

AI 驱动 软件研发 全面进入数字化时代

中国·北京 08.18-19

AI+
software
Development
Digital
summit



数字化时代研发效能 提升策略与系统性方法

张乐 腾讯

▶ 演讲嘉宾



张乐

腾讯研发效能资深技术专家

- 前百度工程效率专家、前京东DevOps平台产品总监兼首席架构师
- 长期在数万人研发规模的一线互联网公司，负责研发效能提升、研发效能度量、敏捷与DevOps实践落地、一站式DevOps平台设计和研发工作
- DevOps运动国内早期布道者与推动者，DevOpsDays 中国区核心组织者
- 国内主流 DevOps、工程生产力、研发效能领域技术大会的联席主席、DevOps 及研发效能专场出品人、Keynote特邀嘉宾
- 《研发效能宣言》发起人及主要内容起草者



目录

CONTENTS

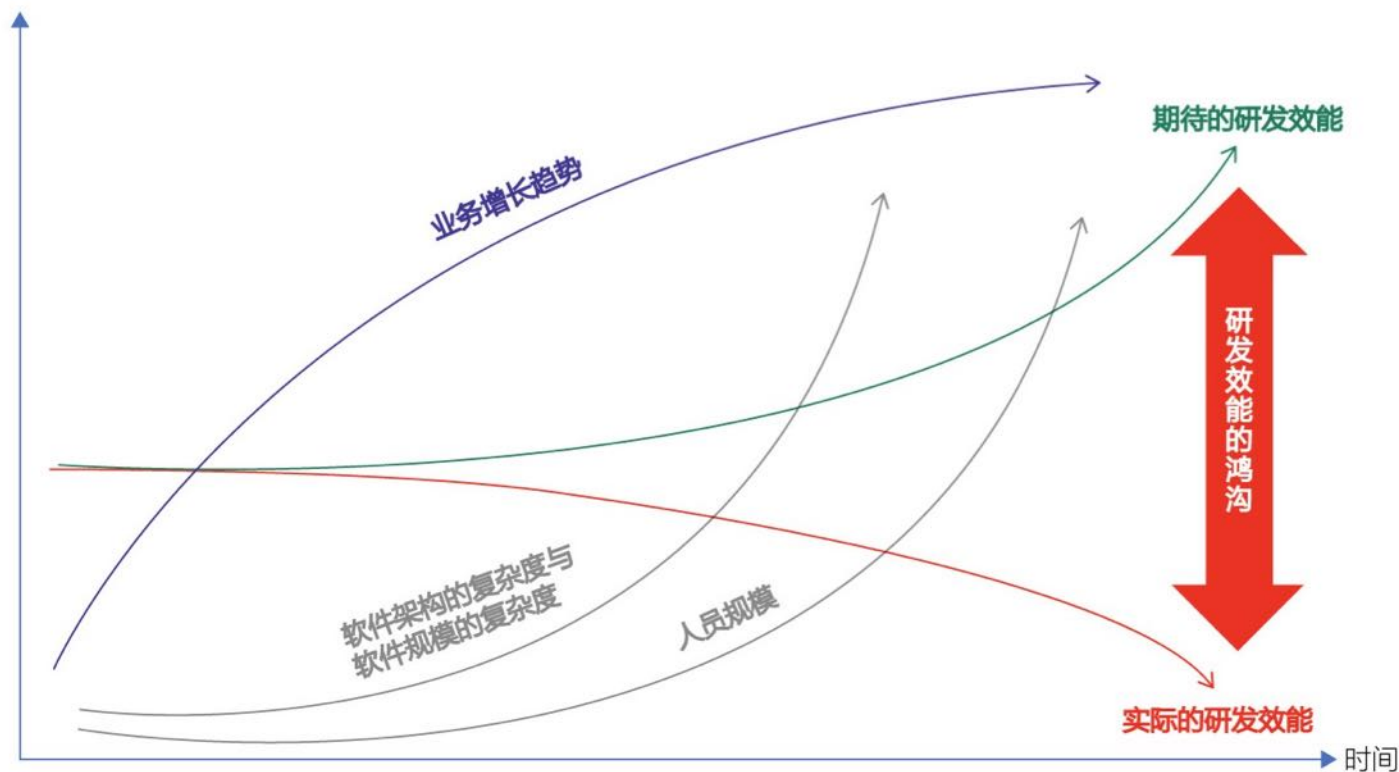
1. 数字化时代研发效能的挑战
2. 行业中研发效能的系统性方法
3. AI2.0时代研发效能提升策略
4. 研发效能黄金三角，升维思考与降维执行
5. 总结和展望

PART 01

数字化时代研发效能的挑战

▶ 数字化时代研发效能的挑战

- 数字化时代，软件研发过程本身也需要实现数字化
- 以往通过大量堆砌人力和资源，“快、糙、猛”开发和交付软件的方式已不可持续
- 降本增效的大背景下，“湖水岩石效应”愈发明显，需要更科学、更可持续的工作方式



内容参考：《软件研发效能提升实践》

根据“熵增定律”，在一个孤立系统里，如果没有外力做功，其总混乱度（熵）会不断增大

我们的软件越做越大、越做越复杂，研发效能的绝对值随着以下因素的增长会有变得越来越差的趋势：

- 软件架构本身的复杂度提升（微服务，服务网格等）
- 软件规模的不断增长（集群规模，数据规模等）
- 研发团队人员规模不断扩大引发沟通协作难度增加

PART 02

行业中研发效能的系统性方法

► 行业中提升研发效能系统性的方法



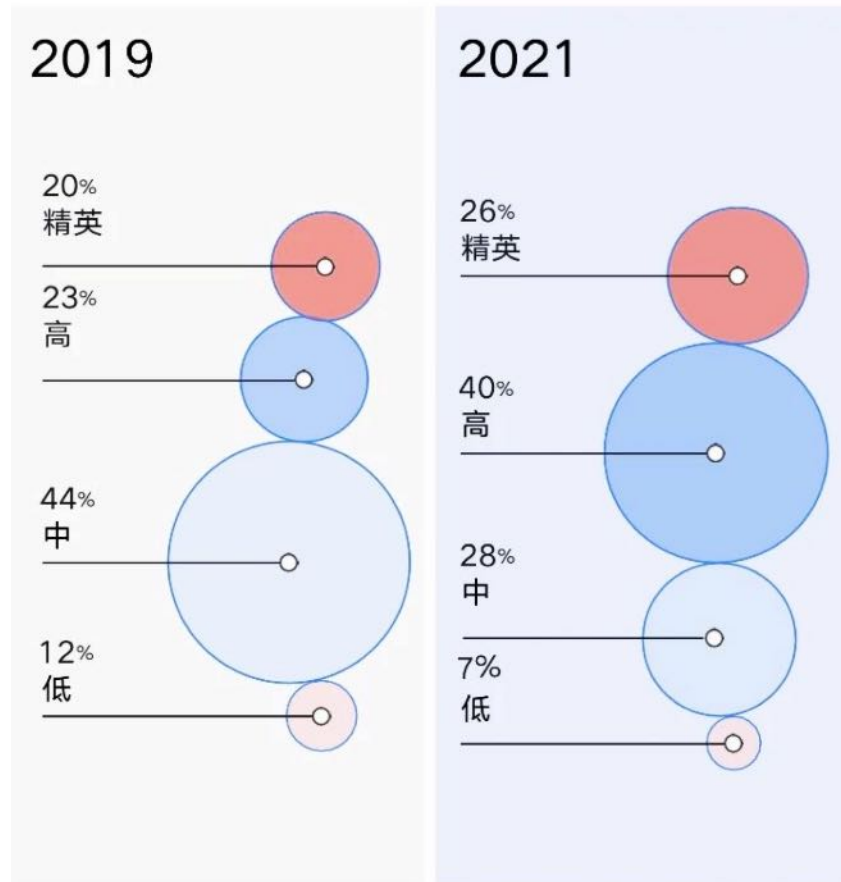
<https://pan.baidu.com/s/1pqncy8hTpmBPreTcxpWNA?pwd=0701> 提取码: 0701

AI驱动软件研发全面进入数字化时代

NiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

▶ DevOps全球调查报告 – 效能指标

软件交付效能指标	精英	高	中	低
🕒 部署频率 对于你所负责的主要应用或服务，你的组织多长时间将代码部署到生产环境中或发布给最终用户？	按需发布（每天多次部署）	介于每周一次到每月一次之间	介于每月一次到每6个月一次之间	小于6个月一次
🕒 变更的前置时间 对于你所负责的主要应用或服务，你的变更前置时间是多少（即，从提交代码到代码成功运行于生产环境中需要多长时间）？	不到1小时	1天到1周之间	1个月到6个月之间	6个月以上
🕒 服务恢复时长 对于你所负责的主要应用或服务，当发生影响用户的服务事件或缺陷（如计划外中断或服务受损）时通常需要多长时间来恢复服务？	不到1小时	不到1天	1天到1周之间	6个月以上
📉 变更失败率 对于你所负责的主要应用或服务，对生产环境的变更或向用户发布的变更有多大比例导致服务下降（例如，导致服务受损或服务中断）并随后需要补救（例如，需要热修复、回滚、向前修复、补丁）	0%-15%	16%-30%	16%-30%	16%-30%



