

神州问学

AI Coding内部探索

开发流程和团队模式的重塑

演讲人：余金辉

神州数码通明湖云和信创研究院 高级AI交付工程师

AiCon

全球人工智能开发与应用大会



个人介绍



余金辉

- ◆ 交付团队 TL
- ◆ 团队内部 AI Coding 推行者

拥有多年一线交付实战经验,聚焦 AI Coding 对团队效能的提升探索,致力于通过 AI 技术赋能研发全流程,推动团队交付能力升级

目录

01 困境：空有“屠龙刀”而无“屠龙技”？

02 探索：从诉求出发，看“技”在何方

03 破局：从“单点突破”到“整体提效”

04 团队新范式：POD模式

极客邦科技 2026 年会议规划

促进软件开发及相关领域知识与创新的传播



参会咨询



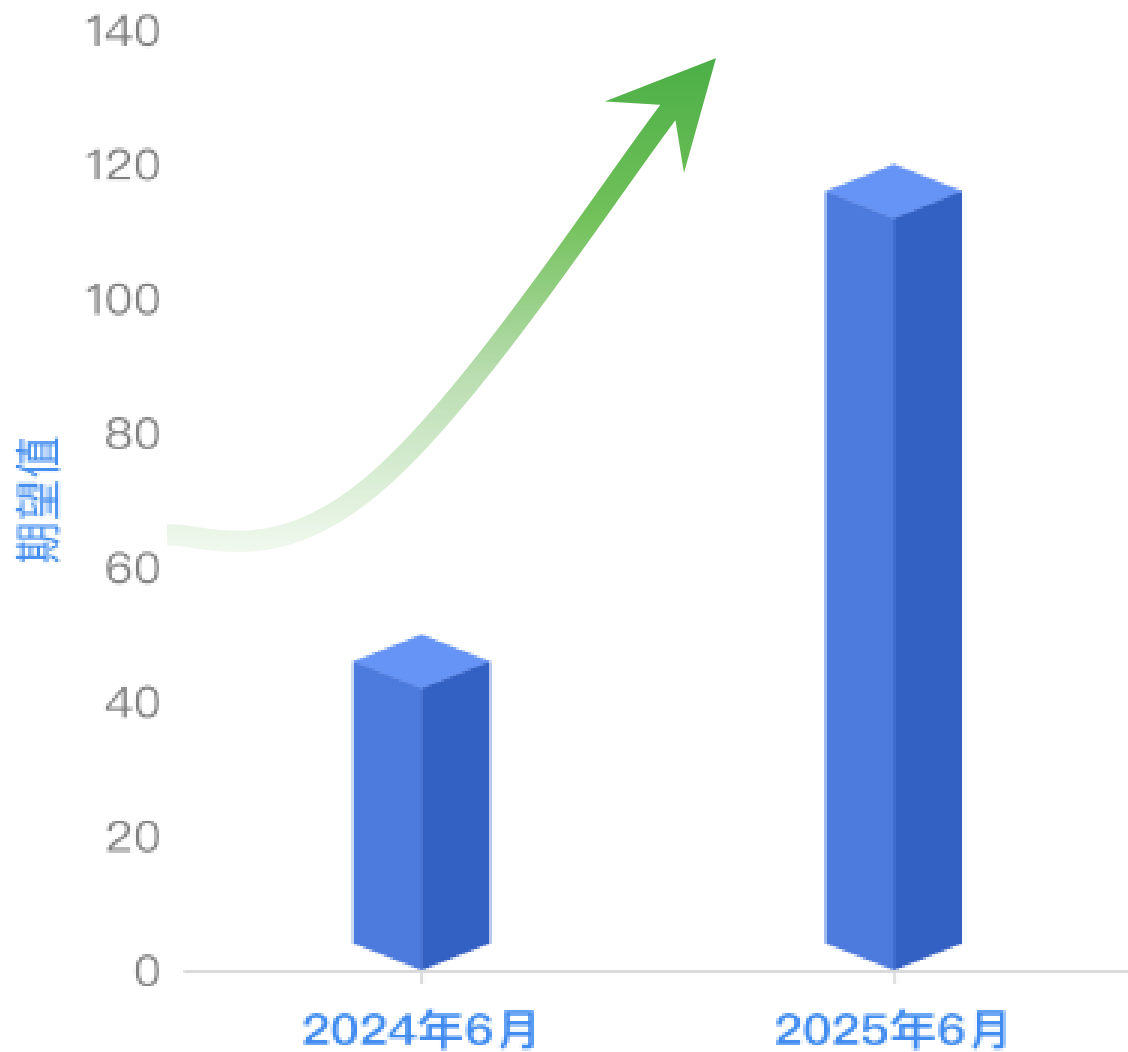
查看会议



01

困境：空有“屠龙刀”而无“屠龙技”？

■ 起因：AI Coding带来的无限遐想



AI工具的爆发式涌现

从神码内部开始统一采购Copilot的一年时间里，AI工具的迭代速度令人瞠目结舌，期间又涌现了Cursor、Trae等热门工具，大大的提升了开发者的工作效率

超级团队的初见端倪

AI工具的不断进化，让更少的人创造更多的价值成为可能。未来可能更多的是以“小而精”的团队进行作战

现状：“宝刀”难“屠龙”，效益难兑现



持续告急

后端人员持续饱和，前端人员一直紧缺。
项目数量并无明显变化，人员紧张程度亦无改善



未见成效

同类型项目在人员投入不变的情况下并没有缩短交付周期，以前是什么样，现在还是什么样





探索：从诉求出发，看“技”在何方

从问卷中看方向

人员分类 (共33人)	使用工具	用途	不足	核心诉求
前端 (8人)	<ul style="list-style-type: none">Copilot: 6人cursor: 3人豆包: 2人Deepseek: 2人文心快码: 1人Codegeex: 1人通义灵码: 1人Trae: 1人	<ul style="list-style-type: none">代码补全: 