

探索软件开发新工序 LLM 赋能研发效能提升

Thoughtworks / 谢保龙 & 黄峰达 (Phodal)



谢保龙

- Thoughtworks 专家级咨询师, Thoughtworks 中国区的工程效能/DevOps解决方案的负责人, CAC(China Agile Coach) 专业敏捷教练, Exin DevOps Master 认证讲师。
- 拥有 15+ 年软件工程从业经验, 目前主要聚焦于国内金融客户的敏捷、DevOps 转型工作以及研发效能相关咨询工作。对敏捷项目管理、研发效能度量与提升、大型企业组织转型等有丰富经验。

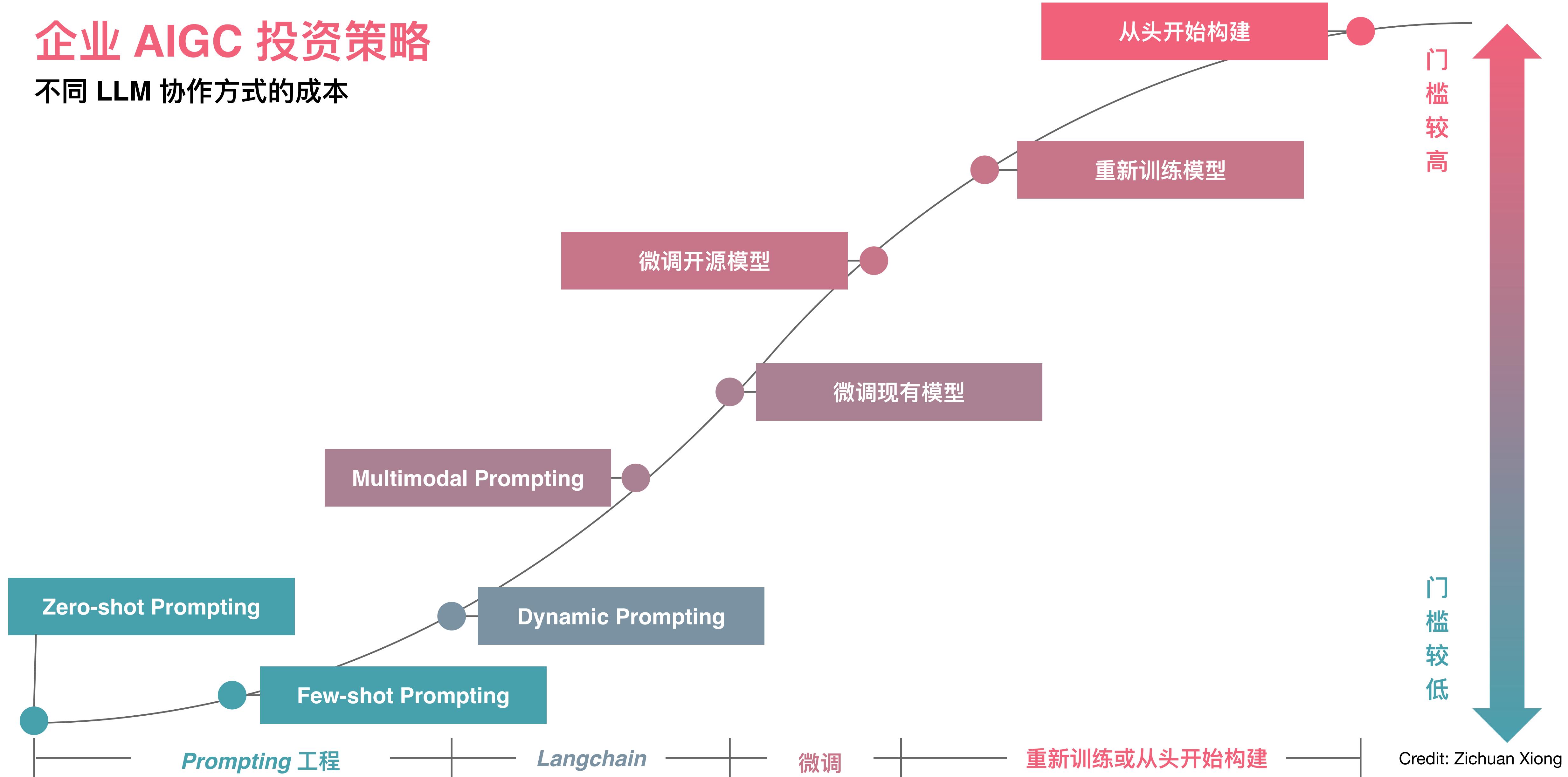


黄峰达 (Phodal)

- 开坑, 开到填不完的知名开源挖坑选手: <https://github.com/phodal/>
 - AI 相关开源项目: ClickPrompt、ChatFlow、Unit Mesh、AutoDev
 - AI 相关文章: 《理解 Prompt》、《Prompt 编写模式》、《Prompt 即代码》、《AI 研发提效的正确姿势: 开源 LLM + LoRA》、《上下文工程: 基于 Github Copilot 的实时能力分析与思考》、《大语言模型友好的 API》等。
- 开源架构治理工具 ArchGuard 核心开发 & 架构师
- Thoughtworks 技术专家, 开源有 Java 重构分析工具 Coca、Mooa 微前端框架、Ledge 平台等

企业 AIGC 投资策略

不同 LLM 协作方式的成本



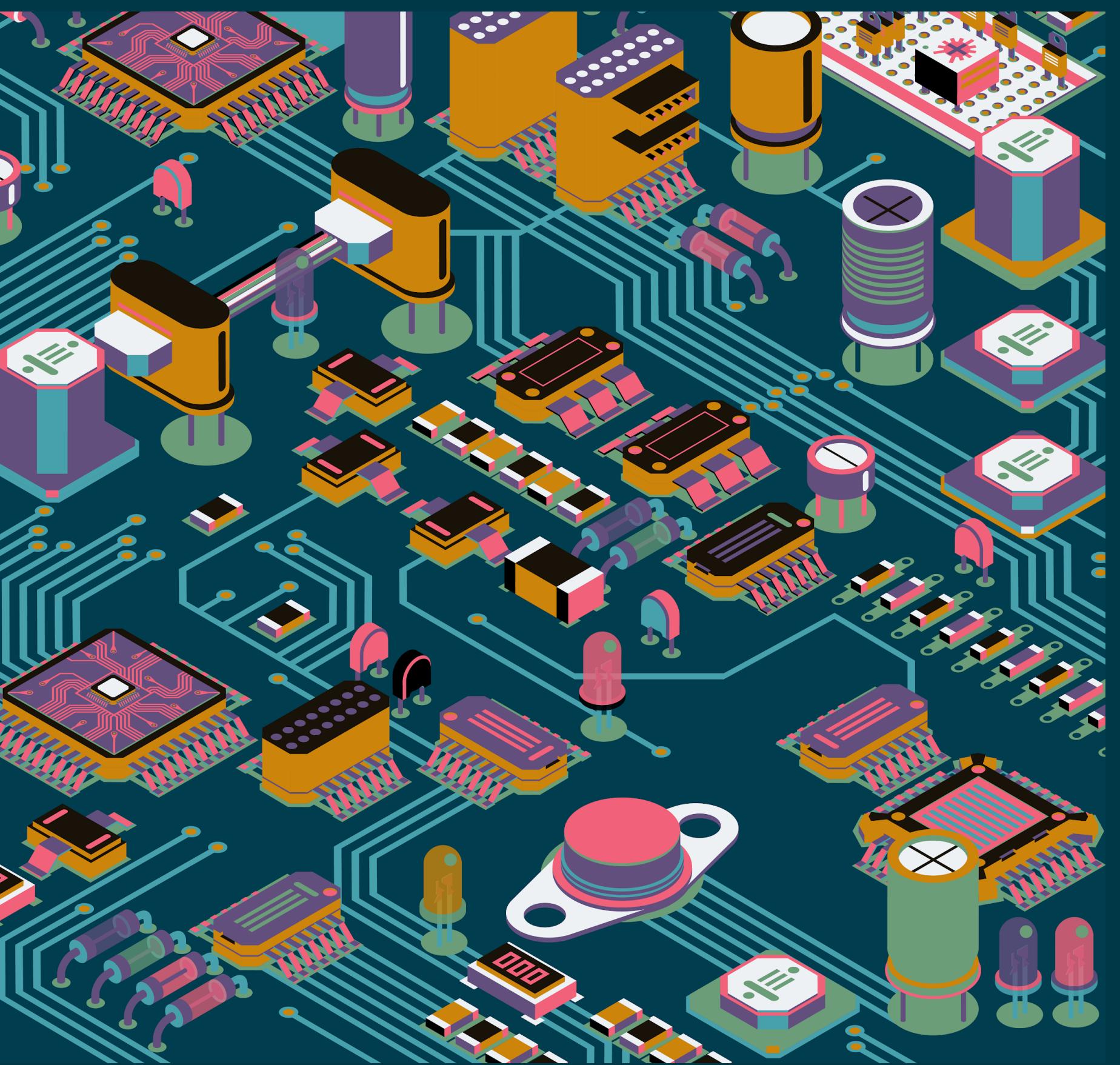
目录



修改自：<https://a16z.com/2023/01/19/who-owns-the-generative-ai-platform/>

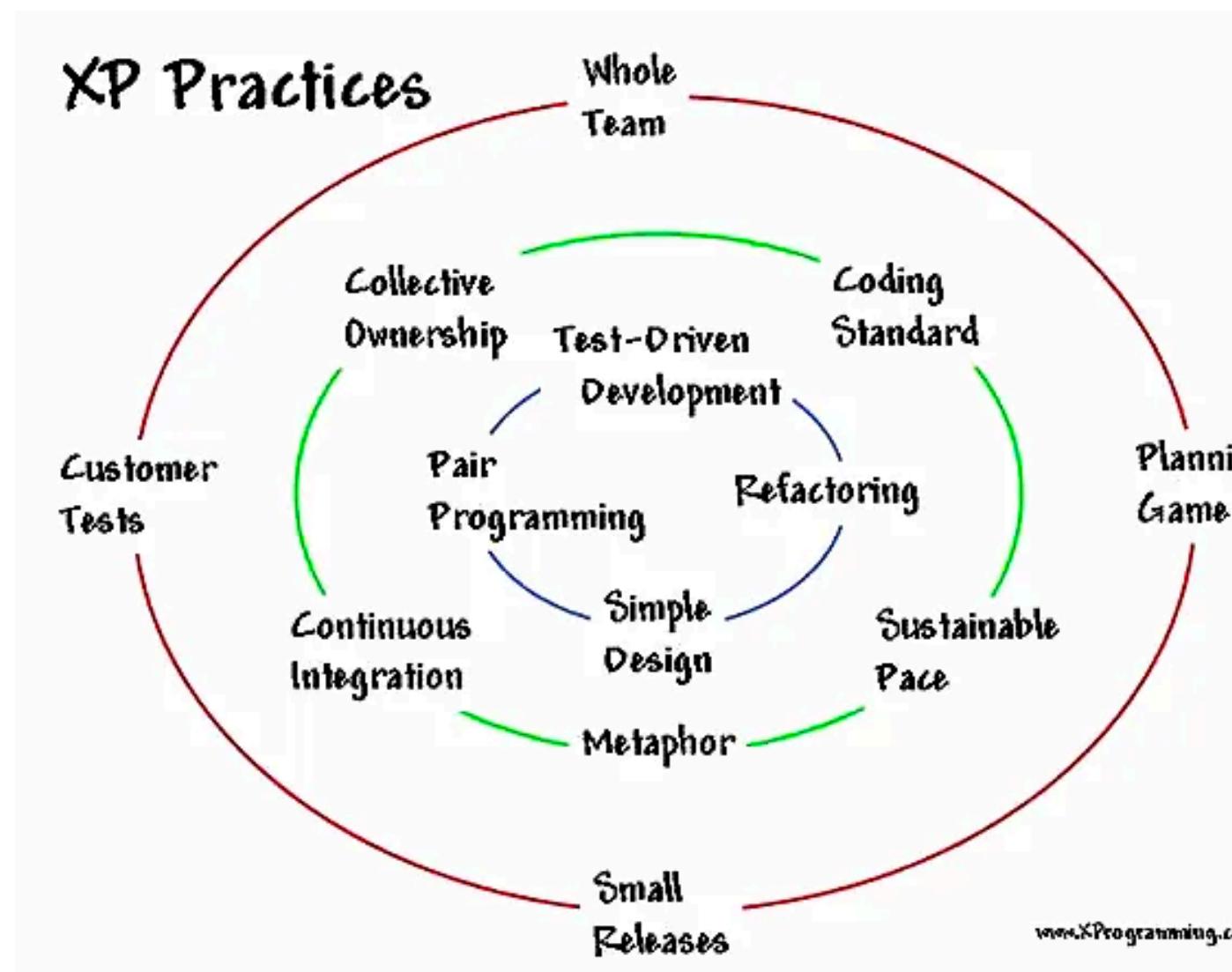
1. 研发效能提升遇到的挑战
 - 梳理研发流程寻找突破点
 - 新机遇带来的新挑战
2. LLM 在研发效能带来的新机遇
 - Discover 阶段：流程工程与“语言”工程
 - Define 阶段：数据工程与 Prompt 工程
 - Delivery 阶段：上下文工程与平台工程
3. LLM + 工序：探索与构建路径
 - Discover 阶段：流程工程与“语言”工程
 - Define 阶段：数据工程与 Prompt 工程
 - Delivery 阶段：上下文工程与平台工程
4. 总结与展望
 - LLM 重塑工具平台
 - LLM 对人才和组织结构的影响

