

Baidu Comate

# 人机协同的AI原生开发范式革命及落地

百度工程效能部

李杨

# 个人简介



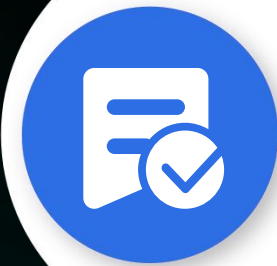
百度工程效能部  
李杨



百度资深研发工程师，文心快码（Baidu Comate）商业化负责人，云端开发平台（iCoding）的技术负责人。



百度一级专利发明人，名下国内外发明专利10余个，已获授权



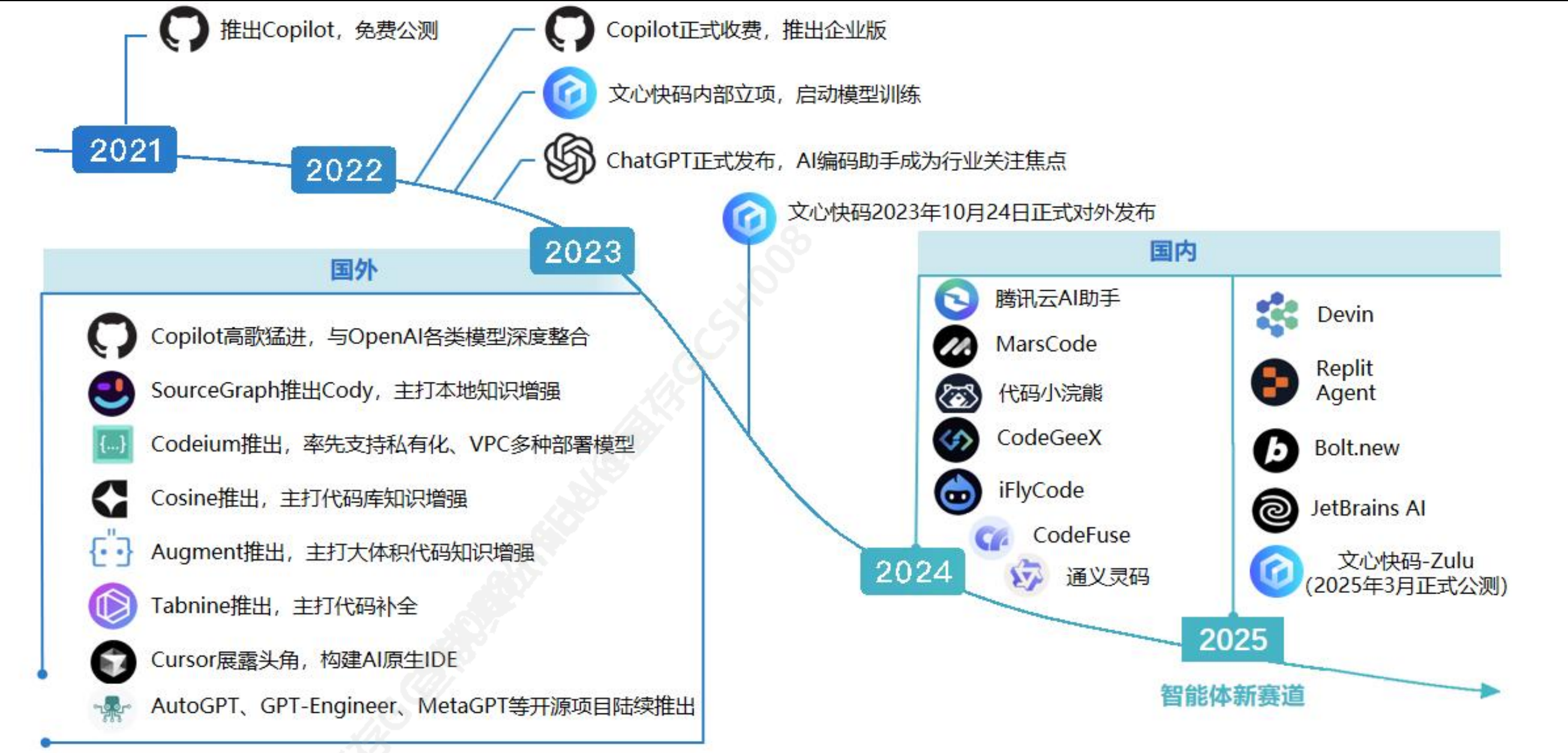
国家重点研发计划『基于人机结对编程与协同进化的智能敏捷开发云平台』技术骨干

# 目录

## CONTENTS

- 01 软件研发正在被重新定义—AI时代的新范式与新机会
- 02 从代码补全到研发智能体—企业AI Coding落地路径与最佳实践
- 03 打造自进化Coding Agent  
Benchmark、Feedback Loop与AgentOps能力飞轮

# 业内普遍看好大模型在研发领域的应用场景



大模型在ToB领域最成功的领域, 50+AI智能编码助手

|                                                                                     |                                                                                      |                                                                                       |                                                                                       |
|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
|  |  |  |  |
| 5亿美元<br>打破历史记录                                                                      | 4亿美元<br>年增长4倍                                                                        | 1亿美元<br>6个月10倍增长                                                                      | 7500万美元<br>7个月时间                                                                      |

用户愿意为AI Coding付费



Devin Ai  
The World's First Fully Autonomous AI Software Engineer

Thomas Dohmke @ashtom

3 Project Padawan: A sneak peek at our SWE agent and how we envision these types of agents will fit into the GitHub user experience. When it's released later this year, Project Padawan will allow you to directly assign issues to GitHub Copilot – and have it produce fully-tested pull requests. The force is strong in this one 🙌

(4/4)

gh repo rename myorg/newname results in myorg/myorg-newname #10633

Describe the bug  
gh version 5.12.0 (2024-11-27)  
2024-11-27T10:06:00Z  
I renamed my repo using 'gh repo rename myorg/newname' and the result was as expected.  
But the result being wrong.  
I renamed my repository to 'myorg/newname' and the result was as expected.  
But the result being wrong.

Steps to reproduce the behavior:  
1. gh repo rename myorg/newname  
2. gh repo rename myorg/newname  
3. gh repo rename myorg/newname  
4. gh repo rename myorg/newname  
5. gh repo rename myorg/newname

Let's take a look at one of the Issues.

1:06 AM · Feb 7, 2025 · 64K Views

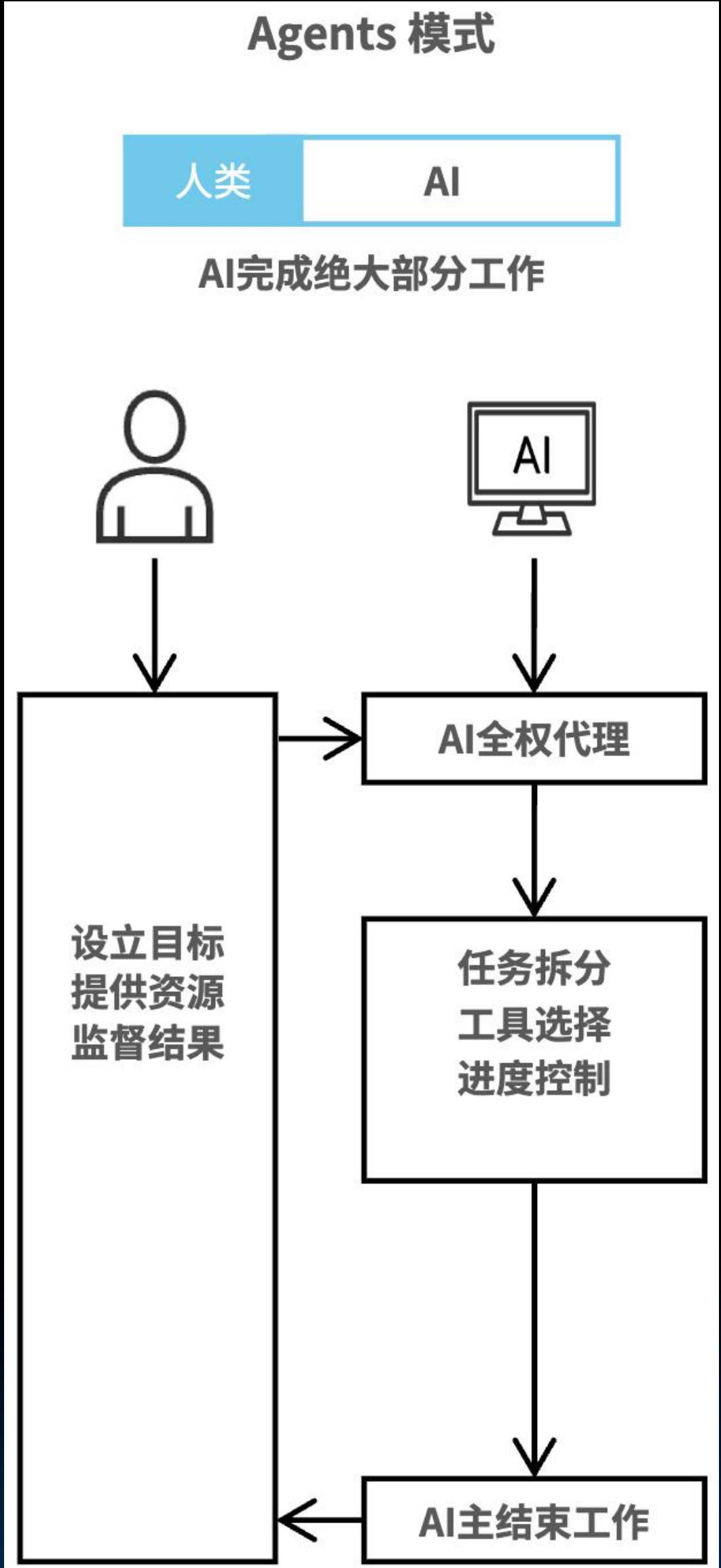
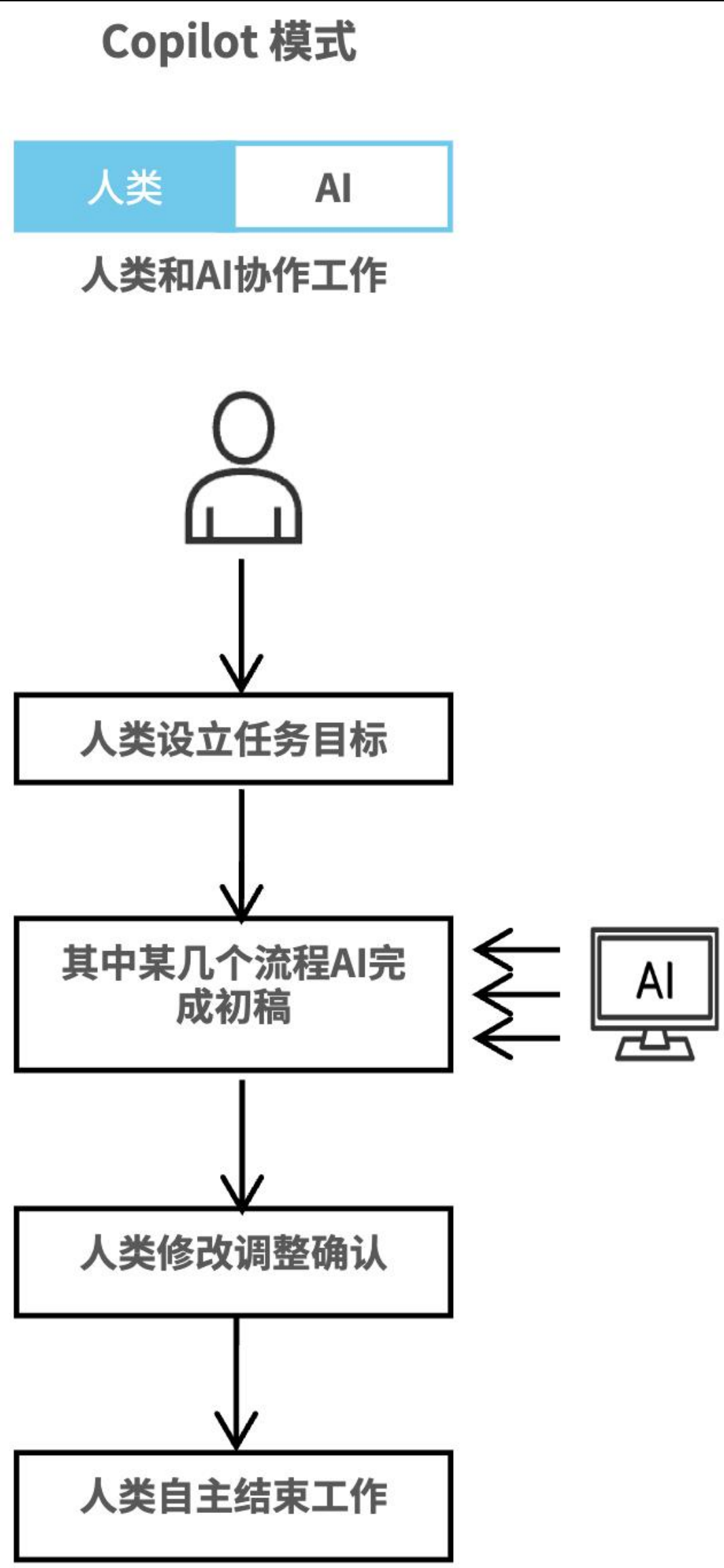
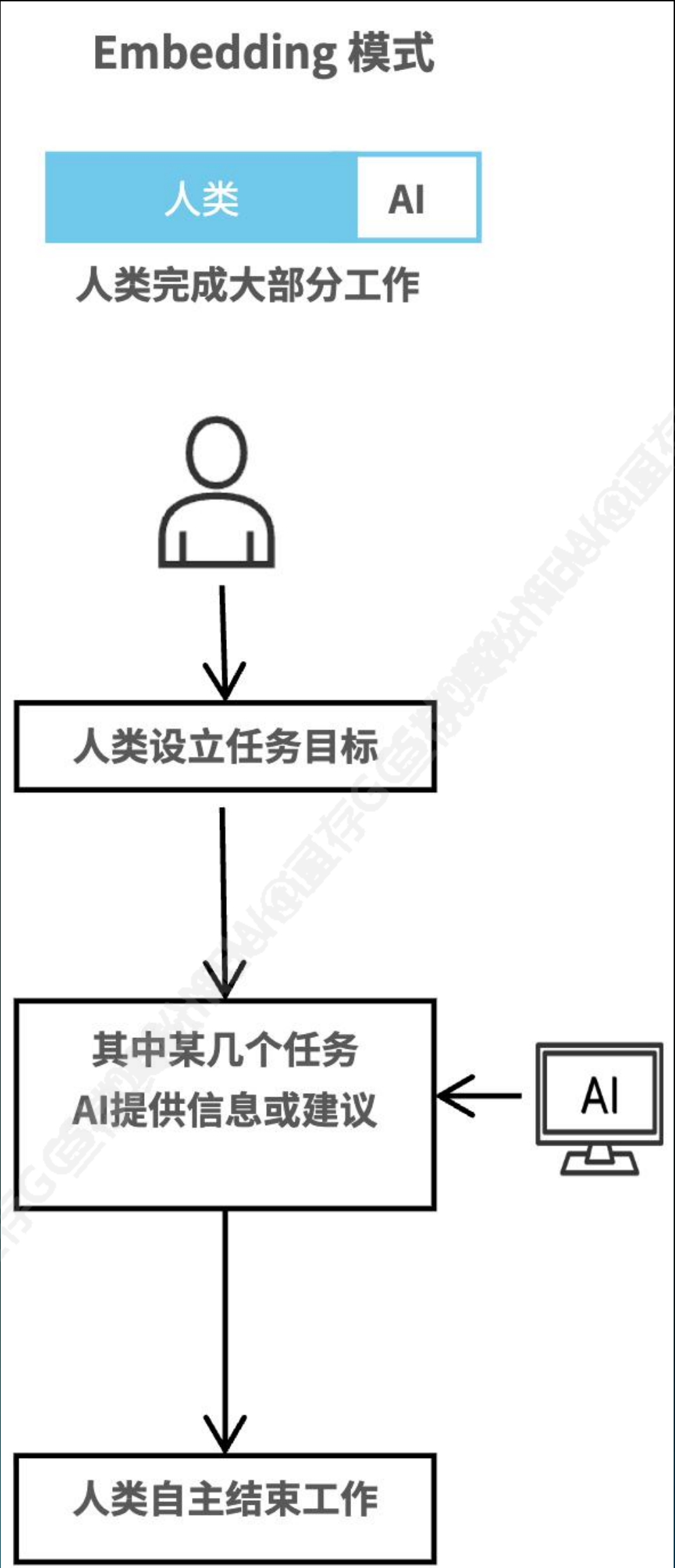
公众号 · 新智元