6人代码优化: 4人代码解读: 4人代码生成: 4人错误排查: 3人问题答疑和学习: 2人页面生成: 1人	<ul style="list-style-type: none">胡编乱造无法理解代码上下文生成的代码需要人后续改造代码和需求不符	<ul style="list-style-type: none">对于简单的页面能快速准确生成Desin to code,代替人去反复描述能够正确理解代码上下文
后端 (25人)	<ul style="list-style-type: none">Copilot: 16人cursor: 7人ChatGPT: 5人通义灵码: 4人Trae: 4人Fitten Code: 2人Codebuddy: 1人cherry studio: 1人Claude: 1人Cline: 1人豆包: 1人Deepseek: 1人文心一言: 1人	<ul style="list-style-type: none">代码补全: 20人代码解读: 18人代码生成: 11人代码优化: 5人错误排查: 4人注释补充: 1人接口文档编写: 1人语言转换: 1人代码评审: 1人	<ul style="list-style-type: none">无法完全理解需求, 需要不断确认缺乏对整体项目的理解, 只局限于1、2个文件生成的代码质量不高	<ul style="list-style-type: none">增强对项目整体的理解能够根据自身的风格编写代码提高生成的代码质量提升业务逻辑理解能力

总结:

- ✓ 主要用途集中在代码补全、解读、优化、生成和排查。
- ✓ 仍然有50%以上的人在项目开发过程中未使用AI生成代码。
- ✓ 普遍反馈代码质量不高、缺乏对项目整体的认知、对需求理解不到位, 即不能奢望通过简单的描述一步到位健壮的实现业务需求
- ✓ 前端对Desin to code的呼声比较高, 至少简单的页面是要能够实现的

从外部交流中看方向



全流程，全链路；有输入，有输出

不局限于某一环节，从以结果为导向的视角来看，整体的提效才是真正的提效；明确每一环节的输入和输出，确定的场景才能更好的实现单点突破

不“东施效颦”，要“因地制宜”

