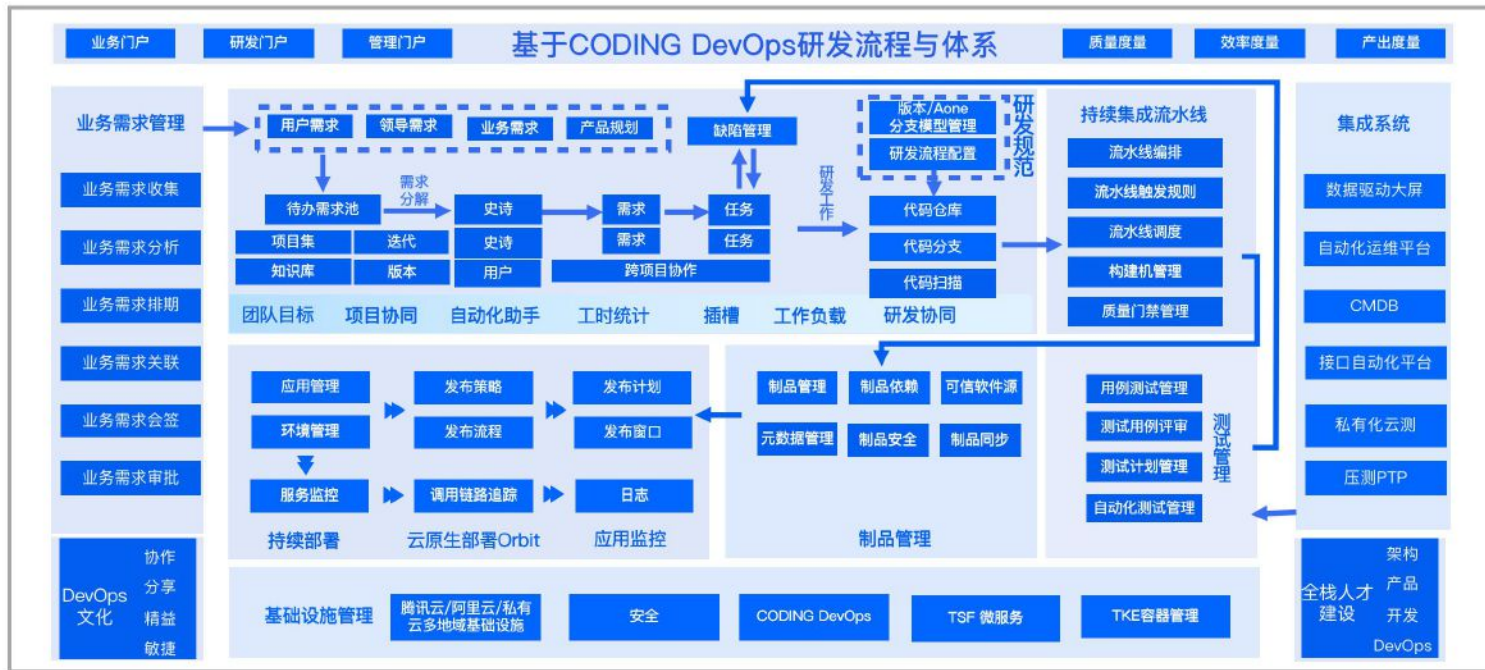
## 实践和分享

- CICD
- 质量内建
- 数据驱动
- 架构升级





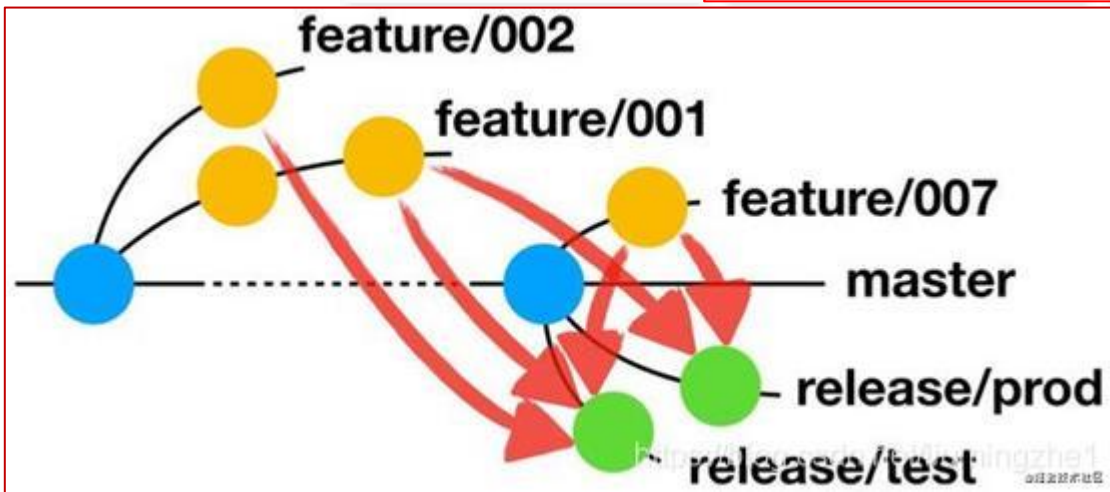
# 插上工具的翅膀



# 研发分支模型

<b>单分支规范</b> <small>系统</small> 支持 master 分支  📄 使用中仓库 49 个	<b>功能分支规范</b> <small>系统</small> 支持 master、feature/* 分支  📄 使用中仓库 21 个	<b>Gitflow 分支规范</b> <small>系统</small> 支持 master、develop、feature/*、release/*、hotfix/* 分支  📄 使用中仓库 5 个
<b>GitLab 分支规范</b> <small>系统</small> 支持 master、feature/*、*-stable、environment/* 分支  📄 使用中仓库 0 个	<b>F&amp;S flow 分支规范</b> <small>团队</small> 适用于多分支迭代并行的中大型开发团队 使用快慢车发布，支持 master、...  📄 使用中仓库 2 个	<b>CD flow分支规范</b> <small>团队</small> 适用于开发与测试中心（Center of Development）一般规模开发团队，支...  📄 使用中仓库 64 个