SWE Agent修复issue

命令行形态集成

研发智能体产品形态多样

# 人类与AI协同模式 - 从工具到员工的范式迁移



# 大模型为软件开发带来全方位的改变

AI是生产力工具，赋能成为更好个人 (Tool → Capability)

AI 是放大器，提升需求、设计、开发、测试、上线等环节的有效产出

AI prototype

Spec研发

TDD

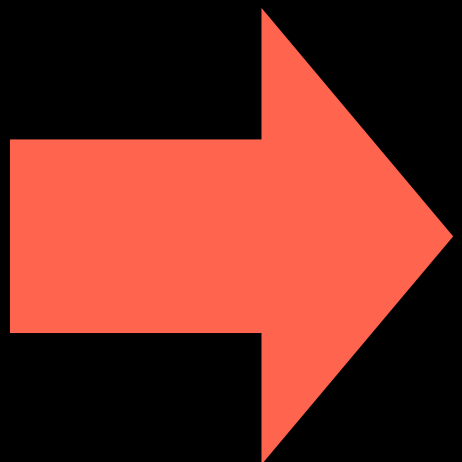
AI能改变生产关系，组织能力 (Role → Boundary Fusion)

AI 是倍增器，角色边界逐渐融合，跨角色沟通成本显著下降

流程重构

角色重定义

能力模型



## AI Native研发团队

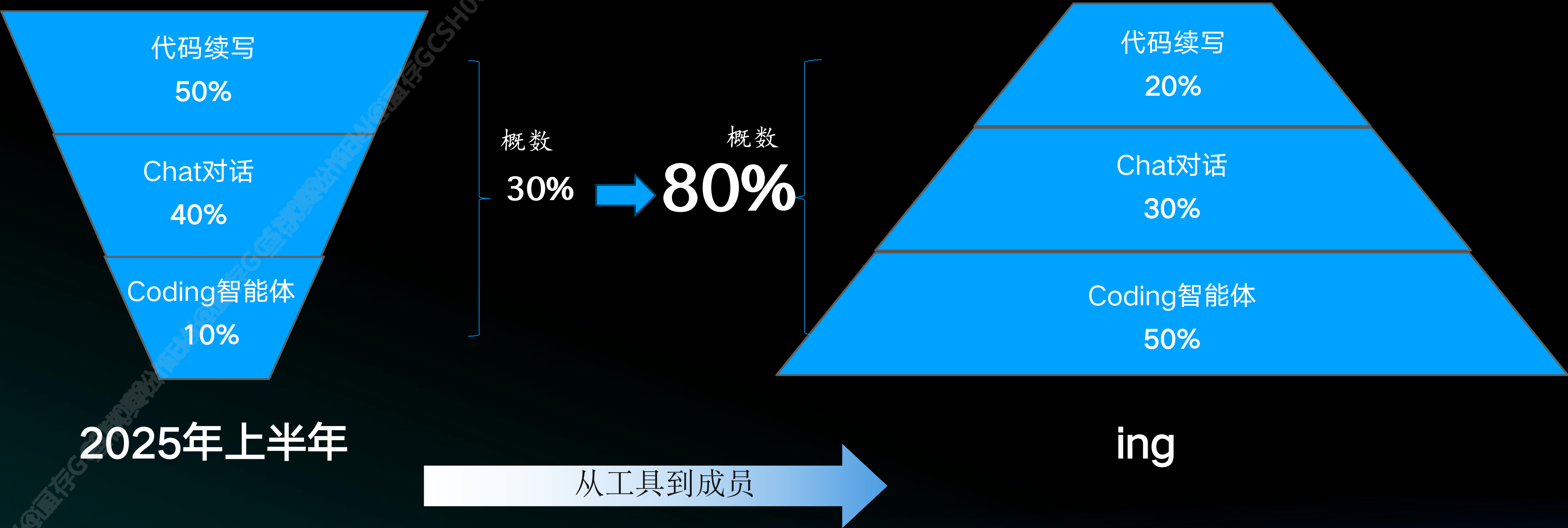
AI是主要生产力

人: 执行者 -> 设计者

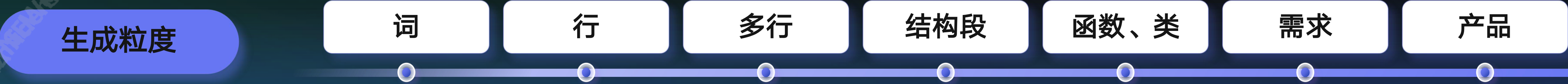
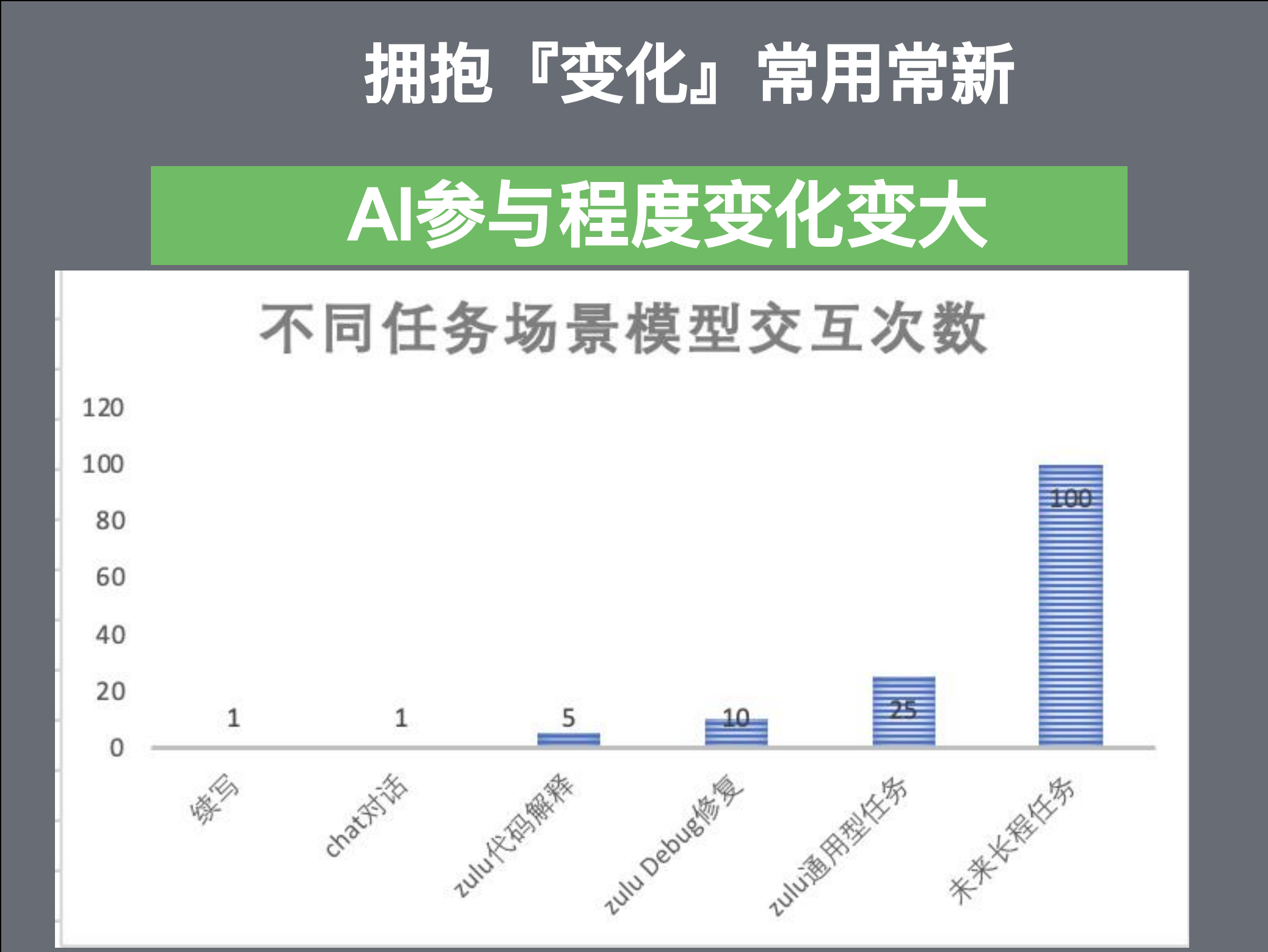
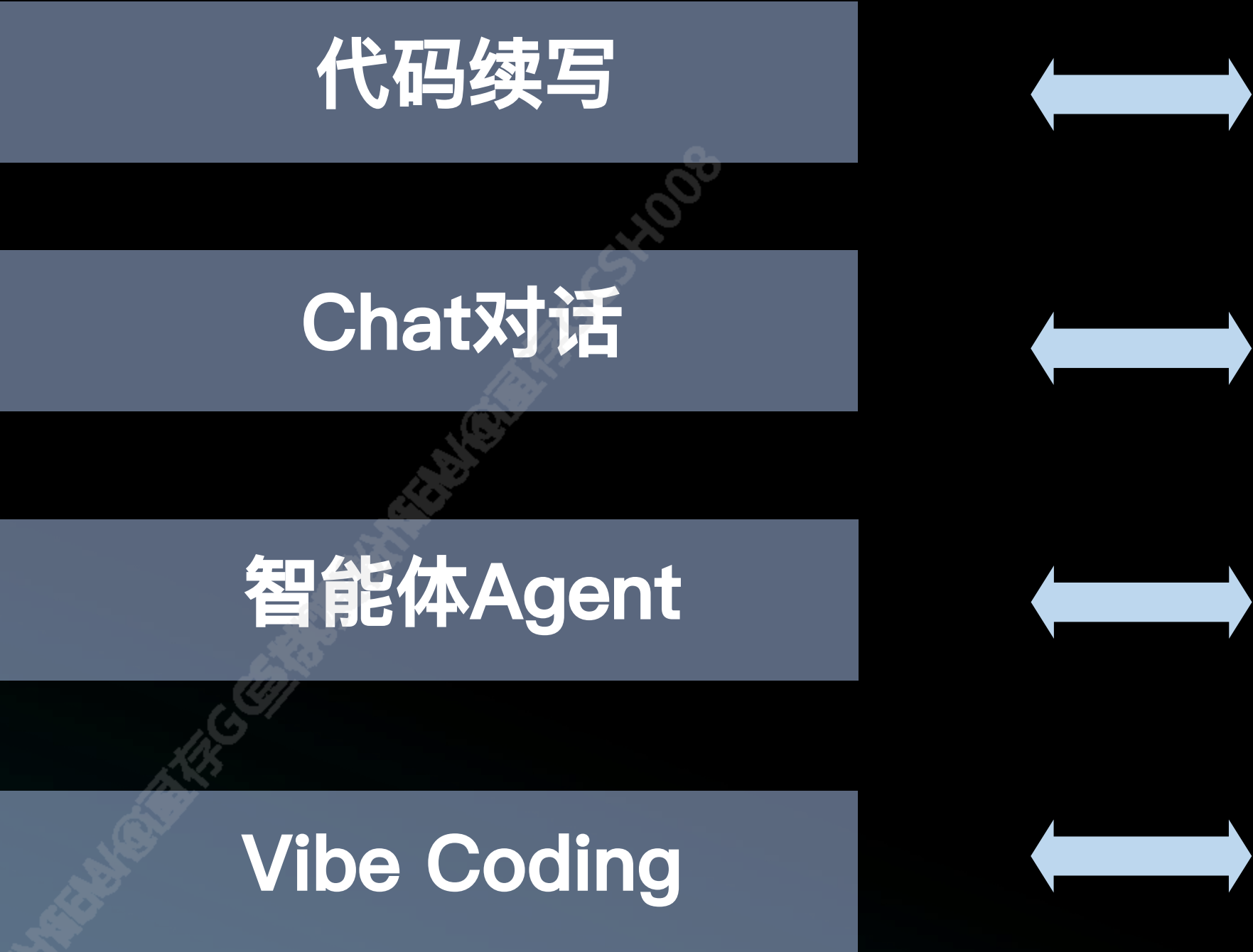
(半)全栈角色