要以自身实际情况为依据，单纯的模仿只会空耗人力物力。以真实的内部诉求作为考量的基础，代码评审和功能测试用例生成值得借鉴，但也要做适当调整

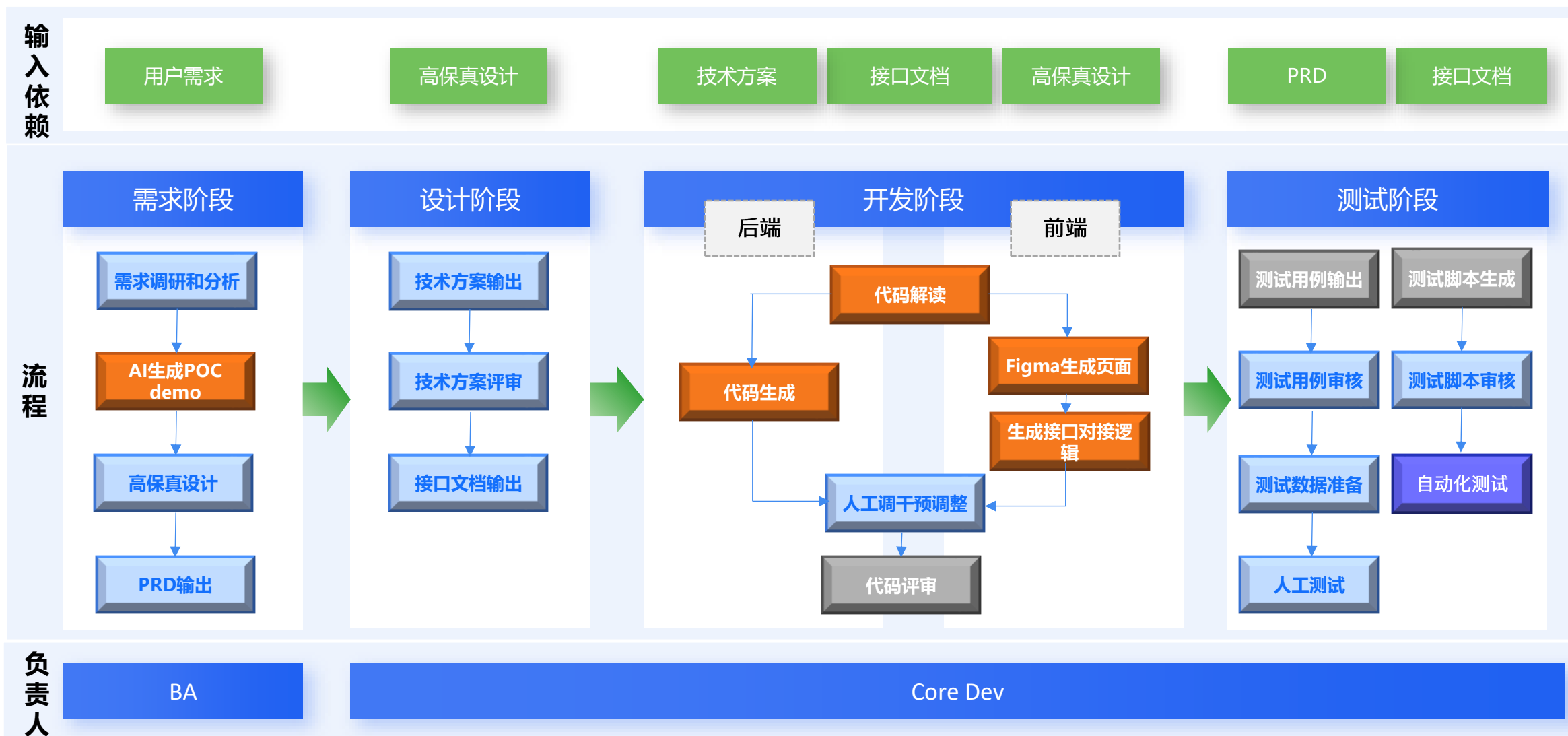
方向

03 破局：从“单点突破”到“整体提效”

