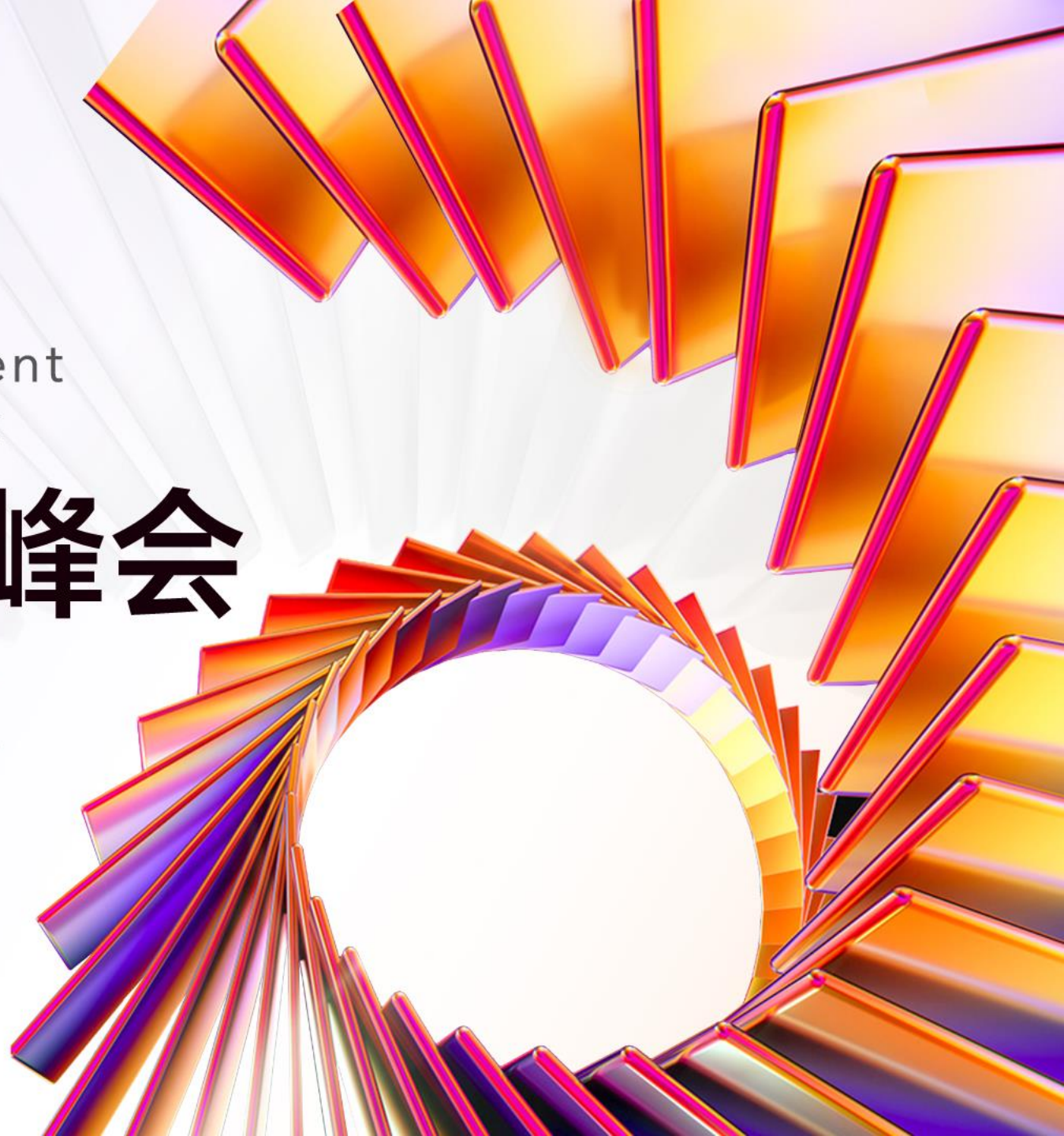
AiDD 8th 2025

第8届 AI+ Development Digital Summit AI+研发数字峰会

拥抱AI 重塑研发

11月14-15日 | 深圳



AiDD × K

2026 AI+ PRODUCT INNOVATION SUMMIT

01.16-17 · ShangHai

AI+产品创新峰会

Track 1: AI 产品战略与创新设计

从0到1的AI原生产品构建

论坛1: AI时代的用户洞家与需求发现

论坛2: AI原生产品战路与商业模式重构

论坛3: AgenticAI产品创新与交互设计

Track 2: AI 产品开发与工程实践

从1到10的工程化落地实践

论坛1: 面向Agent智能体的产品开发

论坛2: 具身智能与AI硬件产品

论坛3: AI产品出海与本地化开发

Track 3: AI 产品运营与智能演化

从10到100的AI产品运营

论坛1: AI赋能产品运营与增长黑客

论坛2: AI产品的数据飞轮与智能演化

论坛3: 行业爆款AI产品案例拆解

2-hour Speech: 回归本质

用户洞察的第一性

——2小时思维与方法论工作坊

在数字爆炸、AI迅速发展的时代，仍然考验“看见”的“同理心”

Panel 1: 出海前瞻

“出海避坑地图”圆桌对话

——不止于翻译:

AI时代的出海新范式

Panel 2: 失败复盘

为什么很多AI产品“叫好不叫座”?

——从伪需求到真价值:

AI产品商业化落地的关键挑战

智能重构产品 数据驱动增长

Reinventing Products with Intelligence, Driven by Data

80+ 业界大咖 汇聚产品大咖的真知灼见

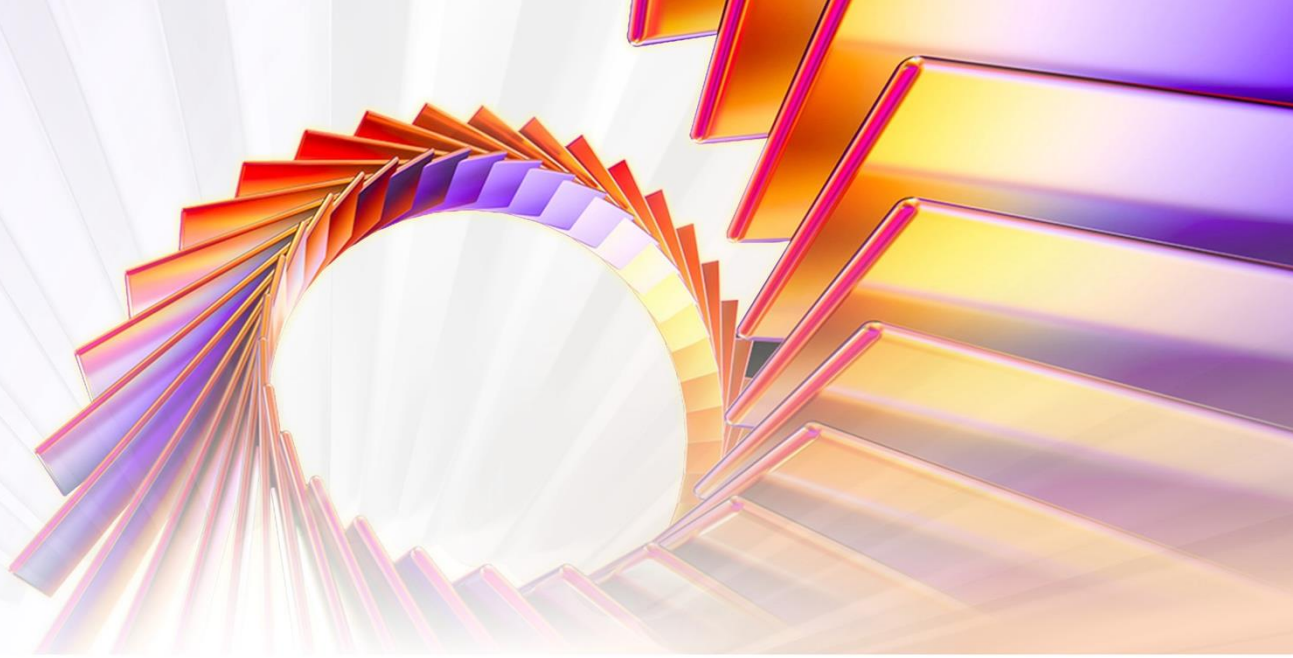
40+ 创新案例 开拓全新AI+产品视角

10+ 分论坛 涵盖产品到商业增长的全链路

800+ 听众规模 共同探索产品的智能重构

扫码查看会议详情

AiDD 8th 2025 | 11月14-15日 | 深圳
第8届 AI+ Development Digital Summit
AI+研发数字峰会
拥抱AI 重塑研发



AI 编程工具选型实践指南
插件、IDE 与终端形态的深度解析

汪晟杰 | 腾讯

汪晟杰

腾讯资深技术产品专家、CodeBuddy 首席产品经理

腾讯资深产品专家，20年工作经验，负责腾讯云开发者AI代码助手产品规划设计与运营，十多年协作SaaS和 SAP 云平台、SuccessFactors HCM、Sybase 数据库、PowerDesigner 等产品的开发经理，在软件架构设计、产品管理和项目工程管理、团队敏捷、AI研发提效等方面拥有丰富的行业经验。

AiDD 8th 2025

PART 01

从代码生成到工程协作
AI 开发的新阶段

目录
CONTENTS

I. 背景

II. 问题/痛点

III. 解决思路/整体方案

IV. 具体实现/技术实践

AI 编程从人机协作到上下文协作

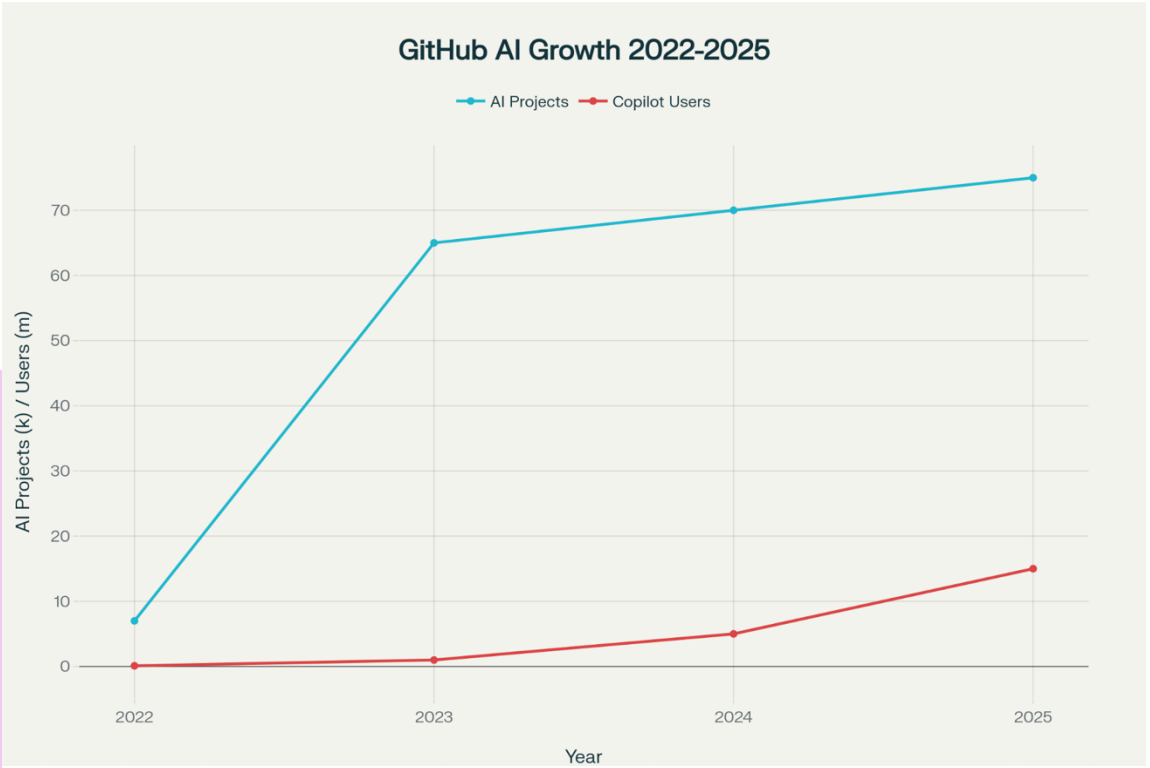
人机协作：
在人机协作的阶段，AI更多的是作为工具，辅助开发者完成特定任务，人作为每一步确认方和修改方。