变化1： AI代码生成占比持续提升，构成正悄然发生变化

AI代码生成占比数值还将继续提升，增速还将加快，达到临界值后团队将产生质变



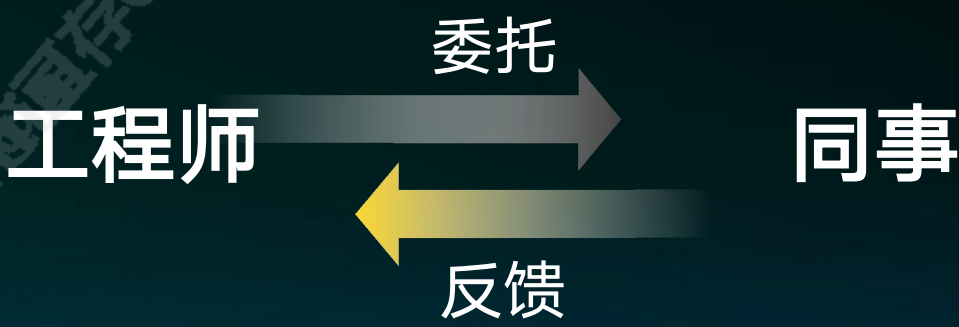
变化2：智能化进程多阶段并存，行业变化快，拥抱变化



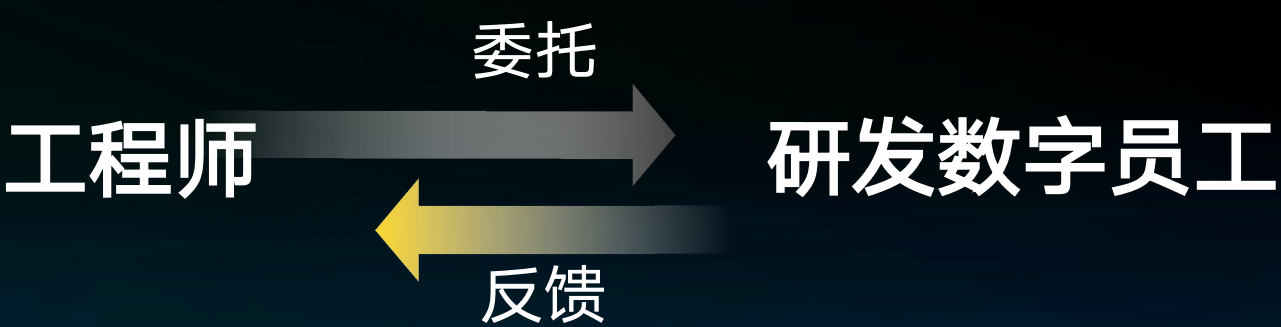
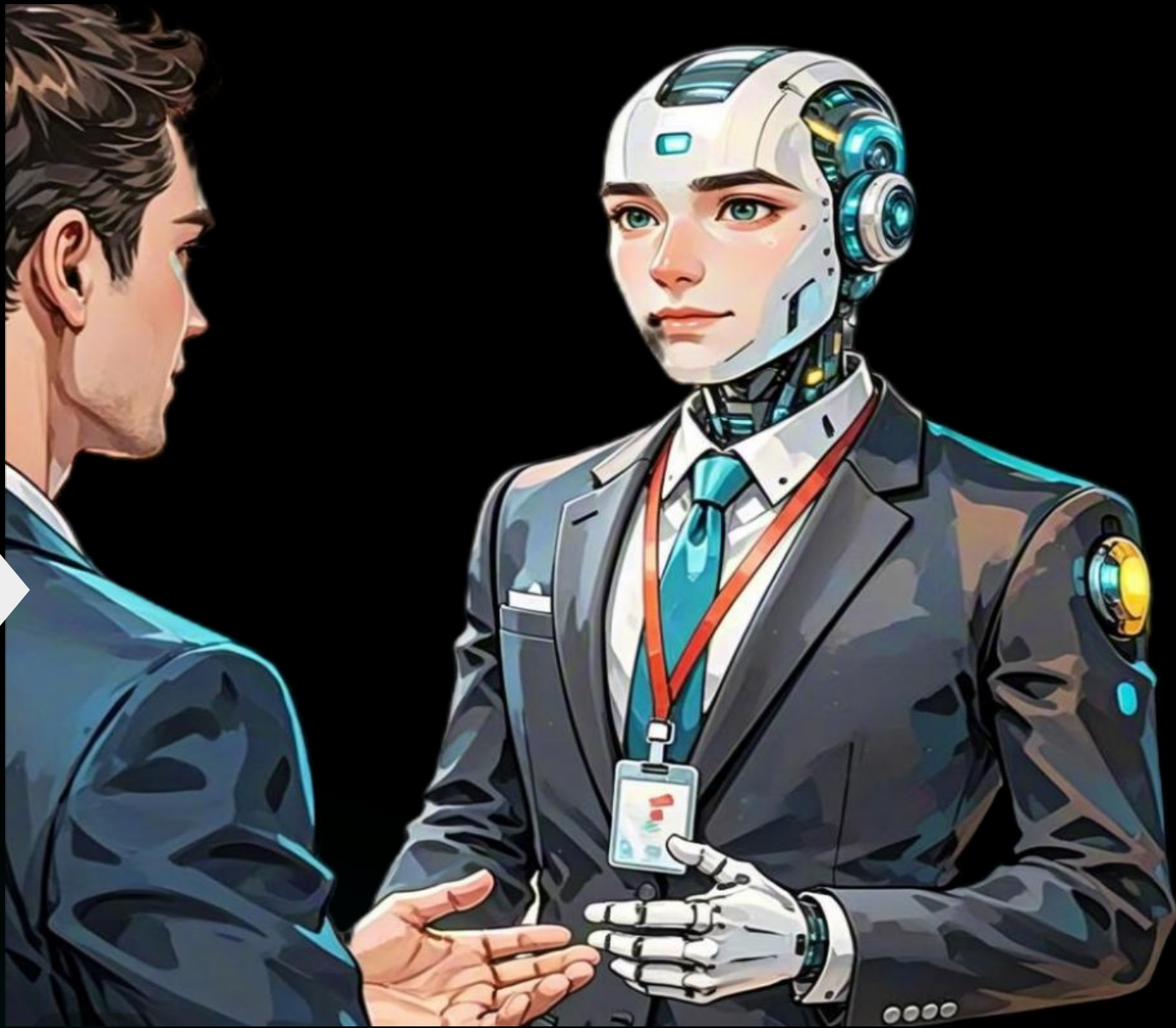
# 变化3：新的产品形态出现，更好实现软件研发场景的协同

最近爆火的Clawdbot，AI正从简单的聊天响应进化为能自主执行任务的智能体

## 工程师日常协同研发



## 人机协同研发新范式



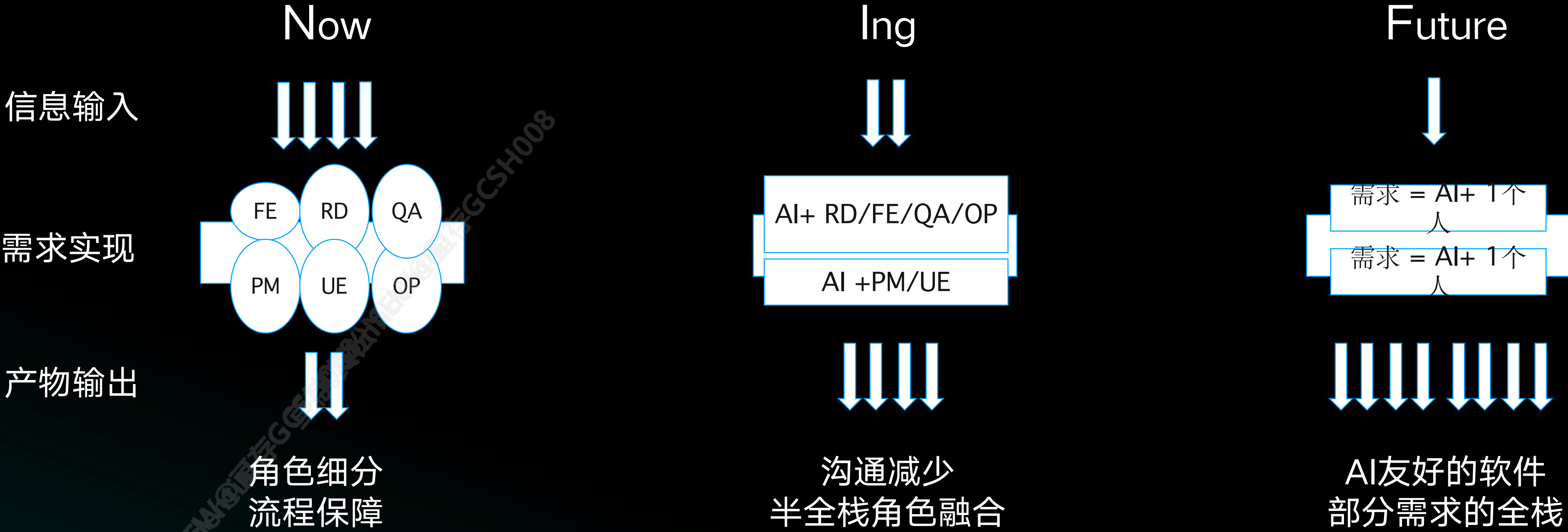
## 类人协作，自主决策并执行闭环

- 研发任务感知  
事件触发 | 主动感知
- 研发任务执行  
多环境联动 | 异步委托与授权
- 研发任务追溯  
研发数字员工可视化工作台

## 人机同权，研发全场景触达



变化4：角色越来越「独」，信息交换「管线」变少，组织变化团队变小



小团队能干大事情，创业公司开始，大公司逐步转变中  
YC 2025年数据显示，孵化项目中24%初创企业， 95%代码由AI生成，团队人均规模3.2人  
Cursor、GenSpark、Devin等公司人比较少，产出非常惊人

# 目录

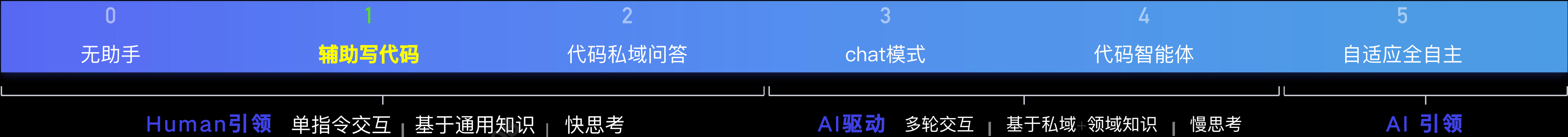
## CONTENTS

- 01 软件研发正在被重新定义—AI时代的新范式与新机会
- 02 从代码补全到研发智能体—企业AI Coding落地路径与最佳实践
- 03 打造自进化Coding Agent  
Benchmark、Feedback Loop与AgentOps能力飞轮