全景图：从需求到上线，走适合团队的提效之路

已实现

实践中














任务实践：找到每一步的最优解



序号	所属交付阶段	场景名称	场景详细描述	使用的AI工具	AI工具使用方式	关键规范约束	工具表现评估	预期效率提升量化	主要挑战
1	编码阶段	Figma 生成页面	由于infcode暂不支持MCP，使用的Figma的JSON描述UI去生成	infcode	-输入：根据以下Figma的JSON描述文件生成一下页面（JSON太长略） -输出1：无法新建文件（如M列） -输出2：输出代码，然后手动新建文件，引入（如N列）	「JSON 与前端实现——映射」：JSON 字段需对应前端 DOM 结构、CSS 样式、JS 交互逻辑，无冗余 / 缺失属性； 「可复用性优先」：组件化拆分、样式变量化，降低前端二次开发与维护成本； 「兼容性兜底」：编码需考虑前端浏览器 / 终端适配，避免 AI 生成不可落地的特性。	准确性：能基本还原 JSON 中定义的页面结构（如框架层级、元素类型），但部分细节存在偏差，例如文本样式（字体大小、颜色）与 JSON 描述有出入，矢量图形路径偶发变形，图片填充比例不符合预设的「COVER」模式。 效率性：生成过程耗时中等，对于包含 100 + 元素的复杂 JSON，存在明显卡顿；批量处理同类组件（如表格多行数据）时，重复元素生成速度较慢，未体现批量优化能力。 稳定性：多数情况下能输出完整页面，但在处理嵌套层级超过 5 层的 JSON 时，偶尔出现元素丢失或位置错乱；相同 JSON 多次生成的结果存在细微差异（如间距误差 1-2px），输出质量一致性不足。 易用性：基础操作流程清晰，但对 JSON 格式的容错性较低，若存在未定义的属性（如自定义动画参数），会直接中断生成并报错，缺乏友好的提示和自动修复功能。	在规范使用标准 JSON 格式的前提下，相比纯手动搭建页面，可节省约 40%-50% 的基础布局时间，但需额外花费 15%-20% 的时间修正生成结果中的细节错误（如样式偏差、层级混乱）。 工作量减少程度有限：对于简单页面（元素数量 < 50），能减少约 50% 重复操作；但复杂页面（含表格、动态组件）的实际工作量减少不足 20%，主要因后期调整成本较高。 质量提升不明显：生成页面的基础结构完整性达标，但细节精度（如对齐、间距一致性）仍依赖人工校验，质量提升指标仅能达到「减少低级排版错误」的基础水平。	工具稳定性：对复杂 JSON 结构的解析能力较弱，嵌套框架、自动布局等属性的还原成功率仅约 70%，极端情况下会出现页面崩溃。 细节还原精度：位置坐标、尺寸比例等数值型属性的还原误差率约 5%-8%，需人工逐元素校准，影响整体效率提升。
			使用mcp，输入地址直接生成页面	copilot	-输入：基于这个 Figma 文件（链接略），1:1 还原视觉细节，用 [React+TS] 写代码，遵循 BEM 规范 + ESLint，实现标注交互，兼容主流浏览器，保证高还原度和优质代码。 -输出：直接生成页面	设计文档规范：Figma 源文件的图层命名、组件库复用规则、交互标注格式 提示词规范：需明确页面类型、风格、组件复用要求、数据接口关联逻辑的描述标准	准确性：生成内容正确率取决于提示词与规范匹配度，若 Figma 源文件组件化程度高 + 提示词明确，页面结构 / 样式的正确率可达 85% 以上；但复杂交互（如动态表单校验）的正确率可能不足 70% 效率性：响应速度较快（简单页面生成耗时≤3 分钟），处理能力支持同时生成 2-3 个页面的并行任务 稳定性：输出质量一致性受 Figma 源文件规范度影响，若源文件组件库统一，生成页面的样式 / 结构一致性达 90%；若源文件混乱，一致性会降至 60% 左右 易用性：学习使用难度较低，熟悉 Figma 操作 + 掌握基础提示词规范后，1-2 天可上手	时间节省百分比：相比纯手动开发，页面搭建环节可节省 60%-75% 的时间（复杂页面节省比例更高） 工作流程省耗度：减少“设计 - 开发”对接的沟通成本（如样式还原争议），沟通耗时可降低 60% 以上 质量提升指标：页面视觉还原度从手动开发的 75%-80% 提升至 90%-95%；	工具稳定性：MCP 服务对复杂 Figma 交互（如嵌套组件、动效）的解析可能出现异常，生成页面需额外调试 规范约束完善度：若提示词 / Figma 规范不明确，易出现生成内容不符合需求的情况 内部集成：MCP 服务与现有项目工程化流程（如代码部署、测试）的对接需适配成本