上下文协作：

在实现过程中，上一轮的完成上下文是下一轮的开始的上下文。AI自主完成

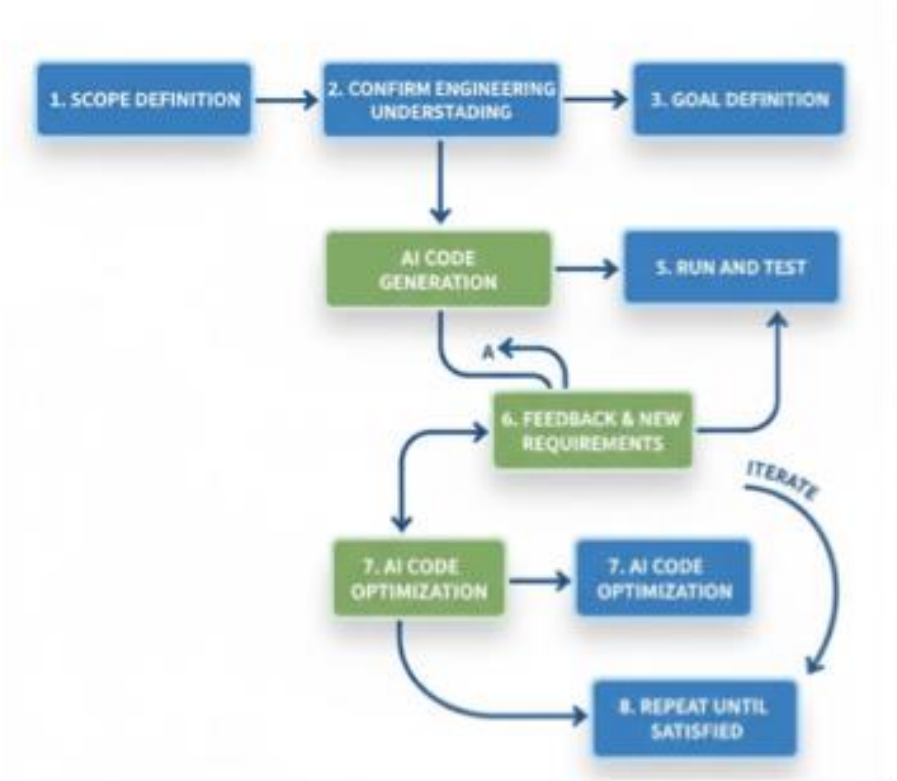
它具有如下特性：

智能感知：AI不仅对当前任务有理解，还能感知整个开发流程的上下文（如代码历史、团队讨论、业务需求等）。
动态反馈：AI根据实时变化的上下文，提供动态建议和反馈，而不是固定的、静态的输入。
闭环优化：AI能够在开发过程中，持续学习并调整其行为，使得每一轮开发都比上一次更加高效和智能。
自动化执行：AI可以根据上下文进行更为复杂的任务执行，例如自动重构代码、优化系统架构、处理技术债务等。



人机协作的关键不再是智能度，而是共享规则与约束的能力

- **明确目标需求**
  复杂度以自然语言表达和澄清你的开发目标，确保每一步都有清晰的方向。
- **AI 生成初版代码**
  借助大模型根据你的需求，快速生成基础实现方案，奠定功能骨架。
- **运行并验证效果**
  实际运行并测试生成的代码，及时了解功能完成度与存在的问题。
- **反馈问题与新想法**
  对测试中发现的问题和新的功能需求，及时进行反馈和补充说明。
- **AI 迭代优化代码**
  让 AI 基于反馈持续重构与完善代码，修正 bug，优化细节，提升整体质量。
- **多轮循环直至理想**
  重复以上流程，持续交互和迭代，直到你对代码和产品完全满意为止。



AI 编程从 Vibe Coding 到氛围编程多种形态

传统编程 (Traditional Coding)

特征

- 传统开发者主导的完全纯手动编码
- 手动调试过程
- 依赖浏览器搜索引擎/开发者社区等辅助开发

效果

- 逐行编码，比较低效

学习曲线及要求

- 陡峭的学习曲线，需了解编程底层技术，需自主学习能力和积累项目中实战经验

氛围编程 (Vibe Coding)

特征

- 基于AI 智能体和AI对话至上，采用自然语言描述需求，实现多文件代码生成，生成执行的应用
- 精准描述需求和任务表达有助于 AI 生成代码质量

效果

- 实现工程级别开发，中等效率

学习曲线及要求

- 学习曲线中等，非程序员(如 产品、设计等小白用户）也可编码，侧重和锻炼表达功能能力

规约编程 (Specification-Oriented Coding)

特征

- 在 Agent 至上，基于规范和设计共识驱动 AI 全栈开发，批量生成业务代码
- 结构化沟通和系统设计，生成代码包含所需的前提和意图
- 多智能体协作，先共识，帮你理清思路，集成系统思维澄清

效果

- 规范文档和共识协作，实现系统级别完整意图的规范化代码，效率高

学习曲线及要求

- 学习曲线高，弥补 Vibe Coding 中的痛点，对规范化、系统化有更高全局要求和把控能力，锻炼编写能充分捕捉意图和价值观的规范能力

未来，基于AI的个人氛围编程，以及过渡和适应专业团队协作的规约编程，两种开发范式并存

软件工程边界消融，AI Coding 重构组织协作方式

AI Coding 产品形态

AI Plugin	AI IDE	AI CLI
智能副驾	一站式 AI 工作台	Agent 操作系统
• 集成至主流IDE • 企业内 AI Coding 快速落地	• “集成开发环境”转为“智能开发环境” • 产设研一体，软件工程全生命周期覆盖	• 命令行交互 • 批处理、多系统异步协作，7x24 运行 • 深度融入云+DevOps与云原生

人机协作的工作范式的转移

	Embedding	Copilot	Agent
			
AI 角色	辅助工具	协作伙伴	自主代理
人类角色	主要执行者	共同执行者	目标制定者
决策权	完全在人类	人类主导	AI 自主决策
典型应用场景	AI 搜索	代码补全	CodeBuddy
协作方式	检索与辅助	实时互动	任务代理
人机比例	80%：20%	50%：50%	20%：80%

目录
CONTENTS

I. 背景

II. 问题/痛点

III. 解决思路/整体方案

IV. 具体实现/技术实践

氛围编程带来的问题

面临的问题

需求与设计环节

“拍脑袋”决策，反复折腾没效率

用户想要啥全靠产品经理猜，用户调研数据堆在 Excel 里没人深挖，产品定义、功能拆解全凭经验。
设计师画完原型、交互、视觉稿，产品一句 “感觉不对” 就得改 N 版，白折腾还说不出具体问题。

架构与开发环节

技术选型 “凭经验” 开发效率低到急

技术选型全靠老工程师拍板，选的框架后面不兼容新功能，得推倒重来。领域建模、架构设计没人带就容易走弯路。
编码、评审环节，代码设计、安全扫码全手动，单元测试生成慢，修复安全问题还得一个个抠

运维与市场环节

问题反馈 “滞后性” 市场变化赶不上

日志分析、故障归因全手动，用户都投诉好几次了，研发才知道问题。
从需求定下来到产品上市，快则半年慢则一年，等卖的时候市场早就变样了。

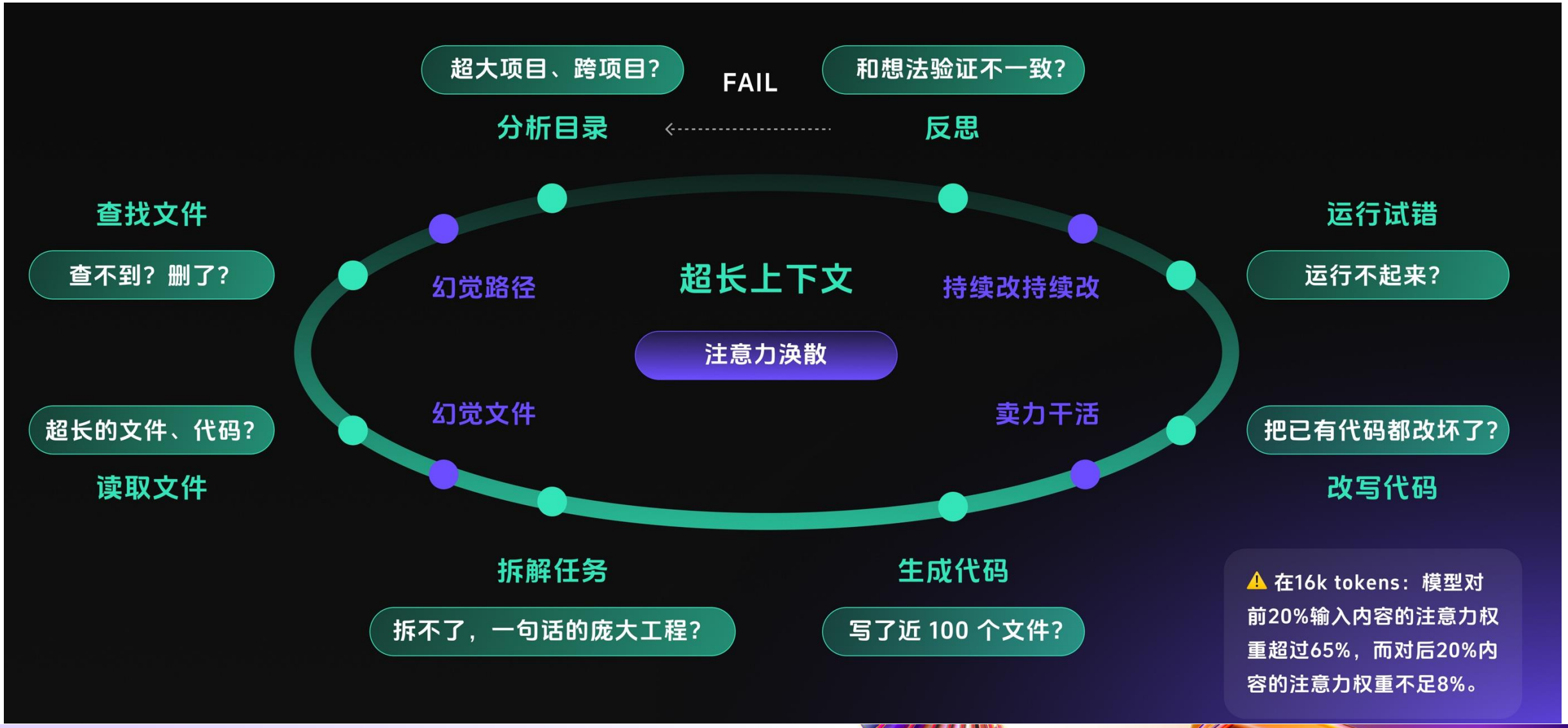
测试与交付环节

上线前测试 “走过场” 交付后锅甩不停

测试用例生成、自动化测试脚本全手动，上线后用户用个冷门场景就崩，锅全甩给研发。
部署脚本、发布更新文档手动写，小批量试产时生产部和研发互相甩锅，说 “设计的不好造”。