# 百度Comate平台产品全景图：定义AI研发应用新范式



# Copilot编码实践| AI加速写代码的速度

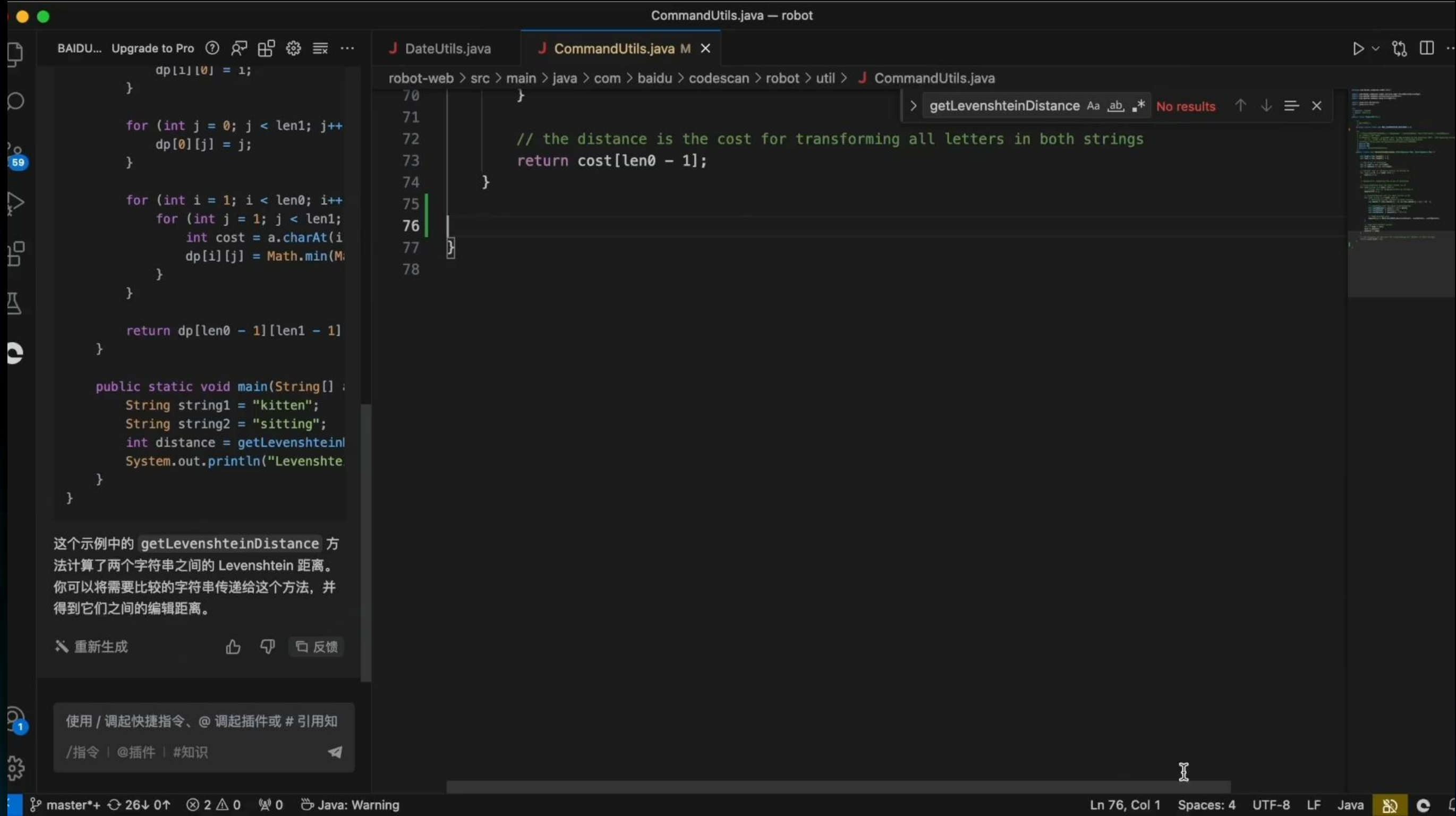


从"工具"走向"Pair Engineer"  
习惯的主动改变

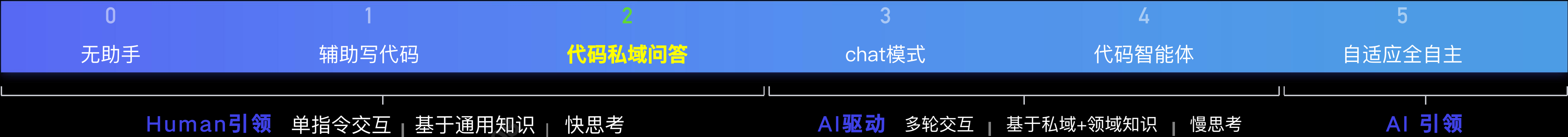
代码续写

自然语言转代码

落地实践：写代码之前先写注释



# Copilot编码实践 | 大模型结合私域知识，打造更懂你的智能编码助手



把你的私域知识告诉AI  
个性化、组织知识沉淀

提供上下文，让AI举一反三

参考本地代码生成新代码

基于文档的问答助手

接口文档生成业务代码

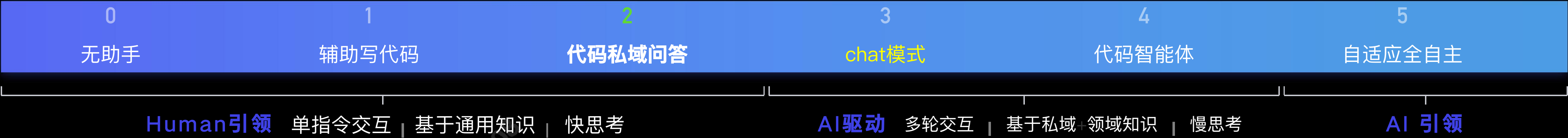
基于链接生成专属代码

1、文字描述为主，多模态不够精确

2、段落标题、目录清晰

```
1 # Linux上GPU预测部署示例
2
3 ## 1 C++预测部署示例
4
5 C++示例代码在[链接](https://github.com/PaddlePaddle/Paddle-Inference-Demo/tree/master/c%2B%2B/cuda_linux_demo)，下面从`流程解析`
6
7 ### 1.1 流程解析
8
9 #### 1.1.1 准备预测库
10
11 请参考[推理库下载文档](https://www.paddlepaddle.org.cn/documentation/docs/zh/develop/guides/05_inference_deployment/inference/)
12
13 ....省略部分内容
14
15 #### 1.1.4 设置Config
16
17 根据预测部署的实际情况，设置Config，用于后续创建Predictor。
18
19 Config默认是使用CPU预测，若要使用GPU预测，需要手动开启，设置运行的GPU卡号和分配的初始显存。可以设置开启TensorRT加速、开启IR优化、开启内存优化。
20
21 ```cpp
22 paddle_infer::Config config;
23 if (FLAGS_model_dir == "") {
24     config.SetModel(FLAGS_model_file, FLAGS_params_file); // Load combined model
25 } else {
26     config.SetModel(FLAGS_model_dir); // Load no-combined model
```