2	开发阶段	接口文档生成代码	基于一份接口文档生成对应的接口调用工具	infcode	在线生成，输入接口文档文件和需求	要求生成接口调用的工具方法和基于接口文档的调用方法	准确性：相对准确，生成了工具方法，基于工具方法和接口文档生成了对应的调用方法，但是生成的内容需要手动调整之后使用，生成的内容有明显的错误 效率：速度一般，在生成阶段总共花了10几分钟 稳定性：1.0.1版本 比之前版本稳定有提升	相对独立的功能，能够基于文件和描述需求直接生成需要的功能，在这种类似的独立场景下效率有显著提升，若是基于生成的代码调整号之后后续的生成将更加准确和快速提升率大约有50%-70%	独立场景和明确需求时表现不错，但是一旦涉及跨多页面或者需要自主去搜索项目中内容时表现不佳 能自主搜索项目中内容，自主完善跨文件搜索表现良好
				copilot		要求生成接口调用的工具方法和对应接口的调用方法，生成的工具虽需手动调整，但无明显错误 效率性：5分钟左右生成完成的 稳定性：稳定	准确性：相对准确，生成了工具方法和对应接口的调用方法，生成的工具虽需手动调整，但无明显错误 效率性：5分钟左右生成完成的 稳定性：稳定		
3	测试阶段	测试用例生成	基于一份prd文档生成一份测试用例	infcode	在线生成，输入prd的txt文件和需求	要求生成一份测试文档	准确性：准确（prd上的要求都又对应的用例），快速生成 效率：快速		
4	编码阶段/测试阶段	测试脚本生成	根据已有接口的接口文档生成自动化测试脚本	infcode	在线生成，指定接口，要求根据接口文档生成可用的自动化测试脚本	请你阅读(文件名)中的(接口)，并仔细阅读其接口文档，生成一份可用的自动化测试脚本	准确性：准确，效果基于接口文档的详细程度 效率：速度一般	生成的测试脚本超过预期，包括了测试脚本本体与配置文件，一共500多条代码，在该场景下对程序员的开发效率有明显提升作用，提升效率大约50%	缺乏整体项目感知度，比较依赖于用户描述的精细度和接口文档本身的精细度
5	开发阶段	代码优化	基于现有python类进行代码优化	infcode		**我的原始 Python 类为DataMigrationService **我的优化目标： ** 1. 遵循单一职责原则（SRP），将一个类拆分为多个。 2. 明确新类之间的依赖关系。 **关于这个类的职责描述：** - 职责 A：根据创建时间范围迁移飞书日程数据到Milvus。 - 职责 B：根据创建时间范围迁移周报数据到Milvus数据库中 - 职责 C：根据创建时间范围迁移飞书会议数据到Milvus - 职责 D：根据创建时间范围迁移飞书任务数据到Milvus	准确性：基本能理解用户的代码优化需求并根据用户的需求进行重构代码，但是重构后的代码出现了多出编译错误，agent识别到了其中一处编译错误并尝试改正，但最终版本还是存在编译错误 效率性：对于项目代码量较少的情况，重构效率较高，但对于代码量较大的项目，重构时间较长 稳定性：1.0.1 版本 稳定性有一定提升	能够节省代码重构时间，但是重构的代码会出现编译报错，对于代码重构之后需要花费时间进行逻辑的检查，能提升30%左右的速度	1、在处理复杂的代码重构任务时表现不佳，对于多个类之间的关系的理解不够深入； 2、模型修改代码速度较慢
				copilot	输入原始python文件	准确性：基本能理解用户的代码优化需求并根据用户的需求进行重构代码，重构后的代码未出现编译错误，且代码结构完善，功能完整 效率性：效率适中 稳定性：十分稳定	能够节省代码重构时间，但是重构的代码仍然需要花费时间进行逻辑的检查，能提升50%左右的速度	1、在处理复杂的代码重构任务时表现一般，容易出现冗余代码逻辑，需要后期人为纠正	
6	开发阶段	代码解读	基于现有代码仓进行代码解读以及审查	infocode	引用项目代码文件	请对当前目录代码进行运行原理解读（主要针对于主要功能、引用依赖、实现原理等），同时对代码进行审查，从安全性、可用性的角度找出不合理的代码段	准确性：相对准确，能够找出通用不合理的位置 效率性：快速	能够节省代码阅读时间，但存在不够深入的情况，能够减少40%左右的阅读时间，审查能够提升50%左右的速度	对于大型及复杂项目，单次完成的准确性和稳定性会比较差
						当前代码构建报错，请完成错误定位并给出修复方案	准确性：在非业务场景下（常见代码问题），能够完成问题定	通用代码问题：能够快速稳定定位和处理问题，能够提升	