一句话需求注定在工程中失败

逻辑漏洞的描述和超长工程代码文件让理解错误率上升



目录
CONTENTS

I. 背景

II. 问题/痛点

III. 解决思路/整体方案

IV. 具体实现/技术实践

代码驱动的规约表达，在 AI 驱动下难以智能化感知

1. 函数规约（Preconditions, Postconditions, Invariants）

假设你在开发一个计算银行账户余额的系统，使用规约编程来确保“存款”和“取款”操作的正确性。

```python
class BankAccount:
    def __init__(self, initial_balance: float):
        self.balance = initial_balance

    # 预条件：存款金额必须大于0
    # 后条件：账户余额会增加相应的金额
    def deposit(self, amount: float):
        assert amount > 0, "存款金额必须大于0"
        self.balance += amount

    # 预条件：取款金额必须大于0且账户余额充足
    # 后条件：账户余额会减少相应的金额
    def withdraw(self, amount: float):
        assert amount > 0, "取款金额必须大于0"
        assert self.balance >= amount, "账户余额不足"
        self.balance -= amount
```

2.模块间规约（Interface Contracts）

在微服务架构中，两个服务之间可能需要遵循一个共享的接口协议。这个协议定义了双方期望的输入输出格式和行为，从而保证它们能够正确协作。

```python
class WeatherService:
    # 输入：城市名称
    # 输出：天气数据字典，包含气温、湿度、天气状况等信息
    def get_weather(self, city: str) -> dict:
        pass  # 模拟天气查询

class NotificationService:
    # 输入：天气数据（字典），包含温度、湿度等信息
    # 输出：发送通知，返回发送状态
    def send_weather_alert(self, weather_data: dict) -
        pass  # 模拟发送通知
```

3. 异常处理规约

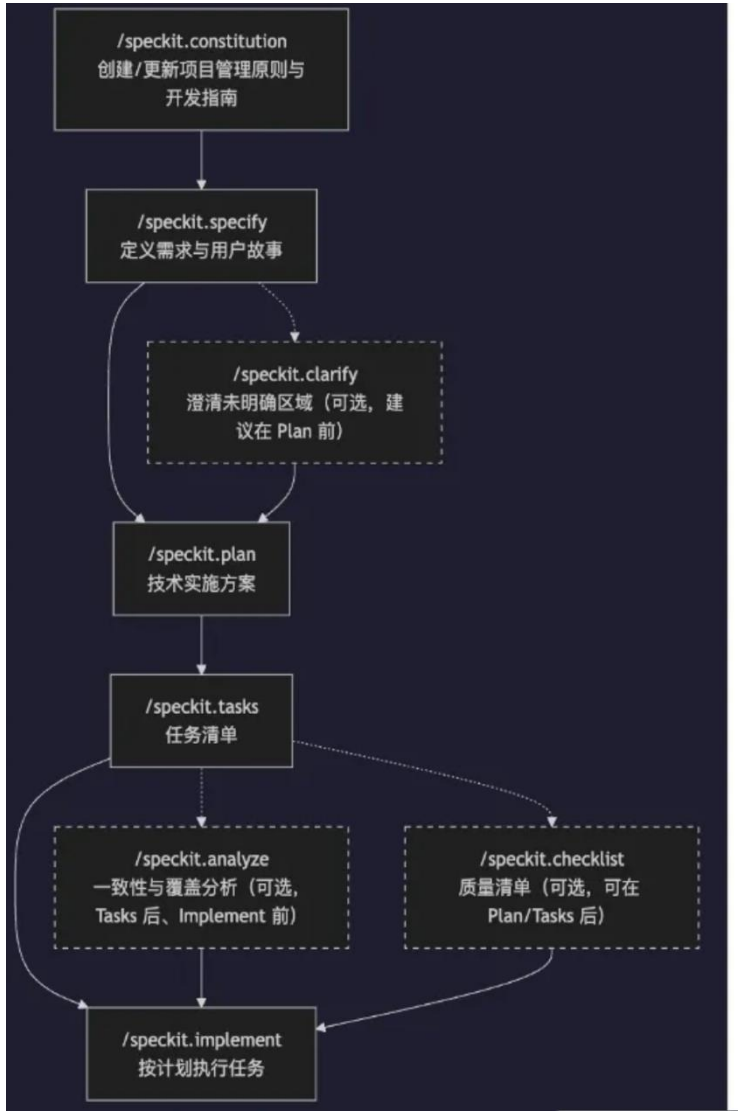
假设我们正在开发一个在线购物系统，用户需要提供有效的信用卡信息进行支付。如果支付失败，我们需要提供一个清晰的错误信息，并要求在支付流程中进行适当的异常处理。

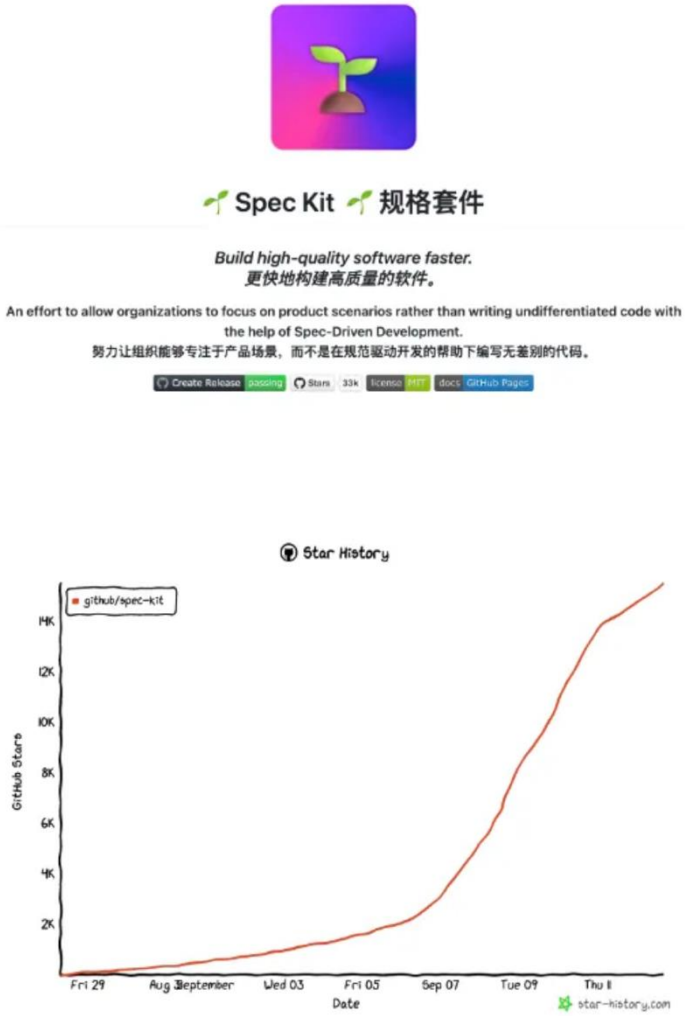
```python
class PaymentSystem:
    def __init__(self):
        self.balance = 1000  # 用户余额

    # 预条件：信用卡信息必须有效，支付金额必须大于0
    # 后条件：支付成功后余额减少
    def process_payment(self, credit_card_info: dict, amount: float):
        assert self._is_valid_credit_card(credit_card_info), "信用卡无效"
        assert amount > 0, "支付金额必须大于0"
        if self.balance < amount:
            raise ValueError("余额不足")
        self.balance -= amount
        return True

    def _is_valid_credit_card(self, credit_card_info: dict) -> bool:
        # 假设验证信用卡逻辑
        return True
```

AI 规约编程的趋势

文件名	仓库数量/采用情况	文件位置	主要支持工具	文件类型	标准化状态
AGENTS.md	超过20,000个开源项目	仓库根目录或子目录	OpenAI Codex, GitHub Copilot, Google Jules, Cursor, Aider, RooCode, Zed, Factory等20+工具	Markdown指令文件	开放标准，正在成为主流
Claude Code Skills	无具体统计数据	作为技能包存在于.claude目录	Claude Code, Claude.ai Web, Claude Desktop, Claude API	技能包（包含SKILL.md + 脚本 + 资源）	Anthropic专有格式
GEMINI.md	无具体统计数据	仓库根目录或.gemini/GEMINI.md	Google Gemini CLI	Markdown指令文件	Google专有格式
Github Copilot Instructions.md	无具体统计数据	.github/copilot-instructions.md 或 .github/instructions/*.instructions.md	GitHub Copilot (包括Agent模式)	Markdown指令文件	GitHub专有格式
CodeBuddy.md	腾讯内部、COdeBuddy 开发者	仓库根目录（基于CodeBuddy.ai规范）	CodeBuddy AI Assistant (VS Code扩展)	Markdown指令文件	CodeBuddy 的 MD 格式

SpecKit

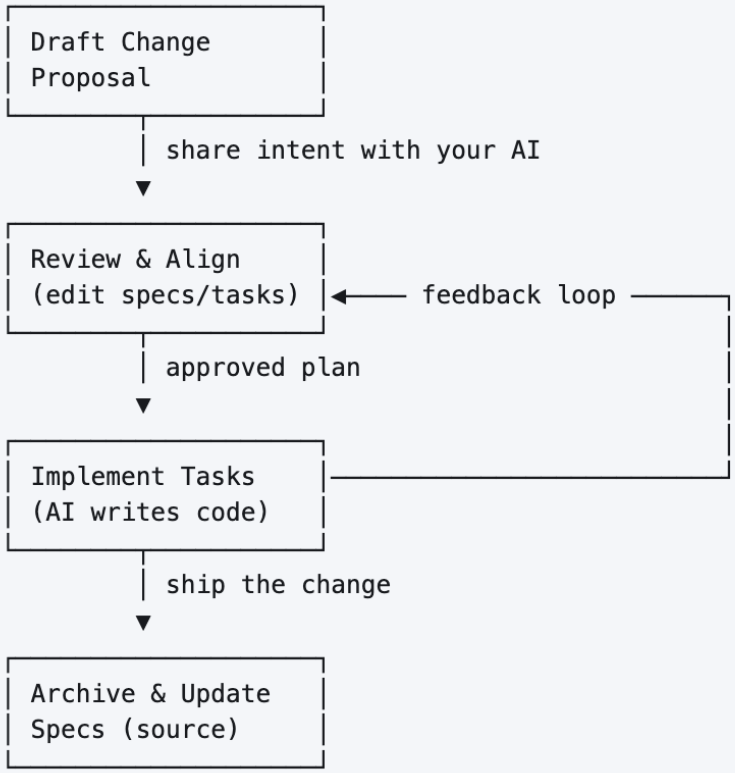





```
● Perfect! The specify CLI is working. Now let me check if CodeBuddy Code is recognized as

● Bash(command: "specify check", description: "Check if CodeBuddy Code is recognized as av
  └

  GitHub Spec Kit - Spec-Driven Development Toolkit

  Checking for installed tools...

  Check Available Tools
  ├── ● Git version control (available)
  ├── ● Claude Code CLI (not found)
  ├── ● CodeBuddy Code CLI (available)
  ├── ● Gemini CLI (not found)
  ├── ● Qwen Code CLI (not found)
  ├── ● Visual Studio Code (available)
  ├── ● Visual Studio Code Insiders (not found)
  ├── ● Cursor IDE agent (available)
  ├── ● Windsurf IDE (available)
  ├── ● Kilo Code IDE (not found)
  ├── ● opencode (not found)
  ├── ● Codex CLI (not found)
  └── ● Auggie CLI (not found)

  Specify CLI is ready to use!

● Excellent! CodeBuddy Code is showing as "available" in the tool check. Now let me test
  CodeBuddy Code:
```