# Copilot编码实践 | 写代码之前问一问，当搜索引擎来用



在动手实现之前，先去问一下AI的实现

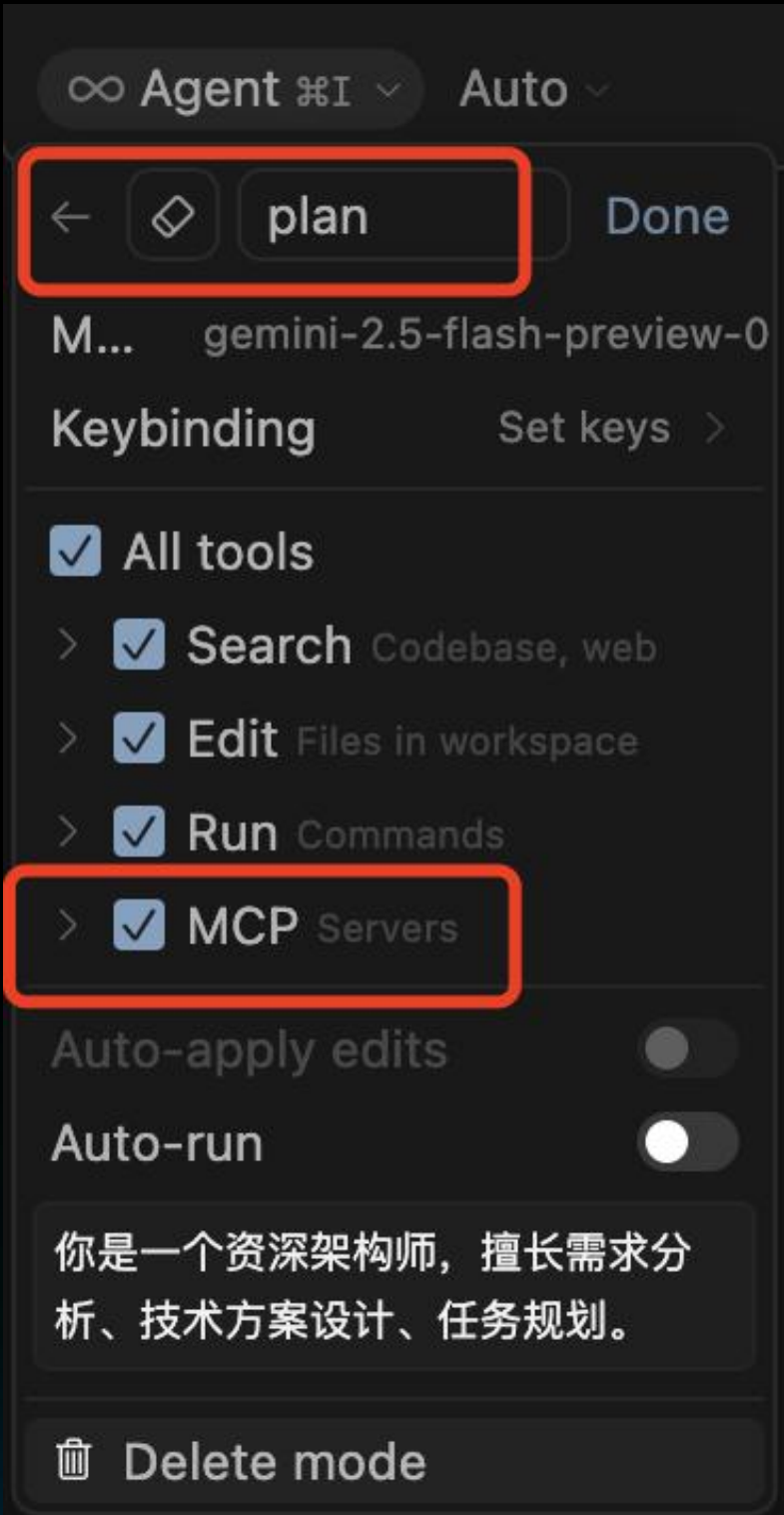
利用ASK/Plan模式做需求澄清  
厂内MCP工具让模型知道私域知识

实现逻辑不了解，问一问

没有思路的时候，问一问

忘记知识的时候，问一问

代码出错的时候，问一问

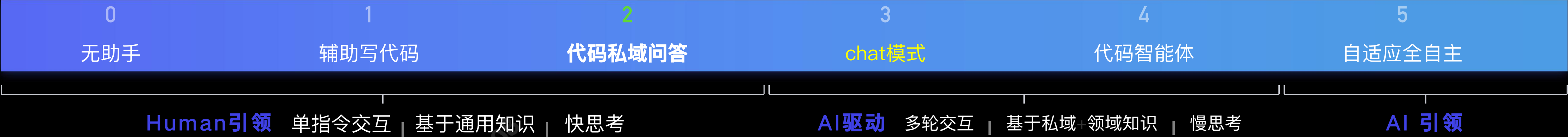


Plan模式



MCP工具

# Agent编码实践 | Coding Agent推进Rules编写，AI在约束下发挥



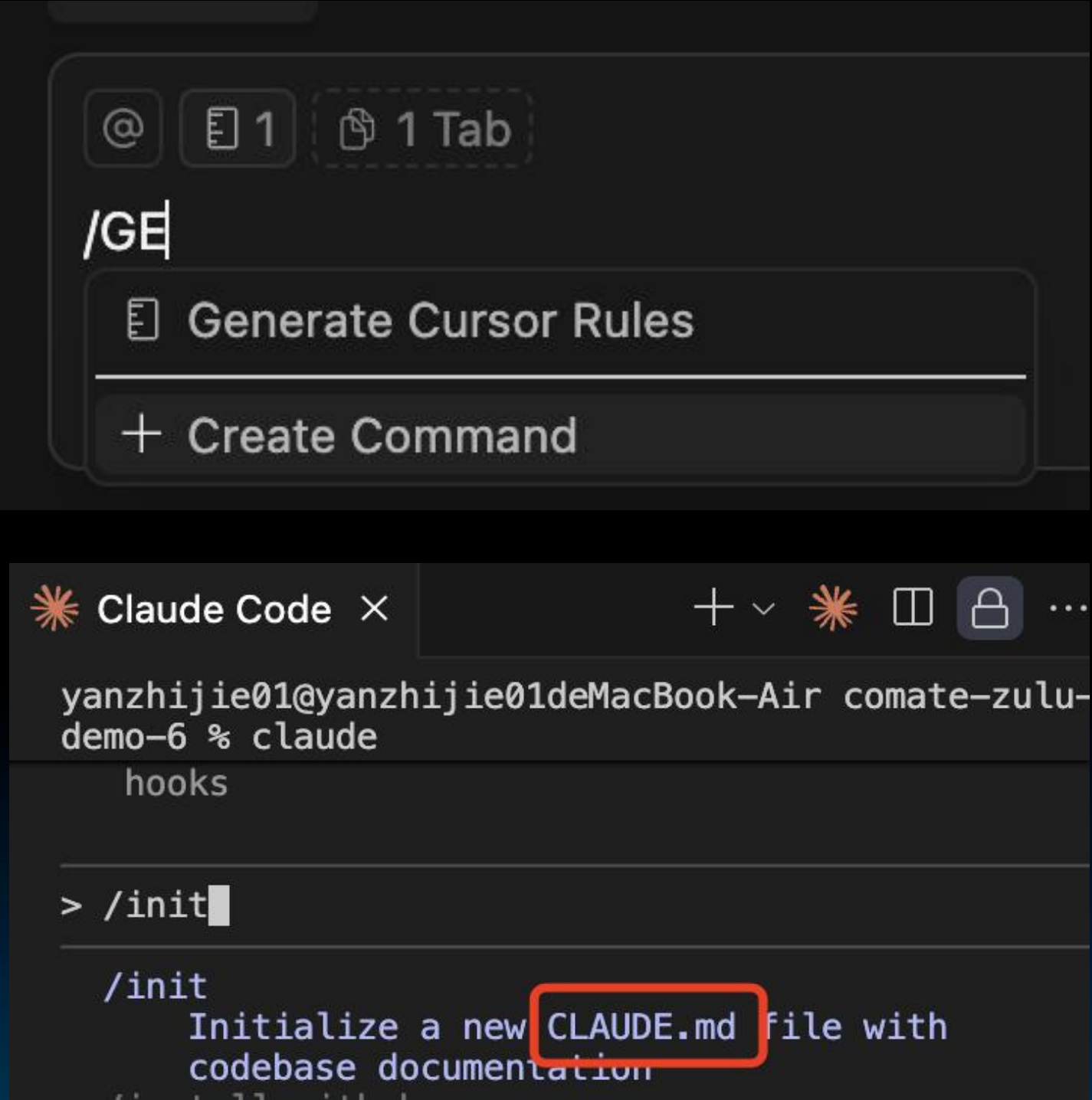
Coding Agent很强大，但很健忘，在实际代码库落地需要Rules的约束和上下文提供

## Rules可以分层

- 技术方案编写规范
- 项目架构（目录、技术栈、业务）
- 代码开发规范（命名、注释、格式、日志、错误码）
- 常见功能开发流程模板和示例
- 代码提交与评审规范

```
plan
├── p-design-tech-solution.mdc // 技术方案编写规范、模板
└── develop
    ├── d-gen-data-center-crud-from-sql.mdc // d-gen 开头的是总结的某类功能的代码生成模板、开发流程
    ├── d-gen-cronjob.mdc // cronjob 开发流程、代码模板;
    ├── d-gen-api.mdc // http api 开发流程、代码模板;
    └── d-test-unit-test.mdc // 单元测试规范、模板
└── review
    ├── r-code-review.mdc // 代码评审规范
    ├── r-git-commit.mdc // commit message 规范
    ├── r-merge-request.mdc // 创建远程合并请求流程
    ├── r-summary-sop.mdc // 总结本次代码修改的标准流程生成 p-gen-xx 文件，供下次开发使用
    ├── g-code-spec.mdc // 项目代码规范
    ├── g-project-hotkey.mdc // 项目中的 hotkey 比如`提示语中出现 cr 则参考 xxx.rules 执行`
    ├── g-project-structure.mdc // 项目结构、技术架构、业务架构...
    └── g-system.mdc // 项目 AI 聊天规范，比如强调多轮对话、引导补全...
```

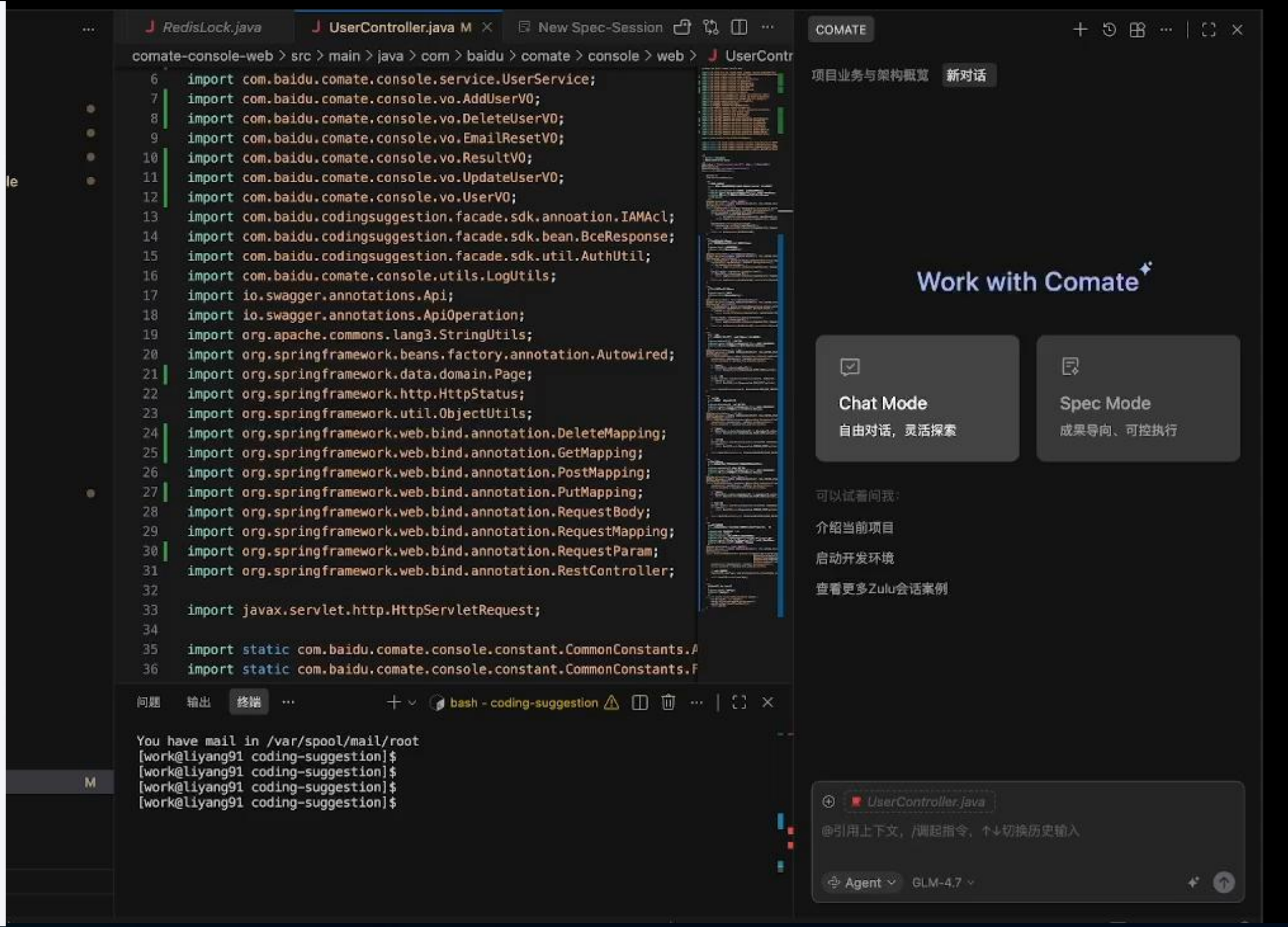
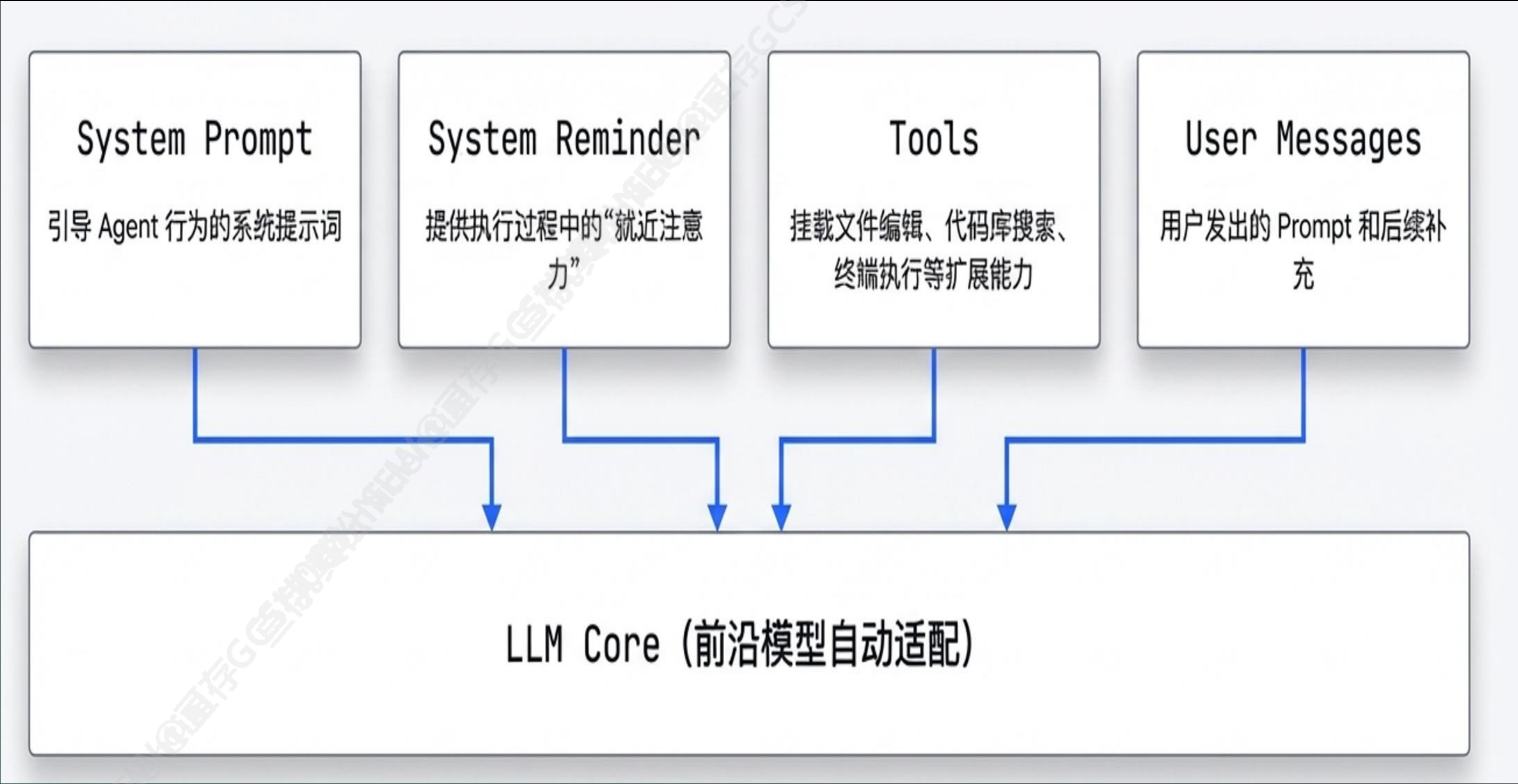
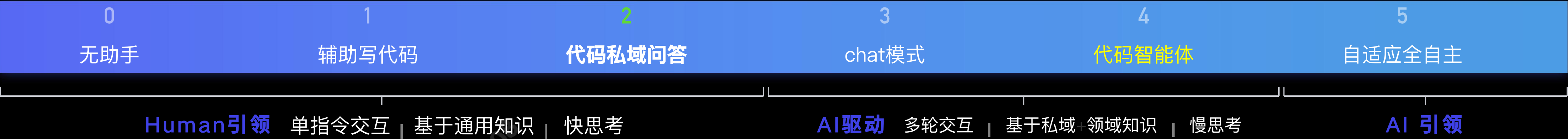
## 善用工具辅助生产



无法掌控的感觉是无法接受  
Coding Agent越强大  
越应该约束

不能要求人人都成为专家  
驾驭AI是有一定门槛  
经验流程要固化

# Coding Agent编码任务端到端生成



Coding Agent 0Day 接入前沿模型，开发者只需要关注业务逻辑

# 工程能力越发重要， 软件研发流程的适配改变



Spec driven 很有效  
从写代码=>写文档  
工程能力是保障

## 知识工程越来越重要

- 1. Prompt工程变成了必要技能
- 2. 需求的Spec也是知识，充分利用AI完善文档，形成闭环
- 3. 充分利用文件系统便于Agent获得知识

### 项目结构

```
...  
packages/  
├─ core-service/  
├─ image-editor/  
└─ example/  
...
```

### 文档结构

```
design/  
├─ core-service/  
├─ image-editor/  
└─ example/
```

## 测试能力是关键因素

- 1. 大模型是概率模型，@1pass不可能
- 2. 确定性的测试能力对生成代码的验证是效果提升的关键因素，AI可以多试几次
- 3. TDD有点太理想，但测试能力是跨过80分危机的良药