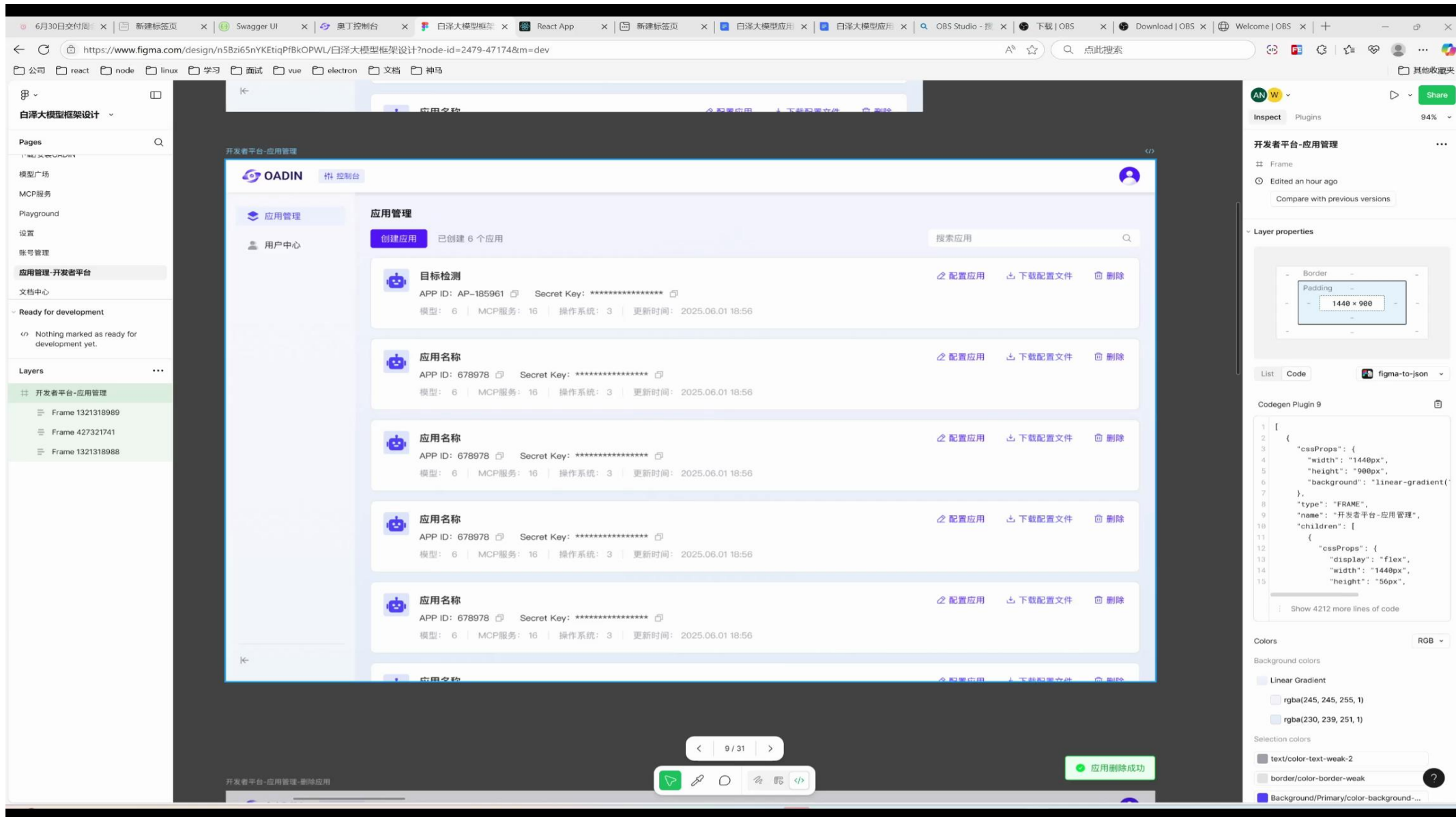


任务实践举例：AI生成POC Demo

1	模块	目的	原效果	提示词	新效果	备注
2	全系统	生成一个基础项目	/	帮我使用react, antd基于vite在当前文件夹搭建一个前端的基础项目, 该基础项目包含后续我会通过这个基础项目进行各种poc级别项目的开发		
3		丰富基础项目信息, 尽可能是常见的形式		1.项目列表页包含常见的查询条件, 列表上列出一些字段和常见的删除、编辑操作; 列表上方有新增操作 2.点击新增按钮自动跳转至新增项目页面, 目录上不需要有新增项目的目录; 3.系统管理包含常见的用户管理、角色管理		
4		在基础项目的基础上描述整体业务需求对模块进行调整		帮我在当前这个项目的基础上完成一个xx行业碳排放核算系统搭建, 该系统需要包含ai对话, 碳排放核算列表页, 碳排放核算数据录入及结果计算, 涉及到数据的部分直接使用mock数据; 整体主题色为浅绿色, 菜单包含: 对话, 碳排放核算。		
5						
6						
7						
8		按照实际业务需求调整AI生成页面		针对AI对话页面, 根据下面的需求进行页面调整 1.对话框包含上传文件及发送按钮, 对话框上面摆放三个小工具, 分别是国民行业代码查询、低碳知识问答、政策库查询; 点击小工具高亮该工具, 在输入框中展示该工具的输入提示词, 再次点击该工具取消调用, 点击其他工具进行替换, 仅支持选择一个工具, 2.点击不同工具输入框进行不同的提示: (1) 国民行业代码查询: 请输入想要查询的“国民行业”和“公司名称”查询相关信息 (2) 低碳知识闯关: 请输入“开始挑战”进行低碳知识闯关 (3) 政策库查询: 请输入想要查询的“政策名称” 3.不调用小工具: 输入框提示语为未输入任何内容时发送按钮点击失效, 输入内容后点击发送按钮跳转至对话页面		