OpenSpec

更少步骤的规约编程的尝试

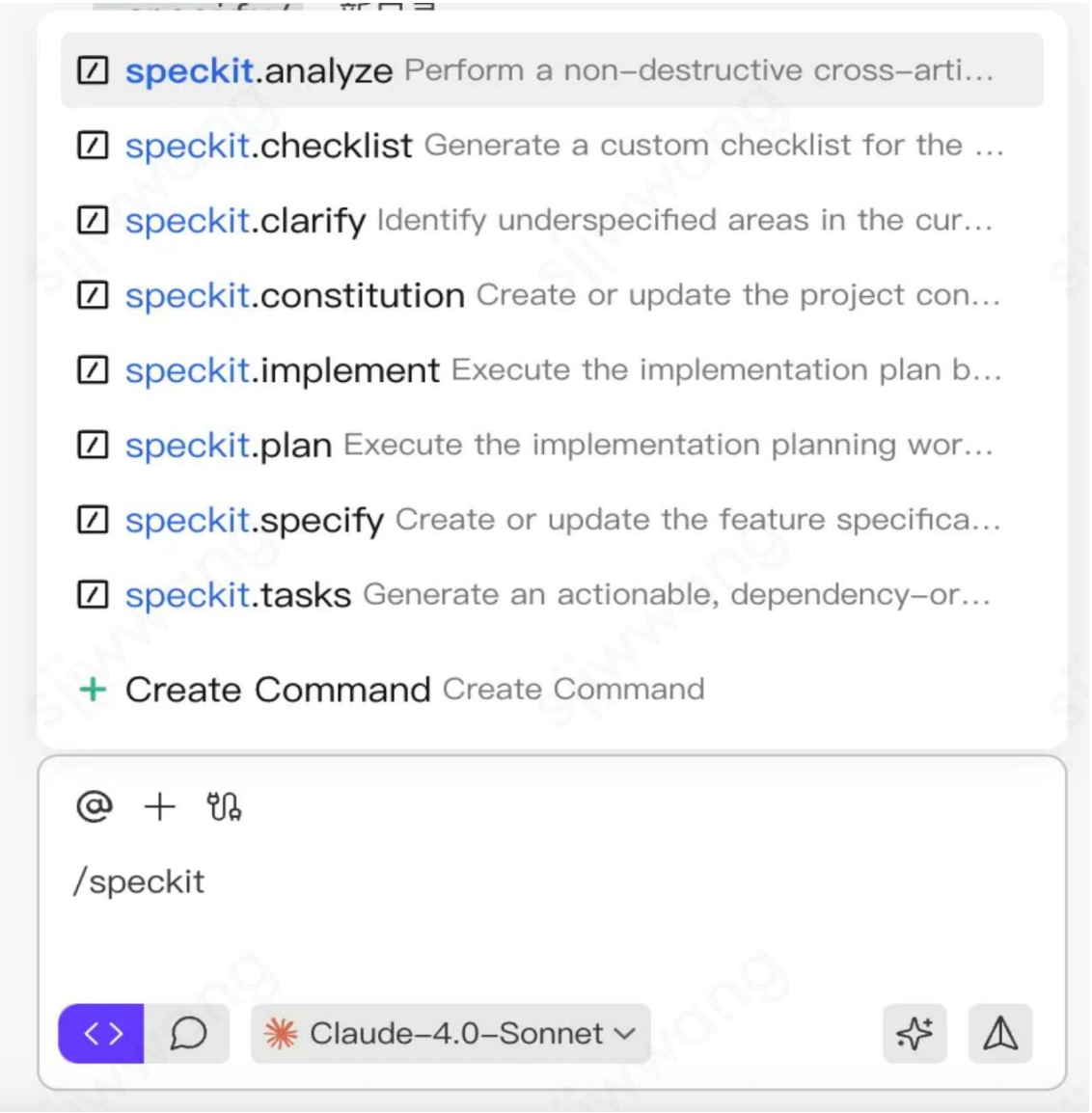


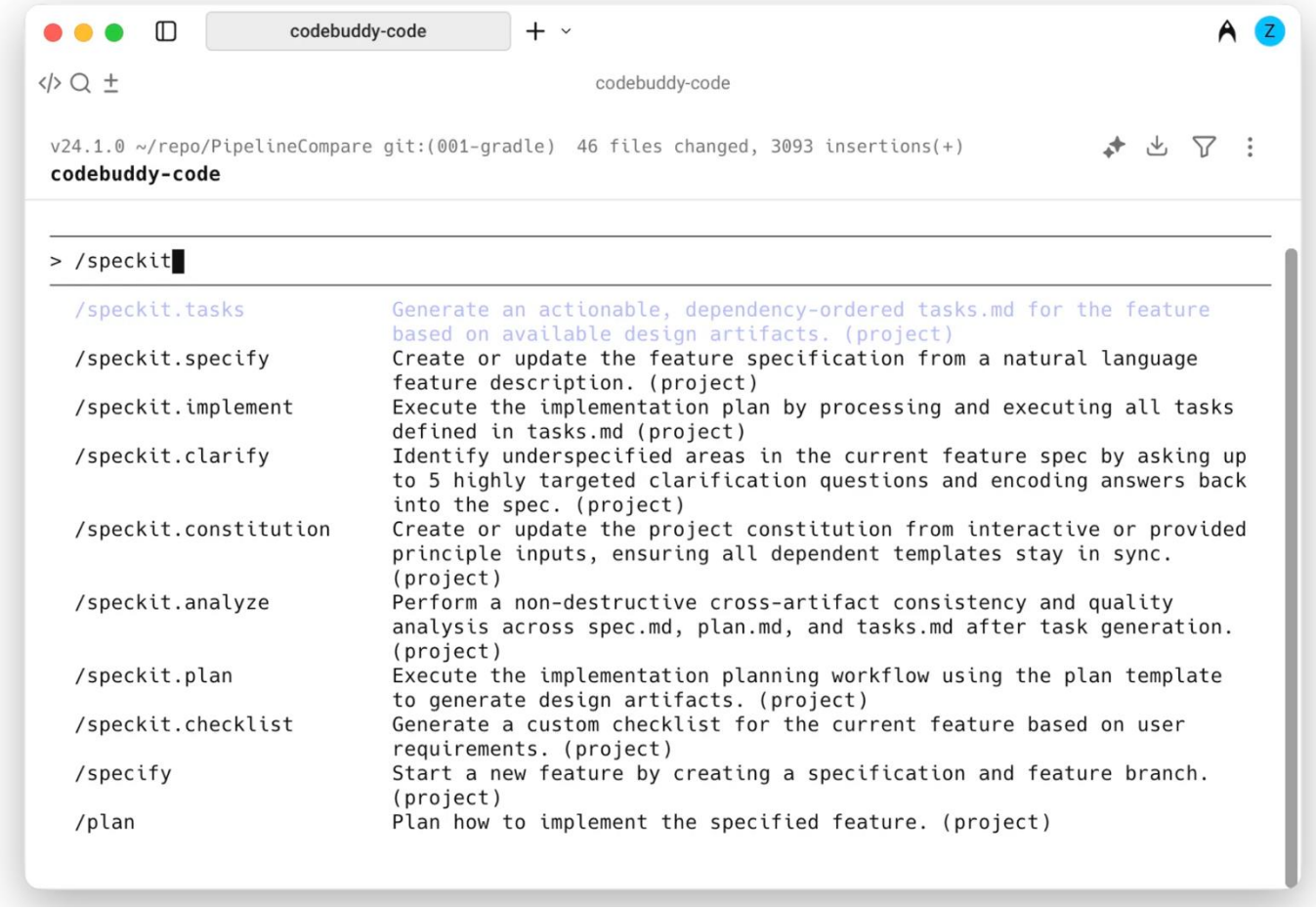
1. Draft a change proposal that captures the spec updates you want.
2. Review the proposal with your AI assistant until everyone agrees.
3. Implement tasks that reference the agreed specs.
4. Archive the change to merge the approved updates back into the source-of-truth specs.

特征：节奏快、切口小、循环短
适用：现有项目功能演进、多功能并行、快速迭代场景
🕑 **时间对比**：OpenSpec 平均快约 **46%**，主要节省在规范生成与任务分解阶段。

```
openspec/
├── specs/spec.md              # 当前稳定规范
├── changes/feature-x/         # 进行中提案
│   ├── proposal.md
│   ├── tasks.md
│   └── spec-deltas/
└── archived/                  # 已归档历史
```

CodeBuddy 支持 Speckit，如虎添翼





CodeBuddy 支持 Speckit，如虎添翼

constitution → specify → plan → tasks → implement 项目原则 → 功能规范 → 实现计划 → 任务分解 → 代码实现