▶ DevOps全球调查报告 – 效能指标

聚类	稳定性		运维效能	吞吐量		%受访者
	服务恢复时间	变更失败率	可靠性	前置时间	部署频率	
起步 (Starting)	一天到一周之间	31%-45%	有时符合预期	一星期到一个月之间	每周一次到每月一次之间	28%
流动 (Flowing)	不到一小时	0%-15%	通常符合预期	不到一天	按需 (每天多次部署)	17%
减速 (Slowing)	不到一天	0%-15%	通常符合预期	一星期到一个月之间	每周一次到每月一次之间	34%
衰退 (Retiring)	一个月至六个月之间	46%-60%	通常符合预期	一个月至六个月之间	每月一次至每六个月一次	21%

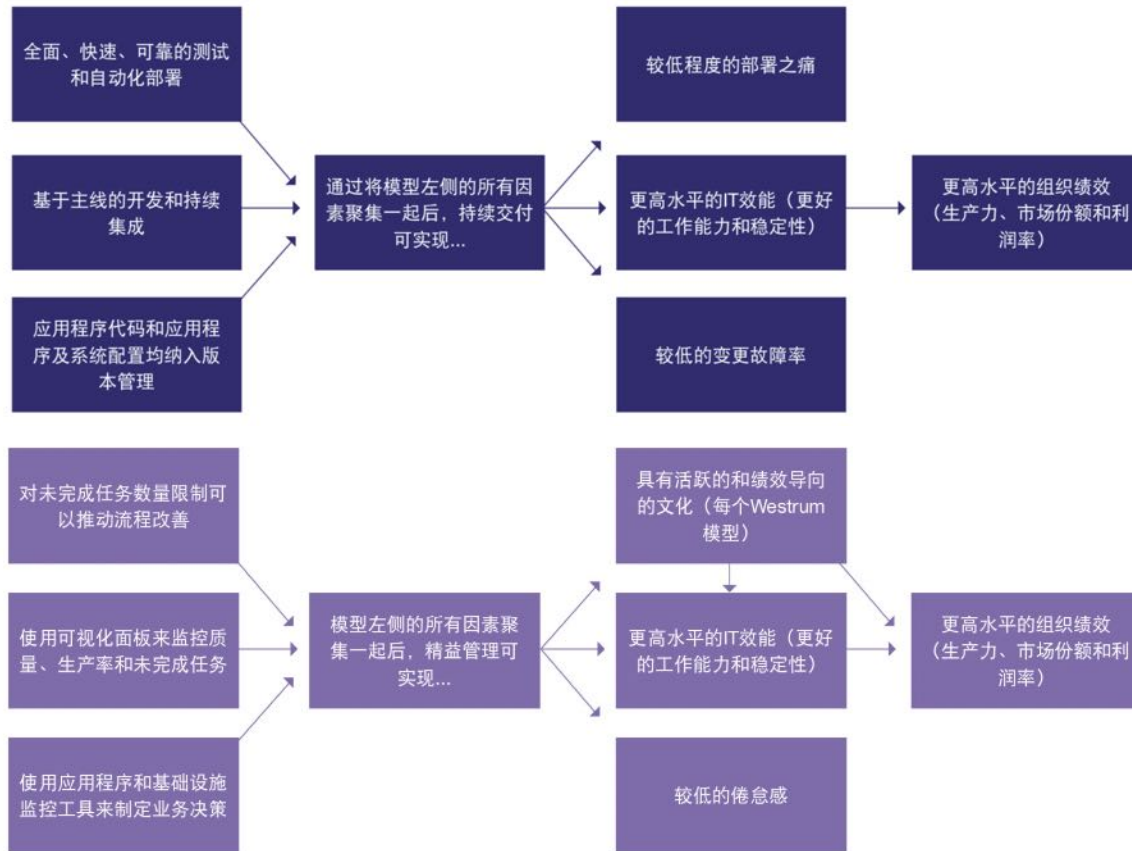
流动聚类展示了高水平的软件交付和运维效能，它们在实践和技术能力方面与其他聚类的区别：

- 松耦合架构：团队可以在不依赖其他团队的情况下对系统设计进行大规模变更
- 版本控制：如何管理对应用程序代码、系统配置、应用程序配置等的变更
- 持续集成 (CI)：将分支集成到主干中的频率
- 持续交付 (CD)：专注于将变更安全、可持续和高效地投入生产的能力
- 提供灵活性：公司对员工的工作安排的灵活性（不应消极看待弹性工作安排，而是根据产出对员工进行评估）