研发效能提升遇到的挑战

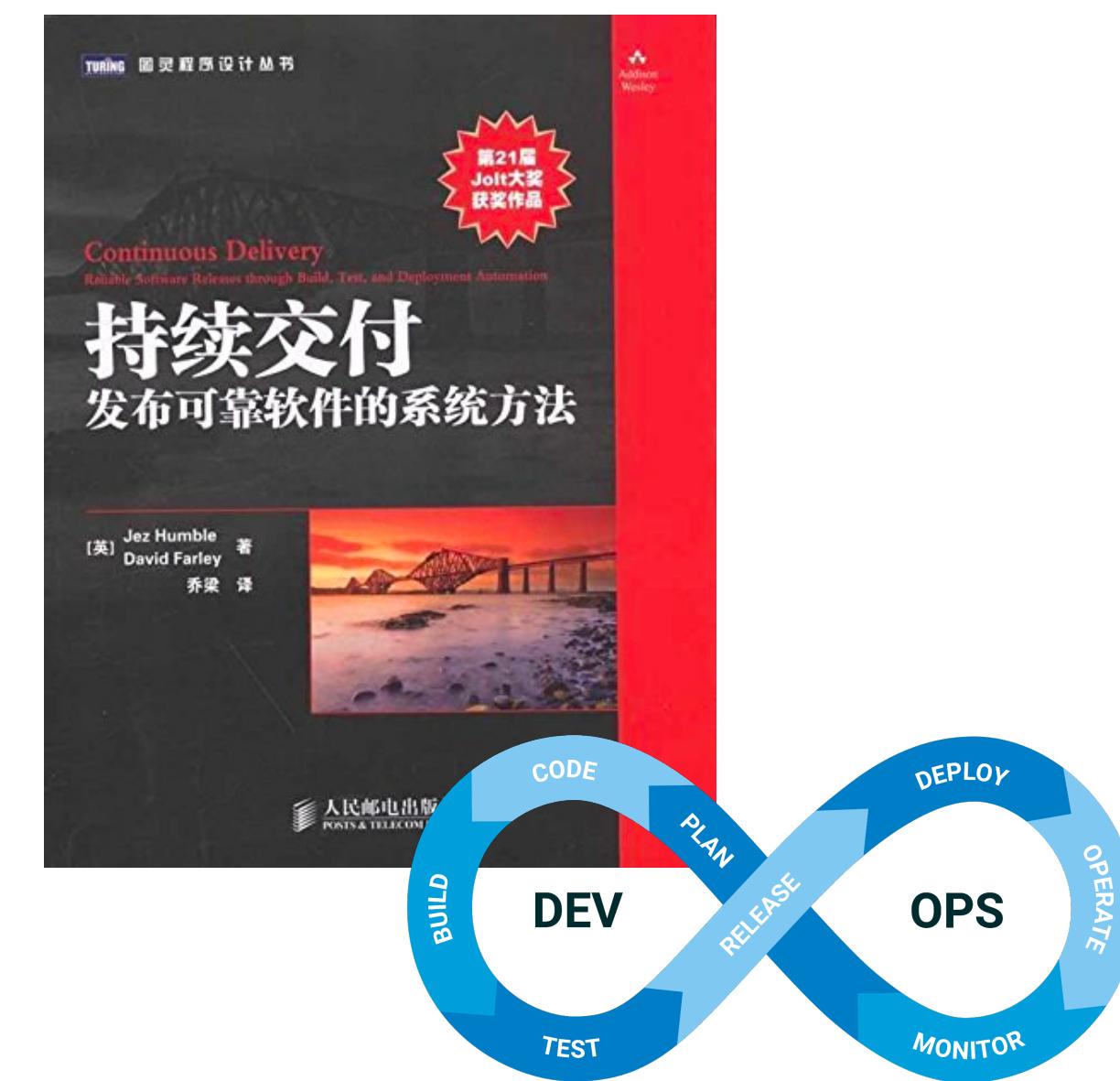
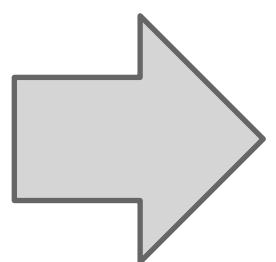


从极限编程到研发效能

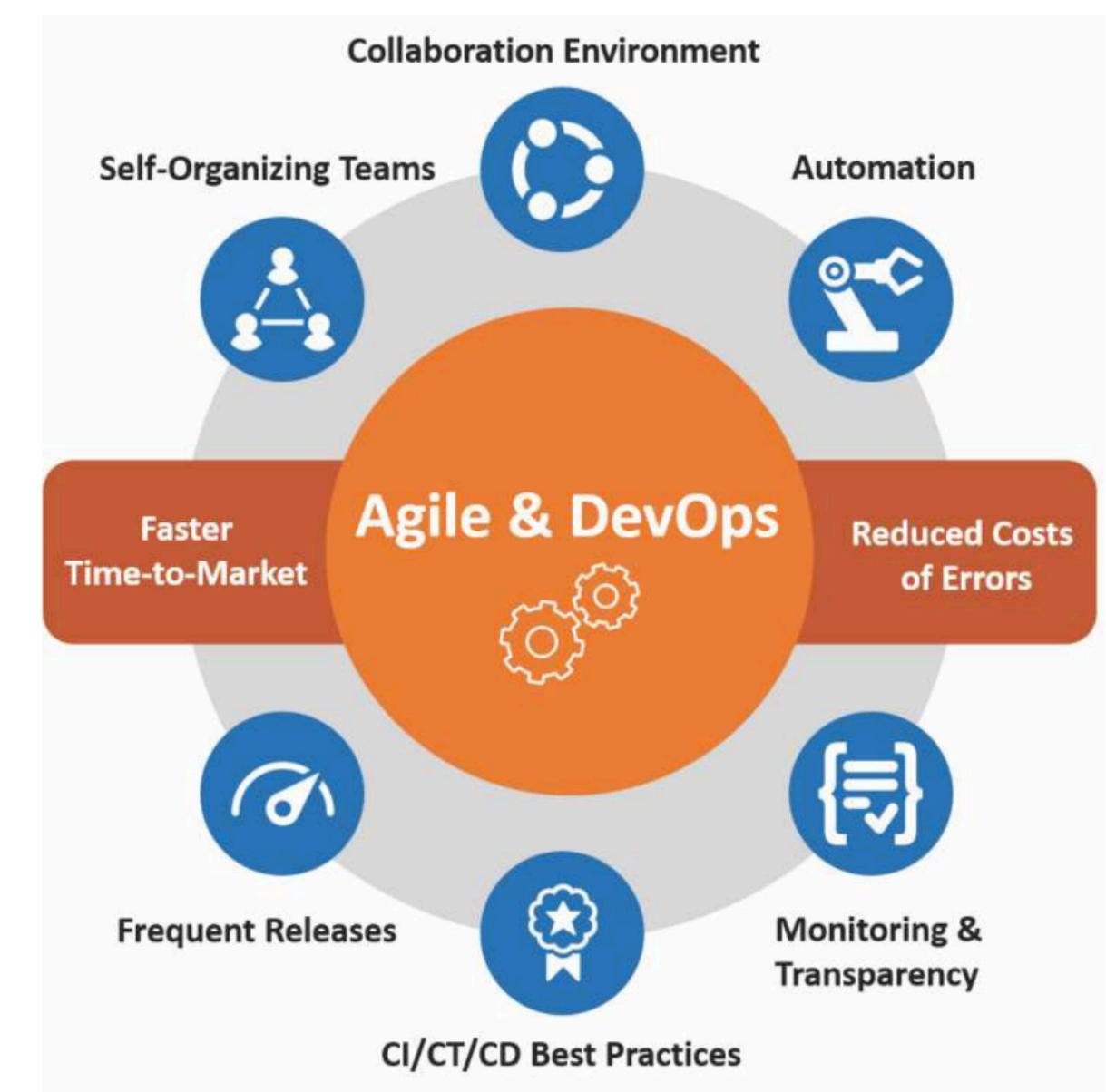
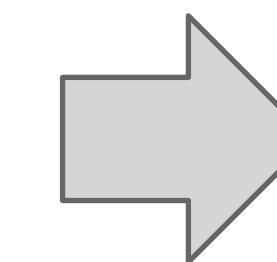
什么变了？什么没有变？ —— 工具变的成熟，思想与内核的实践没有变



1996
极限编程



2009~2010
DevOps 和持续交付



2020
研发效能

《Adapting Agile and DevOps Principles for Business Agility》

经典方法 局限性

传统的工程实践依赖于人能力和意愿，提升个人能力是一个漫长的过程且难以标准化。



如果重写比重构快？

**Any fool can write
code that a computer
can understand. Good
programmers write
code that humans can
understand.**

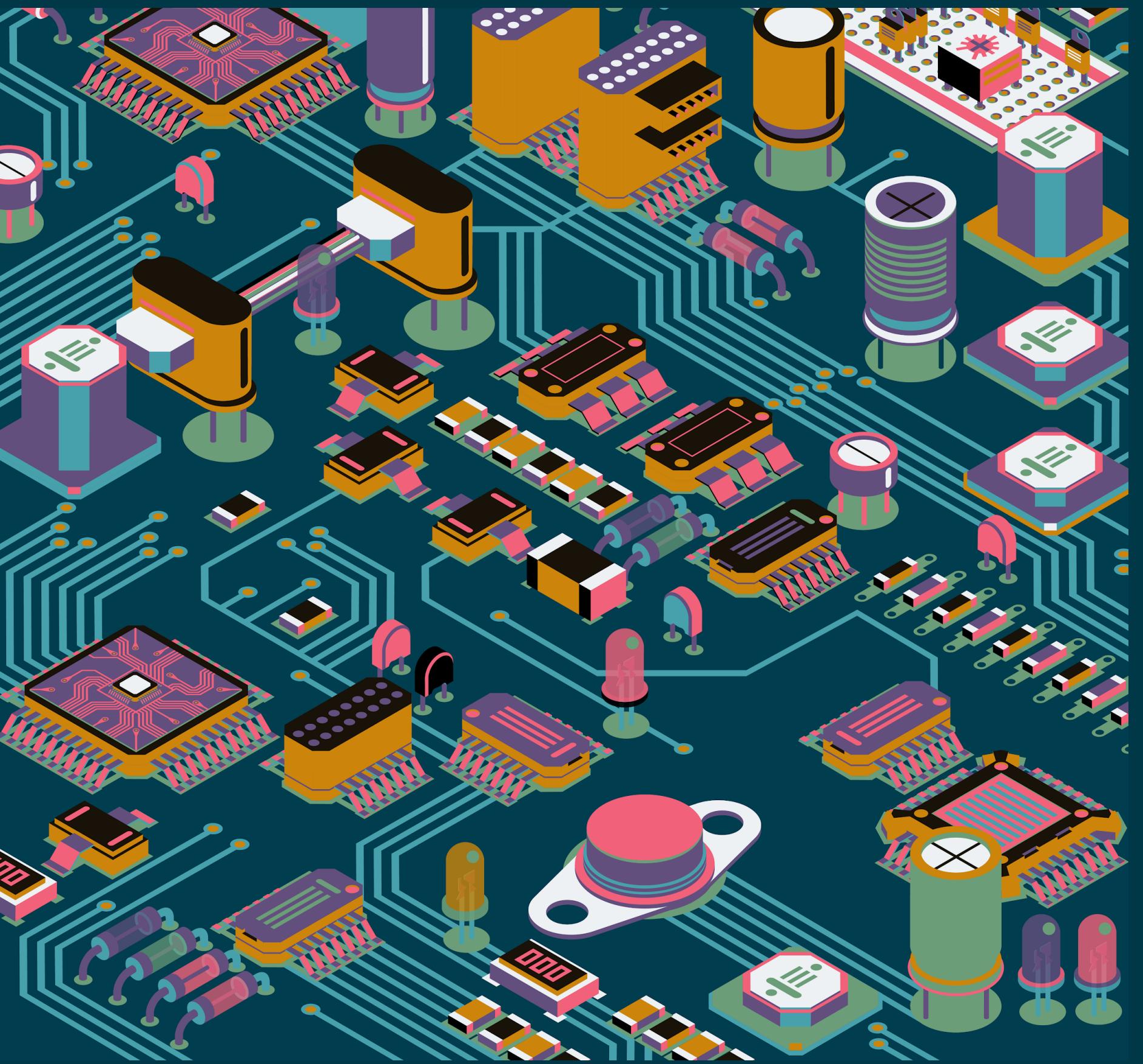
Martin Fowler



Martin Fowler

Thoughtworks 首席科学家，著有《重构：改善既有代码的设计》《分析模式：可复用的对象模型》《领域特定语言》《企业应用架构模式》等经典著作

LLM 在研发效能 带来的新机遇



SDLC Co-piloting with AI

基础模型与人工智能生产力工具正在重塑技术人员的工作方式。

现在，类似于 ChatGPT 和 Microsoft Copilot 这样的人工智能突破正在彻底改变软件行业。

基于人工智能的工具与基础大语言模型可以增强软件开发在设计、需求、测试、发布和运维等各个环节中的能力，提高质量和效率。

产品设计

Ideation  

Prototype   

Research   

Design   

需求分析

Facilitation 

Documentation  

Collaboration  

运营
交付和运维
开发



基础模型

开源

 Stable Diffusion

 CODEGEN

 StarCoder

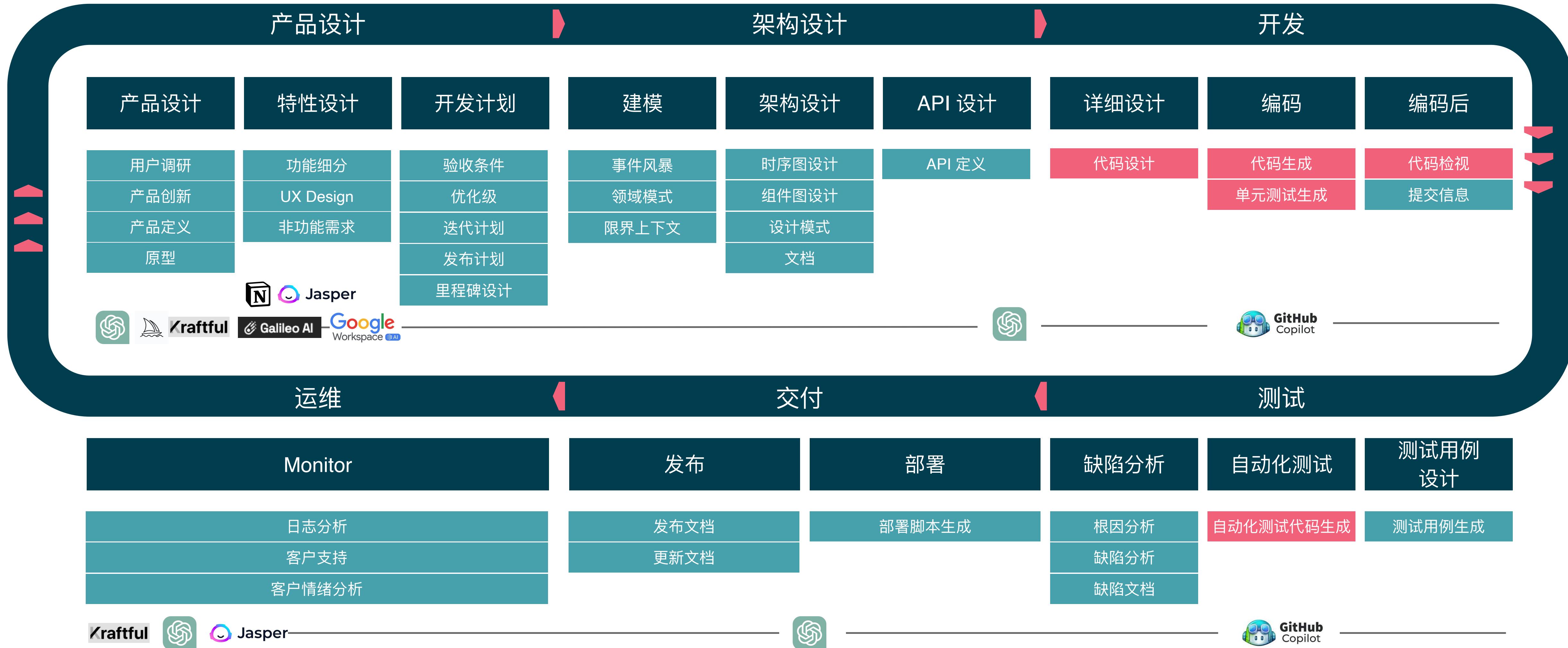
闭源

CODEX

 OpenAI
GPT - 4

梳理研发流程寻找突破点

需要拆到足够小的工序，才能看到明显的效果



ChatGPT + Copilot 端到端使用评估

研究目的

ChatGPT + Copilot 在企业应用中，从需求、编码、测试端到端的辅助能力和提效作用评估

研究过程

第1步：选择一个真实的项目需求: Thoughtworks OKR工具

第2步：使用 ChatGPT 来完成领域术语定义和竞品分析启发，再到MVP特性设计启发

第3步：使用 ChatGPT 来完成特性的需求分析，展示可视化的用户旅程、功能需求拆分及格式化编写、验收标准定义

第4步：尝试使用 ChatGPT 来把验收标准转化为UML设计图

第5步：使用 ChatGPT 生成 API，在通过Copilot 生成代码

第6步：...