阶段	命令	目的
1	/speckit.constitution	定义项目的开发宪章（指导原则）
2	/speckit.specify	编写初始功能规格文档
3	/speckit.checklist	验证规格质量是否达到“可交付”标准
4	/speckit.clarify	根据 checklist 结果进行需求澄清
5	/speckit.plan + /speckit.tasks	进入实现规划与任务分解阶段
6	/speckit.implement	代码产出

```
spec-kit/
├── src/                                    # 核心源代码目录
│   └── specify_cli/                        # Specify CLI 主要实现
│       └── __init__.py                     # CLI 工具的完整实现，包含所有命令和AI代理集成
├── templates/                              # 模板系统 - SDD工作流的核心
│   ├── commands/                           # AI代理命令模板
│   │   ├── specify.md                      # 规范创建命令 - 将自然语言转换为结构化规范
│   │   ├── plan.md                         # 实现计划命令 - 从规范生成技术实现计划
│   │   ├── tasks.md                        # 任务分解命令 - 将计划分解为可执行任务
│   │   ├── implement.md                    # 实现执行命令 - 执行任务列表构建功能
│   │   ├── constitution.md                 # 项目原则命令 - 建立项目治理原则
│   │   ├── clarify.md                      # 澄清命令 - 结构化问题解决规范中的模糊性
│   │   ├── analyze.md                      # 分析命令 - 跨工件一致性和覆盖率分析
│   │   └── checklist.md                    # 检查清单命令 - 生成质量验证清单
│   ├── spec-template.md                    # 功能规范模板 - 定义用户故事和功能需求结构
│   ├── plan-template.md                    # 实现计划模板 - 技术架构和实现细节结构
│   ├── tasks-template.md                   # 任务模板 - 可执行任务列表结构
│   ├── checklist-template.md               # 检查清单模板 - 质量验证清单结构
│   ├── agent-file-template.md              # 代理文件模板 - AI代理配置文件结构
│   └── vscode-settings.json                # VS Code 设置模板 - 开发环境配置
├── scripts/                                # 自动化脚本系统
│   ├── bash/                               # POSIX Shell 脚本
│   │   ├── common.sh                       # 公共函数库 - 仓库路径、分支管理、功能目录查找
│   │   ├── create-new-feature.sh           # 功能创建脚本 - 自动分支创建和目录结构设置
│   │   ├── setup-plan.sh                   # 计划设置脚本 - 实现计划初始化
│   │   ├── update-agent-context.sh         # 代理上下文更新 - 同步项目信息到AI代理
│   │   └── check-prerequisites.sh          # 先决条件检查 - 验证工具和环境
│   └── powershell/                         # PowerShell 脚本 (Windows支持)
│       ├── common.ps1                      # PowerShell 公共函数库
│       ├── create-new-feature.ps1          # PowerShell 功能创建脚本
│       ├── setup-plan.ps1                  # PowerShell 计划设置脚本
│       ├── update-agent-context.ps1        # PowerShell 代理上下文更新
│       └── check-prerequisites.ps1         # PowerShell 先决条件检查
├── docs/                                   # 文档系统
│   ├── index.md                            # 文档主页
│   ├── installation.md                     # 安装指南
│   ├── quickstart.md                       # 快速开始指南
│   ├── local-development.md                # 本地开发指南
│   ├── docfx.json                          # DocFX 文档生成配置
│   └── toc.yml                             # 文档目录结构
├── .github/                                # GitHub 工作流和配置
│   └── workflows/                          # CI/CD 管道
│       ├── release.yml                     # 发布工作流 - 自动化版本发布和包分发
│       ├── docs.yml                        # 文档构建工作流
│       ├── lint.yml                        # 代码质量检查工作流
│       └── scripts/                        # 发布脚本
│           ├── create-release-packages.sh  # 创建发布包 - 为每个AI代理生成模板包
│           ├── create-github-release.sh    # GitHub 发布创建
│           ├── generate-release-notes.sh   # 发布说明生成
│           ├── get-next-version.sh         # 版本号管理
│           └── update-version.sh           # 版本更新脚本
├── .devcontainer/                          # 开发容器配置
│   ├── devcontainer.json                   # 容器配置 - VS Code 扩展和环境设置
│   └── post-create.sh                      # 容器后创建脚本 - AI工具安装
```

Fast Path Workflow

小缺陷开发模式的推荐的 SpecKit 指令工作流

场景类型	推荐流程	是否生成 spec.md
复杂 bug（跨模块）	Clarify → Plan → Tasks → Implement	否（可引用旧 spec）
小 bug（微逻辑、UI、配置）	Plan → Tasks → Implement	否，直通快修流程
配置错误 / 更新脚本	Tasks → Implement	否，仅单步任务

CodeBuddy IDE 产设研实战 – Design Spec

假设：

AI能否准确解析Figma设计稿并生成可用代码？

验证方案

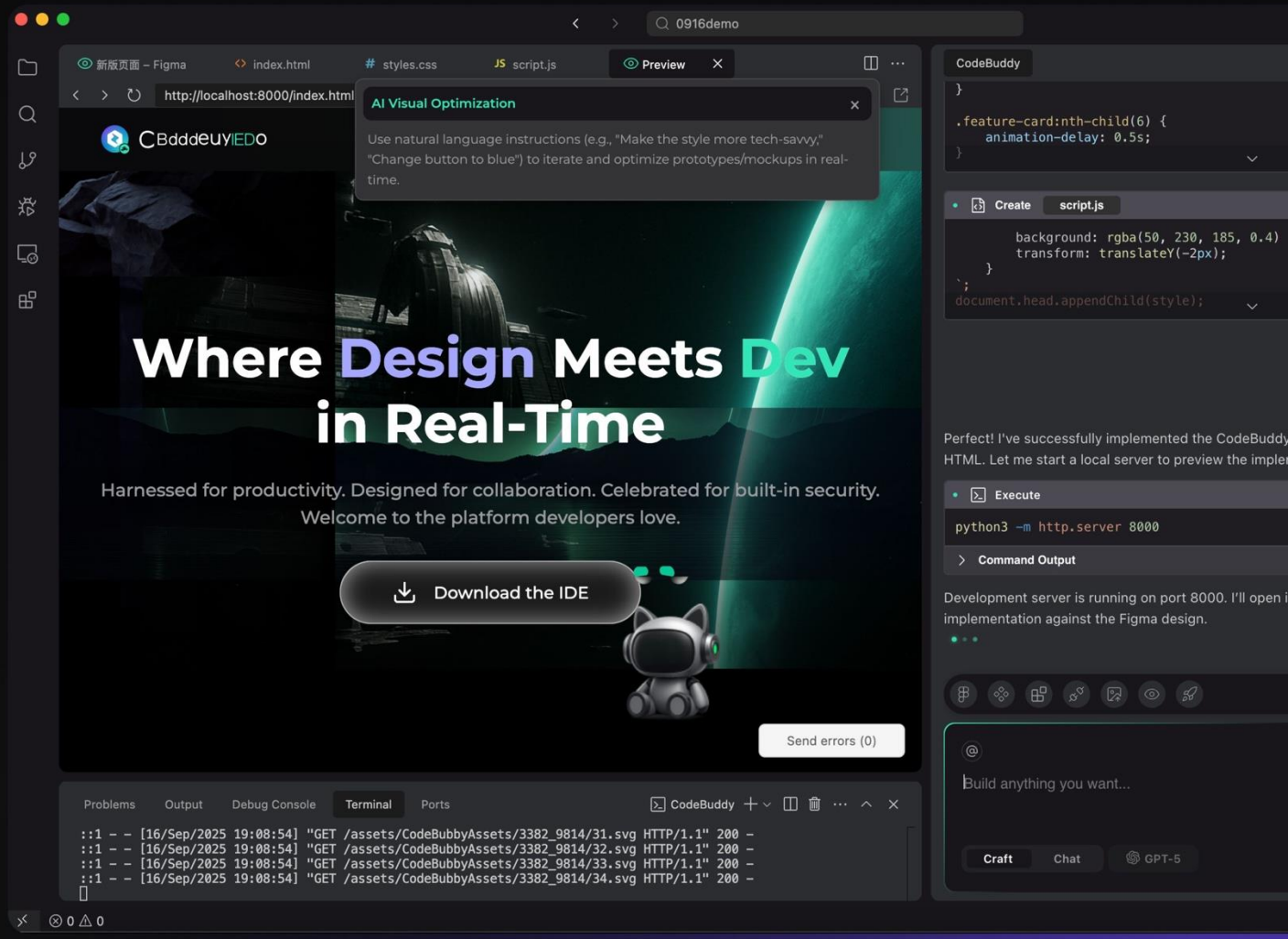
构建Craft Agent原型，测试100+组件转换准确率

发现痛点：纯AI生成代码结构混乱，需引入规则引擎

决策权衡

产品决策：规则为主 + AI辅助 vs 纯端到端AI生成。

背后的思考：这不仅是技术选型，更是对“用户价值”的权衡。纯AI方案追求极致的首次生成速度，但可能牺牲长期维护性。我们选择后者，因为CodeBuddy服务的是“项目生命周期”，而非“单次任务”。这一定位决定了我们更看重代码的可维护性、可预测性，从而为企业客户提供稳定价值。



CodeBuddy IDE 产设研实战 – Arch Spec

假设：

Spec Coding 能否在实际项目中提升交付效率？

验证方案

对比实验：传统开发 vs Spec Coding项目周期

结果：需求变更响应速度提升，Bug率降低

决策权衡

产品决策：平衡Rules的“灵活性”与“规范性”。

背后的思考：这是产品“普适性”与“专业性”的经典矛盾。我们的理念是：提供强大的默认规则集（开箱即用），同时开放完整的自定义能力（灵活扩展）。这确保了产品既能快速入门，又能随团队成长而演进，避免了成为“玩具”或“铁笼”。