▶ DevOps全球调查报告 – 结构方程模型

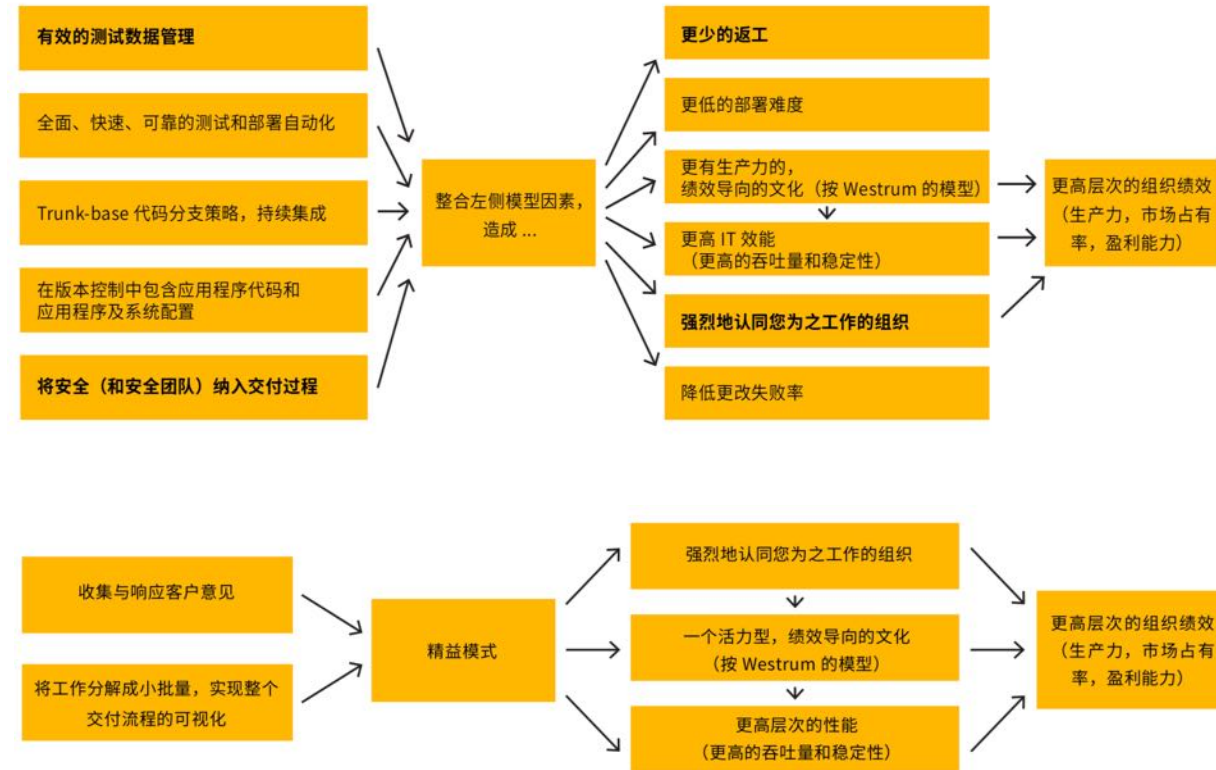
2015

- 创建结构方程模型
- 研究持续交付和精益管理实践如何影响IT效能和组织绩效



2016

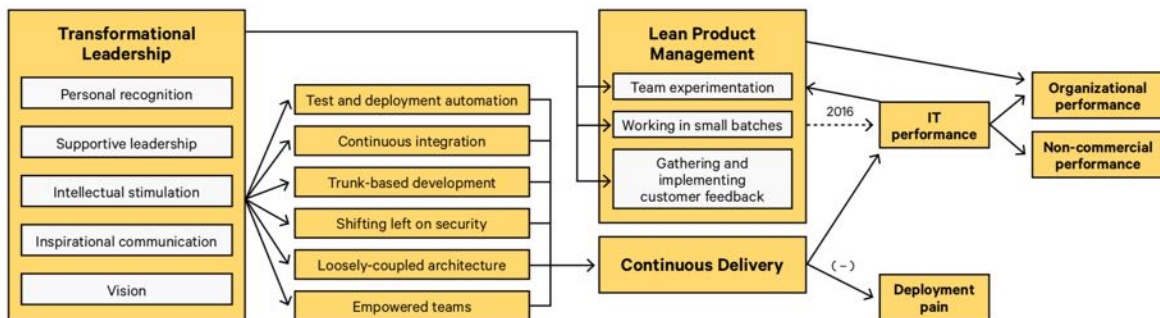
- 增强结构方程模型
- 将测试数据管理以及安全纳入模型研究



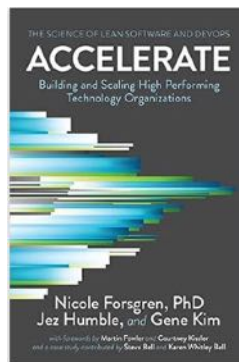
▶ DevOps全球调查报告 – 结构方程模型

2017

- 持续增强结构方程模型
- 将变革领导力以及松耦合的架构和团队纳入模型研究



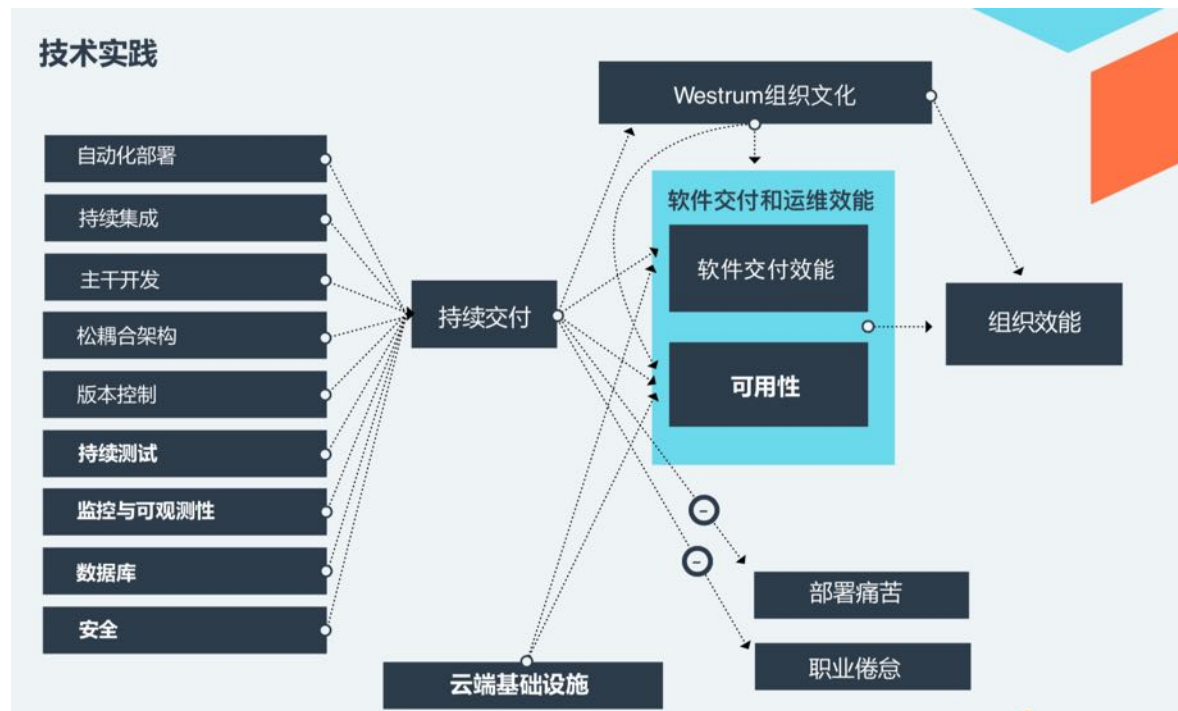
Dimensions of transformational leadership



《加速：企业数字化转型的24项核心能力》

2018

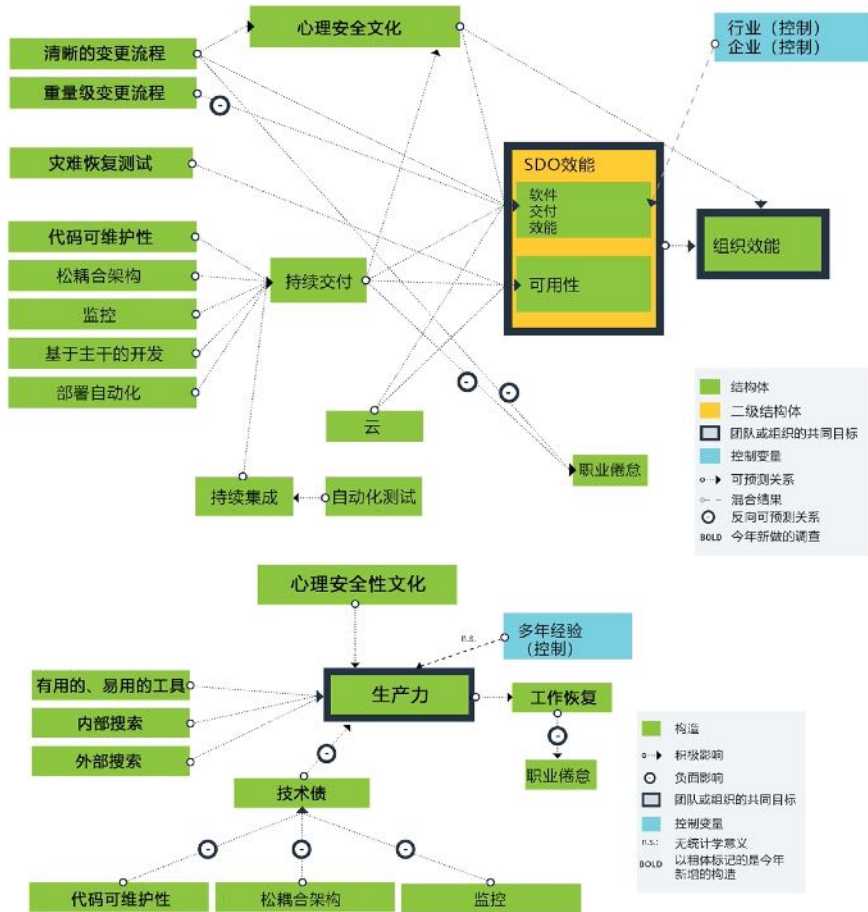
- 持续增强结构方程模型
- 将持续测试、监控与可观测性、云基础设施纳入模型研究