第x步：使用 ChatGPT 生成发布文档

研究结论

辅助能力评估：

整体提升 20%~30% 效率

- 其中重复性工作，如需求格式、代码编写、测试用例、单元测试提升较为明显

L1 <10% L2 10-30%

| 特性分析 | | | 架构设计 | | | 代码实现 | | |
|------|------|------|------|------|-------|------|------|-----|
| 特性设计 | 功能拆分 | 定义AC | 架构设计 | 领域模型 | API接口 | 详细设计 | 编写代码 | 编码后 |
| L2 | L2 | L2 | L1 | L1 | L2 | L1 | L2 | L2 |

局限性：

- 每次反馈结果都不一致，需要人工来判断这个结果是否“合理/有用”，能否进入下一步；
- 需要结合经验迭代优化给出更细的细节要求提示，生成结果才会更好；
- ...

产品设计：0-1 产品设计启发（手动）

GPT 现状：生成只是看上去可用的，还需要人工校准，贴合公司组织结构，与组织的实际情况贴合

提示词

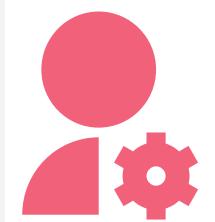


我在设计一个 **OKR 系统**，想实现一个 **员工对齐团队目标，发布 OKR 的功能**，
需要考虑什么点？

设计一个 **OKR 系统**需要考虑以下几个点：

- 设定目标：确定团队的长期目标和短期目标，确保员工可以理解目标，并能够将其转化为可量化的 OKR。
- 制定 OKR：制定关键结果，确保它们是可衡量的、具有挑战性的，并且与团队和公司的目标保持一致。
- 分配 OKR：将 OKR 分配给每个员工，并为每个 OKR 设定期限和优先级。
- 监控进展：确保每个员工都能够及时更新他们的进展，并向团队报告他们的进展情况。
- 跟踪结果：跟踪每个员工达成他们的 OKR 的进度，并为整个团队的 OKR 设定一个跟踪系统。
- 提供反馈：为员工提供实时反馈，并在必要时提供支持和指导，以确保他们能够达成他们的 OKR。
- 评估绩效：使用 OKR 系统来评估每个员工的绩效，并将其与他们的 OKR 相关联。
- 持续改进：定期审查 OKR 系统的效果，并进行必要的调整，以确保它能够支持团队和公司的目标。

人工校准



OKR基础功能清单

- 制定OKR，并支持**设置对齐**，包括：垂直对齐和水平对齐
- 透明多维度的OKR信息，包括：时间、部门、进度状态等
- **更新OKR进度与内容**，并带有“信心指数”功能
- 对OKR进行**评分**，并能灵活设置评分标准
- **管理日常任务**，并将此任务与OKR建立关联

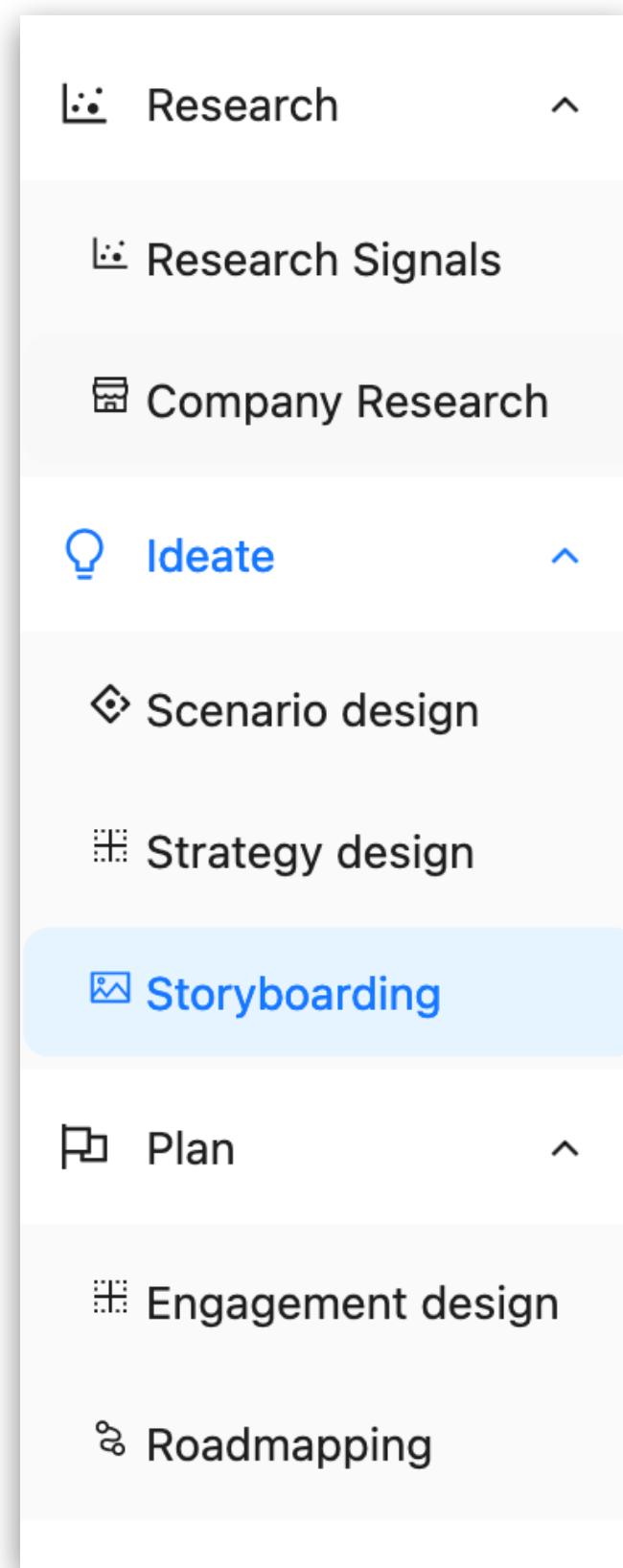
OKR管理工具加分项

- **清晰呈现OKR对齐效果**
- **灵活设置OKR使用层次**，包括：组织、部门、团队、个人等
- 对OKR执行过程提供“**社交化**”特性，包括：提醒、赞赏、评论、勋章等
- **可视化数据报表**
- **多种产品形态**，且实现数据同步，包括：网页端、桌面端、移动端等

产品设计：0-1 产品设计启发（工具化）

产品经理 Copilot：帮助产品经理、需求分析师进行创意和策略方面的研究与分析

产品经理日常



GPT 3.5 + LangChain + 微调模型 (产品领域)

Crystal Boba: Storyboarding

Scenario and story description (multiple lines allowed):
员工对齐团队目标，发布 OKR

Create storyboard Generate at least as many scenes as needed Illustration style: Photo-realistic

Team Meeting to Discuss Goals
The team gathers in the conference room to discuss their objectives for the quarter and how they align with the company's overall goals. Each member shares their ideas and concerns about the upcoming projects. They agree to create an OKR framework to stay on track and measure their progress.

Creating the OKRs
The team splits into smaller groups to create their individual OKRs. They reference the company's mission, vision, and values to ensure each objective is aligned with the organization's goals. They collaborate to create clear, measurable, and specific objectives that will challenge and motivate them throughout the quarter.

Sharing and Discussing the OKRs
With the individual OKRs created, the team reconvenes to review and discuss each objective. The team leader provides constructive feedback on each objective to ensure they are SMART (Specific, Measurable, Achievable, Relevant, and Time-bound). The team considers the feedback and makes changes as needed to ensure their OKRs are realistic and attainable.

OKRs Presentation to Management
At the end of the quarter, the team celebrates the successful completion of their OKRs. They reflect on the challenges they faced and the lessons they learned. They highlight the positive outcomes and how their hard work positively impacted the company's overall goals.

Celebrating Success
At the end of the quarter, the team celebrates the successful completion of their OKRs. They reflect on the challenges they faced and the lessons they learned. They highlight the positive outcomes and how their hard work positively impacted the company's overall goals.

出自：<https://www.boba-ai.com> (By Thoughtwoker)

标准实践
(转换化为 prompt)

Heuristic Evaluation Report

Design System: [Design System Name]
Date: [Date of Evaluation]
Evaluators: [Evaluator Names]

Summary

This heuristic evaluation assessed the [Design System Name] against [Number of Heuristics] usability heuristics, focusing on components, patterns, and guidelines that constitute the design system. [Number of Evaluators] evaluators participated in the evaluation process, identifying a total of [Number of Issues] usability issues. This report outlines these issues, along with recommendations for addressing them to improve the design system's usability and consistency.

Usability Issues and Recommendations

- Issue:** Inconsistent button styles
 - Heuristic:** Consistency and standards
 - Severity:** Major
 - Recommendation:** Standardize button styles across the design system, ensuring that their appearance, hover states, and active states are consistent.
- Issue:** Insufficient color contrast in text elements
 - Heuristic:** Visibility of system status
 - Severity:** Critical
 - Recommendation:** Adjust text colors to meet accessibility guidelines (e.g., WCAG 2.1 Level AA), ensuring sufficient contrast between text and background colors for better readability.
- Issue:** Lack of responsive design patterns
 - Heuristic:** Flexibility and efficiency of use
 - Severity:** Major
 - Recommendation:** Introduce responsive design patterns and guidelines for adapting components and layouts to different screen sizes and devices.

懂点 prompt 的产品专家

架构设计：领域模型设计与代码化输出

人工优化的点：根据我们的理解（上下文）完善了模型

提示词

我在设计一个 OKR 系统，请根据下面的需求，根据你的理解设计领域模型，并使用 **PlantUML** 返回。

作为一个团队负责人，我希望能够创建和更新团队的长期和短期目标，以便能够将它们转化为可量化的 OKR。
作为一个员工，我希望能够创建和更新我的 OKR，以便将我的目标对齐到团队和公司的目标上。
作为一个团队负责人，我希望能够分配 OKR 给每个员工，并设置期限和优先级，以便能够跟踪整个团队的进展情况。
作为一个员工，我希望能够及时更新我的 OKR 进展情况，并向团队报告我的进展情况，以便能够保持团队的透明度和协同性。
作为一个团队负责人，我希望能够跟踪每个员工 OKR 的进度，并将其与他们的绩效相关联，以便能够评估每个员工的绩效。
作为一个团队负责人，我希望能够定期审查 OKR 系统的效果，并进行必要的调整，以便能够持续改进整个系统的效果和价值。
作为一个团队负责人，我希望能够提供支持和指导，以帮助员工达成他们的 OKR，以便能够支持员工的发展和团队的成功。
作为一个员工，我希望能够获得实时反馈和支持，以便能够不断地改进我的表现和达成我的 OKR
###

AI 生成的初步领域模型

The diagram shows a class hierarchy in UML notation. At the top is **Company**, which has a relationship to **Team**. **Team** has relationships to **Employee** and **OKR**. **Employee** has relationships to **OKR** and **PerformanceCycle**. **OKR** has relationships to **Goal** and **Employee**. Below the classes is a code editor window showing the generated PlantUML code for the classes and their interactions.

人工优化后的领域模型

This diagram shows a refined version of the domain model. It includes additional classes like **PerformanceCycle** and **Objective**. The associations between classes like **Team** and **Employee** are more clearly defined with multiplicity and roles. A large circular arrow at the bottom indicates a feedback loop between **OKR**, **Objective**, and **KeyResult**.