### /test

Included in your Q Developer Agent subscription

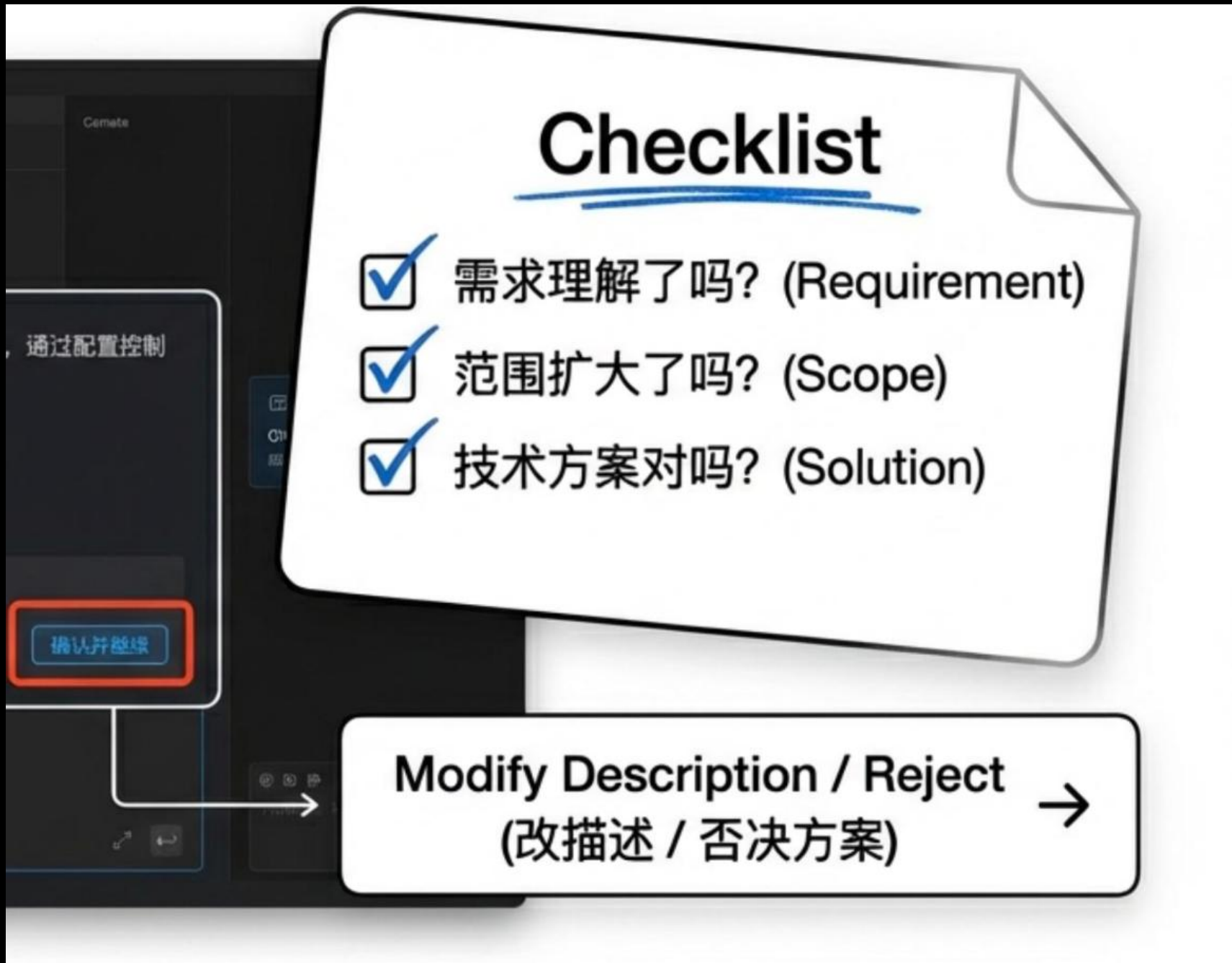
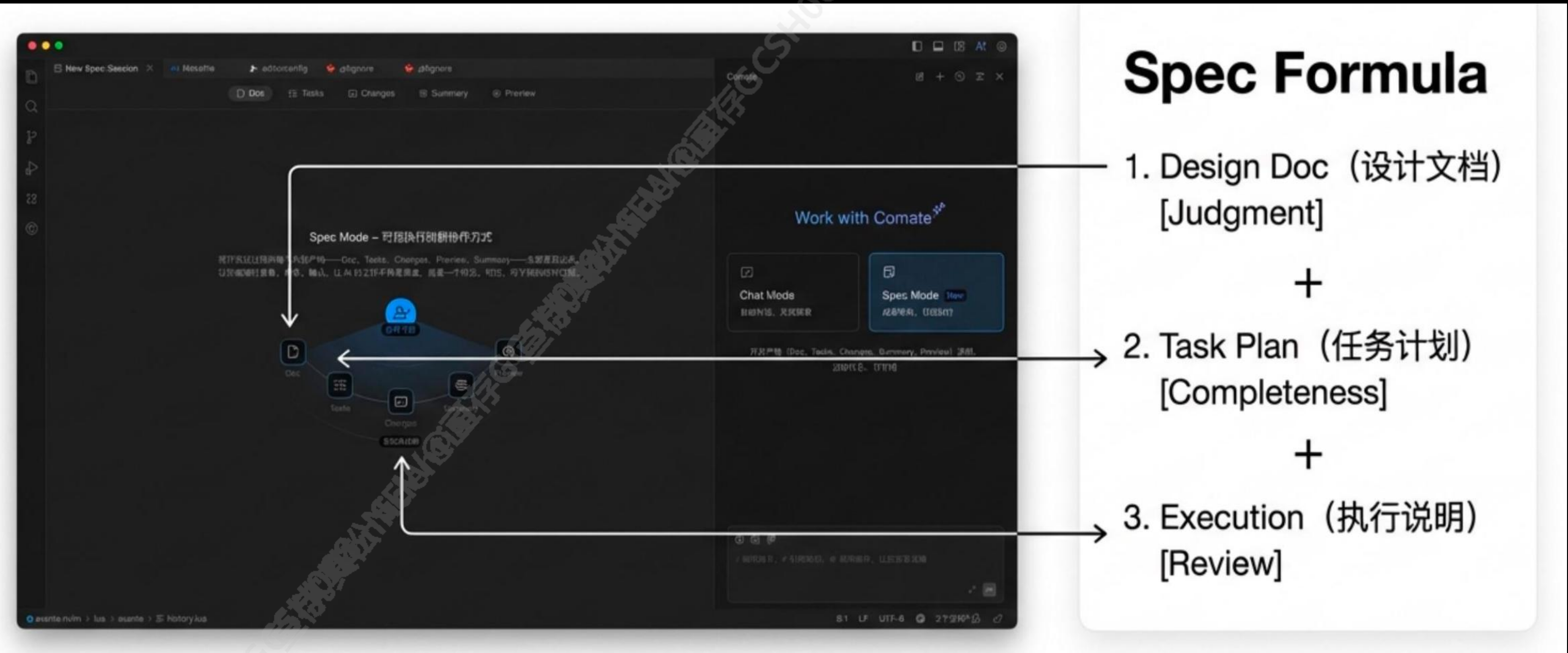
I can generate unit tests for your active file.

After you select the functions or methods I should focus on, I will:

1. Generate unit tests
2. Place them into relevant test file

To learn more, check out our user guide.

# Agent编码实践|工程能力越发重要， 软件研发流程的适配改变

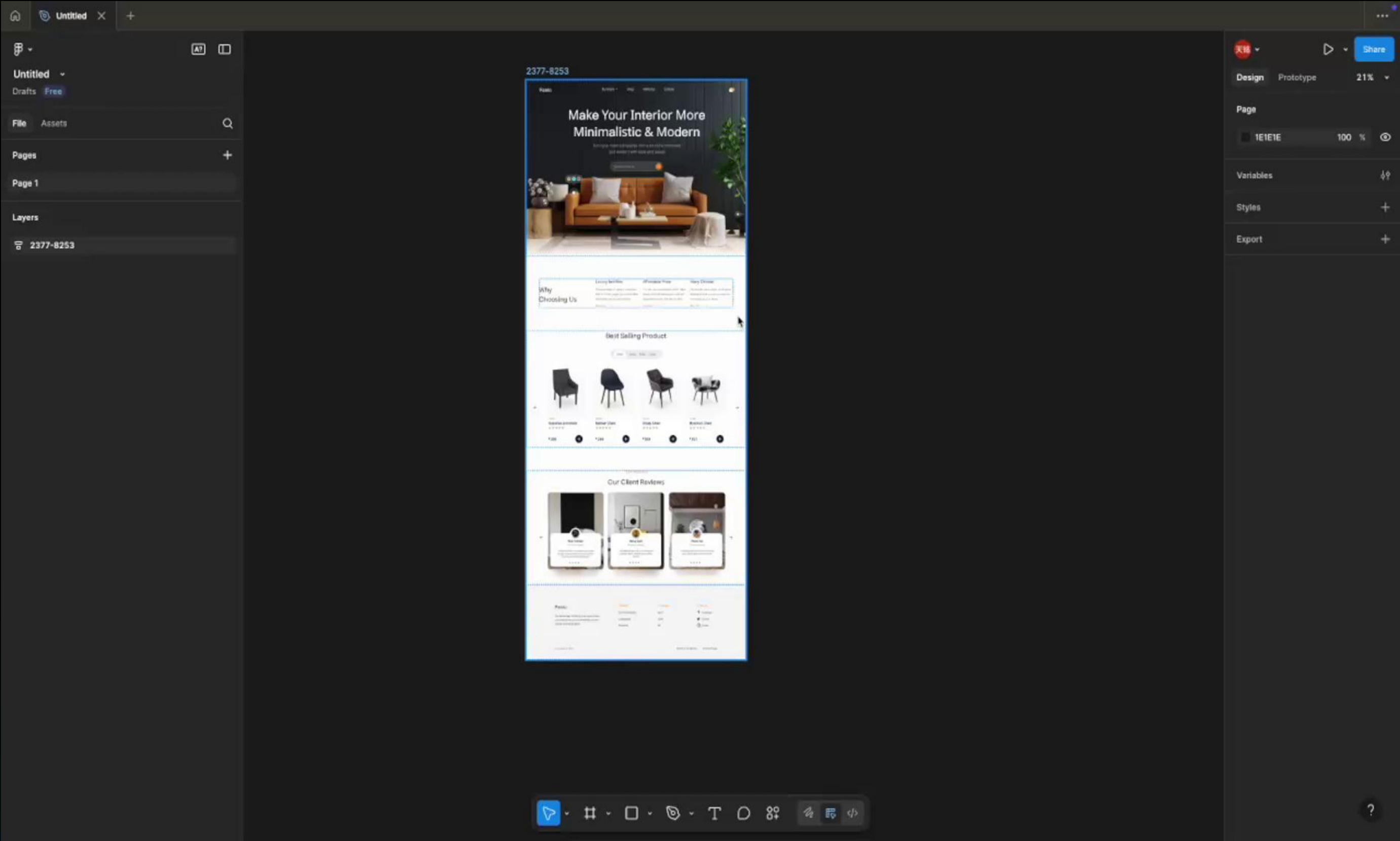


Spec: 让AI写之前先把打算怎么做写出来

# Agent设计实践 | F2C应用落地广泛，先标准化再智能化

充分利用  
大模型  
现有能力

先标准化  
再智能化



# Agent实践同DevOps平台的结合|编译修复从IDE、云端到CR，多形态

研发伙伴

首页

帝霸哥

文思匠

单测侠

智测师

添加数字员工

帝霸哥

Debug工程师

一键修复

感谢您的信任，我要开始工作啦。我将大致按照以下4个步骤来执行，当然具体流程可能存在差异：

搭建独立修复环境

分析错误日志

定位问题并修复

执行编译验证修复结果

这个过程大概需要2-10分钟，执行完成我会通知你呢。

查看结果

我已经修复成功啦！您可以查看修复结果，如果没问题可以直接采纳，我会帮您提交iCode CR

查看详情

正在工作中，请不要打扰我哦

baidu/bcloud/bussiness-Chan...

已采纳

概述

任务过程

结果预览

Base

1

environment:

image: DECK\_STD\_CENTOS7

tools:

- go: 1.18.latest

build:

command: make -f Makefile

cache:

# http://buildcloud.baidu.com/bcloud/9-bcloud\_subcmd#9.15-cache

environment:

image: DECK\_STD\_CENTOS7

tools:

- go: 1.22.latest

build:

command: make -f Makefile

cache:

# http://buildcloud.baidu.com/bcloud/9-bcloud\_subcmd#9.15-cache

自动修改文件完成

执行编译自动验证完成

自动提交CR完成

# 构建“工具+知识+Skills+Agents”的开放研发新生态



# MCP+Skills 扩展研发全流程



MCP Server

已安装 MCP 市场

配置

在市场中搜索 MCP Server

Memory

增强 Agent 持久记忆。

9 Tools

create\_entities

create\_relations

添加

Sequential-Thinking

一种结构化的问题解决工具，可以对复杂的推理任务进行逐步分析、思维修正和分支逻辑。

1 Tools

sequential\_thinking

添加

GitHub

全面的 GitHub API 集成，支持存储库管理、文件操作、问题跟踪、拉取请求处理以及跨代码、问题和用户的高级搜索功能。

26 Tools

create\_or\_update\_file

push\_files

添加

Figma

提供对 Figma 设计数据的访问权限，通过获取和简化 Figma API 响应以获得最佳上下文，实现从设计文件生成准确的代码。

添加

OneTool

效率工具

创建 Skill

全部 我的

搜索 Skill 名称或描述...