▶ DevOps全球调查报告 – 结构方程模型

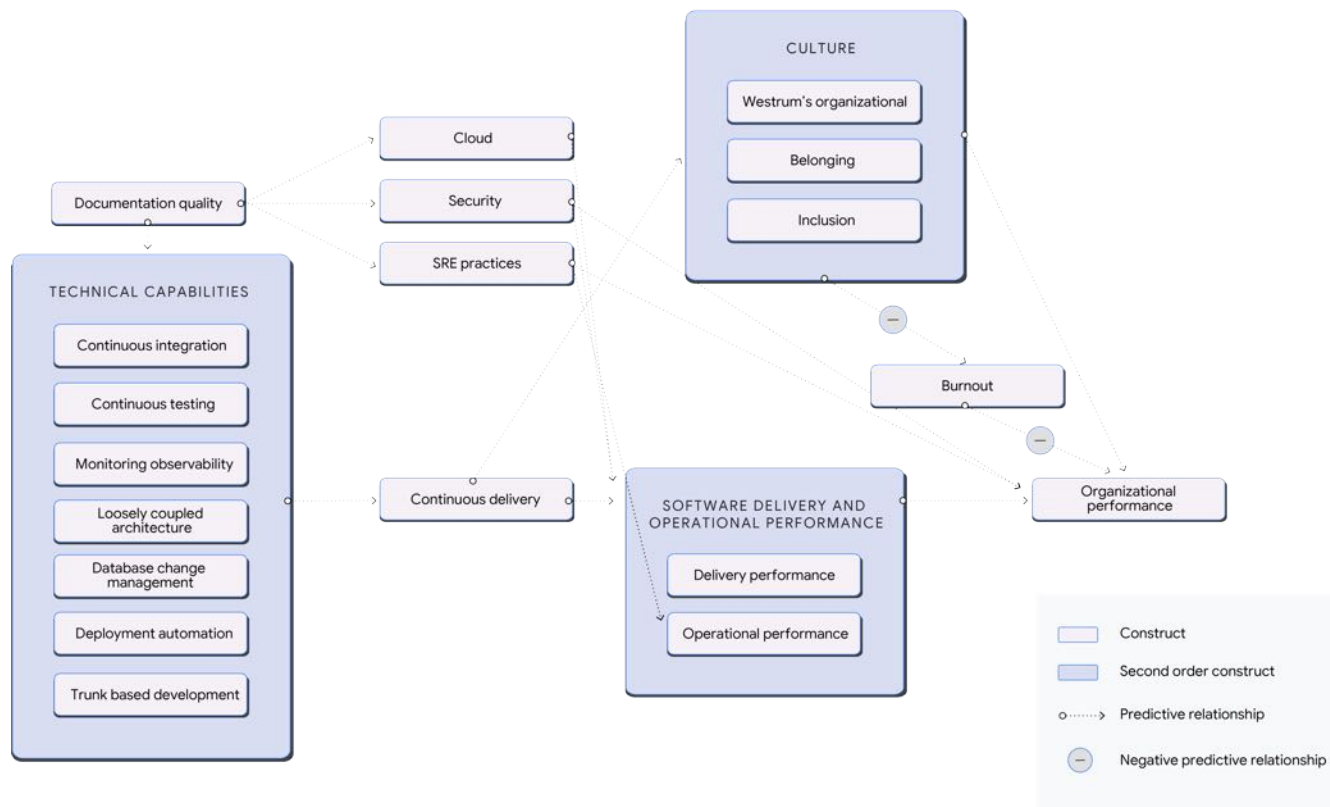
2019

- 持续增强结构方程模型
- 分解为软件研发和运维效能、生产力两个模型进行研究



2021

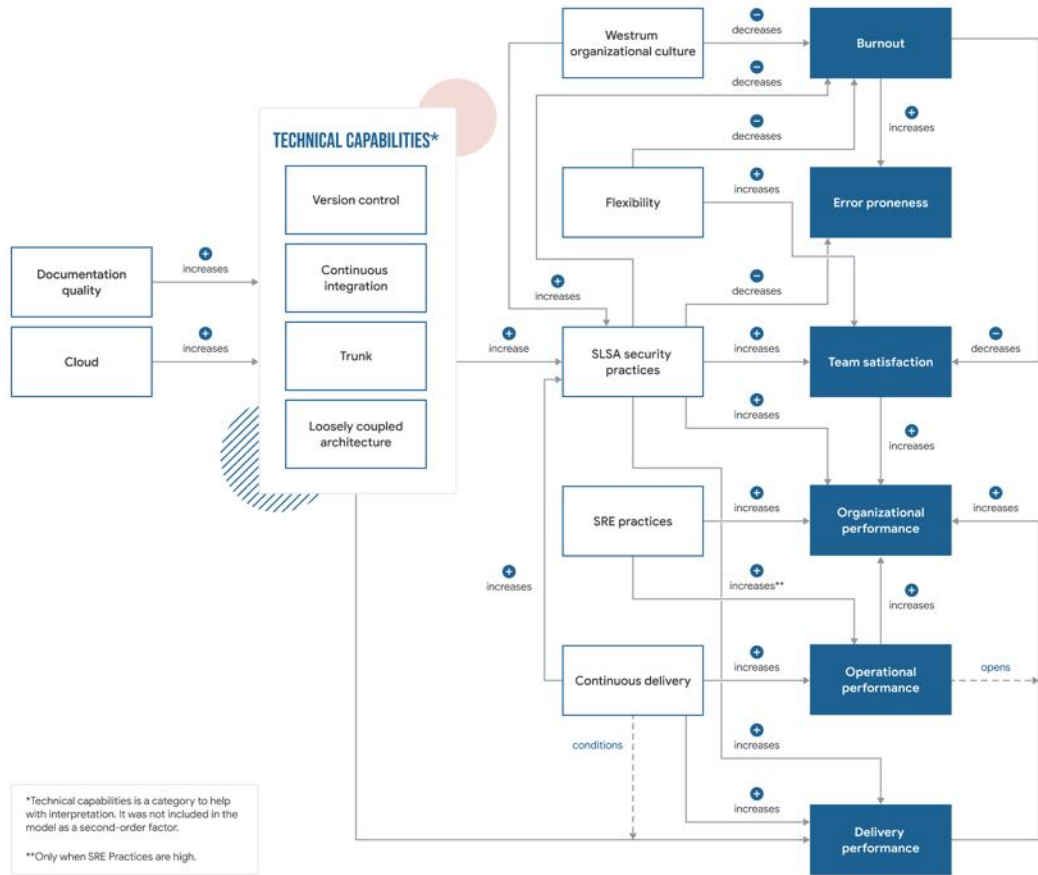
- 持续增强结构化方程模型
- 将文档质量、SRE、归属和包容的文化等纳入模型研究



▶ DevOps全球调查报告 – 结构方程模型

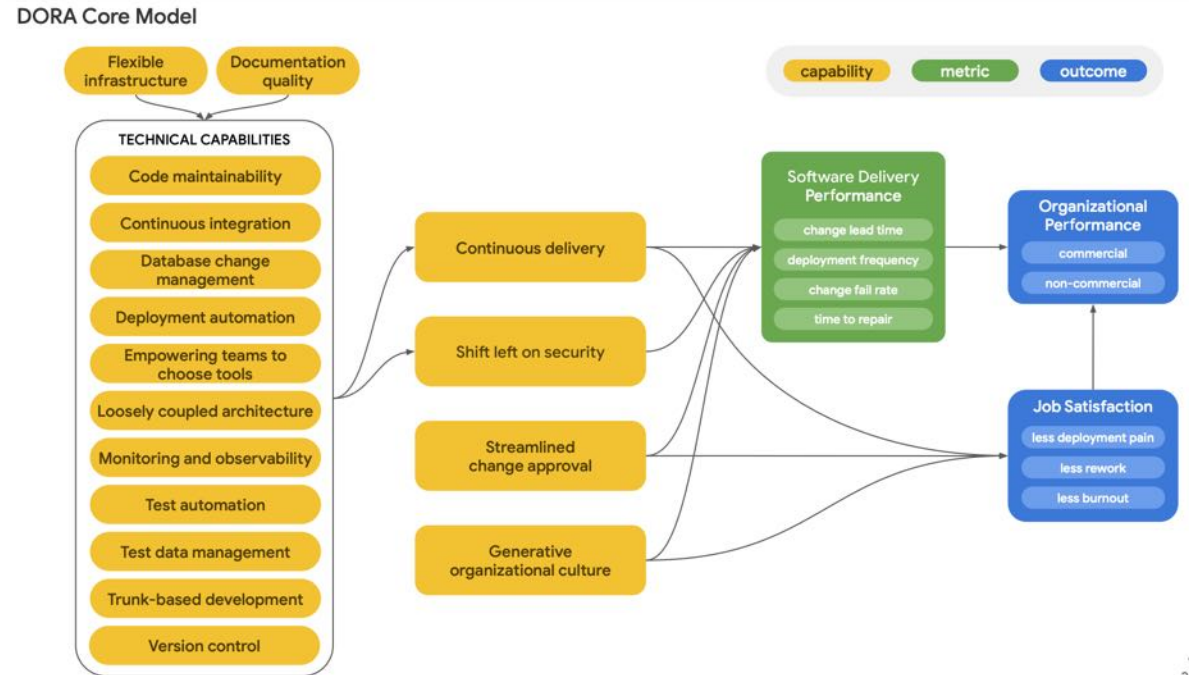
2022

- 持续增强结构方程模型
- 将软件供应链的安全、团队上下文环境等纳入模型研究



2023

- 提炼DORA Core模型
- 基于历年的研究结果，突出最稳定、最受支持的实体及预测路径



version 0.2
2023-06-26

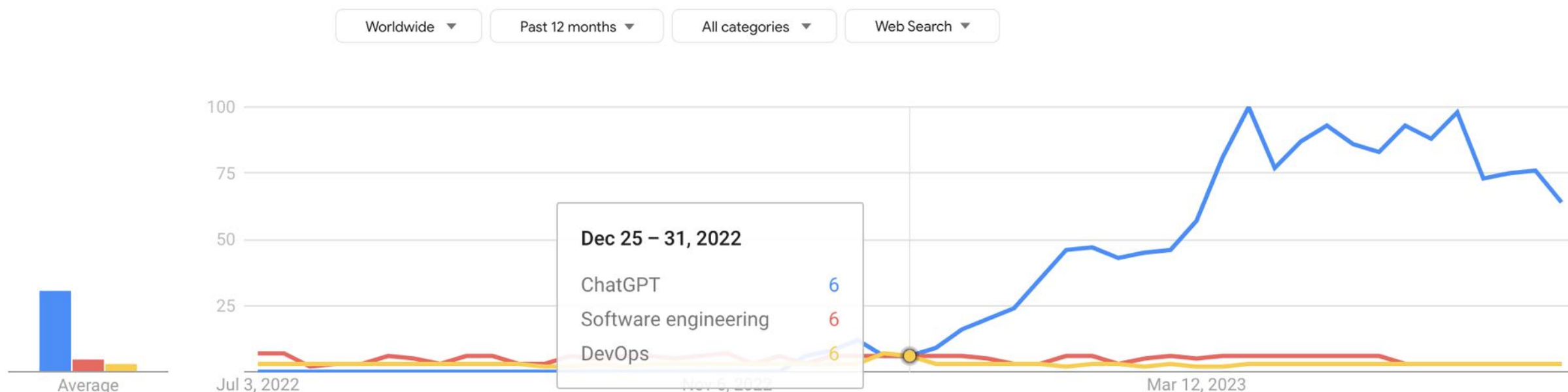
PART 03

AI2.0时代研发效能提升策略

▶ AI2.0时代研发效能提升策略

OpenAI 报告：ChatGPT将会影响全行业中80%工作岗位

- GPT会是像蒸汽机或印刷机一样的通用技术，影响无所不在
- 在各种专业测试和学术基准上，GPT-4的表现与人类水平相当
- 横跨整合行业，学历越高、收入越高，受到冲击的可能性越大
- 编程领域约 62.3% 的活动，可通过 GPT 或 GPT 驱动的系统，将人工执行或完成特定活动所需的时间，降低至少50%