注：结合自研、开源的 AutoDev 插件绘制

QCon
全球软件开发大会

InfoQ
uele

架构设计：面向 API 管理

团队需要将 API 规范提供给 AI，否则生成的 API 就不符合规范

提示词

AI 生成的初步 API

我在设计一个 OKR 系统，请根据下面的需求，根据你的理解设计 API。要求：

1. API 功能不重复（包含 request 和 response）
2. 以表格的形式返回 API

需求：

###

作为一个团队负责人，我希望能够创建和更新团队的长期和短期目标，以便能够将它们转化为可量化的 OKR。
作为一个员工，我希望能够创建和更新我的 OKR，以便将我的目标对齐到团队和公司的目标上。

作为一个团队负责人，我希望能够分配 OKR 给每个员工，并设置期限和优先级，以便能够跟踪整个团队的进展情况。

作为一个员工，我希望能够及时更新我的 OKR 进展情况，并向团队报告我的进展情况，以便能够保持团队的透明度和协同性。

作为一个团队负责人，我希望能够跟踪每个员工 OKR 的进度，并将其与他们的绩效相关联，以便能够评估每个员工的绩效。

作为一个团队负责人，我希望能够定期审查 OKR 系统的效果，并进行必要的调整，以便能够持续改进整个系统的效果和价值。

作为一个团队负责人，我希望能够提供支持和指导，以帮助员工达成他们的 OKR，以便能够支持员工的发展和团队的成功。

作为一个员工，我希望能够获得实时反馈和支持，以便能够不断地改进我的表现和达成我的 OKR

###

| API名称 | API功能 | 请求方法 | 请求参数 | 返回参数 |
|-----------------------|------------|------|---|---|
| /teams/create | 创建团队 | POST | team_name (string), long_term_goal (string), short_term_goal (string) | team_id (int), team_name (string), long_term_goal (string), short_term_goal (string) |
| /teams/update | 更新团队信息 | PUT | team_id (int), team_name (string), long_term_goal (string), short_term_goal (string) | team_id (int), team_name (string), long_term_goal (string), short_term_goal (string) |
| /employees/okr/create | 创建员工 OKR | POST | employee_id (int), objective (string), key_results (string), deadline (date), priority (int), team_id (int) | okr_id (int), employee_id (int), objective (string), key_results (string), deadline (date), priority (int), team_id (int) |
| /employees/okr/update | 更新员工 OKR | PUT | okr_id (int), objective (string), key_results (string), deadline (date), priority (int) | okr_id (int), employee_id (int), objective (string), key_results (string), deadline (date), priority (int), team_id (int) |
| /teams/okr/assign | 分配 OKR 给员工 | POST | okr_id (int), employee_id (int) | okr_id (int), employee_id (int), team_id (int) |
| ... | ... | ... | ... | ... |

开发：结合 GitHub Copilot 编写代码和测试

复制或者编写注释到 IDE 中，可自动结合上下文信息（打开的 tab、代码等）生成

```
// ObjectiveDTO: { "name": string, "description": string, "type": string }
public class ObjectiveDTO {
    private String name;
    private String description;
    private String type;

    public ObjectiveDTO() {}

    public ObjectiveDTO(String name, String description, String type) {
        this.name = name;
        this.description = description;
        this.type = type;
    }

    public String getName() {
        return name;
    }

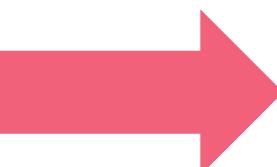
    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getType() {
        return type;
    }
}
```

通过Copilot 生成后
进行校准



```
@Test
void should_create_objective_when_perform_update_objective() throws Exception {
    ObjectiveDTO objectiveDTO = new ObjectiveDTO("objective", "description");

    String updateJson = new ObjectMapper().writeValueAsString(objectiveDTO);
    client.perform(MockMvcRequestBuilders.post(urlTemplate: "/teams/1/objectives")
        .contentType("application/json")
        .content(updateJson))
        .andExpect(MockMvcResultMatchers.status().isOk())
        .andExpect(MockMvcResultMatchers.jsonPath(expression: "$.id").isNotEmpty())
        .andExpect(MockMvcResultMatchers.jsonPath(expression: "$.name").value(expectedValue));

    assertEquals(expected: 1, objectiveRepository.findAll().size());
}

// PUT /teams/{teamId}/objectives/{objectiveId}
// ObjectiveDTO: { "name": string, "description": string, "type": string }
// ObjectiveResponse: { "id": string, "name": string, "description": string, "type": string }
@Test
void should_update_objective_when_perform_update_objective() throws Exception {
    Objective objective = new Objective("objective", "description", "type");
    objectiveRepository.save(objective);

    ObjectiveDTO objectiveDTO = new ObjectiveDTO("objective1", "description1", "type1");

    String updateJson = new ObjectMapper().writeValueAsString(objectiveDTO);
}
```

根据当前的习惯
生成

AI辅助提效的新挑战

虽然生成式AI技术及工具已经在软件生命周期的各个环节中体现出了提效的可行性，但想要在大型科技组织中端到端落地、实现提效的突破，还面临很多挑战。大部分企业尚在探索过程中，于是我们产生了两个假设。

企业规模化软件过程提效的挑战

信息安全和信创化的强制要求

大型企业尤其是银行，面临最强的信息安全监管及信创化、国产化要求，需要国产生态中能力突出的大模型

开源大模型表现偏弱、自己训练成本高

目前可私有化部署的大模型，其自然语言理解和代码生成的水平与GPT有一定差距；根据大语言模型论文做一些粗略的数学计算，如果用的是Facebook LLaMA，训练成本（不考虑迭代或者出错）大约是400万美元，如果是谷歌PaLM，大约2700万美元。