筛选 全部 效率工具 开发编程 数据分析 DevOps 测试安全 内容创作 调查研究 生活方式 其他场景

图流程可视化

图流程可视化 代码库 <https://console.cloud.baidu-int.com/devops/icode/repos/baidu/ps-...>

开发编程

7 钟仁杰 更新于 2026-02-26 15:46:58

Golang性能分析

golang代码如果需要进行性能分析，可以使用该工具分析，skill会给出合理的建议供开发者使用 提供8大领域的41项...

开发编程

81 毛康 更新于 2026-02-12 18:45:28

百度VectorDB开发工具

该仓库是百度向量数据库接口技能库，提供建表、写入、检索、索引与报错排查能力。

开发编程 数据分析

66 朱晨畅 更新于 2026-02-12 13:17:53

根据PRD生成技术文档

当用户提供 PRD 文档并要求编写技术设计、架构设计或实现方案时使用。当用户要求将产品需求转换为技术规格说...

开发编程

188 徐传鹏 更新于 2026-02-13 16:08:51

GP2根据表生成api 接口

基于sql生成api 接口

开发编程

17 祝国杰 更新于 2026-02-11 17:15:34

代码鼓励师

这项技能为程序员在各种场景下提供幽默且鼓舞人心的支持。当用户对编码感到沮丧、压力或自我怀疑时，当遇到...

开发编程 内容创作

183 郭瑞彪 更新于 2026-02-13 16:13:40

电商直播日志精简

实验代码清理

GoCodeReviewer

# SubAgent: 适配研发规范和业务逻辑的Workflow

创建Agent

编辑Agent

## 什么是 Subagent?

Subagent是 Comate 主Agent可以灵活委派任务的专业化Agent助手。它通过在主对话外拆解复杂任务，助力主Agent实现多线协同。

每个Subagent在独立的上下文窗口中运行特定类型的工作，并将结果精准返回给主Agent，确保主对话上下文的绝对纯净。



### 上下文隔离

每个子代理拥有自己的上下文窗口。耗时较长的调研或探索任务不会占用主对话的上下文空间。

### 并行执行

可同时启动多个子代理。能够在代码库的不同部分并行工作，无需等待低效的顺序执行完成。

### 专门领域能力

通过自定义提示词、工具访问权限和特定模型，为子代理精准配置专属特定领域的任务能力。

### 灵活可复用

支持开发者自定义配置子代理，实现一次构建，即可在多个项目之间无缝复用。

Subagents

个人

项目

创建 Subagent

+

API\_Controller\_梳理器

梳理项目中的Controller，形成完整的API列表文档

+

API梳理智能体

负责梳理项目中的API接口，生成完整API文档的智能体

项目

创建 Subagent

选择 Subagent 类型

个人 Subagent  
仅个人可用

项目 Subagent  
使用该项目的所有人都能用

输入描述信息 (必填)

0/999

取消

AI创建

Comate支持项目及个人  
Subagent自定义

# 如何衡量AI Coding对组织效率的提高

衡量组织效率，指引组织落地

采纳率、AI代码生成占比的实时跟踪

按照组织、个人查看详细内容

查看各个语言、功能使用情况



采纳率 ①

31.40%

环比: 15.52% ▾

采纳率行数 ①

4.4K

环比: 37.12% ▾

推荐行数

13.9K

代码生成占比 ①

52.48%

环比: 69.56% ▲

变更行数

8.3K

使用用户数 ①

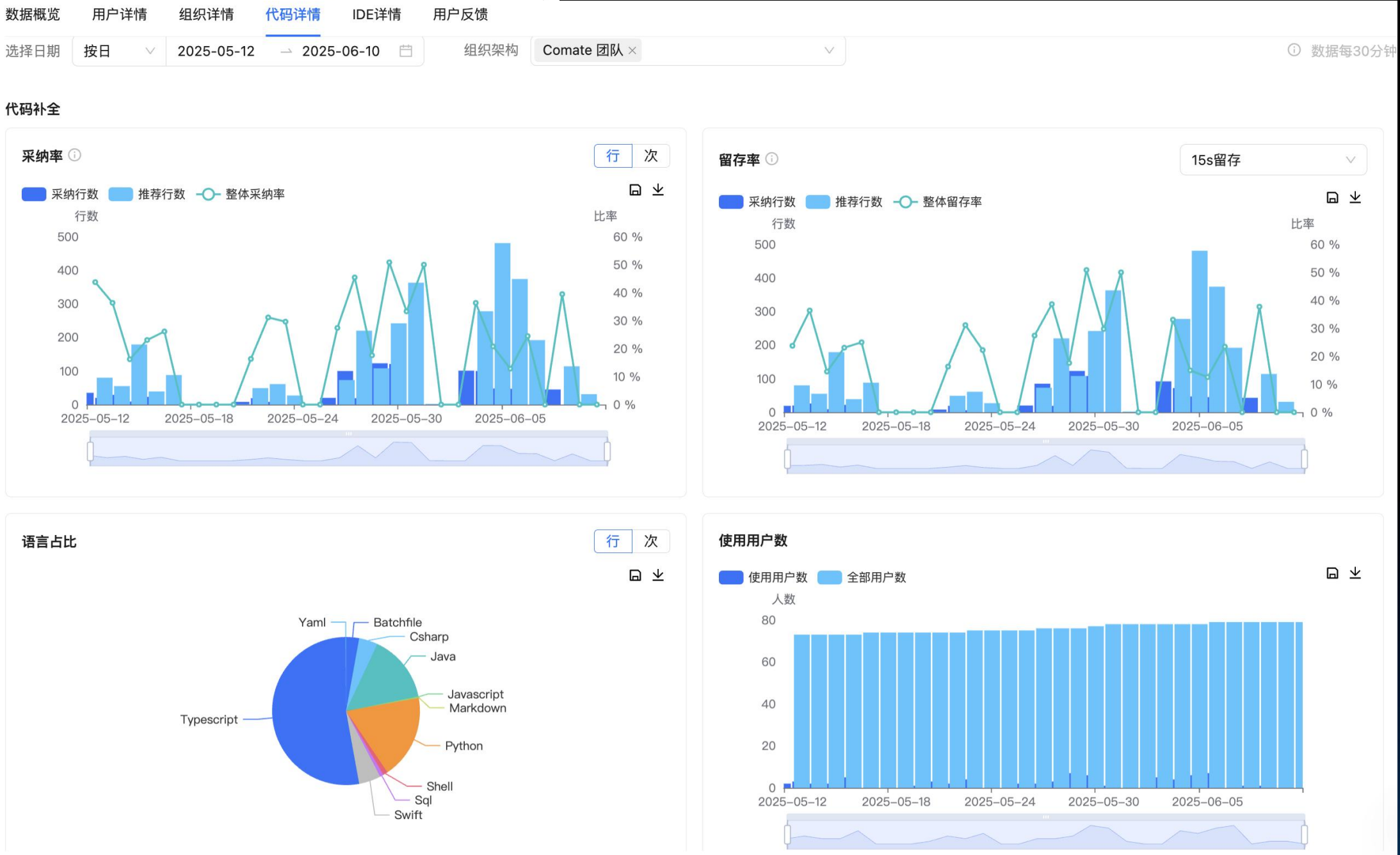
13 / 79

环比: 35.00% ▾

组织明细

⬇

| 子部门名称 | 采纳率 ▾  | 代码补全采纳率 ▾ | 智能回答采纳率 ▾ | 采纳行 ▾ | 推荐行 ▾ | 代码生成占比 ▾ | 变更行  | 使用用户数 | 采纳用户数 | 未使用用户数 | 总使用 |
|-------|--------|-----------|-----------|-------|-------|----------|------|-------|-------|--------|-----|
| 测试一部  | 3.47%  | 0%        | 3.47%     | 263   | 7.6K  | 95.99%   | 274  | 2     | 1     | 4      | 542 |
| 研发一部  | 79.8%  | 25.97%    | 93.45%    | 2.4K  | 3.0K  | 45.8%    | 5.3K | 4     | 3     | 12     | 440 |
| 研发二部  | 18.15% | 18.67%    | 16.13%    | 55    | 303   | 7.77%    | 708  | 1     | 1     | 2      | 197 |
| 算法一部  | 0%     | 0%        | 0%        | 0     | 0     | 0%       | 0    | 0     | 0     | 0      | 0   |
| 运维一部  | 0%     | 0%        | 0%        | 0     | 0     | 0%       | 0    | 0     | 0     | 2      | 0   |
| 运维二部  | 0%     | 0%        | 0%        | 0     | 0     | 0%       | 0    | 0     | 0     | 0      | 0   |



# 目录

## CONTENTS

- 01 软件研发正在被重新定义—AI时代的新范式与新机会
- 02 从代码补全到研发智能体—企业AI Coding落地路径与最佳实践
- 03 打造自进化Coding Agent  
Benchmark、Feedback Loop与AgentOps能力飞轮

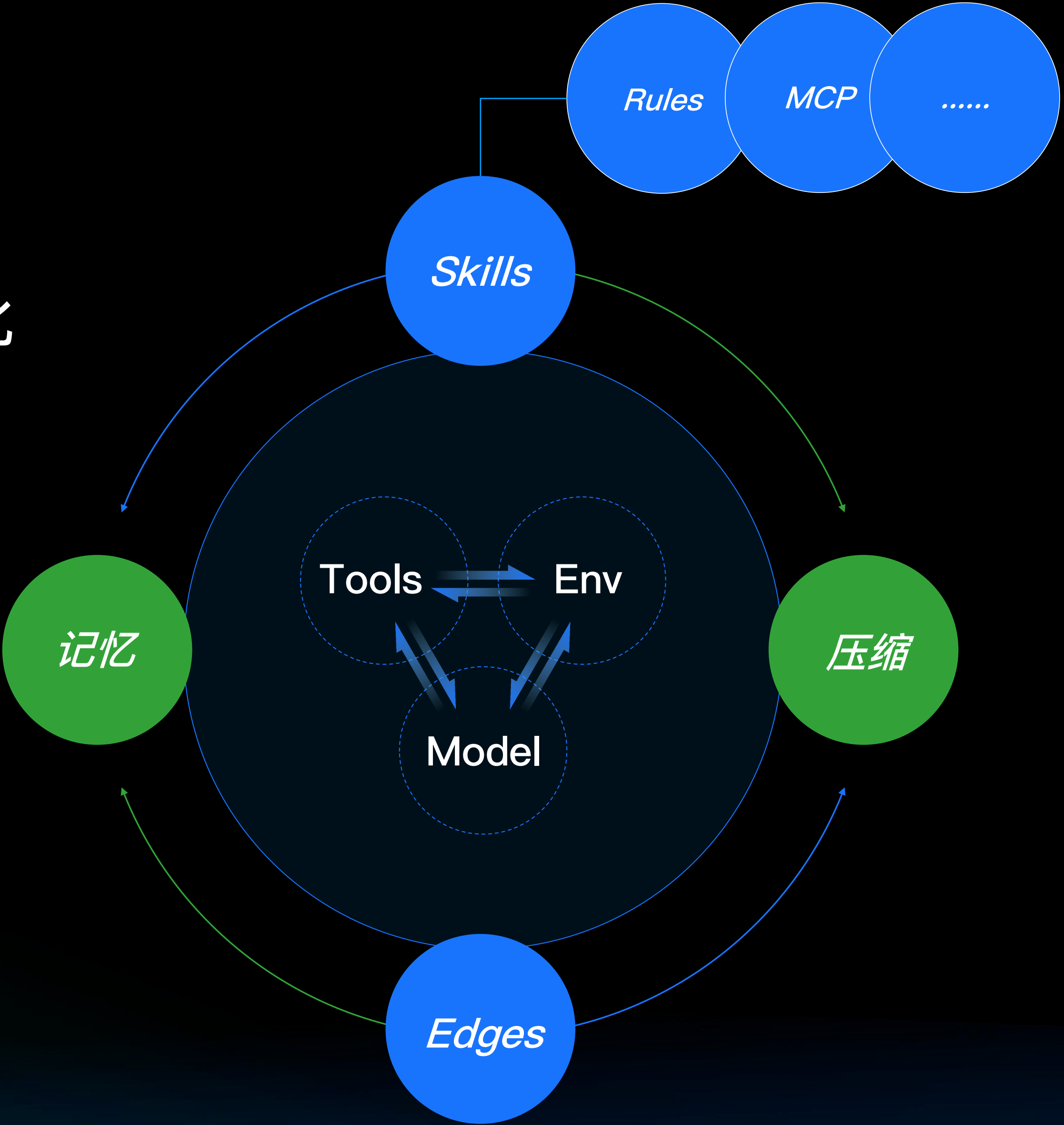
# Agent Loop的基本框架

## 核心Loop：驱动Agent运转，感知外部环境变化

包含模型、工具以及Agent所运行的环境。很多Agent框架忽略模型本身「特性」的变化，尝试打造统一的框架适配所有模型，Comate的实践让我们清醒认识到这是错误的。因此，Agent框架必须能够发现模型的「变化」。

## 外层Loop：为Agent提供扩展性、鲁棒性

通过记忆、Skills等Context来为Agent提供更多接触环境、感知环境的手段，从而大幅延长Agent的「触角」。通过「压缩」和「Edges（边界情况处理）」来让Agent具备自愈能力。