▶ 立足当下、要面向未来，实现可持续发展

70%

持续改进现有流程

第一地平线 HORIZON 1

基于现有软件工程的流程和专业分工进行增强和改进，利用大模型技术进行需求及意图理解、文档生成、代码理解、代码生成及重构、缺陷检测及修复、测试用例及脚本生成、故障定位及分析、客户支持及知识库管理...



20%

逐步改变现有模式

第二地平线 HORIZON 2

10%

彻底变革到新范式

第三地平线 HORIZON 3

立足当下：积极拥抱新技术/工具，持续优化

面向未来：尝试新模式、探索新范式

PART 04

研发效能黄金三角，升维思考与降维执行

研发效能的升维思考



效能度量

效率

吞吐量
交付周期

质量

缺陷密度
部署成功率

体验

满意度
NPS

...

效能实践

【管理维度】

关注流程、人员和文化等方面，以帮助组织实现更高效、更高质量、协作和创新的研发过程

- 结构化需求分析
- 敏捷&精益协作
- 可视化管理
- 价值流管理

- 测试 / 安全左移
- 测试门禁 / 质量红线
- 应用/数据库变更流程

- 标准化/简化研发流程
- 高效能团队模式
- 开放、协作和支撑创新的团队文化

【工程维度】

强调使用自动化、一致性的方式来执行研发活动，提高效率、降低风险、提升质量和可靠性

- 代码分支模型
- 自动化构建/测试
- 自动化环境治理
- 自动化部署发布
- LLM提示工程

- 代码评审 / 结对编程
- 静态测试 / 动态测试
- 持续集成 / 持续交付
- 低风险部署和发布

- 自动化研发活动，减少手工、事务性、重复性工作，聚焦到业务逻辑和创造性活动

【技术维度】

侧重于应用和基础设施的架构设计与实现，结合创新技术，支持高效交付、高可用、高可扩展性

- 模块化 / 松耦合
- 云原生 / AI原生
- 低代码 / 无代码
- 智能化软件开发

- 关注点分离/可测试性
- 监控和可观测性
- 智能化软件测试
- AIOps/智能化运维

- 降低复杂性，降低认知负担 [声明式API]
- 降低耦合/依赖，避免繁琐沟通协同和集成

效能平台

【DevOps工具链】

支撑管理实践、工程实践、技术实践的落地，并将能力集成到对研发人员友好的效能平台

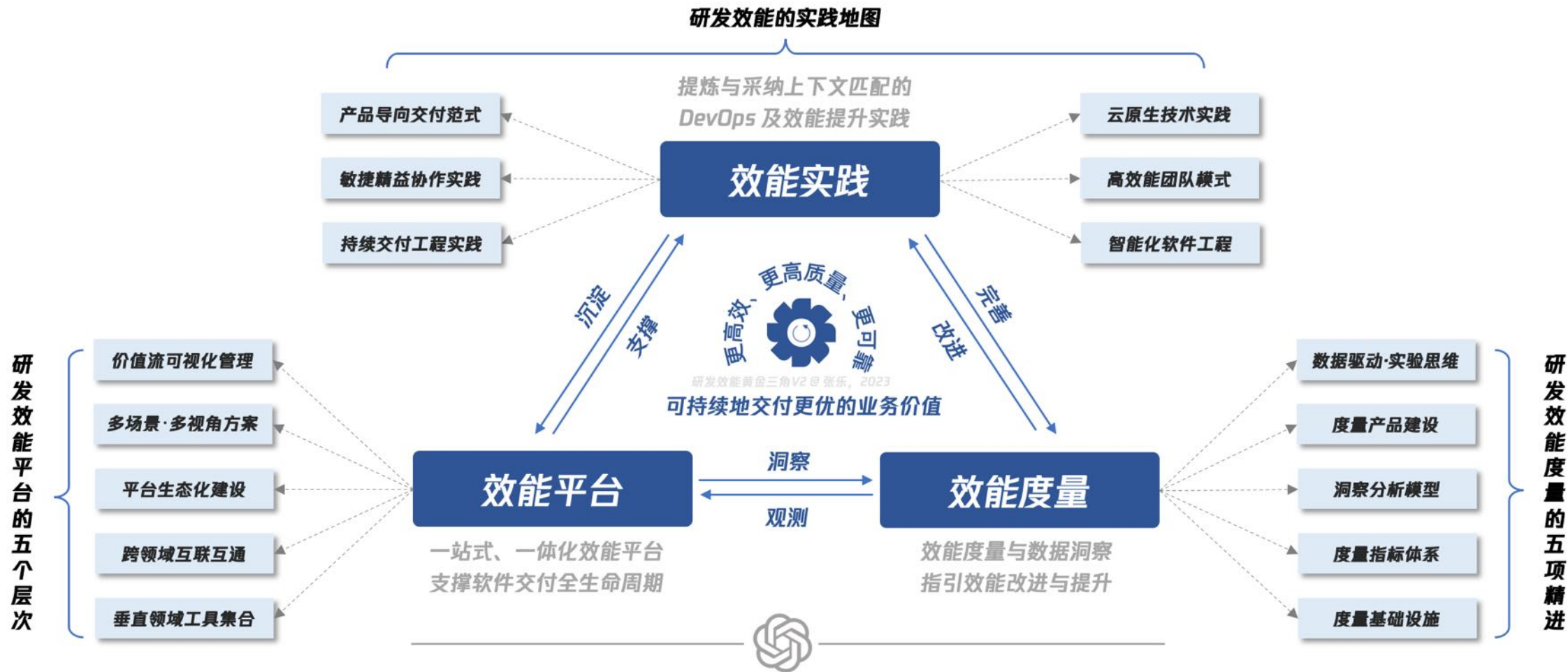
- 敏捷协作工具
- 代码托管工具
- 自动化测试工具
- 自动化部署工具
- 基于LLM的效率工具

- CI / CD流水线
- 代码 / 安全扫描工具
- 监控 / 日志 / 调用链
- 基于LLM的质量工具

- 一站式研发效能平台
- 贴合场景的解决方案
- 基于LLM的智能助手

研发效能的升维思考 @ 张乐, 2023

研发效能的黄金三角



研发效能黄金三角V2.0 @张乐, 2023

▶ 研发效能的降维执行 – 举例：AI辅助编程

理解

代码理解 / 代码解释
相似代码搜索 / 克隆检测
代码缺陷 / 代码漏洞检测
代码评审 / 代码问题定位



生成

生成代码 / 生成注释
生成测试 / 生成断言
代码补全 / 代码续写
代码重构 / 缺陷修复

< 使用自然语言解释代码 >

```
def largest_integer_in_list(l):  
    """  
    Returns the largest integer in a list  
    """  
    if len(l) == 0:  
        return None  
    else:  
        max_int = l[0]  
        for i in l:  
            if i > max_int:  
                max_int = i  
        return max_int
```

< 基于注释生成代码 >

```
21  
22 # Get the Goodreads key from the environment variable.
```

< 基于被测代码生成测试代码 >

```
1  
2 def common_prefix(a, b):  
3     """Return the common prefix of two lists."""  
4     if len(a) < len(b):  
5         return common_prefix(b,a)  
6     for i in range(len(a)):  
7         if a[i] != b[i]:  
8             return a[:i]  
9     return a  
10  
11  
12
```