与企业内部工具进行结合

碎片化的应用提效效果有限，把AI无缝集成到BizDevOps工具链中，技术难度尚未可知。

开发 AI 辅助研发提效的局限性

基于 GPT 模型的工具不符合信息安全要求

目前大多工具基于OpenAI GPT构建，不能私有化部署，不符合信息安全的强制要求；需要寻找能够私有化部署且水平相当的替代品。

公开LLM 针对专业领域表现不佳，适用性差

缺乏专业知识，对于某些专业领域缺乏足够的理解。它受训练数据中的影响，容易产生偏见和错误。

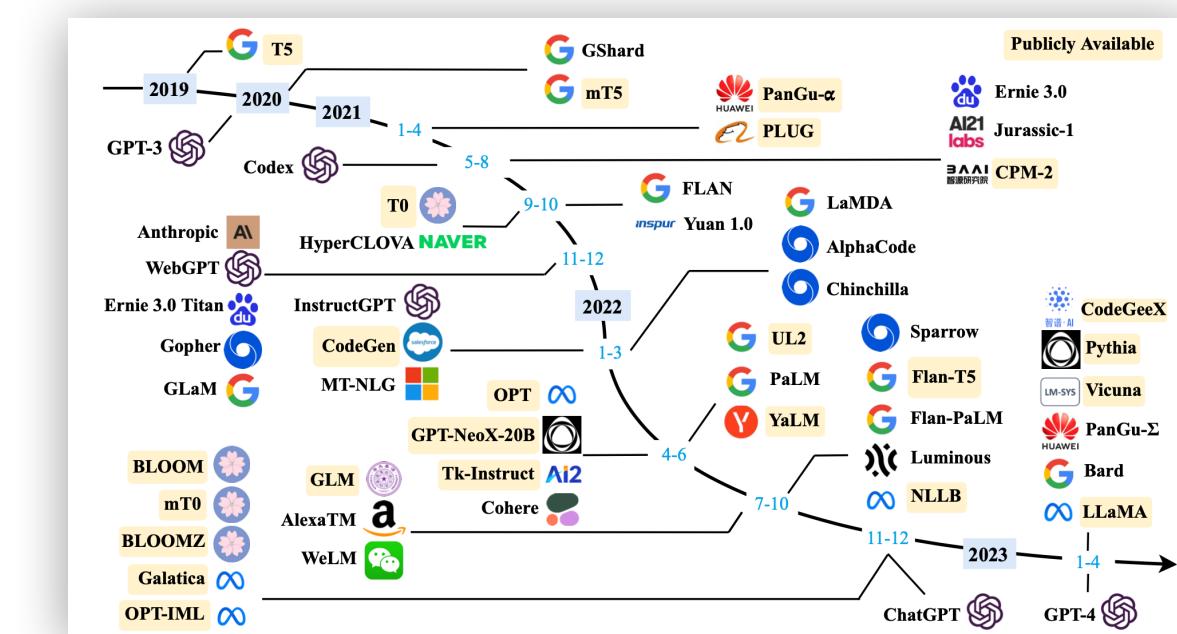
LLM 工具碎片化

各类工具都是在一个个分散的工作节点上辅助，使用时来回切换工具的成本很高，整体端到端地提效不明显。

2 个关键假设

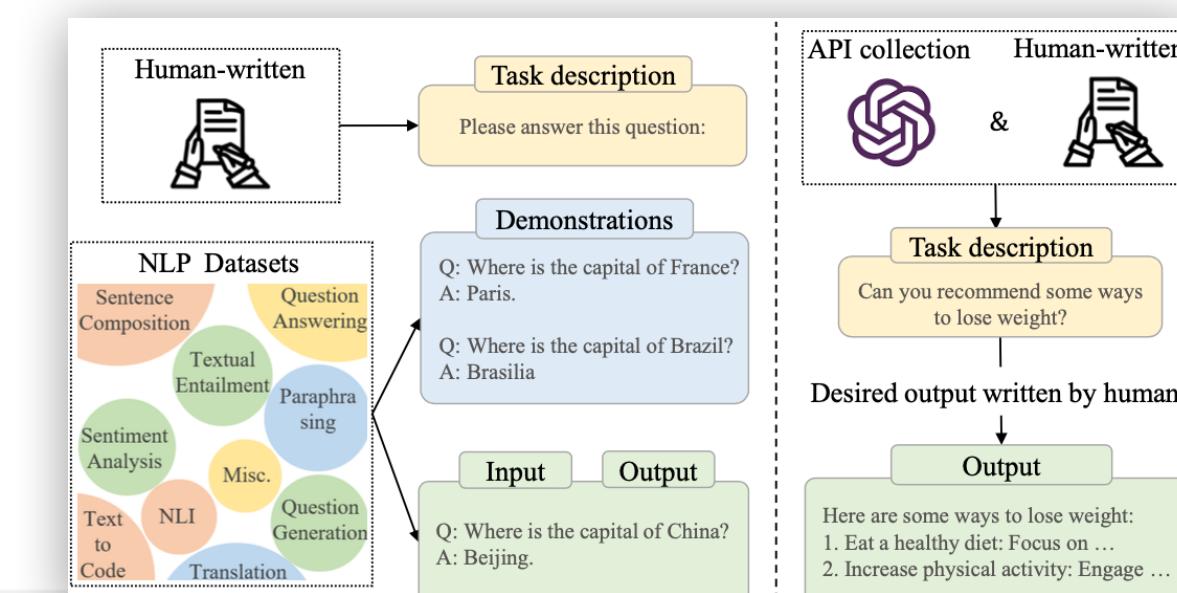
假设1

每个大型企业都将有私有化的大语言模型

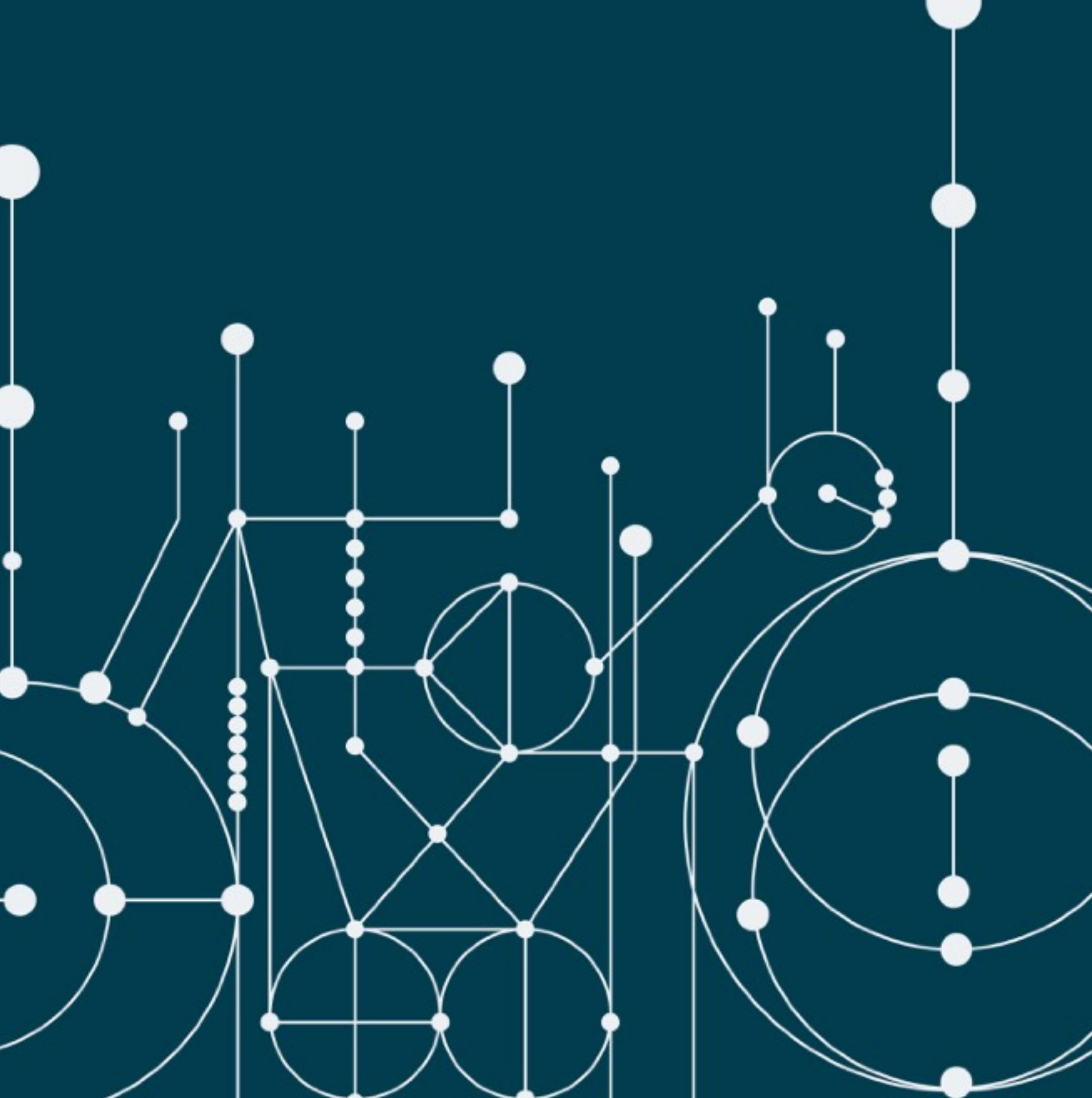


假设2

私有化的主流方式：开源 LLM + 微调



LLM + 工序 探索与构建路径



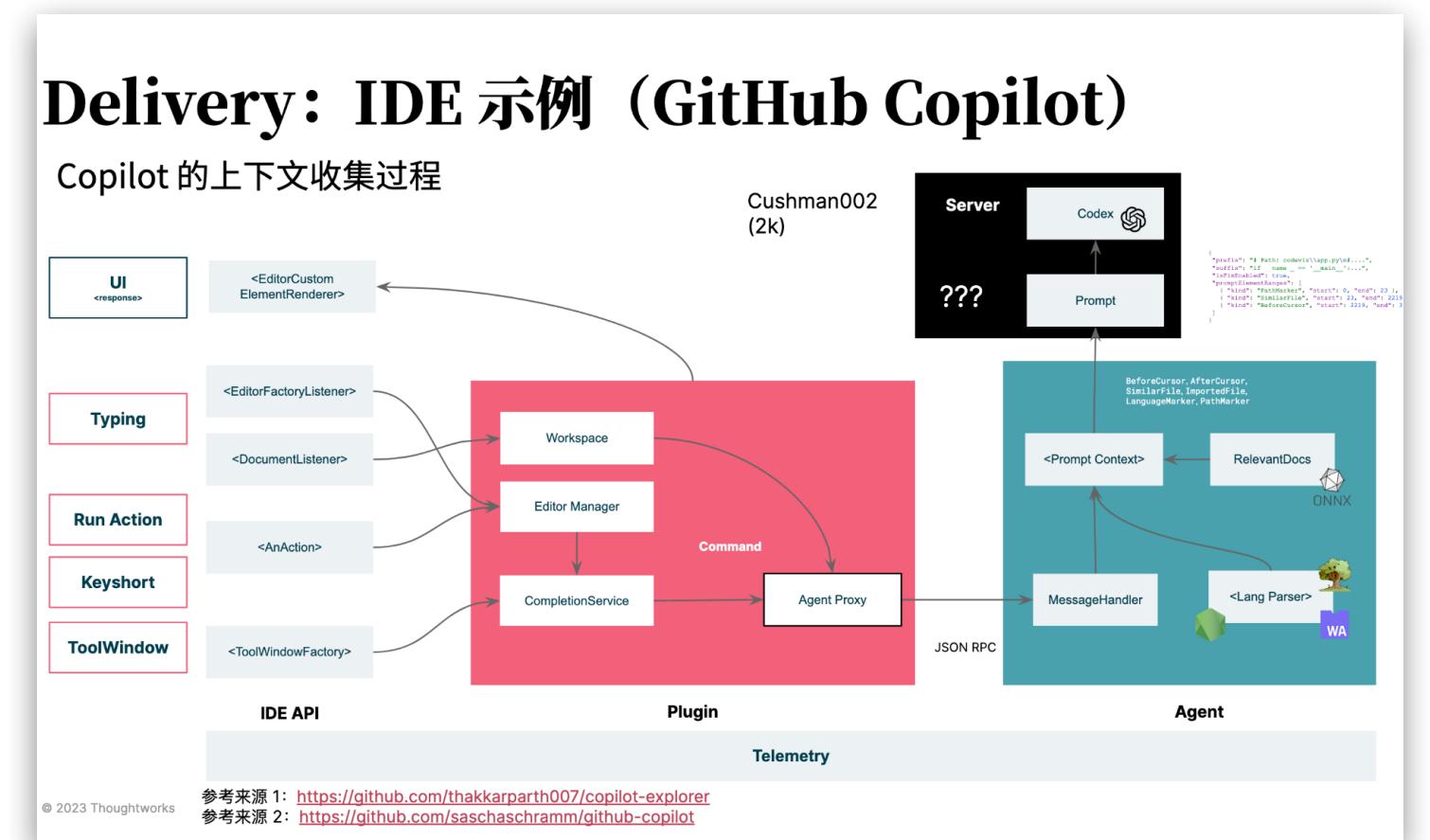
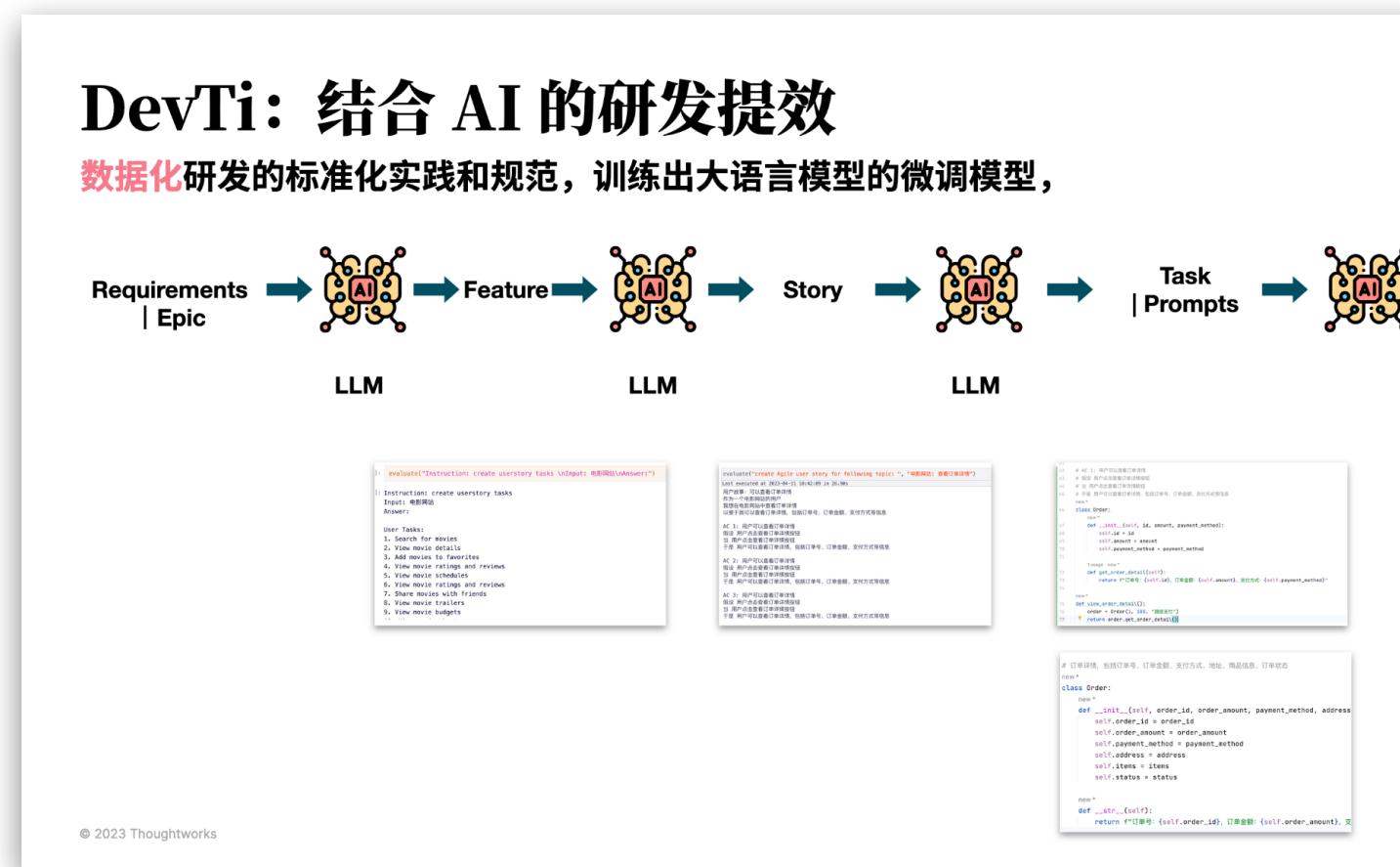
前期探索过程：在摸索中前进

从“0”开始，要学习和解决的问题、挑战太多太多了。

微调与研发过程结合

工具设计和整合

分析成功 AI 产品



现有的开源模型能满足需求吗？

合理的指令（输入和输出）应该是怎样的？

微调的成本和工作量在哪？

<https://github.com/unit-mesh/devti>

产品形态应该是怎样的？（聊天、交互式）

与现有工具对比竞争力在哪？（端到端？）

工具是否需要重写一遍？

<https://github.com/unit-mesh/auto-dev>

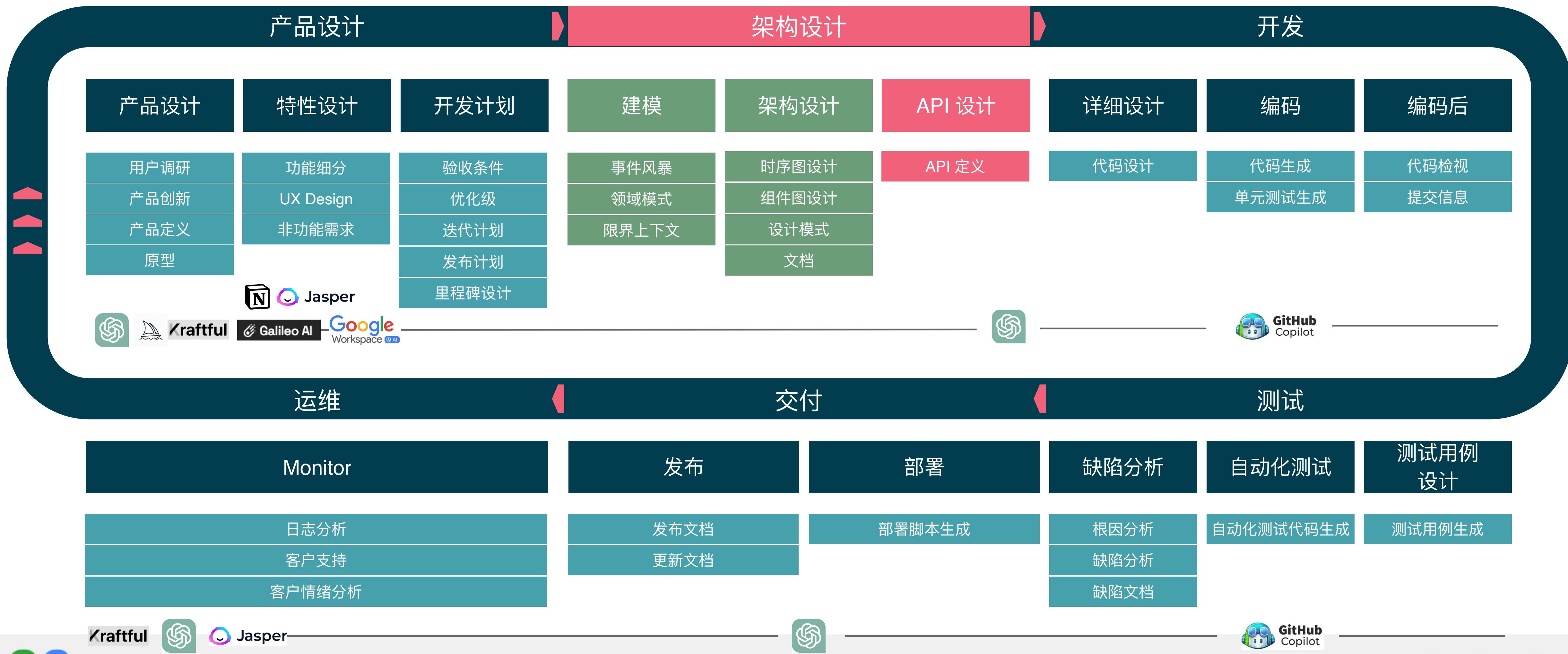
如何高效构建上下文？（2K）

如何提升速度？（缓存、减少请求）

如何提升准确率/接受率？

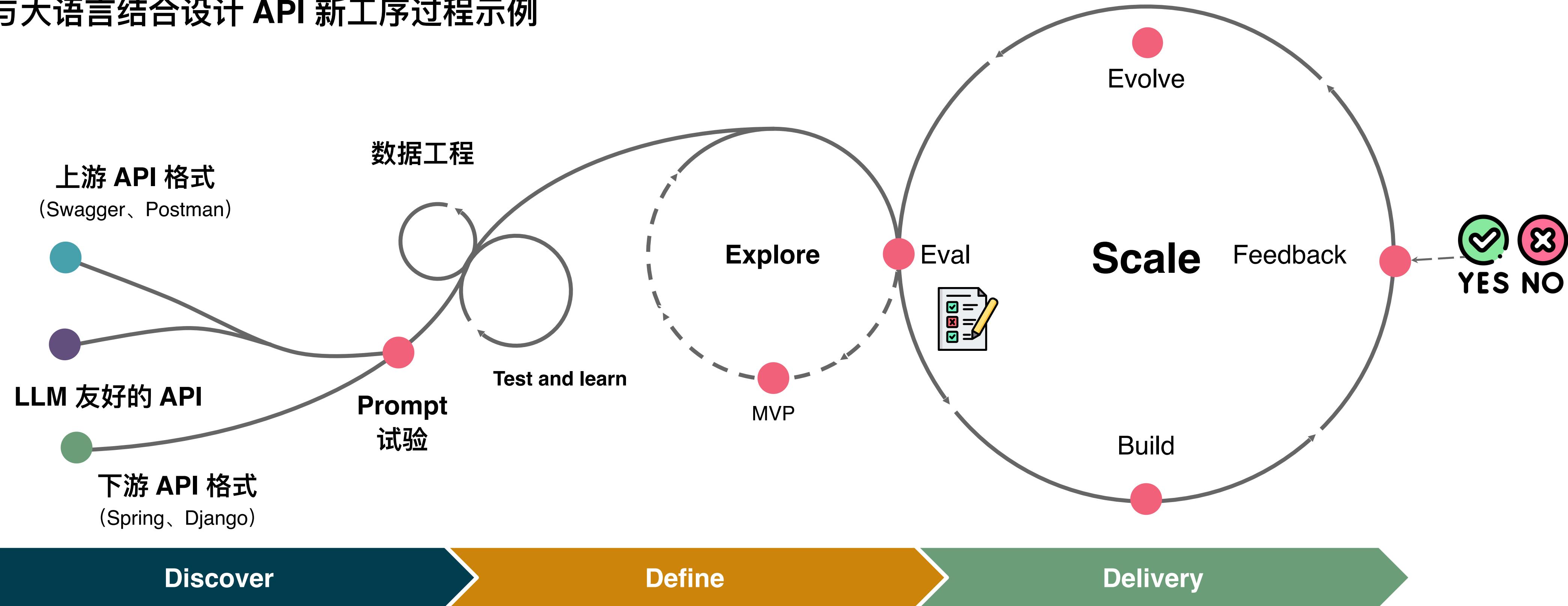
SDLC 中的高价值部分

在私有化模型的能力之内，**最适合探索**的是什么？



大语言模型友好的工序

架构与大语言结合设计 API 新工序过程示例



流程与“语言”工程

- 软件开发过程资产化
- 资产的“语言建模”

数据与 Prompt 工程

- MVP 与微调试验
- 增量引导指标设计

上下文与平台工程

- 上下文工程
- 持续反馈的平台工程

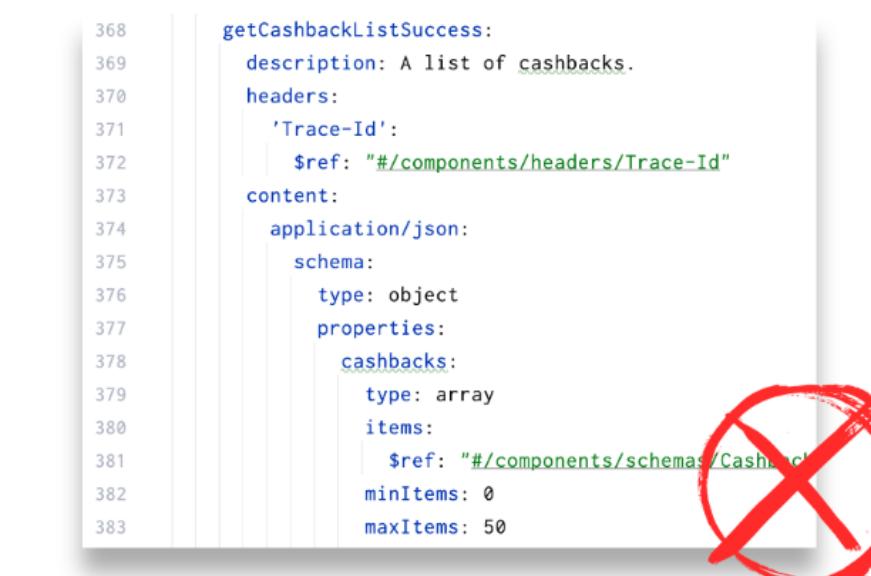
Discover 探索阶段

大模型友好的数据格式 (API 示例)