# 打造一个通用Agent产品时，构建Agent Loop面对的核心问题

对模型解题能力的持续观测，动态调整Agent框架。

1. 随着模型的「解题」能力越来越强，构建Agent核心原则从「流程定义」变为「状态感知」，不再需要教模型怎么做，只需要告诉模型现在发生了什么
2. 针对Claude 4.6系列、GPT-5.3/5.4系列模型对与编码的细节处理已经足够优秀，完全不需要Spec Driven Development、Superpowers等开发模式/脚手架，反而拖累模型效果，浪费上下文

构建有效的评测级，重点不再是解题分数而是发现异常值。

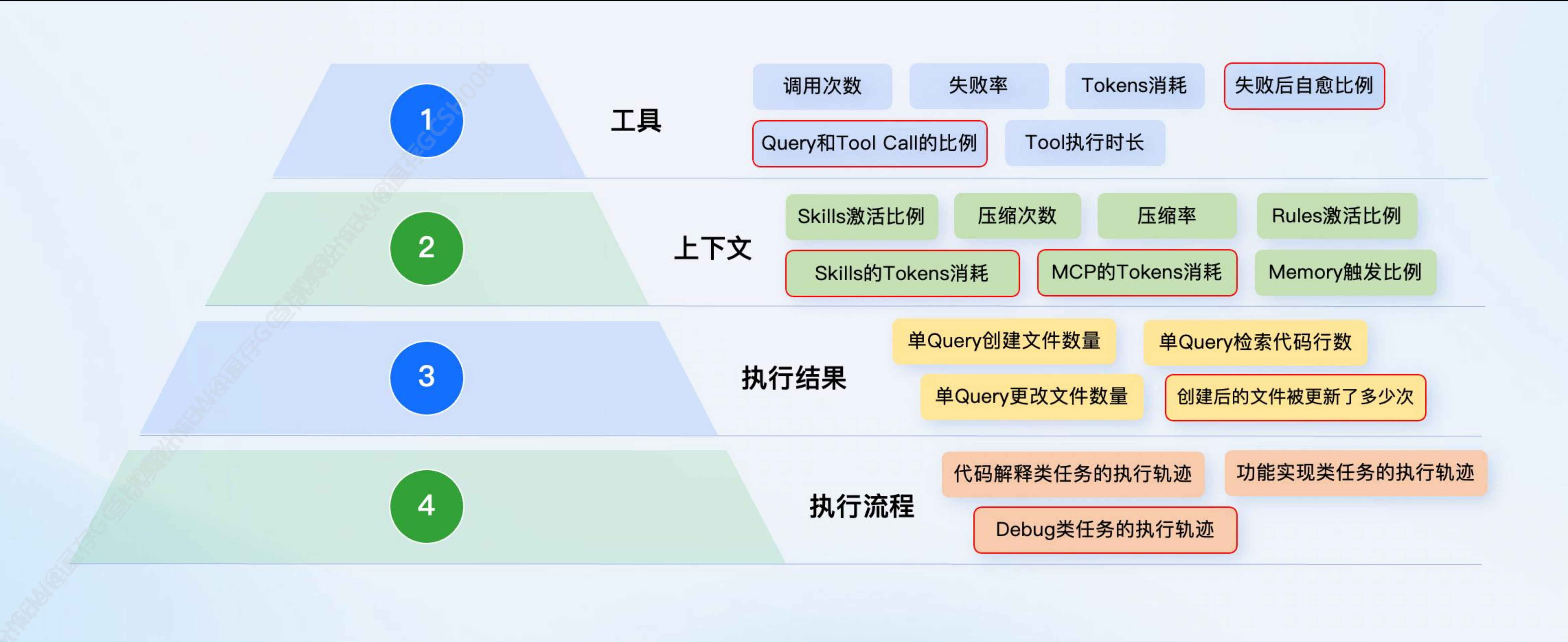
1. 单纯做面向SWE-Bench等通用方法的评测无法真正体现Agent的效果，评测集和对应的评判标准体现了Agent的「调性」
2. 评测单纯看分数是不够的，需要发现异常值，通过异常值发现模型的偏好

「人」也是Agent的Tool、Context，需要打破角色分工。

1. 传统软件研发协作流程将开发者分成前端、后端、测试等各个角色，这种协作适用于「稳定态」的软件，而基于模型构建的Agent应用是「混沌态」，无法有效适用

# 构建线上Agent执行数据的观测体系

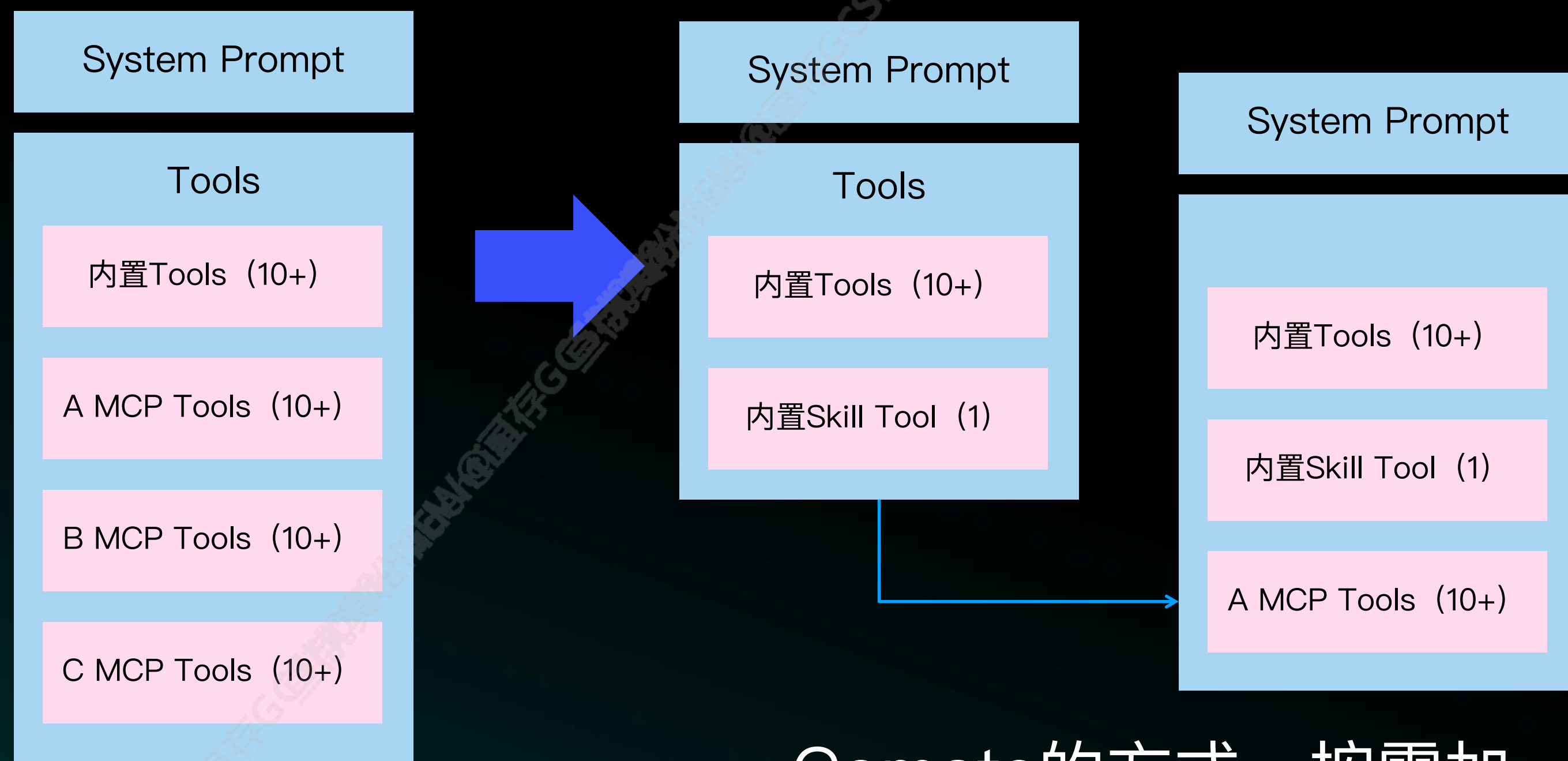
Feedback Loop: 让 Agent 的行为可观测



## 实践：MCP和Skills的Tokens消耗对比强烈，推动我们快速优化

## ■使用Skills的方式实现MCP动态加载，节省98%的Tokens消耗

- ✓ Comate自动为每个MCP Server生成Skill风格的描述，并作为一个虚拟的Skill加载
- ✓ 模型根据Query自动判断是否需要加载整个MCP Server的Tools



# Comate的方式， 按需加载

## 业内主流的方式

```

graph TD
    1[用户在 Comate MCP 配置界面安装 MCP 服务器] --> 2[|]
    2 --> 3[▼]
    3 --> 4[Comate 自动为每个 MCP 服务器生成对应的 Skill 配置  
(名称 + 高级描述, 记录服务器能力概要)]
    4 --> 5[|]
    5 --> 6[▼]
    6 --> 7[每次请求时, skill 工具 description 中嵌入所有 MCP Skill 的轻量列表  
(模型可见: 服务器名称 + 能力概要)]
    7 --> 8[|]
    8 --> 9[▼]
    9 --> 10[模型根据任务判断: 是否需要使用某个 MCP 服务器?]
    10 --> 11[否]
    10 --> 12[是]
    11 --> 13[(不加载)]
    12 --> 14[调用 skill(name=<server_name>)]
    14 --> 15[|]
    15 --> 16[▼]
    16 --> 17[Comate 实时拉取该 MCP 服务器最新工具列表  
返回所有工具的完整 Schema 定义]
    17 --> 18[|]
    18 --> 19[▼]
    19 --> 20[模型阅读 Schema, 选择合适工具  
调用 use_mcp_tool(server_name, tool_name, arguments)]
    20 --> 21[|]
    21 --> 22[▼]
    22 --> 23[Comate 执行 MCP 工具调用  
返回结果给模型]

```

# 从业务本身挖掘评测集，善用Git Blame

## 为Agent提供作为评测集的代码库

- 必须是一个Git仓库，且有良好的Commit Message
- 告诉Agent希望提取什么类型的Case作为评测

## Agent通过分析Git Logs提取信息

- Agent开始逐个分析Commit判断是否适合作为评测集
- 如果适合则抽象给出评测的Query、验证方法

Agent

## 启动另一个Agent进行验证

- 检查无误则启动另一个独立Agent进行验证
- 验证结果经过Agent和人工检验

## 人工检查评测集的合理性并修正

- 检验Agent提取Case的合理性
- 请注意，一条评测Case一般要跨多个Commit，需告知Agent这样的情况存在

# 评测结果的度量，分析流程和异常值 —— 四象限分析法

$$\text{Overall Score} = 0.6 \times \text{Outcome Score} + 0.4 \times \text{Execution Score}$$

## Outcome Score (结果质量 · 权重 60%)

- *Test Pass* (实际测试通过, 0.5)
- *Functional Correctness* (功能正确性, 0.2)
- *Completeness* (完整性, 0.1)
- *Patch Quality* (代码质量, 0.1)
- *Regression Risk* (回归风险, 0.1)

## Execution Score (执行效率 · 权重 40%)

- *Problem Understanding* (理解能力, 0.25)
- *Agent Execution Quality* (执行质量, 0.75)
  - *Investigation*
  - *Tool Usage*
  - *Control Flow*
  - *Task Management*
  - *Validation*
  - *Craftsmanship*

### OUTCOME 轴定义

Test 通过  $\text{outcome} = 0.5 + \text{llm\_score}/2$ , 范围 [0.5, 1.0]

Test 不通过  $\text{outcome} = 0 + \text{llm\_score}/2$ , 范围 [0, 0.5]

```
llm_score = (  
    functional_correctness +  
    completeness +  
    patch_quality +  
    regression_risk  
) / 4
```

### EXECUTION 轴定义

核心指标  $\text{execution} = \text{agent\_execution\_quality}$  (LLM评分)

高 Execution  $\text{execution\_quality} \geq 0.7$  (执行规范、高效)

低 Execution  $\text{execution\_quality} < 0.7$  (存在循环/混乱)

agent\_execution\_quality 含:

- Investigation & Localization
- Tool Usage / Control Flow
- Task Management / Validation
- Execution Craftsmanship

# 把人放到Loop里的两层含义

## 全员转型 —— 每个人都成为Agent Engineer

1

- 不只是开发者在用Agent, PM、UE、测试、售前都在用IDE写代码 —— 角色边界正在消失
- 转型的核心不是"学会写Prompt", 而是从"执行者"转变为"问题提出者", 能力瓶颈从编码速度变成了提出好问题的能力
  - 协作模式从"大Feature拆小Task分给多人"变为"一人 + Agent一竿子捅到底"

2

## 把人看作Agent的工具和Context

- 理解中人是Agent的使用者、监督者。但在实践中, 人本身就是Agent Loop的一部分 —— 人提供的判断、审美、业务知识是Agent无法自己生成的Context
  - 当Agent遇到需要产品决策、视觉判断、业务逻辑确认的节点时, "调用人"和"调用Tool"没有本质区别

# Baidu Comate

## 拥抱AI原生研发新范式

感谢聆听



扫码加好友(李杨)

AI原生研发新范式更深入交流



欢迎扫码了解

用科技让复杂的世界更简单