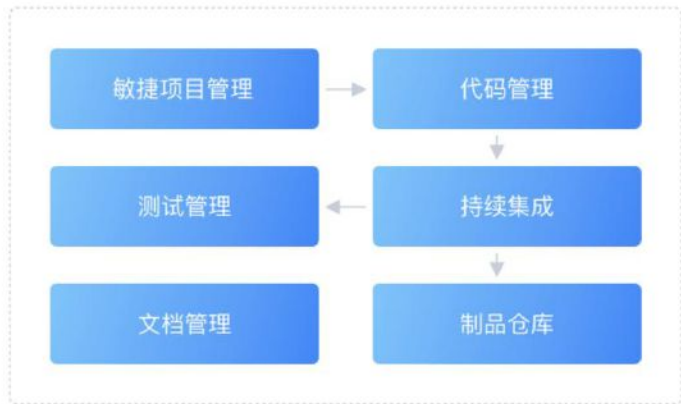
Aone Flow



# CICD



## CODING 项目



## CODING 部署控制台



# CICD

## 开发与运维人员共享

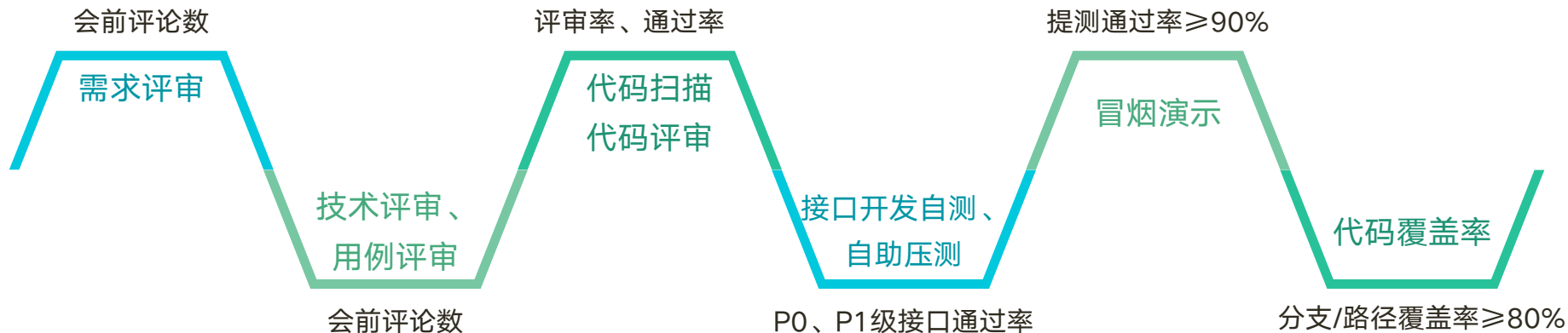
- 构建模版
- Deploy模版(gitops)
- 统一制品库

The screenshot displays the Design-Center interface, which is used for managing applications and deployments. The interface is divided into several sections:

- Left Sidebar:** Contains navigation links for '项目概览' (Project Overview), '项目协同' (Project Collaboration), '代码仓库' (Code Repository), '研发规范' (Development Standards), '持续集成' (Continuous Integration), '持续部署' (Continuous Deployment), 'Kubernetes', '云服务器' (Cloud Servers), '静态网站' (Static Websites), '制品库' (Artifact Repository), '测试管理' (Test Management), and '文档管理' (Document Management).
- Application List:** A table listing applications such as 'svelte', 'mac-dev-playbook', 'cross-node-guide', 'chrome-aws-lambda', 'rundack', and 'pyra-check', each with a link to its deployment process.
- Deployment Details (CODING-mytest):** Shows the deployment process for 'CODING-mytest', including the manager 'Ruth Clark', the start time '2020-02-12 15:23', and the number of deployment processes '23'.
- Deployment List:** A table showing the status of various deployment processes, including '执行中' (Running), '已完成' (Completed), and '失败' (Failed), along with the application name, deployment process name, duration, creation time, and creator.