CodeBuddy IDE 产设研实战 – 规模验证 – MCP & Sandbox

假设：

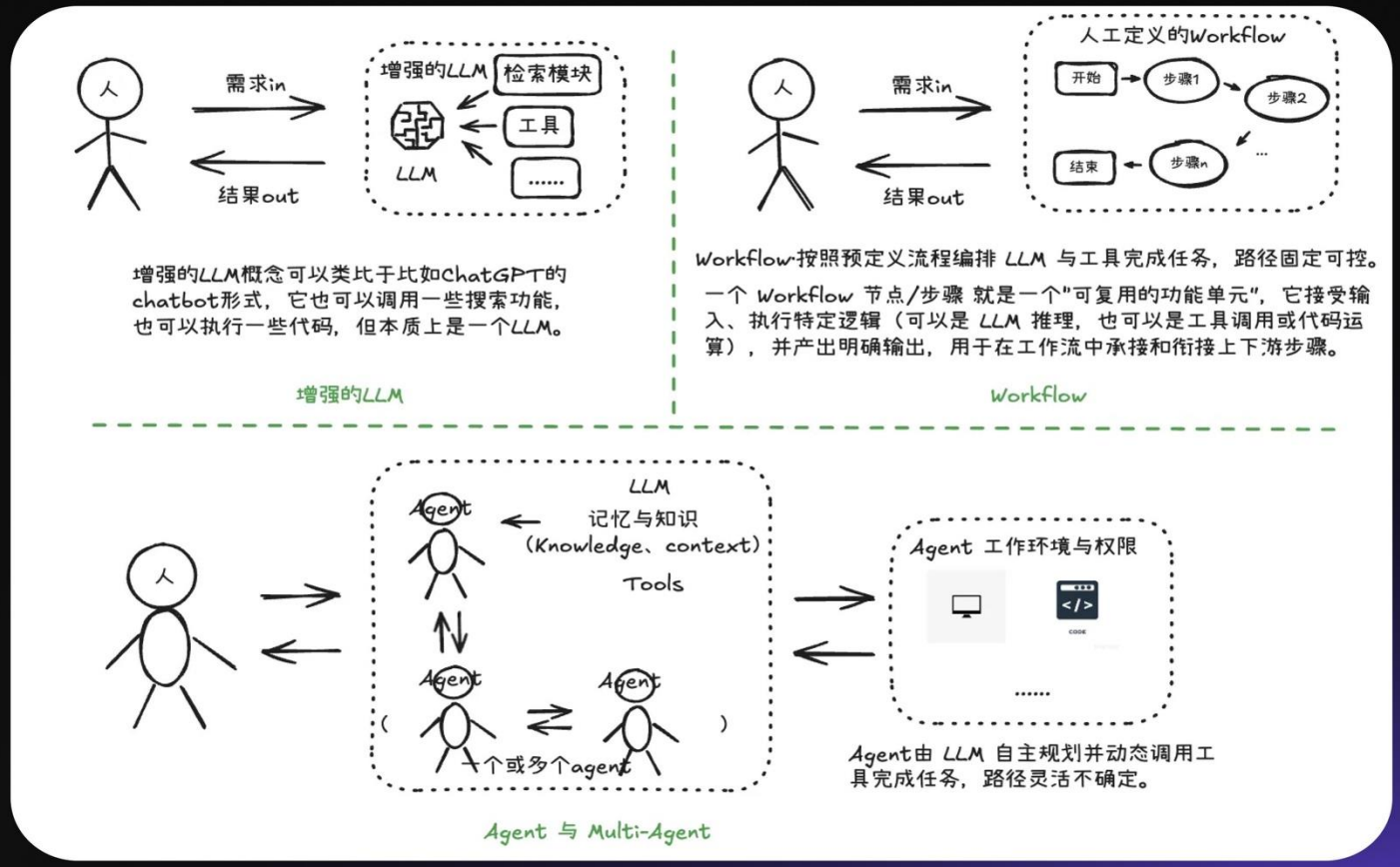
智能体工作流能否支撑企业级复杂项目？

验证方案

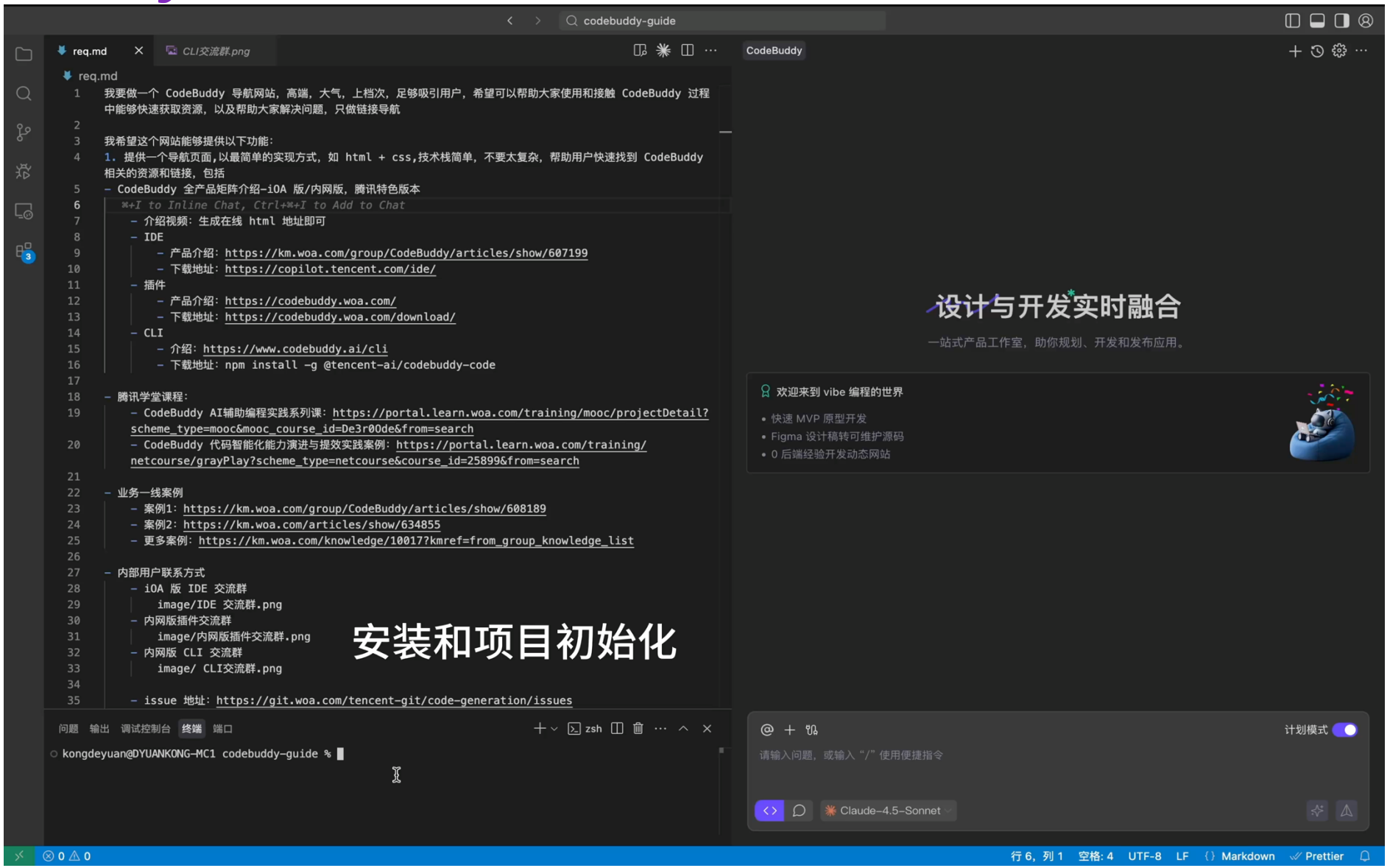
腾讯云业务中台：涉及20+系统集成、高并发要求
结果：自动化生成核心模块代码，人工干预率<15%

决策权衡

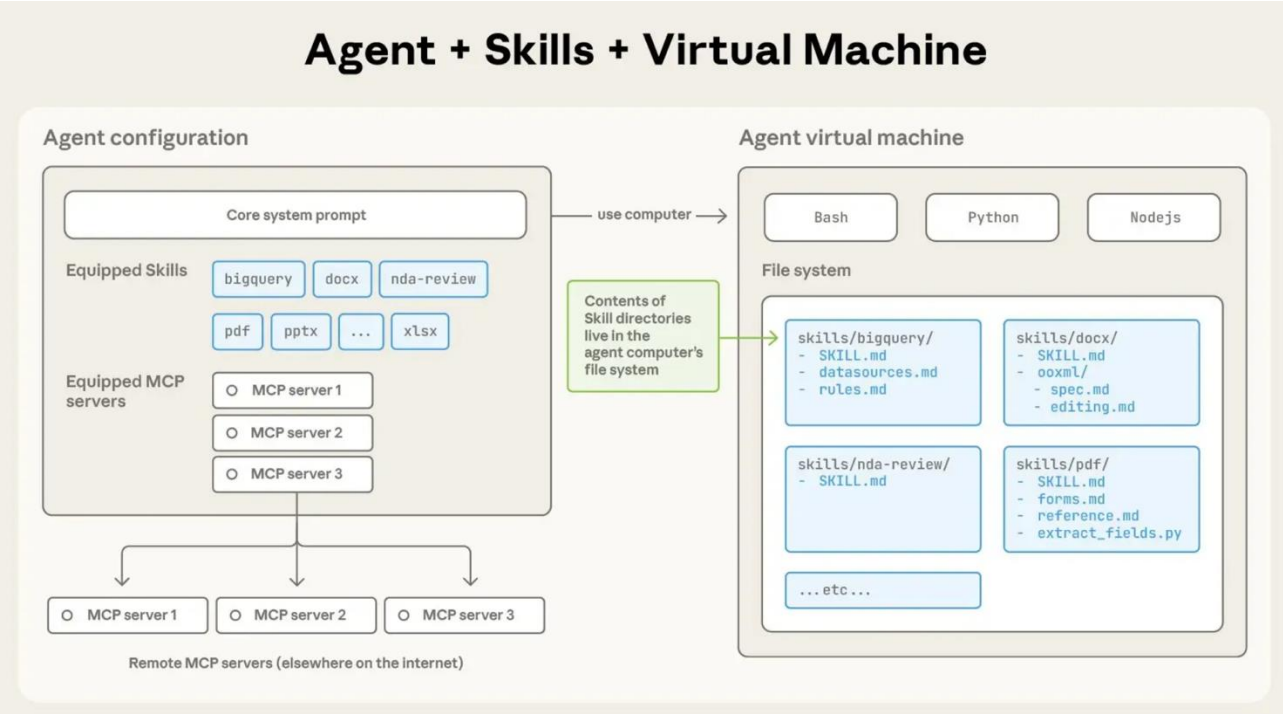
产品决策：引入MCP协议和沙箱环境。
背后的思考：面对企业级需求，产品的核心价值从“功能强大”转向 “安全可控”和“生态集成”。MCP协议的选择，体现了CodeBuddy的生态化、平台化野心——不做所有工具，而是成为连接一切工具的智能中枢。沙箱环境则是对“信任”这一企业核心诉求的产品化回应



CodeBuddy IDE 产设研实战 – 规模验证 – 演示



CodeBuddy IDE 产设研实战 – 效果验证 Skills



效果稳定输出，是一种上下文工程约定和指导。

实战演示

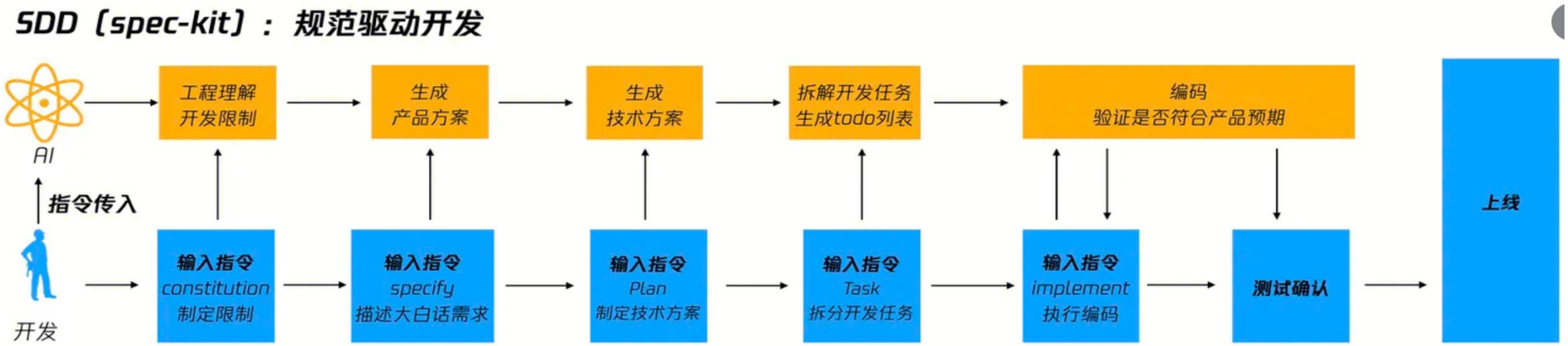

```
-zsh ⌘1 | xiaohongshu-mcp-darwin-arm64 ⌘2 | node ⌘3

12306-mcp-server                awesome-cursorrules                 demo                            kdy-cli-agent                   super-mall
AI-For-Beginners                bilibili-mcp-js-main-trial2.zip     demo-kdy                        kdy-k1                          system-prompts-and-models-of-ai-tools
AIIDE-Demo                      bilibili-mcp-server                 demo1                           kdy.txt                         tech-decision-prompt.md
CSIG进行时                      cankao-claude-code                  demo1111                        kdykdy                          test
CodeBuddy-IDE-WebSite           chatbot                             deyuankong                      kkkkk                           testD2C.zip
IDE-Docker                      chonggou                            gemini-cli                      kkkkkkk                         test_cases
Kiro                            claude-code                         genie-codebuddy-code            mcp                             tiyan
QQ_Music                        claude-skills                       genie-codebuddy-repo            mcp-kdy                         toupiao
TerminalCoding.sh               codebuddy-ccpm                      genie-ide                       md                              travel
Xcode-demo                      codebuddy-cli                       git                             mowen-mcp-server                trpc-promot
admin-user-management           codebuddy-code                      github-spec-kit-codebuddy       prompt-eng-interactive-tutorial vibe-coding
agent-practice                  codebuddy-course                    go-travel                       ruoyi-go                        vibe-coding-guide
agent-rules                     codebuddy-docs                      gongfeng                        sepcs-rules                     waveforge-vibe-coding
agents.md                       codebuddy-guide                     gongfeng-codebuddy-cli          spec-driven-development-rules   wework-bot-mcp-server
ai-doc-search                   codebuddy-guide-website             gongfeng-codebuddy-code         spec-rules                      wxauto-mcp
authentication-service-java     codebuddy-mall                      java                            std.txt                         xiaohongshu-login-darwin-arm64
authentication-service-java.zip codebuddy-official                  java-ent-service                student-system                  xiaohongshu-mcp-darwin-arm64
awesome-claude-code             codeguide-starter-pro               java-ut-carts                   subo-codebuddy-repo             xiaohongshu-mcp-darwin-arm64.tar.gz
awesome-codebuddy               coze-studio                         kdy                             super-mail                      员工礼品选款系统
kongdeyuan@DYUANKONG-MC1 awesome-repos % chmod +x xiaohongshu-mcp-darwin-arm64

kongdeyuan@DYUANKONG-MC1 awesome-repos % chmod +x xiaohongshu-mcp-darwin-arm64
./xiaohongshu-mcp-darwin-arm64

zsh: killed     ./xiaohongshu-mcp-darwin-arm64
kongdeyuan@DYUANKONG-MC1 awesome-repos % chmod +x xiaohongshu-mcp-darwin-arm64
./xiaohongshu-mcp-darwin-arm64

INFO[0000] Registered 12 MCP tools
INFO[0000] MCP Server initialized with official SDK
INFO[0000] 启动 HTTP 服务器: :18060
[GIN] 2025/11/04 - 04:02:35 | 204 |      16.042µs |             ::1 | OPTIONS  "/mcp"
[GIN] 2025/11/04 - 04:02:35 | 200 |       589.5µs |             ::1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:02:35 | 204 |       7.25µs |             ::1 | OPTIONS  "/mcp"
[GIN] 2025/11/04 - 04:03:06 | 200 |     617.834µs |             ::1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:03:06 | 202 |      23.625µs |             ::1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:03:06 | 200 |     137.833µs |             ::1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:03:12 | 200 |    2.431375ms |             ::1 | POST     "/mcp"
INFO[0272] MCP: 检查登录状态
WARN[0272] failed to load cookies: failed to read cookies from tmp file: open cookies.json: no such file or directory
[GIN] 2025/11/04 - 04:03:26 | 200 |      8.13566s |             ::1 | POST     "/mcp"
INFO[0345] MCP: 检查登录状态
WARN[0345] failed to load cookies: failed to read cookies from tmp file: open cookies.json: no such file or directory
[GIN] 2025/11/04 - 04:04:37 | 200 |  6.250359583s |             ::1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:06:39 | 200 |         328µs |       127.0.0.1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:06:39 | 200 |     325.833µs |       127.0.0.1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:06:39 | 200 |      389.75µs |       127.0.0.1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:06:39 | 202 |      57.083µs |       127.0.0.1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:06:39 | 202 |      24.416µs |       127.0.0.1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:06:39 | 202 |      36.916µs |       127.0.0.1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:06:39 | 200 |     685.167µs |       127.0.0.1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:06:39 | 200 |      687.75µs |       127.0.0.1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:06:39 | 200 |     685.375µs |       127.0.0.1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:06:48 | 200 |    1.696541ms |       127.0.0.1 | POST     "/mcp"
[GIN] 2025/11/04 - 04:06:48 | 200 |    1.815459ms |       127.0.0.1 | POST     "/mcp"
```