欢迎使用Comate智能编码伙伴

通过自然语言对话，辅助编码过程



记住快捷键，快速唤起Comate

可以试着问我

⌘ 为选中的代码生成单测

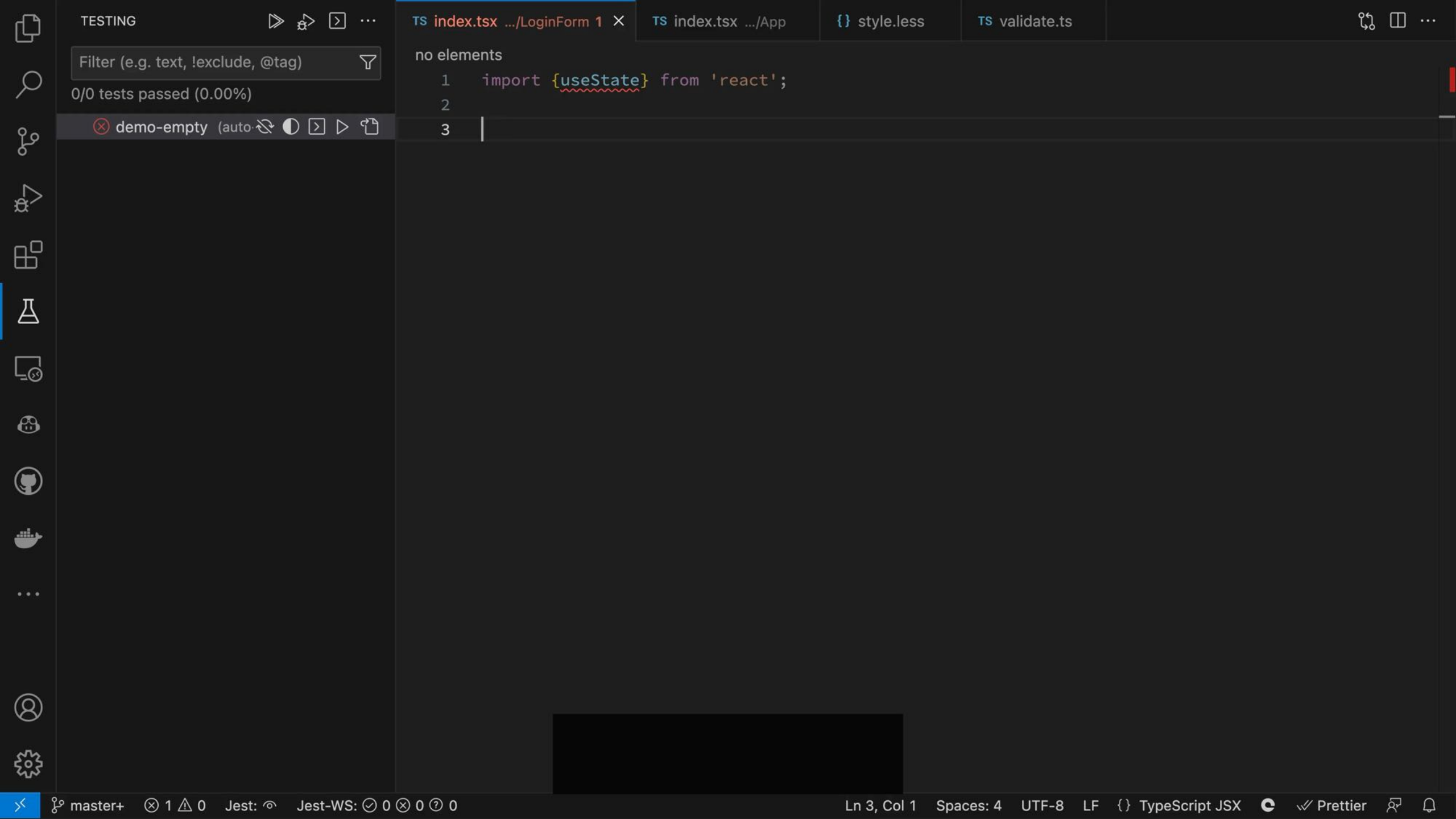
⇧ 为选中的代码生成注释

了解更多智能编码能力，请查看Comate用户手册

src > main > java > baidu > comate > com > Demo.java

```
1 package comate.com;
2
3 import java.security.SecureRandom;
4 import java.util.UUID;
5 public class Demo {
6
7     final static SecureRandom random = new SecureRandom();
8
9     函数注释 | 行间注释 | 生成单测 | 代码解释
10    public static void main(String[] args) {
11        String activationCode = generateKey();
12        System.out.println("生成的激活码: " + activationCode);
13    }
14
15    函数注释 | 行间注释 | 生成单测 | 代码解释
16    public static String generateKey() {
17        String CHARACTERS = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
18        StringBuilder code = new StringBuilder(16);
19        for (int i = 0; i < 16; i++) {
20            int randomIndex = random.nextInt(CHARACTERS.length());
21            char randomChar = CHARACTERS.charAt(randomIndex);
22            code.append(randomChar);
23        }
24        return code.toString();
25    }
26 }
```