用于微调的数据集既要适合于 LLM 生成，又要适合于人类阅读，还要适合于工具化集成

Markdown 表格

JSON, Yaml



368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383

```
getCashbackListSuccess:  
  description: A list of cashbacks.  
  headers:  
    'Trace-Id':  
      $ref: "#/components/headers/Trace-Id"  
  content:  
    application/json:  
      schema:  
        type: object  
        properties:  
          cashbacks:  
            type: array  
            items:  
              $ref: "#/components/schemas/Cashback"  
            minItems: 0  
            maxItems: 50
```

HTTP Request (IntelliJ IDEA)



```
1  ### Send POST request with json body  
2  POST https://httpbin.org/post  
3  Content-Type: application/json  
4  
5  {  
6    "id": 999,  
7    "value": "content"  
8  }  
9  
10  ### Send POST request with body as parameters  
11  POST https://httpbin.org/post  
12  Content-Type: application/x-www-form-urlencoded  
13  
14  id=999&value=content
```

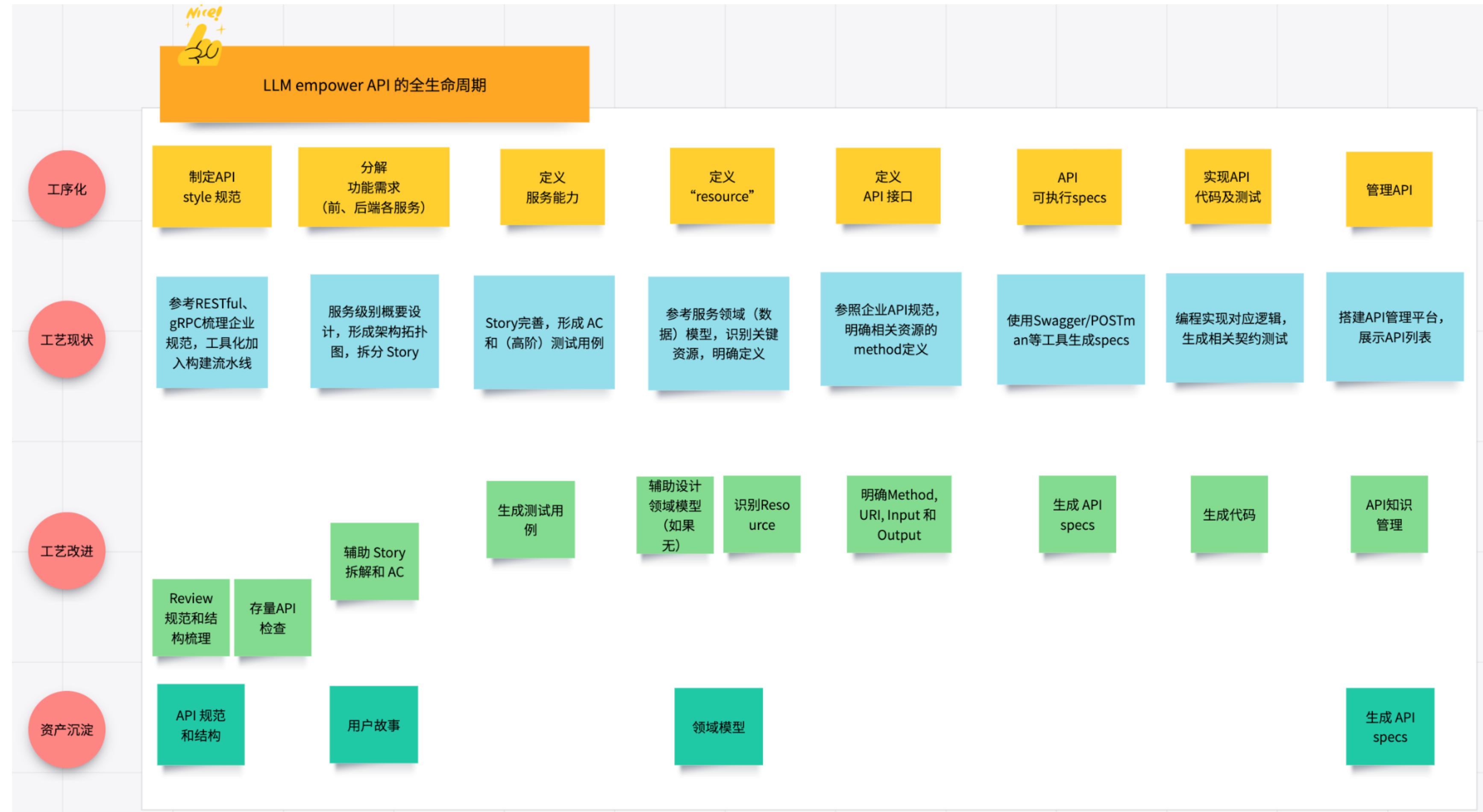
适合于解析，但是阅读困难 解析难度低，也适合于阅读 适合于阅读和工具化，但是实现起来复杂

© 2023 Thoughtworks

44

软件工程过程资产化 (API 示例)

基于私有化 LLM 的新工序一部分取决于：已有的软件过程资产（基于架构规范）



挑战：

1. 定义概念
2. 标准化流程（工序）
3. 审视规范
4. 过程资产



新机遇？

资产的“语言建模”

探索适合于大语言模型的资产代码化形式

功能需求

| 用例名称 | 前置条件 | 后置条件 | 主成功场景 |
|--------|--------------|-------------|-------|
| 提交发票申请 | 用户已登录并填写发票信息 | 发票申请提交成功 | 用户输入正 |
| 批量查询发票 | 用户已登录 | 返回满足条件的发票列表 | 用户输入正 |
| 查询发票详情 | 用户已登录 | 返回指定发票的详细信息 | 用户输入正 |

建模

```
1 @startuml
2 class Invoice {
3     -invoiceId: InvoiceId
4     -status: InvoiceStatus
5     -applyTime: DateTime
6     -buyerName: BuyerName
7     -sellerName: SellerName
8     -amount: Money
```

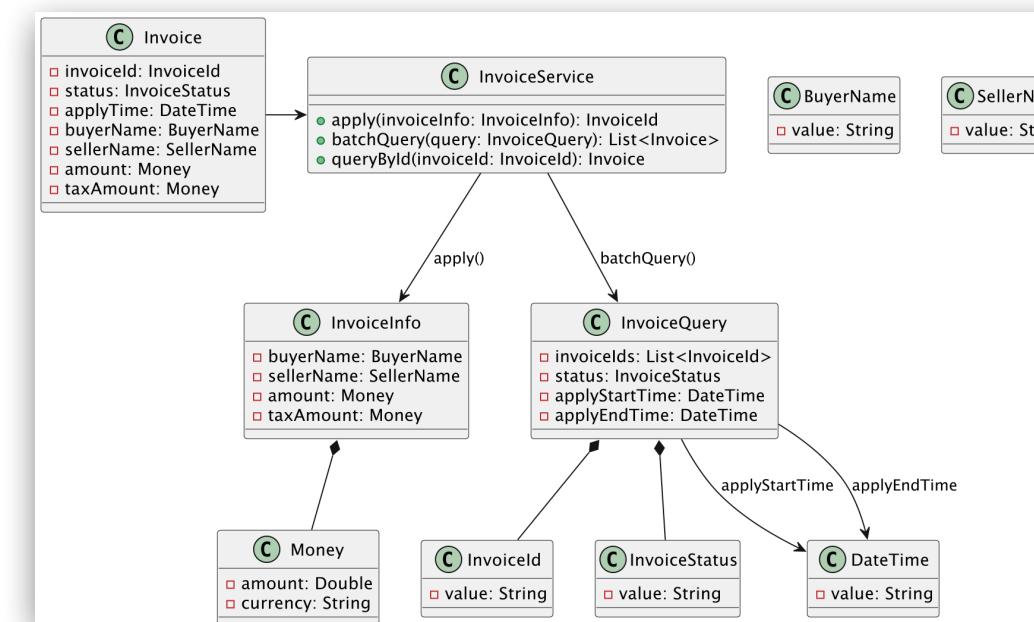
API Spec

| API | Method | Description | Request | Response | Error Response |
|---|--------|-------------|---------|----------|----------------|
| /v3/alipay/ebpp/invoice/apply/inv/upload | PUT | 发票回传接口适用 | | | |
| /v3/alipay/ebpp/invoice/apply/result-sync | POST | ISV向支付宝同 | | | |
| /v3/alipay/ebpp/invoice/apply/status/notify | PUT | 发票申请状态 | | | |

源格式

可视化

| 用例名称 | 前置条件 | 后置条件 | 主成功场景 | 扩展场景 |
|--------|--------------|-------------|-----------------------------|----------------------|
| 提交发票申请 | 用户已登录并填写发票信息 | 发票申请提交成功 | 用户输入正确的发票信息，系统保存发票信息并分配发票编号 | 用户输入错误的发票信息，系统提示错误信息 |
| 批量查询发票 | 用户已登录 | 返回满足条件的发票列表 | 用户输入正确的查询条件，系统返回符合条件的发票列表 | 用户输入错误的查询条件，系统提示错误信息 |
| 查询发票详情 | 用户已登录 | 返回指定发票的详细信息 | 用户输入正确的发票编号，系统返回此发票的详细信息 | 用户输入错误的发票编号，系统提示错误信息 |



| API | Method | Request | Response | Error Response |
|--|--------|---|----------|------------------------|
| /v3/alipay/boss/fnc/invoice/apply | POST | [200: {"resultset": "123"}, 500: {}] | | 400: {"error": String} |
| /v3/alipay/boss/fnc/invoice/batchquery | POST | [200: {}, 500: {"value": ""}] | | 400: {"error": String} |
| /v3/alipay/boss/fnc/invoice/query?invoice_id | GET | invoice_id: 123 [200: {}, 500: {"value": ""}] | | 400: {"error": String} |

Define 定义阶段

```
evaluate("Instruction: 帮我设计一组 API: \nInput: Investment, Customer \nAnswer:")
Last executed at 2023-05-01 17:54:48 in 18.48s
Instruction: 帮我设计一组 API:
Input: Investment, Customer
Answer:
以下是一组 API 的示例:

API:
| API | Method | Description | Request | Response | Error Response |
| --- | --- | --- | --- | --- | --- |
| /investments | GET | 获取投资组合 | 200 {"id": String, "type": String, "balance": Decimal} | 400: { "error": String} |
| /investments/{id}/balance | GET | 获取投资组合余额 | {"id": String} | 200 {"balance": Decimal} | 400: { "error": String} |
| /investments | POST | 创建投资组合 | {"id": String, "type": String, "balance": Decimal} | 201 {"id": String, "type": String, "balance": Decimal} | 400: { "error": String} |
| /investments/{id}/balance | PUT | 更新投资组合余额 | {"id": String} | 200 {"balance": Decimal} | 400: { "error": String} |
| /investments/{id}/type | PUT | 更改投资组合类型 | {"id": String, "type": String} | 200 {"type": String} | 400: { "error": String} |
| /investments/{id}/transfer | PUT | 投资组合转移 | {"from": String, "to": String} | 200 {"message": String} | 400: { "error": String} |
| /investments/{id}/description | GET | 获取投资组合描述 | {"id": String} | 200 {"description": String} | 400: { "error": String} |

evaluate("Instruction: 设计一组银行的 API \nInput: Customer \nAnswer:")
Last executed at 2023-05-01 17:55:33 in 22.77s
Instruction: 设计一组银行的 API
Input: Customer
Answer:
# Customer

API | Method | Description | Request | Response | Error Response |
--- | --- | --- | --- | --- | --- |
| /accounts | GET | Get all customer accounts | 200 {"accounts": [{"id": String, "balance": Number}]} | 400: { "error": String} |
| /accounts/{id}/balance | GET | Get account balance | {"id": String} | 200 {"balance": Number} | 400: { "error": String} |
| /accounts/{id}/transactions | GET | Get account transactions | {"id": String, "start_date": Date, "end_date": Date} | 200 {"transactions": [{"id": String, "date": Number}]} | 400: { "error": String} |
| /accounts/{id}/transfer | POST | Transfer money between accounts | {"from_id": String, "to_id": String, "amount": Number} | 200 {"message": String} | 400: { "error": String} |
| /accounts/{id}/payee | GET | Get payee list | {"id": String} | 200 {"payee": [{"name": String, "address": String, "id": String}]} | 400: { "error": String} |
| /accounts/{id}/payee/{payee_id}/list | GET | Get payee details | {"id": String, "start_date": Date, "end_date": Date} | 200 {"name": String, "address": String, "id": String} |
| /accounts/{id}/recipient | GET | Get recipient account | {"id": String} | 200 {"recipient": [{"name": String, "address": String, "id": String}]} | 400: { "error": String} |
| /accounts/{id}/recipient/{recipient_id}/list | GET | Get recipient details | {"id": String}
```