关于 SDD 编程 + CodeBuddy 全系工具 落地方案

经验总结

- **明确表达目标与动机**（What/Why），**规避过早技术选型**（不要在规格阶段锁死 How）。
- **反复迭代与澄清**，先完善规格，再进入实现。
- **先验证计划**（可行性、契约、测试口径）再编码。
- **让代理/自动化负责细节实现**，人侧把控规格质量与决策。

未来 – SDD 编程



PART 02

工具选型实践与与智能协作的未来展望

三种形态的专业形态

AI编码辅助
(CodeBuddy Plugin)

特征
- 传统开发者主导，AI打辅助的编码模式
- 手动调试过程，依赖于开发者的技术基础
- 通过对话给出建议，手动复制改动（不信任AI托管）

效果
- 编码过程推荐行生成
- 通过对话解决问题，需要多次修复试错

专业性
- 较高，需了解编程底层技术，需自主学习能力和积累项目中实战经验，靠自身经验来驱动AI辅助编码

全智能程度

较弱，AI打辅助，单轮/多论改动后需要人的确认

规约编程驱动产设研编码平台
(CodeBuddy IDE)

特征
- 基于AI 智能体和AI对话至上，采用自然语言描述需求，实现多文件代码生成，生成执行的应用
- 精准描述需求和任务表达有助于 AI 生成代码质量

效果
- 可视化的工程级别开发，但需等待AI完成看效果，期间不能协同完成。中等效率

专业性
- 中等，需要学习IDE的操作，非程序员(如 产品、设计等小白用户）也可编码，侧重和锻炼表达功能能力
- 依赖于IDE安装，需要适配各种操作系统
- 应用规模偏小，较大项目替换成本较高，且不敢乱用

全智能程度
- 中等

软件智能体
(CodeBudddy Code – CLI)

特征
- 面向软件工程全链路而生
- 面向无GUI环境下的软件智能体运行
- 覆盖所有的环境，企业级通用适配：一套命令适配所有环境要求
- 自动化与可组合性：无缝接入 CI/CD、DevOps
- 面向 AI 编程未来：Agent + CLI 驱动研发自动化
- 高性能与安全合规：支持沙箱隔离运行

效果
- 结合规约文档，并行完成任务、修复缺陷到版本提交合并发布自动化
- 结合软件工程端到端的场景丰富，结合DevOps覆盖面更广

专业性
- 高，面向专业开发者、专业的软件开发模式

全智能程度
- 较高，可以切换YOLO/SOLO模式，可以结合云端环境启动Background agent

未来，三种产品形态会一直并存，满足不同需求

回归专业工程全链路智能体形态 - 软件工程趋向自主智能

AiDD 8th 2025

编码调试自动化

•[异步] 静态分析与日志收集：自动检测质量问题，捕获并分类错误日志

[Hooks] 自动化测试与覆盖率：文件保存后，可以根据规则生成测试数据、运行单元测试，甚至异步完成覆盖率报告

智能调试工具：集成断点、变量监控与调用栈分析

评审阶段与构建发布代码自动化

•AI代码审查与CI流水线：预筛选代码，自动编译、测试、打包

多环境部署与回滚：支持开发、测试、预发环境一键切换

版本与产物管理：自动生成版本信息并进行安全检查

业务验证自动化

- 端到端与接口测试：模拟用户流程，验证API功能与性能
- 单元测试和行为测试（BDD）验证功能正确性

从辅助编码到自主智能体的工具的跃迁

智能编码辅助应用开发

专业开发团队的项目，需要业务人员借助IDE可视化能力，来观察确认AI的每一步步骤的完成，如：

- 编码辅助补全
- 产设研一体的应用开发
- 数据分析工具
- 语法可视化及编译调试
- MVP 原型验证项目

企业生产应用

大规模复杂工程级生产应用，需要继续保持团队协同开发到发布的软件工程体系。AI需要另一种形态，参与驱动开发，并得到更好效率提升

- 企业级应用架构复杂
- 从编码到版本管理发布
- 需求/缺陷等并行开发自动化
- 与业务团队协作和规范化代码治理、评审、纠错

产品形态

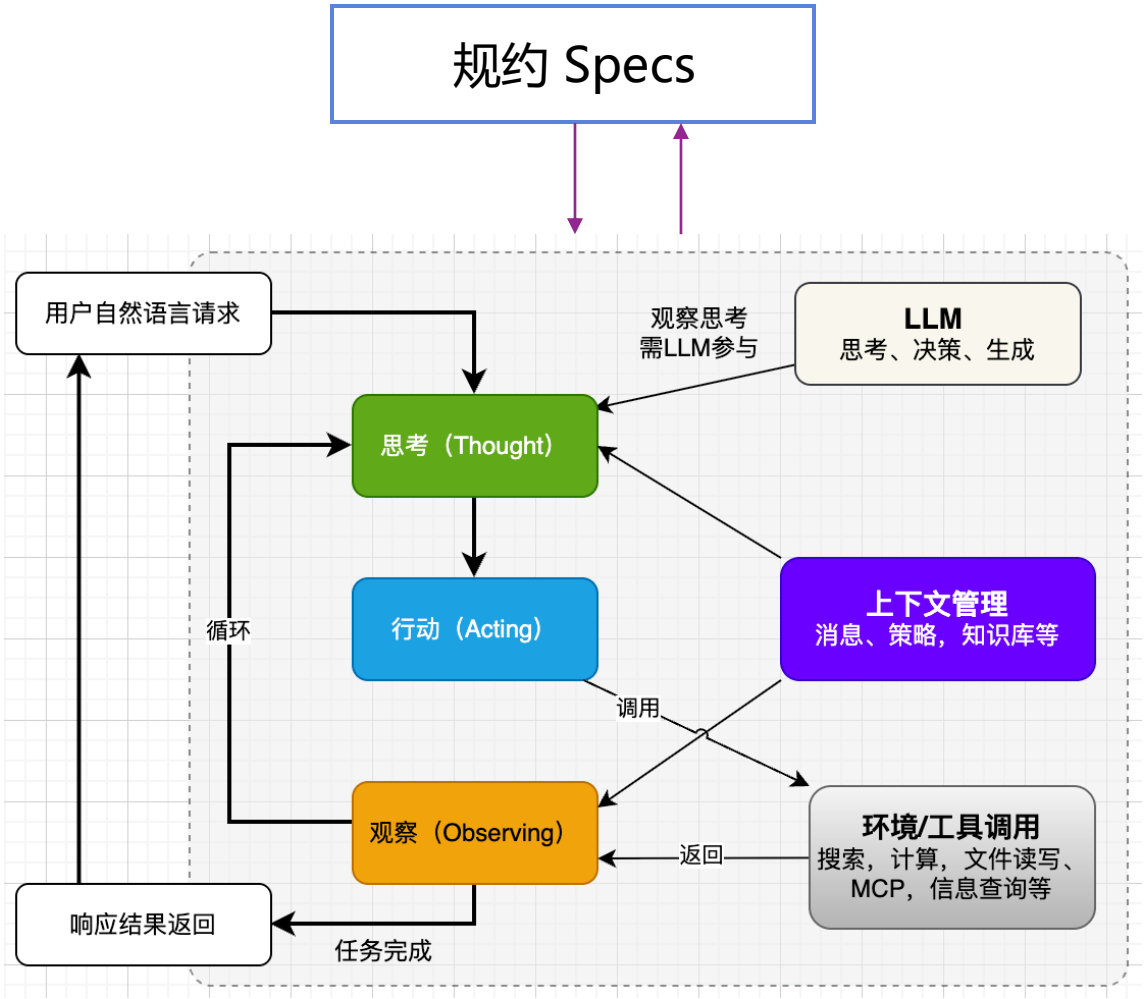
需要IDE能力，集成开发资源，面向代码加载语言LSP协议，帮助开发者智能辅助开发和运行调试，过程中AI作为辅助生成