通过「代码解释能力」解释已有的方法



▶ 研发效能的降维执行 – 举例：AI辅助编程（案例）

01

代码解释

可给出整个方法、函数的功能和完整逻辑说明，也支持选中任意代码给出解释

02

技术问答

在对话框中输入研发中遇到的问题，第一时间获得解答

03

实时续写

通过分析上下文逻辑关系，智能生成方法、函数、判断、循环体等完整的代码块

04

注释生成代码

直接在注释中用自然语言描述所需功能，自动生成完整函数

05

代码生成注释

分析方法、函数、类，生成符合规范的文档注释，或识别函数的逻辑逐行增加注释

06

生成单元测试

支持对任意方法、函数一键生成单元测试，也支持对多文件进行批量生成

07

代码优化

支持对长函数等坏味道代码进行优化

08

代码修复

识别代码中的潜在错误，并自动进行修复

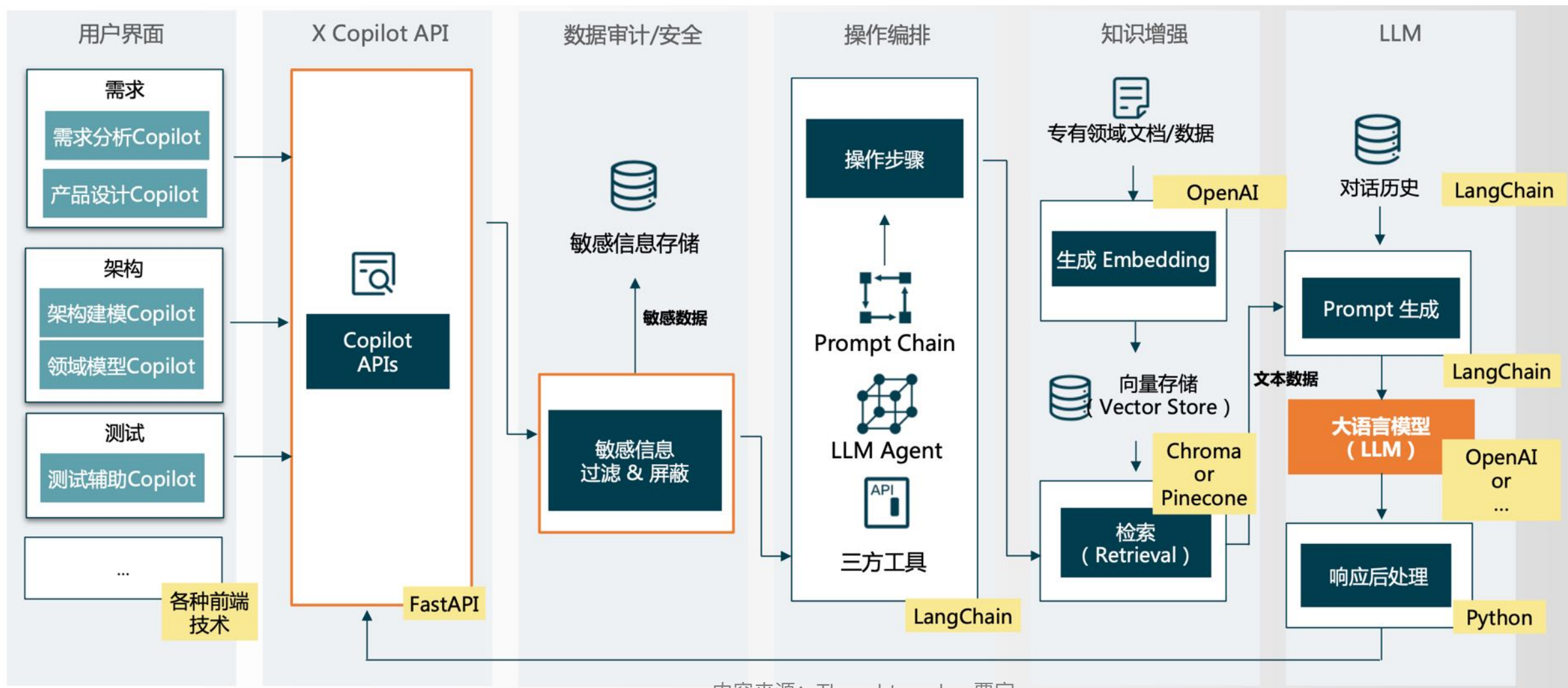
<http://comate.baidu.com/>

▶ 研发效能的降维执行 – 举例：提示工程



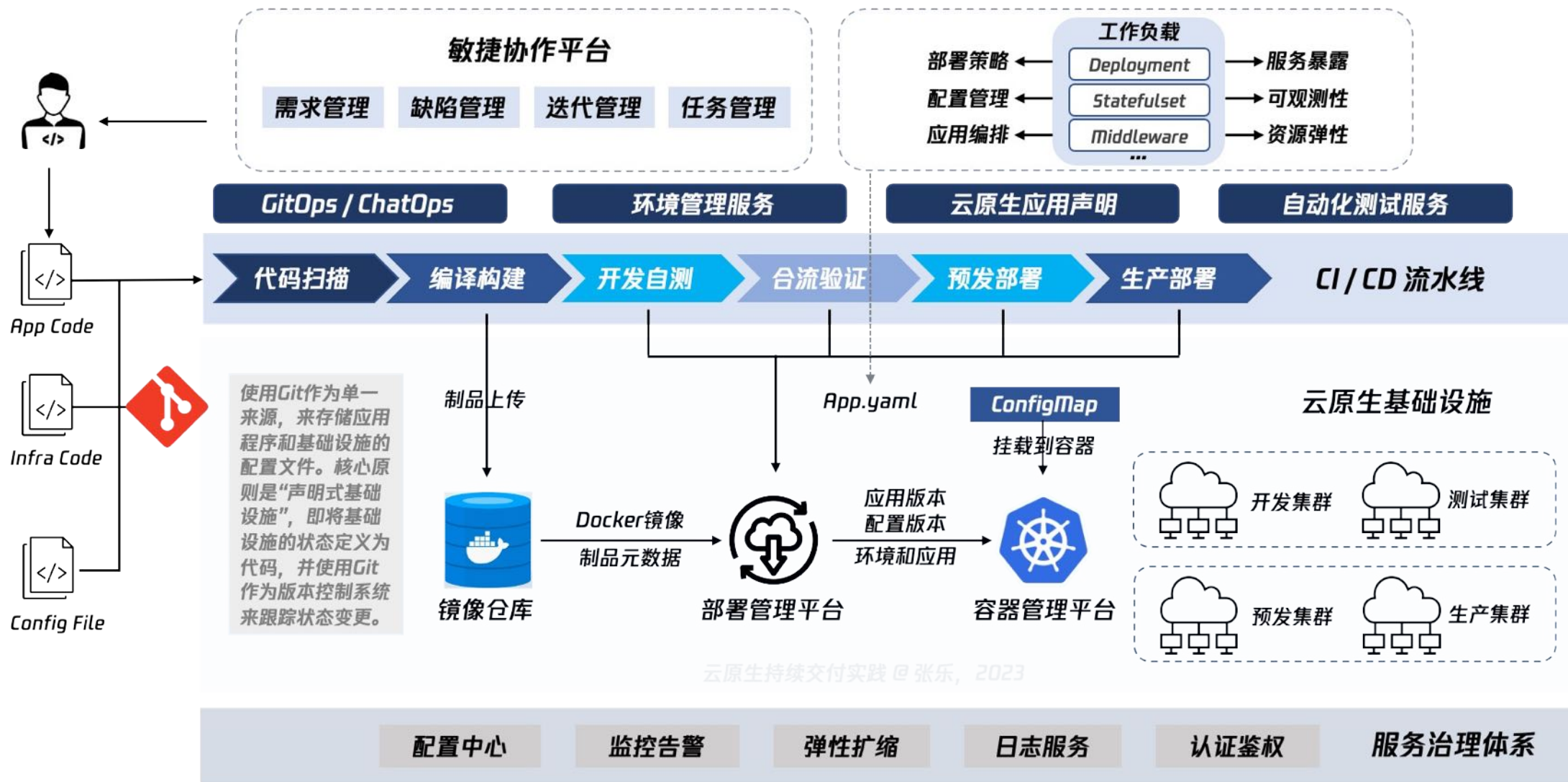
<https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/>