- Deployment Flow:** A detailed view of a deployment process, showing the '基础信息' (Basic Information) and the '阶段' (Stages) of the deployment. The stages include 'Find Image from Tags', 'Patch(Manifest)', 'Find Artifacts From Resource(Manifests)', and 'Patch(Manifest)'.



# 质量内建



测试覆盖率

测试覆盖率

系统设置

测试覆盖率

opp-meeting

+

5543

控制台

测试覆盖率

搜索

执行

Id	分支	类型	功能分支	基线提交	被测提交	分支覆盖率	路径覆盖率	执行日期	执行人	执行状态	失败原因	操作
2-2-1700012606615_merge	release/daily	功能	feature/opp-58035	146cc1	810f7a	43.54%	05.11%	2023-11-15 09:46:46	5543	完成	-	<div>查看报告</div> <div>最近执行记录</div> <div>重新执行</div>





# 变更发布

## 变更宣言

- 所有的变更都应被记录，以便于跟踪和回溯。
- 我们知道自己不知道，因此会主动协调相关干系人全面评估变更影响和风险。
- 变更影响将在实施前且尽早地通知受影响用户。
- 对“生产保持敬畏”和完整清晰的变更步骤清单，是最有效的风险管控措施。
- 我们追求技术上的精进，致力于把所有的变更实现自动化、智能化。
- 技术验证和业务验证，是变更实施后不可省略的环节。
- 事前对变更技术方案进行测试，是变更成功的最有力保证。

Coding DevOps 机器人 BOT

变更类型: 紧急变更申请

变更申请人: 强

变更原因: 解决 APP 端 报表无效数据返回问题

变更对象: gbss

变更版本: V230901.40f01【紧急升级】

变更内容: 【Coding 链接】 <https://infinitus-devops.coding.net/p/gbss/releases/113503/issues>

变更策略: 滚动升级

变更时间: 2023-08-31 19:30--20:30

变更影响: 变更期间部分业务可能出现无法正常使用的情况

风险评估和应对: 风险较小, 已经过测试。如有问题, 将回滚版本。

变更步骤:

1、通过 Coding/Jenkins 触发应用 CI/CD 流水线

2、按照版本升级配置清单 checklist 更新配置项 (数据库对象、MQ、nacos、网关导入、定时任务、菜单/功能配置等)

变更后验证: 登录系统任意访问相关业务, 验证核心功能。

Coding DevOps 机器人 BOT

请您点击以下链接确认是否执行本次变更?