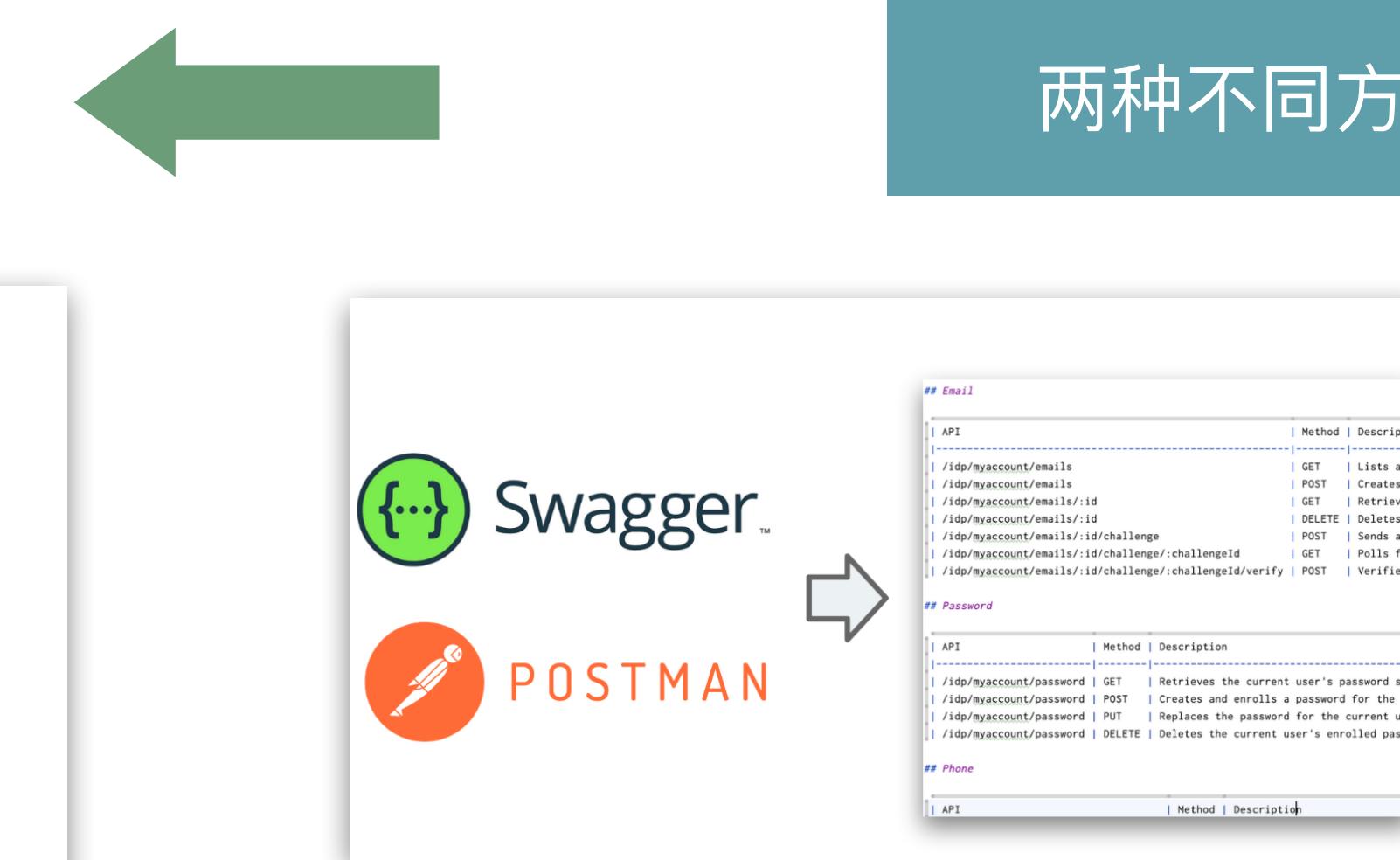
MVP 与微调试验：数据处理与 Prompt 优化

对已有数据进行标准化，生成指令格式数据集（反复多次）

预期数据结果



```
{  
  "instruction": "生成 RESTful API 接口",  
  "input": "查看账户收入, 添加收入记录, 修改收入记录, 删除收入记录",  
  "output": "## Enrichment API\n\nIncomes| API | Method | Description | Request | Response | Error Response |\n--- | --- | --- | --- | --- | --- | --- |\napi/incomes/?page=1&account=uuid | GET |  
| page: 1, account: uuid | [200: {}, 401:  
[\t{\t\t\"code\":  
\"<string>\",\t\t\"message\":  
\"<string>\",\t\t\"request_id\":  
\"<string>\t},\t\t\"code\":  
\"<string>\",\t\t\"message\":  
\"<string>\",\t\t\"request_id\":  
\"<string>\t}]] | 400: {\"error\":  
String} |\n| ...
```



基于已有数据生成

1. 收集公开数据集 (HuggingFace..)
2. 收集和公司内部的私有数据 (整理)

OpenAI 蒸馏

1. 内部数据集不够时, 可以模拟场景
2. 降低人工成本

设计增量引导指标

作为分阶段的微调目标：5% -> 10%

量级评估

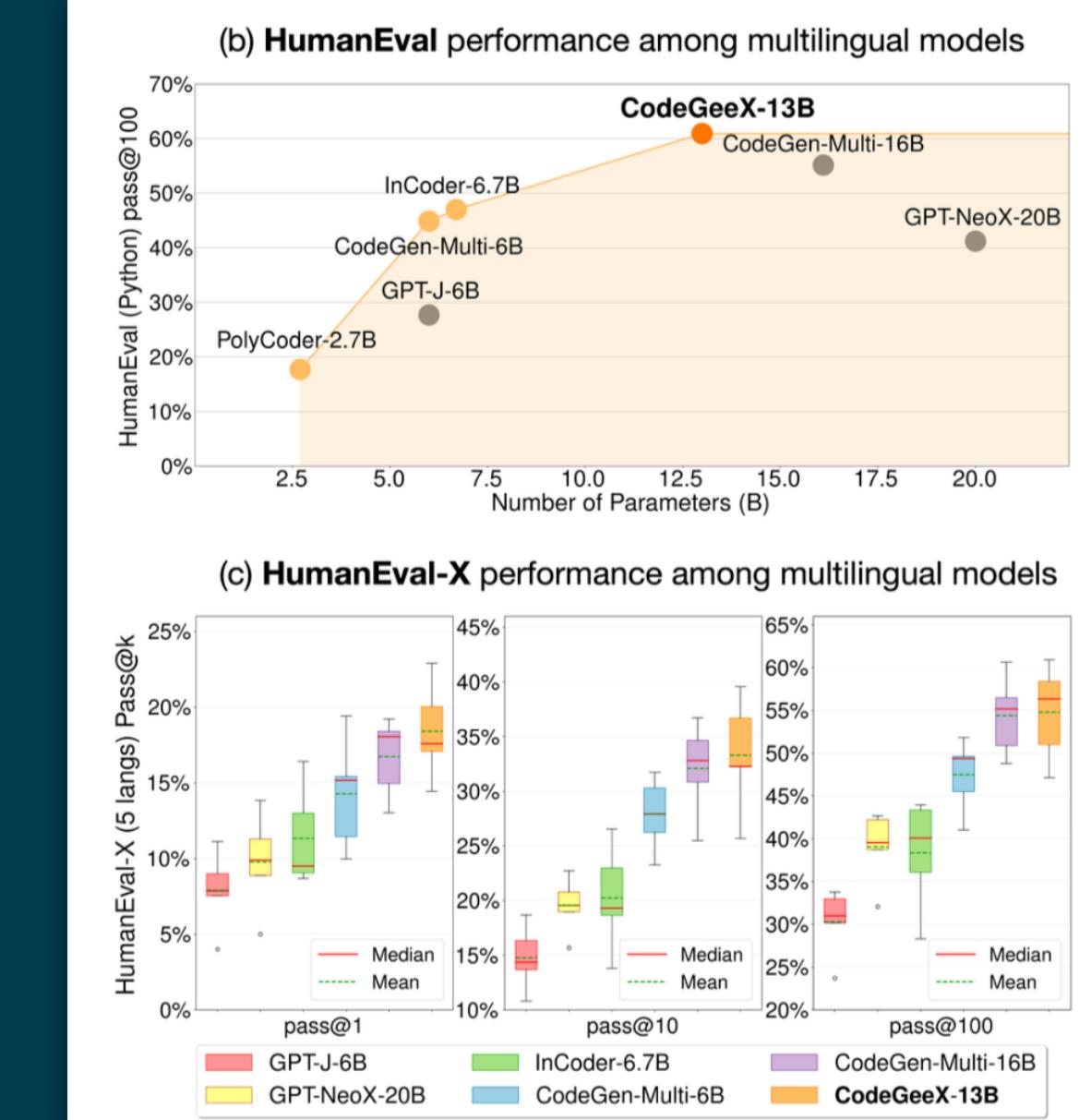
| 场景 | 数据 | 数据来源 |
|------------------|---------------------------------------|-----------------------------|
| 辅助功能需求 (AC编写) | 1k, 3k 个用户故事 微调及生成结果对比 | 结合公开搜集到的应用类型，调用 OpenAI 进行生成 |
| 辅助编码 (特定上下文) | 10k, 20k, 40k, 80k 个代码片段 微调及生成结果对比 | 公开的代码数据集 |
| 辅助API设计 | 1k, 3k 个 API spec 结果对比 | 业界各银行的开放银行API资料数据格式化后作为数据集 |

基于 ChatGLM (6B)、LLaMA (7B) 微调试验

3k 能达到初步可用

HumanEval 示例
(评估代码生成质量)

API Eval



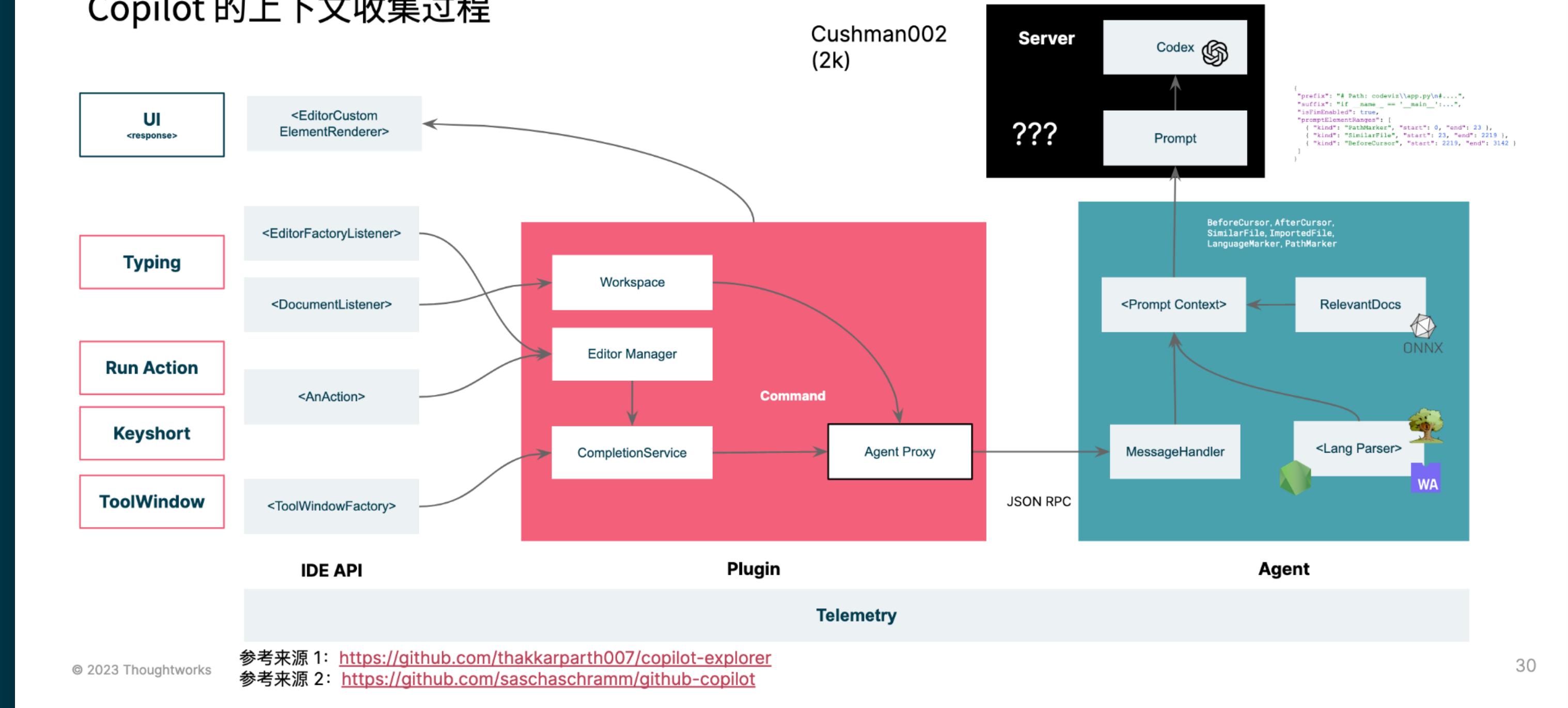
手写高质量的大模型测试用例
(TODO)

图出自：<https://arxiv.org/abs/2303.17568>

Delivery 交付阶段

Delivery: IDE 示例 (GitHub Copilot)

Copilot 的上下文收集过程



有限 token 的上下文工程：提供足够丰富的上下文

上下文的构建应该被当作是一个工程、策略，而不是一个简单的事情。

```
ProductController.java  Product.java  ProductRepository.java x
19  @Autowired
20  private ProductJpaPersistence repository;
21
22  public Long save(Product product) {
23      ProductDataEntity productDataEntity =
24          repository.save(mapper.map(product, ProductDataEntity.class));
25      return productDataEntity.getId();
26  }
27
28
29  public Product findById(Long productId) {
30      Optional<ProductDataEntity> product = repository.findById(productId);
31      return product.map(productDataEntity -> mapper.map(productDataEntity, Product.class));
32  }
33
34  public List<Product> findAll() {
35      // ...
36      List<Product> result = Lists.newArrayList();
37      // ...
38      Iterable<ProductDataEntity> iterable = repository.findAll();
39      // ...
40      iterable.forEach((productDataEntity) -> result.add(mapper.map(productDataEntity, Product.class)));
41      // ...
42      return result;
43  }
44
45  return Lists.newArrayList(repository.findAll().stream()
46      .map(productDataEntity -> mapper.map(productDataEntity, Product.class))
47      .collect(Collectors.toList()));
48
```



```
{
  "type": "prompt",
  "prompt": {
    "prefix": "// Path: src/main/java/com/dmall/productservice/infrastructure/repositories/ProductRepositor
    "suffix": "return Lists.newArrayList(repository.findAll().stream()\n      .map(productDataEntity -> mapper.map(productDataEntity, Product.class))\n      .collect(Collectors.toList()));",
    "isFimEnabled": true,
    "promptElementRanges": [
      {
        "kind": "PathMarker",
        "start": 0,
        "end": 99
      },
      {
        "kind": "SimilarFile",
        "start": 99,
        "end": 1567
      },
      {
        "kind": "SimilarFile",
        "start": 1567,
        "end": 2481
      },
      {
        "kind": "BeforeCursor",
        "start": 2481,
        "end": 3983
      }
    ]
  }
}
```