▶ 研发效能的降维执行 – 举例：企业AI应用参考架构

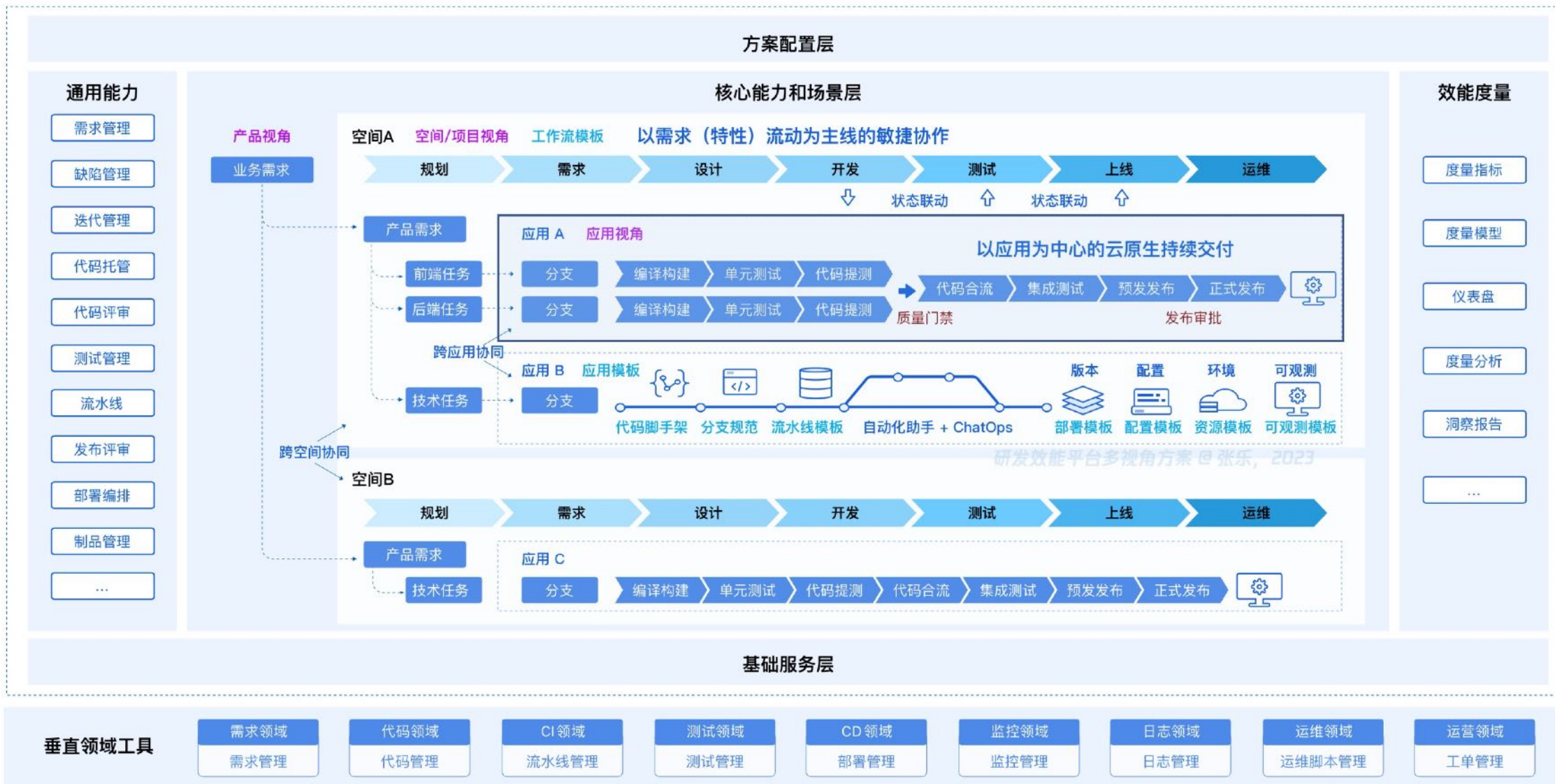


内容来源：Thought works-覃宇

研发效能的降维执行 - 举例：云原生持续交付



研发效能的降维执行 - 举例：效能平台



研发效能的降维执行 – 举例：效能度量

度量的受众

- 企业管理者
- 基层管理者
- 一线工程师

度量的对象

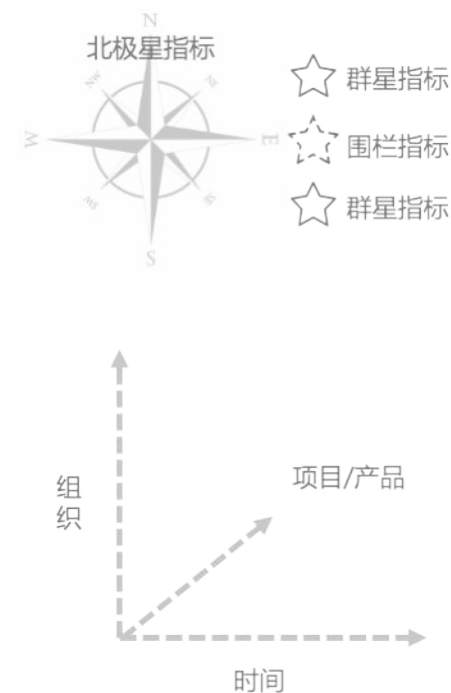
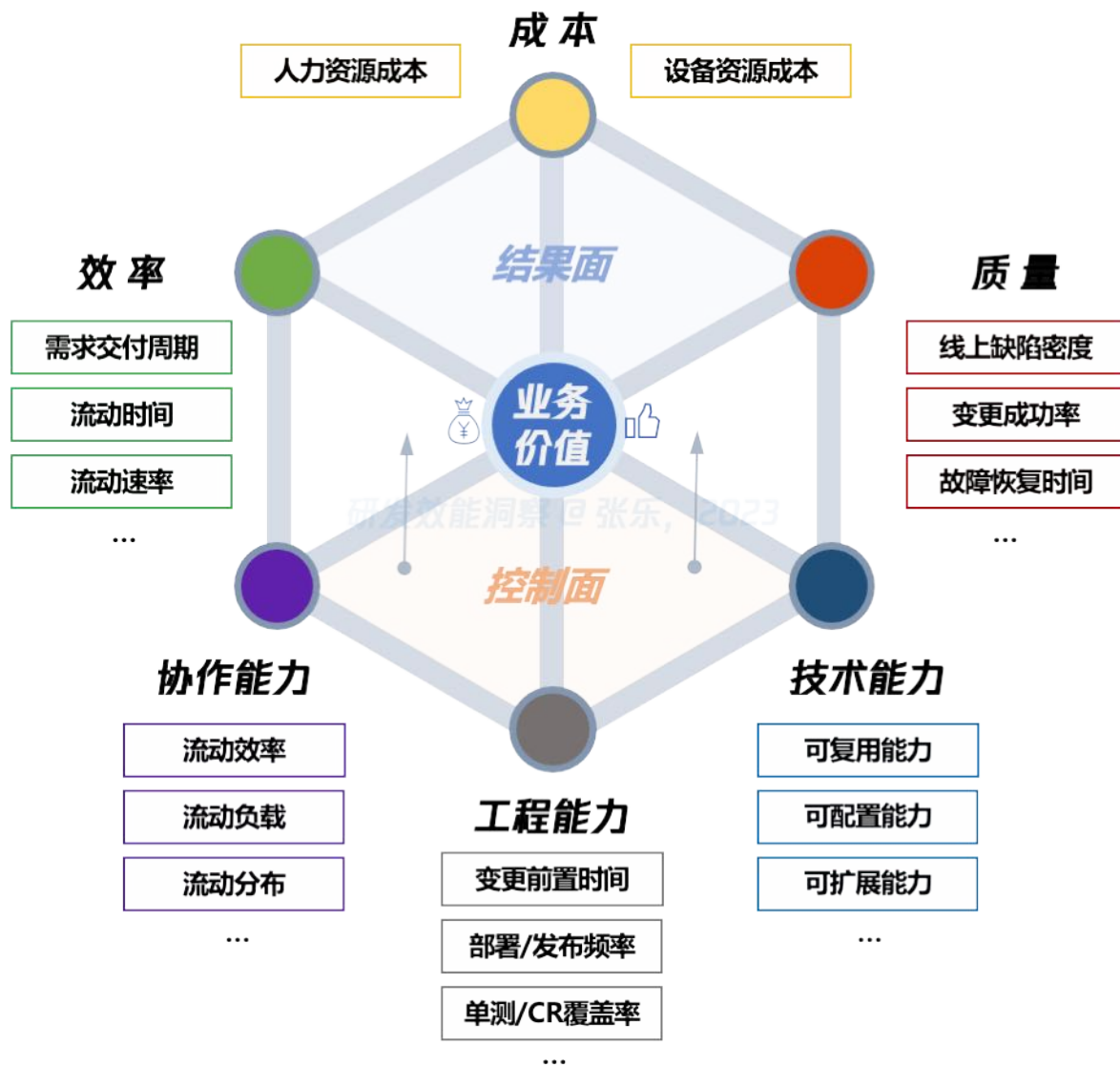
- 业务结果
- 研发产出
- 研发过程
- 组织/团队

度量的维度

- 多 (数量维度)
- 快 (效率维度)
- 好 (质量维度)
- 省 (成本维度)
- 强 (能力维度)

度量的目的

- 了解：认知的第一步
- 评价：对比产生评价
- 改进：对差距做改进
- 控制：确保在区间内
- 预测：面向未来预判



度量指标体系设计要点

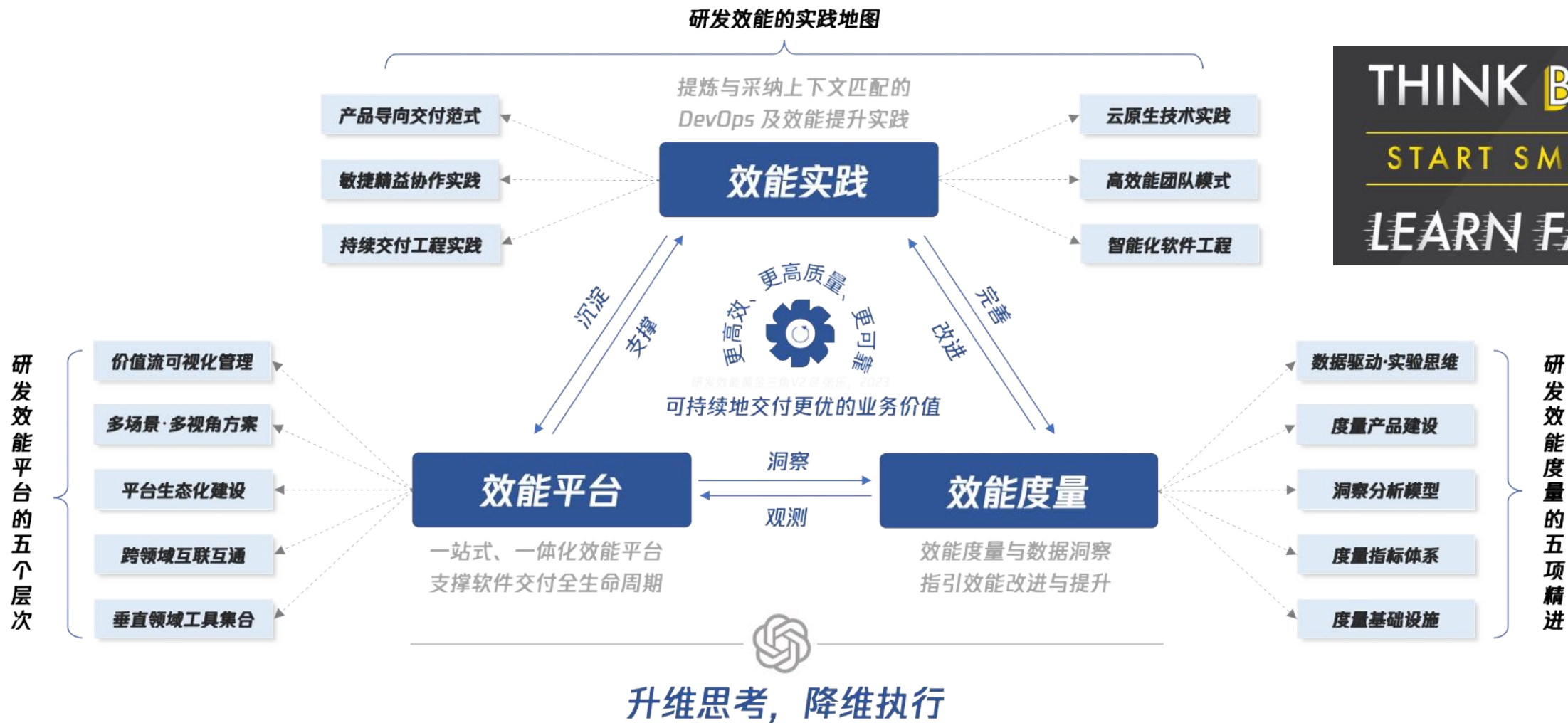
- 以结果为导向，不依赖活动导向代理指标
- 将价值流中的研发产出，关联到业务结果
- 为技术和业务提供一套通用语言和成功标准
- 结果性指标评估效果，过程性指标定位瓶颈
- 先导性指标事前干预，滞后性指标事后复盘

PART 05

总结和展望



▶ 数字化时代研发效能提升策略与系统性方法



研发效能黄金三角 V2.0 @张乐, 2023

感谢聆听