任务实践举例：Figma生成页面



场景实践：用效率说话

方法论

细分场景

软件工程领域没有“银弹”，我们需要归纳场景，在特定的场景使用特定的工具和特定的方法去解决问题

规范流程

好的输入才有好的输出，这就对我们整个交付流程的输出物要求更高，需要更高质量的PRD、技术方案、接口文档等

对比效率

选取32名后端，11名前端，分场景，并遵守规范，比较使用AI工具和不使用AI工具的效率差异，形成最佳实践和Benchmark

+

工具



Copilot



通义灵码



TRA E

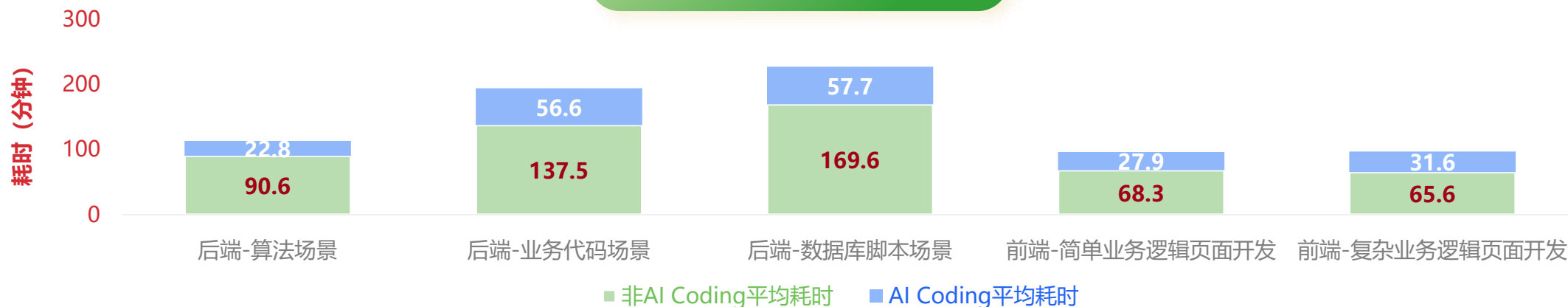
InfCode

From Tokfinity®

InfCode

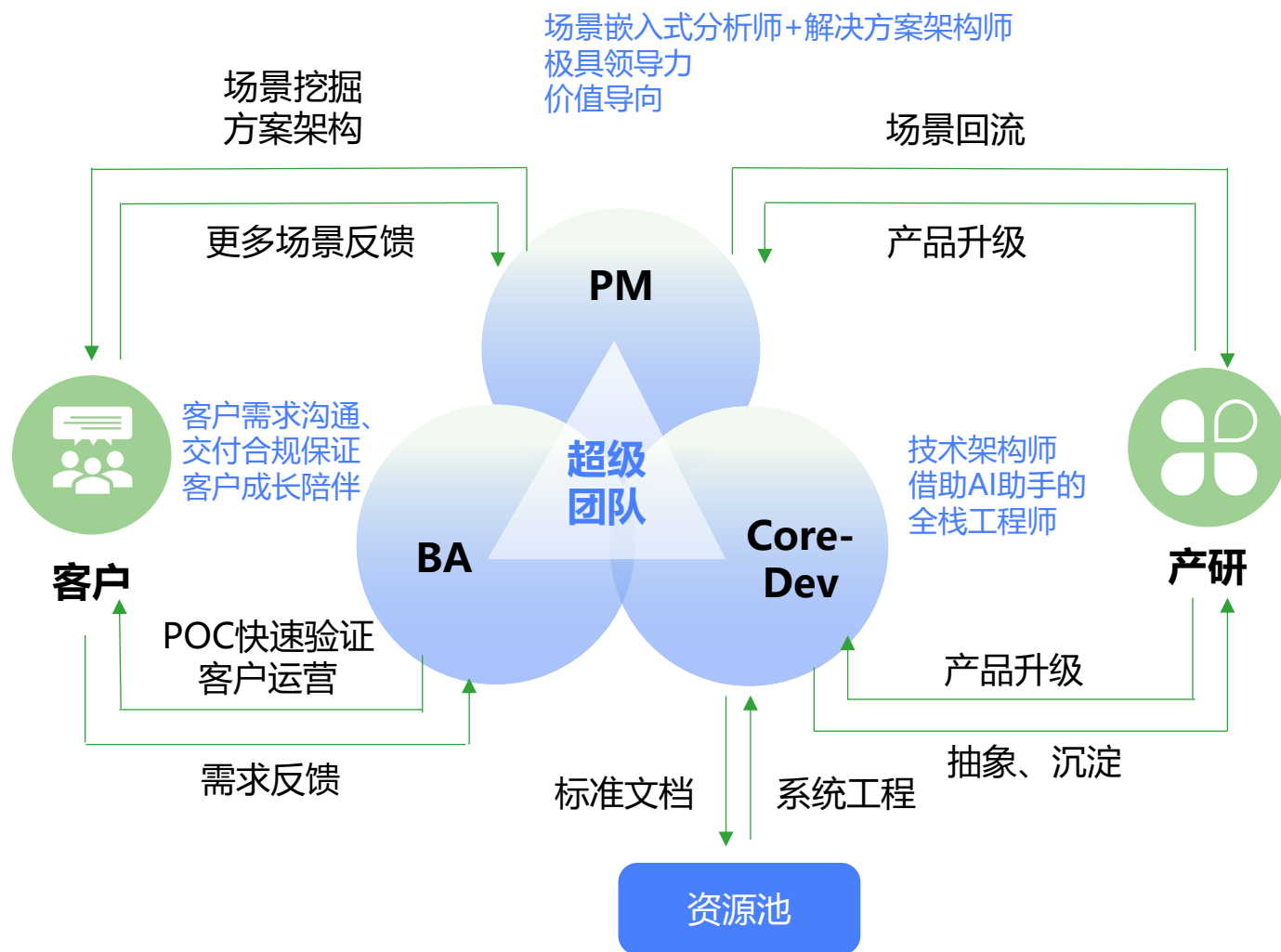
效果

五大场景AI Coding 实践对比



04 团队新范式：POD模式

超级团队蓝图



从超级团队/个人到超级能力

- ✓ 技术能力: AI原生+工程卓越
- ✓ 协同能力: 跨域协同 + AI赋能
- ✓ 业务能力: 懂业务 + AI驱动价值
- ✓ 学习能力: 快速迭代 + AI辅助成长
- ✓ 韧性能力: AI预警+稳定交付

交付更快 质量更高
价值更准 团队更优

神州问学：企业级智能体中台



场景Scale

行业知识Scale

行业工具Scale

精度提升

ROI提升

问学AI服务

文档审核 知识库 内容生成 AI4BI 多模态语义 企业情报 知识搜索 超级员工