2048 上限

```
1 // Path: src/main/java/com/dmall/productservice/infrastructure/repositories/ProductRepositor
2 // Compare this snippet from src/main/java/com/dmall/productservice/domain/product/Product.j
3 //
4 // package com.dmall.productservice.domain.product;
5 //
6 // public class Product {
7 // ...
8 // }
9 //
10 package com.dmall.productservice.infrastructure.repositories;
11
12 @Repository
13 public class ProductRepository {
14
15     protected static final ModelMapper mapper = new ModelMapper();
16
17     @Autowired
18     private ProductJpaPersistence repository;
19
20     public Long save(Product product) {
21         ProductDataEntity productDataEntity =
22             repository.save(mapper.map(product, ProductDataEntity.class));
23         return productDataEntity.getId();
24     }
25
26     public Product findById(Long productId) {
27         Optional<ProductDataEntity> product = repository.findById(productId);
28         return product.map(productDataEntity -> mapper.map(productDataEntity, Product.class));
29     }
30
31     public List<Product> findAll() {
32         // ...
33         List<Product> result = Lists.newArrayList();
34         // ...
35         Iterable<ProductDataEntity> iterable = repository.findAll();
36         // ...
37         iterable.forEach((productDataEntity) -> result.add(mapper.map(productDataEntity, Product.class)));
38         // ...
39         return result;
40     }
41
42     return Lists.newArrayList(repository.findAll().stream()
43         .map(productDataEntity -> mapper.map(productDataEntity, Product.class))
44         .collect(Collectors.toList()));
45 }
```

prefix
(文件路径 + 其它文件 + 光标前代码)

```
return Lists.newArrayList(repository.findAll().st
  .map(productDataEntity -> mapper.map(productDataEntity, Product.class))
  .collect(Collectors.toList()));
```

suffix
(光标后代码)

IDEA 潜在的相关信息

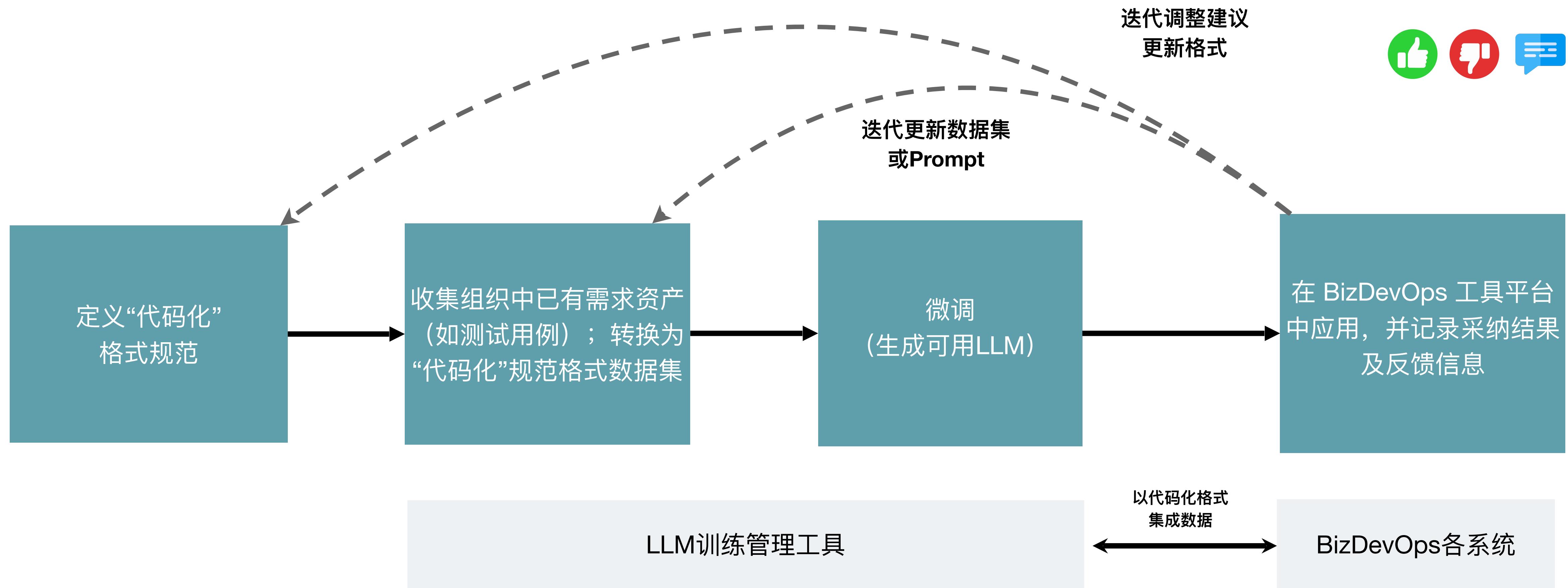
生成的 Prompt
(反复计算)

参考来源 1: <https://github.com/thakkarparth007/copilot-explorer>

参考来源 2: <https://github.com/saschaschramm/github-copilot>

持续反馈的平台工程

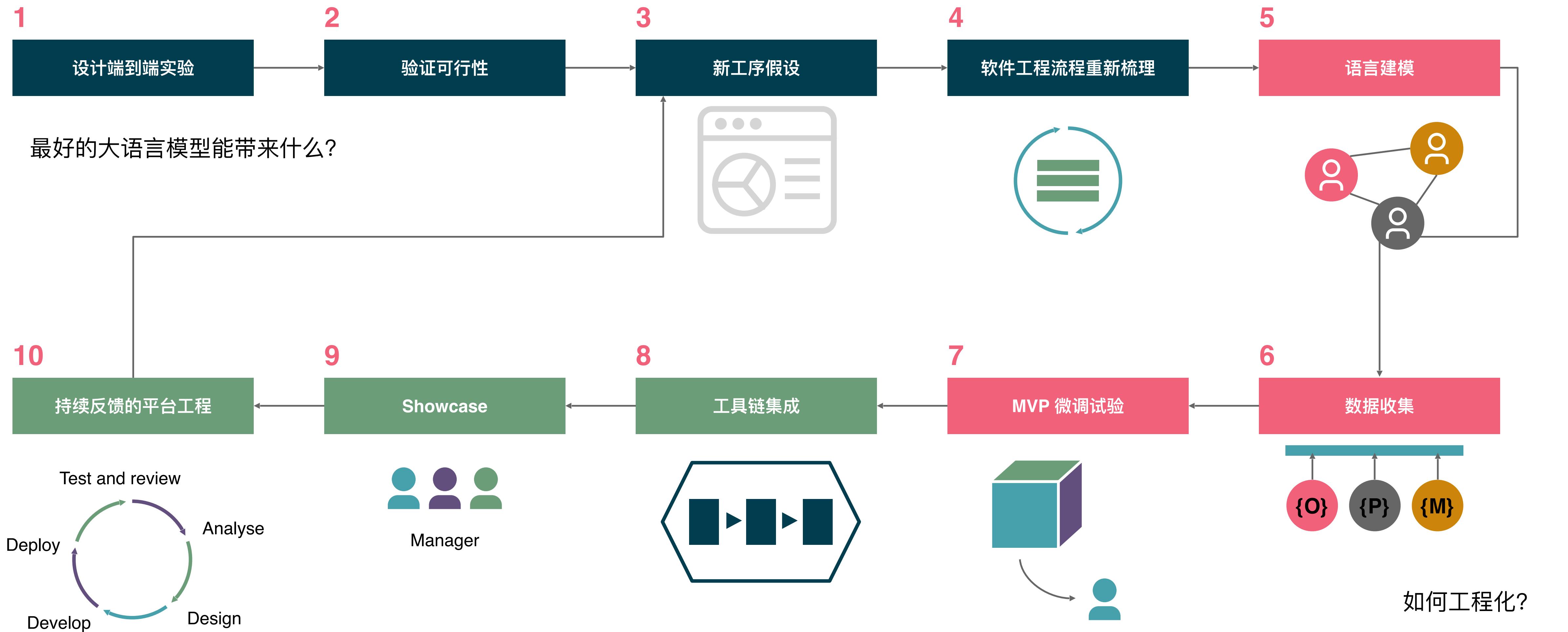
与已有工具集成，构建起自动调优闭环，持续收集数据和反馈





-----未来展望

总结一下

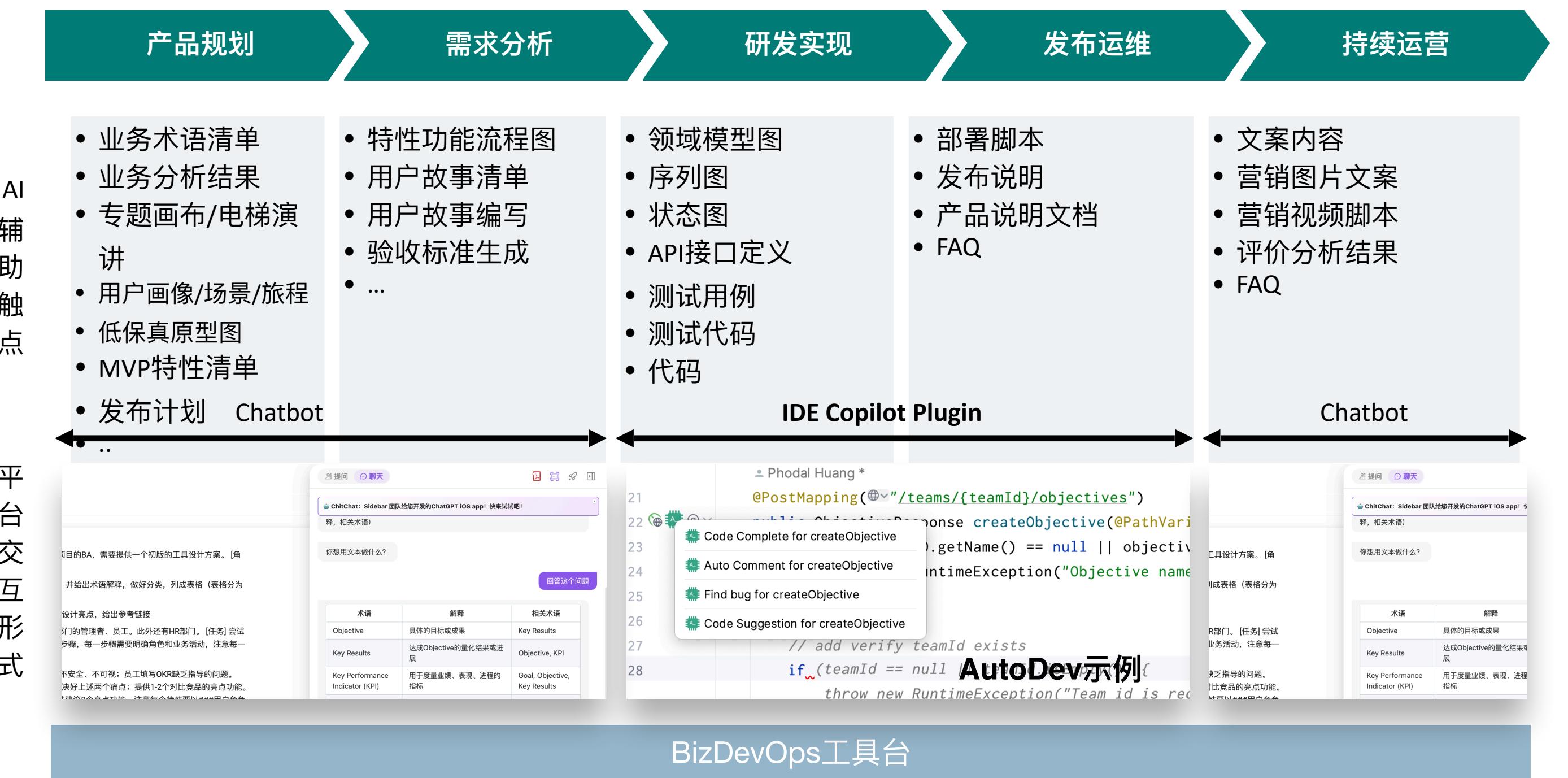


AI 无缝集成到研发工具链

探索如何把 AI 无缝集成到 BizDevOps 工具链中，评估技术可行性和实施难度



Hackathon

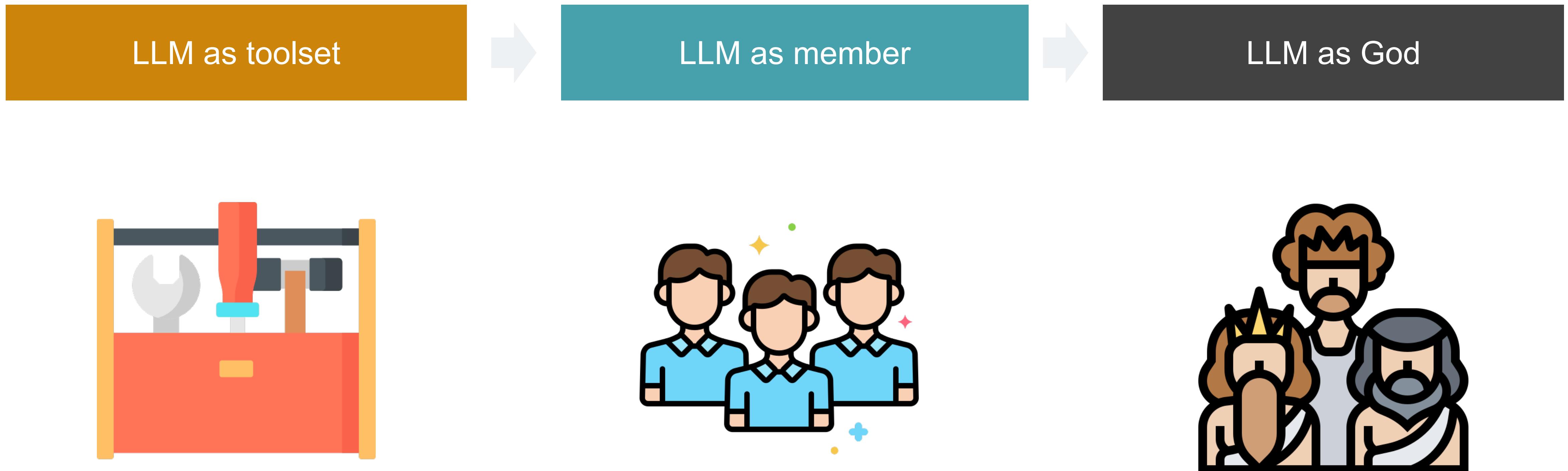


PS: 先验证公开可用的 LLM 带来的提升

<https://github.com/prompt-engineering/chat-chess>

LLM 对人才和组织结构的影响？

取决于我们如何看待 LLM



欢迎一起探讨：<https://github.com/unit-mesh>

THANKS

软件正在重新定义世界

Software Is Redefining The World