<https://infinitus-devops.coding.net/p/gbss/ci/job/1518798/build/46/pipeline?step=20>

@ (Rain)

Coding DevOps 机器人 BOT

请您点击以下链接确认本次版本是否已通过测试?

<https://infinitus-devops.coding.net/p/gbss/ci/job/1518798/build/46/pipeline?step=26>

@ ..... (Abbie) @, (Hardy) @, (Anson)

一般变更申请

重大变更申请

紧急变更申请

构建成功

手动触发

构建成功

陈顺生 手动触发

构建成功

手动触发

checklist 确认 BOT

发布清单: [https://doc.weixin.qq.com/sheet/e3\\_AMYAnZeAHoiar07DL9Qzqm01p1NF?scode=ACAA9AfwAagHq35P05AMYAnZeAHo&tab=BB08J2](https://doc.weixin.qq.com/sheet/e3_AMYAnZeAHoiar07DL9Qzqm01p1NF?scode=ACAA9AfwAagHq35P05AMYAnZeAHo&tab=BB08J2)  
尊贵的开发负责人, staging 已测试通过, 请您确认【生产应用发布前】的清单是否已检查?

若清单通过检查, 请您点击以下的链接进行人工确认动作  
<https://infinitus-devops.coding.net/p/gbss/ci/job/2406942/build/39/pipeline?step=32>

@ (Ray) @, (Sam) @, (Cactus) @, (Ethan) @, (Jerry)