问学Agent App Builder/Ops

应用形态 生成型应用 对话型应用 增强型应用 智能体应用 数据型应用 Manus式Agent应用 Canvas式写作应用

挂载 Prompt 知识库 工具 MCP workflow Routine 记忆

体验 多轮对话增强 对话开场白 指令 预设问题 内容审查 下一步问题建议 语音对话

Model&DatasetOps 数据生成 模型微调 模型评测 RLHF 后训练

AnalysisOps 语义建模 指标维度 归因分析 权限管控 SQL解析

AgentOps SOP Routine MCP 工具 长短记忆

DocOps 版面识别 语义切片 表图理解 多模态 Pipeline

企业特性 多租户隔离 可观测性 权限管控 单点登录 日志监控 统一API网关

问学HICA推理加速

问学HISO异构算力调度

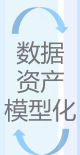


问学算力服务及问学一体机



通专融合

企业数据资产



数据治理

MCP Hub



企业现有数字化系统

AI for Process

极客邦科技 2026 年会议规划

促进软件开发及相关领域知识与创新的传播



参会咨询



查看会议



THANKS

智能未来，探索AI无限可能

Intelligent Future, Exploring the Boundless
Possibilities of AI

AiCon

全球人工智能开发与应用大会



扫码，预约专家一对一讲解