产品形态

AI 全链路参与团队一起，并行完成各项任务。而IDE给开发者使用，对于AI而言，异步化、智能化、无干预、连接软件工程各个环节，CLI会是更好的一种产品形态。

代码智能体 | Craft 设计理念

- 基于 ReAct 范式，结合多种扩展组件，系统性提升智能体的自主编码能力
  - 环境/工具集定义：项目探索，代码编辑，命令执行，MCP扩展等
  - 思考模式：调用工具前，进行思考过程，包括：任务拆分，错误分析，需求理解，反思等
  - 上下文管理：阶段式消息总结、策略调整和记忆管理等
  - 工程加速：Prompt Caching, 减少重复计算，提升响应效率



人与智能体的协作 趋向于 自主化、管理化、异步化

从 AI Pair Programmer 到 Autonomous Contributor

AI Pair Programmer：智能体与开发者共同工作，提供建议和自动化的辅助功能。

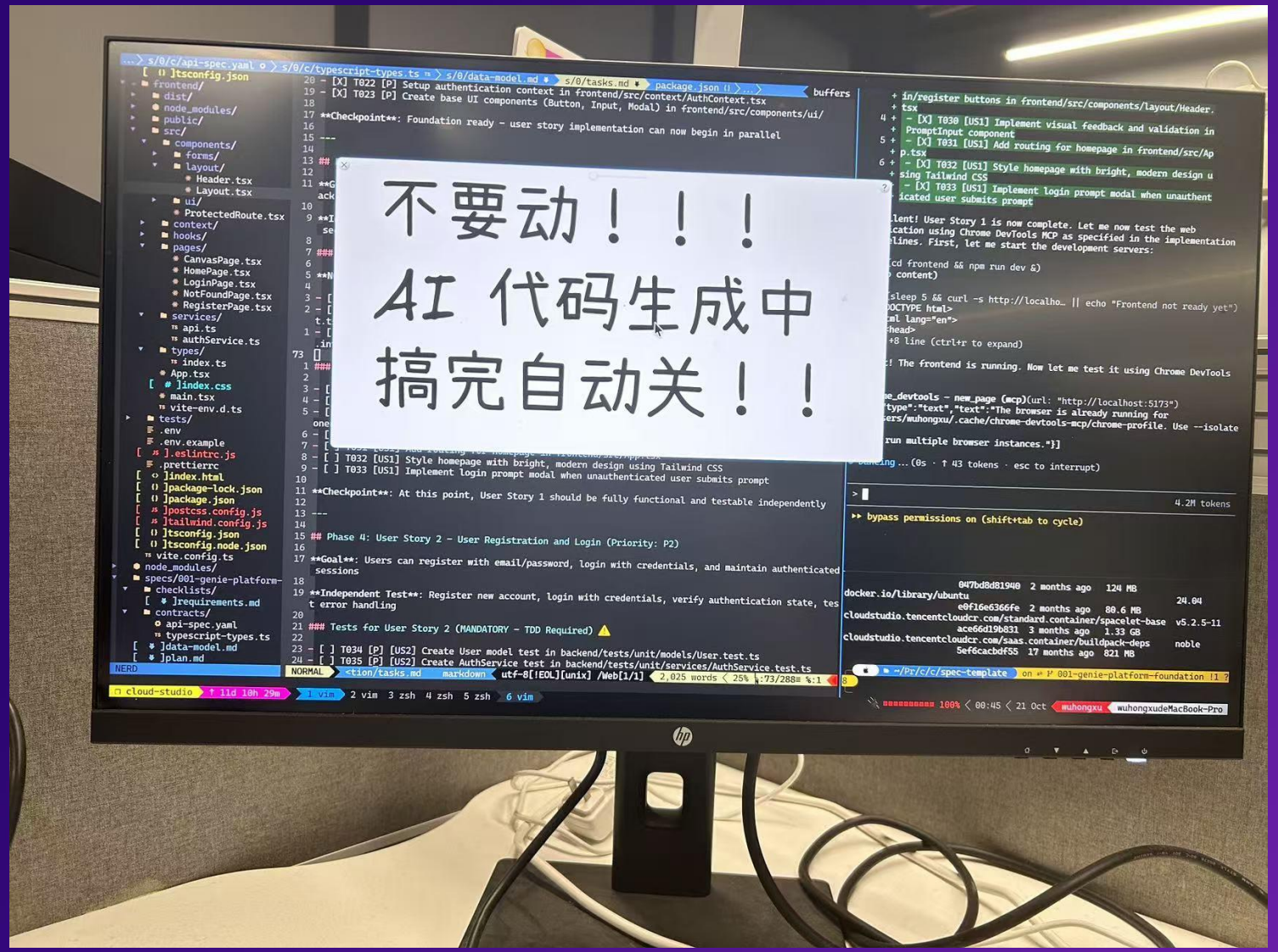
Autonomous Contributor：AI 不仅辅助开发者，还能自主承担开发任务，完成代码编写、修改、甚至部署等工作。

AI 智能化 CLI 会是软件工程 3.0 的基石

自主 AI 驱动软件全链路，全新范式

层级	核心能力
L1	文件级代码生成 专注于代码补全和对话式提供代码生成意见。
L2	任务级自动化 工具能基于描述性提示处理功能开发
L3	项目级自动化 集成平台实现需求收集→代码生成→PR创建→部署的多步骤自动化
L4	AI 软件工程师 从产品需求到生产部署的全流程自动化，使非技术人员也能快速创建完整软件产品
L5	AI开发团队 将出现协作式AI代理系统。多个AI代理可分工协作处理不同开发环节。

最终的软件开发工程师电脑，可能会这样...



产品理念：
让 AI 能严格按照规约完成既定任务、需求。

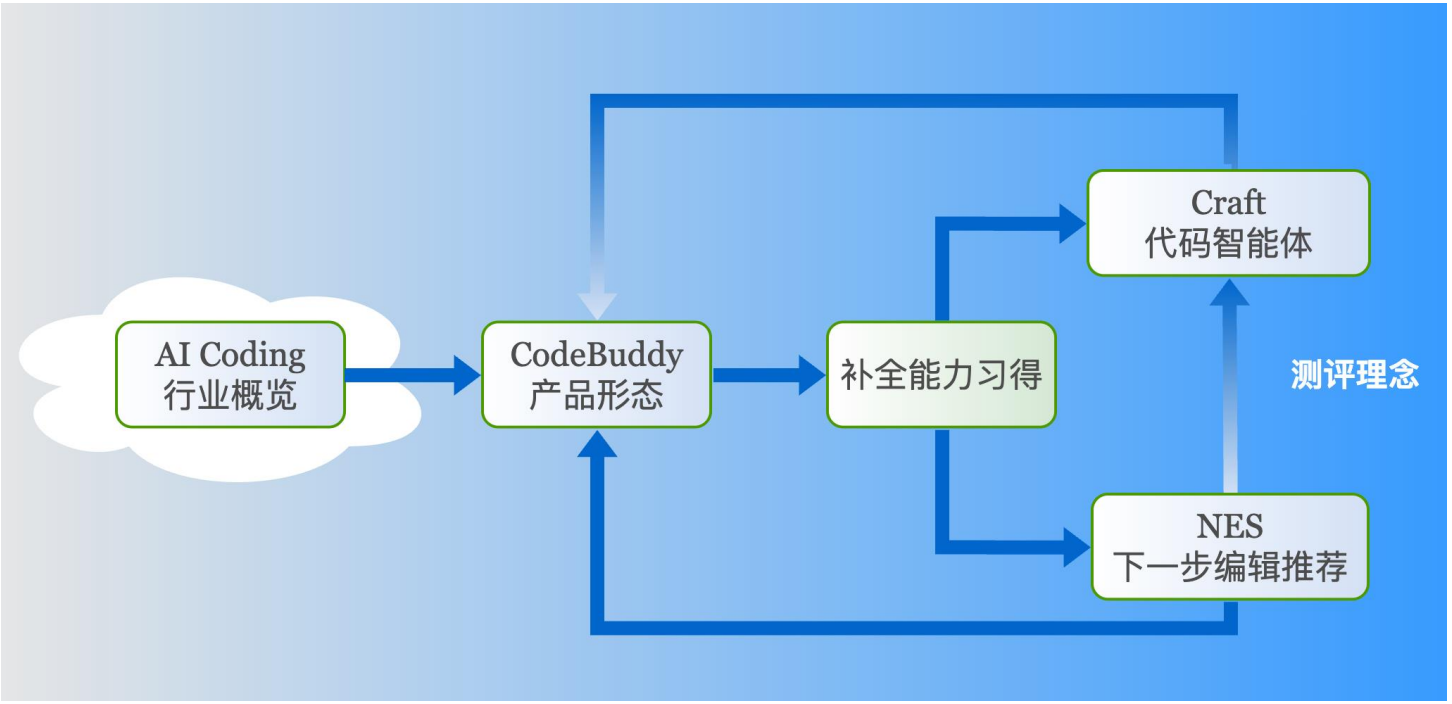
当规约清晰，我们只需要必要的时候确认即可，也可以不确认，你自己干吧。

PART 03

总结

给多种形态的粒度，按需选择

- AI Coding 继续站在LLM产业浪头
- CodeBuddy 的plugin、IDE、CLI 三种产品形态，致力于为“全民”编码，构筑最佳用户体验
- 关于1）传统补全能力，2）人机协作的编辑代码推荐，3）AI为主、用户验收的Craft模式的一些实践



科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

AiDD 峰会

AiDD峰会 上海站
AI+研发数字峰会
时间：2026.05.22-23

AiDD峰会 北京站
AI+研发数字峰会
时间：2026.08.21-22

AiDD峰会 深圳站
AI+研发数字峰会
时间：2026.11.20-21

AiDD峰会详情

AiDD × AI+产品峰会

AiDD × 上海站
AI+产品创新峰会
时间：2026.01.16-17

AiDD × 北京站
AI+产品创新峰会
时间：2026.08.23-24

产品峰会详情

AiDD × K

2026 AI+ PRODUCT INNOVATION SUMMIT

01.16-17 · ShangHai

AI+产品创新峰会

Track 1: AI 产品战略与创新设计

从0到1的AI原生产品构建

论坛1: AI时代的用户洞家与需求发现

论坛2: AI原生产品战路与商业模式重构

论坛3: AgenticAI产品创新与交互设计

Track 2: AI 产品开发与工程实践

从1到10的工程化落地实践

论坛1: 面向Agent智能体的产品开发

论坛2: 具身智能与AI硬件产品

论坛3: AI产品出海与本地化开发

Track 3: AI 产品运营与智能演化

从10到100的AI产品运营

论坛1: AI赋能产品运营与增长黑客

论坛2: AI产品的数据飞轮与智能演化

论坛3: 行业爆款AI产品案例拆解

2-hour Speech: 回归本质

用户洞察的第一性

——2小时思维与方法论工作坊

在数字爆炸、AI迅速发展的时代，仍然考验“看见”的“同理心”

Panel 1: 出海前瞻

“出海避坑地图”圆桌对话

——不止于翻译:

AI时代的出海新范式

Panel 2: 失败复盘

为什么很多AI产品“叫好不叫座”?

——从伪需求到真价值:

AI产品商业化落地的关键挑战

智能重构产品 数据驱动增长

Reinventing Products with Intelligence, Driven by Data

80+ 业界大咖 汇聚产品大咖的真知灼见

40+ 创新案例 开拓全新AI+产品视角

10+ 分论坛 涵盖产品到商业增长的全链路

800+ 听众规模 共同探索产品的智能重构

扫码查看会议详情

第8届AI+研发数字峰会
AI+ Development Digital Summit

感谢聆听！

扫码领取会议PPT资料