# 稳定性保障

日常保障

- 监控告警
- 安全众测
- 智能推荐配置规格
- 故障注入

业务高峰保障

- 活动保障方案
- 云商保驾护航
- 一键弹性扩缩容

平台修炼

- 自研AIOPS平台迭代
- 可观测平台集成

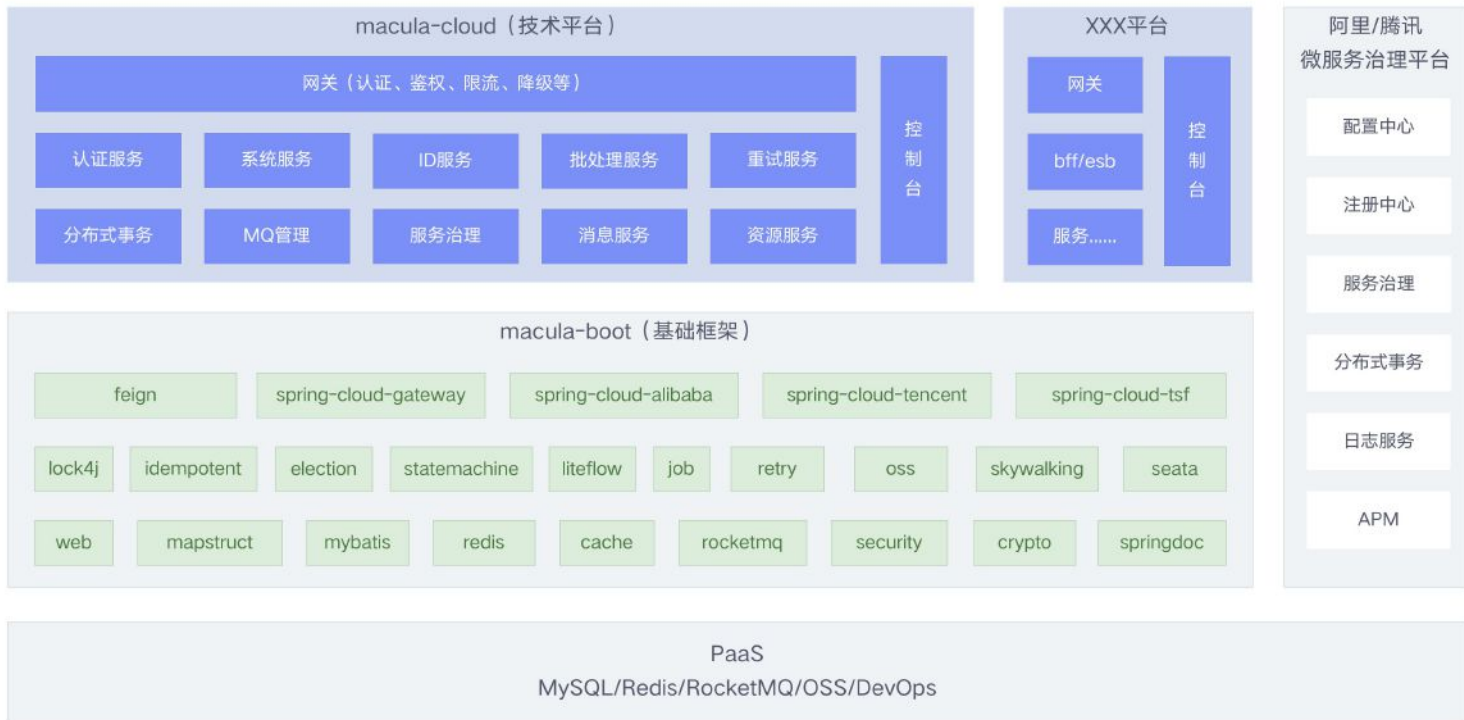




# 敏态和稳态-一切为了快速的响应

## 自研架构优势

- 行业公司特性
- 统一技术栈
- 快速上手、复用
- 避免重复造轮子

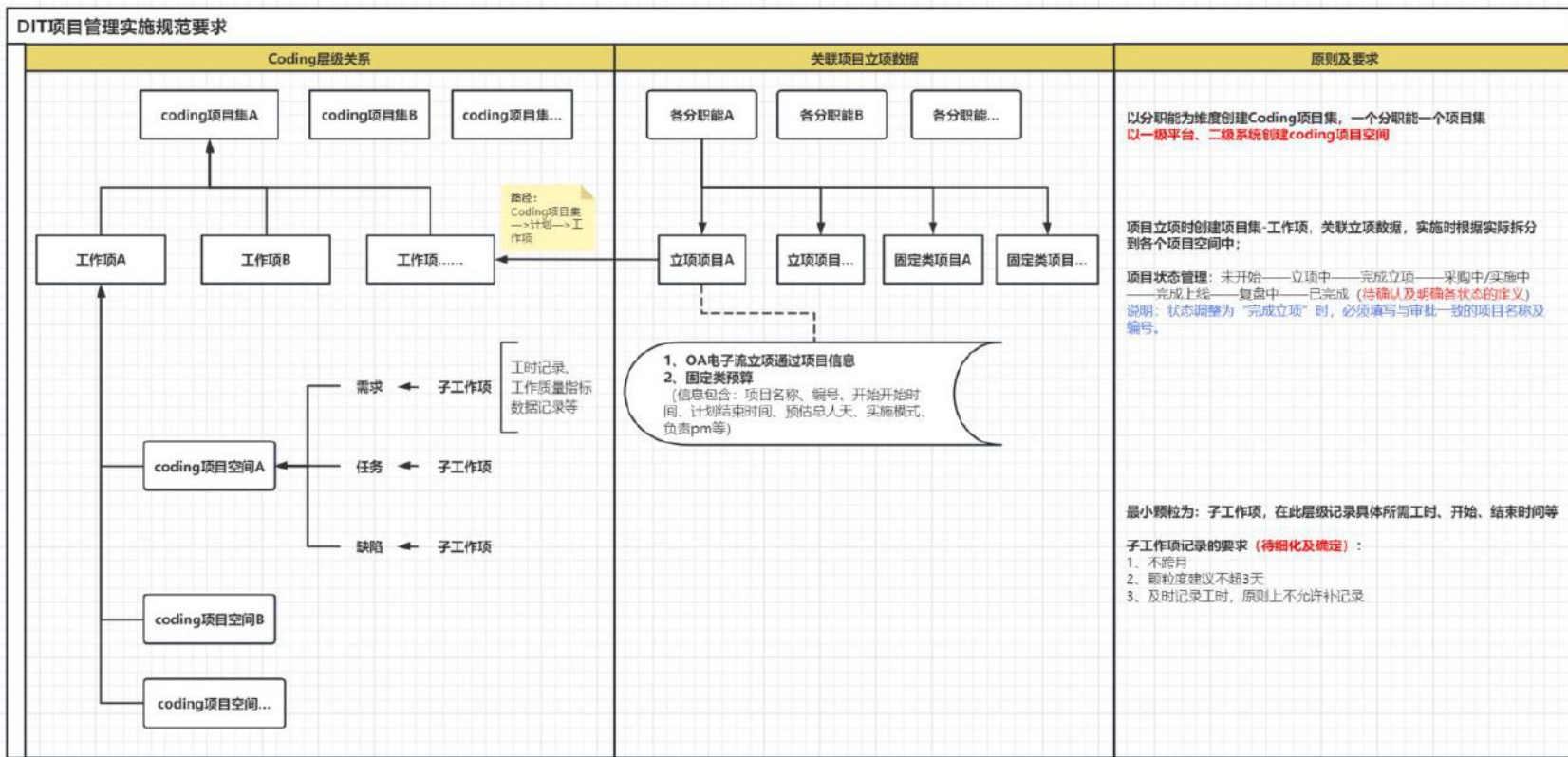




# 打通Biz流程

## 单个立项

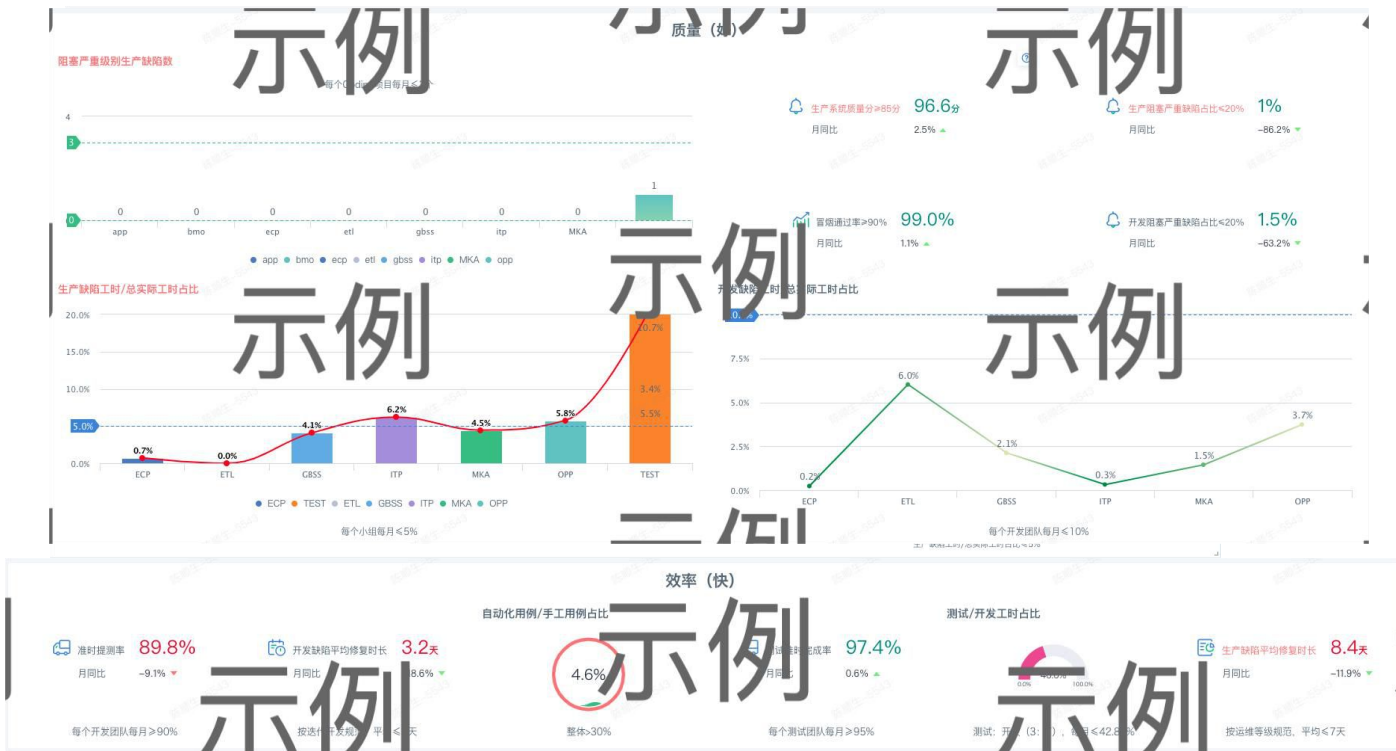
- 平台投入
- 供应商结算
- 财务复盘
- 业务复盘
- ROI复盘



# 数据驱动-效能提升看得见

## 自研效能大屏

- odc管理大屏
- odc结算大屏
- odc人员大屏
- sow管理大屏
- 版本发布大屏
- 产品质量大屏
- 迭代冲刺大屏
- 价值交付大屏





# 收益和思考

- 收益
- 思考





# 取得的收益

- 为公司创造价值，长在业务上

成本中心 --> ROI收益增强（经营思维）

- 需求端到端流程，提高人效重塑组织

部门壁垒 --> 协同流动、透明负载

被动等 --> 主动要

- 有计划的交付节奏，敢于承诺

迭代不清晰 --> 迭代3周-->2周

- 不用熬夜发布

“忐忑”上线 --> 灰度尝试

系统平台	开发缺陷严重级别占比	生产质量分	生产缺陷关闭周期
平台1	-59%	+6%	-54%
平台2	-56%	+3%	-1%
平台3	-53%	+9%	-12%
平台4	-58%	+21%	-2%



# 几点思考

## DevOps on AI

1. 智能文档生成
2. 智能代码生成、审查
3. 智能测试
4. ...

```
Question:
帮我创建一个用户表，用mybatis-plus写个mapper

Answer:
Sure, I can help you create a user table and write the mapper using MyBatis-Plus. Here's how you can do it:

1. Create a user table in your database. Here's an example using MySQL:

CREATE TABLE user (
  id BIGINT(20) NOT NULL AUTO_INCREMENT,
  name VARCHAR(50) NOT NULL,
  age INT(11) NOT NULL,
  email VARCHAR(50) NOT NULL,
  PRIMARY KEY (id)
);

2. Add the MyBatis-Plus dependency to your project. You can do this by adding the following to your Maven pom.xml:

<dependency>
  <groupId>com.baomidou</groupId>
  <artifactId>mybatis-plus-boot-starter</artifactId>
  <version>3.4.3.1</version>
</dependency>

3. Create a user entity class that maps to the 'user' table.

@Data
@Table(name="user")
public class User {
    @TableId(type = IdType.AUTO)
    private Long id;
    private String name;
    private Integer age;
    private String email;
}

4. Create a user mapper interface that extends BaseMapper<User>.

public interface UserMapper extends BaseMapper<User> {
}

That's it! You can now use the UserMapper to perform CRUD operations on the 'user' table. For example, to insert a new user:

User user = new User();
user.setName("John Doe");
user.setAge(30);
user.setEmail("john.doe@example.com");
userMapper.insert(user);

I hope this helped! Let me know if you have any questions.
```

