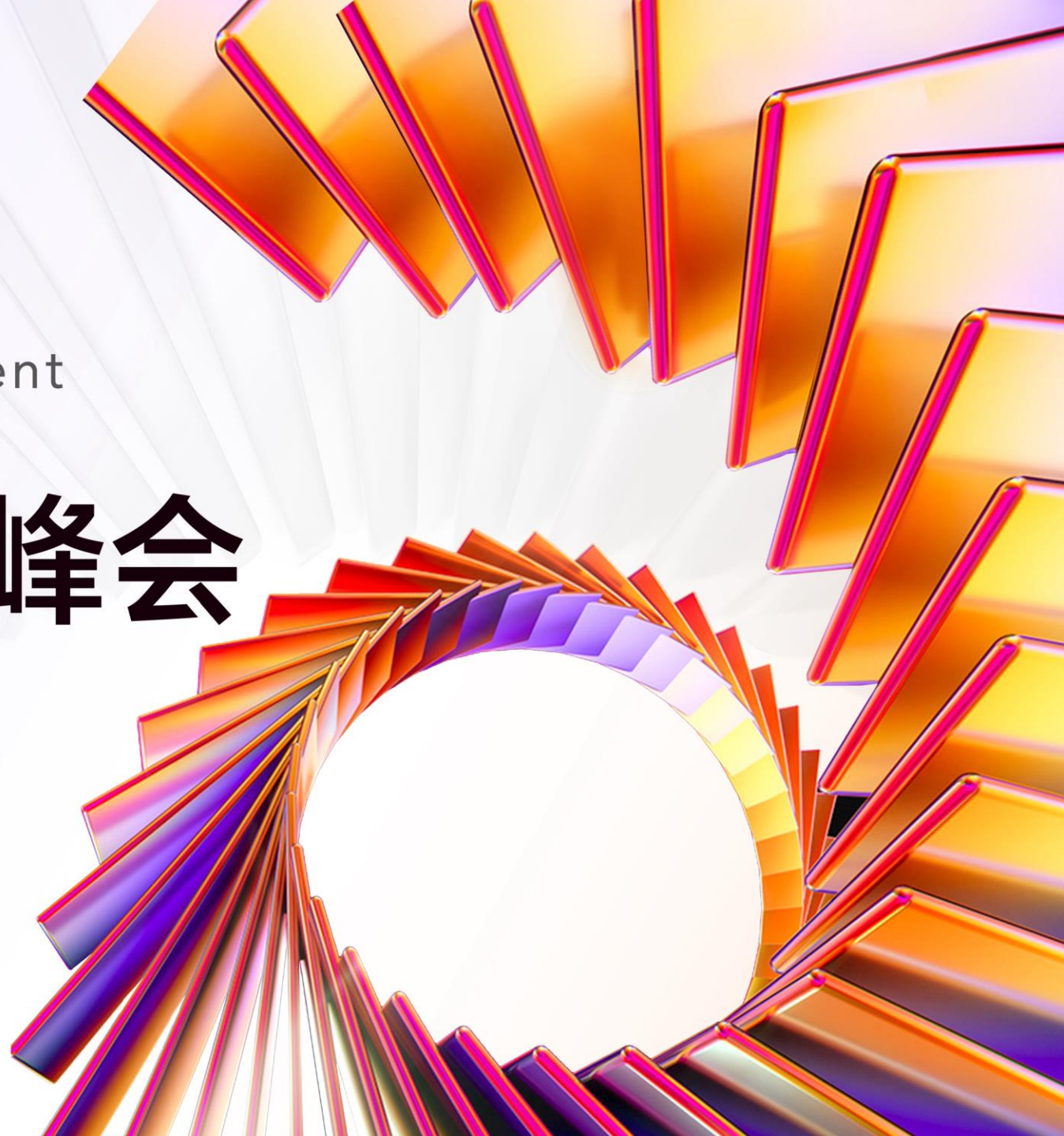
AiDD 7th 2025

第7届 AI+ Development Digital Summit AI+研发数字峰会

拥抱AI 重塑研发

8月8-9日 | 北京站



AiDD 8th 2025

第8届AI+研发数字峰会

拥抱AI 重塑研发 AI+ Development Digital Summit

查看会议详情

下一站预告 11/14-15 | 深圳站 12/19-20 | 上海站

深圳站论坛设置

智能装备与机器人

超越“编程 Copilot”

下一代知识工程

智能网联与汽车智能化

AI 测试工具开发与应用

AI 基础设施和运维

数据智能及其行业应用

可信 AI 安全工程

大模型和 AI 应用评测

多 Agent 协同框架

从智能测试到自主测试

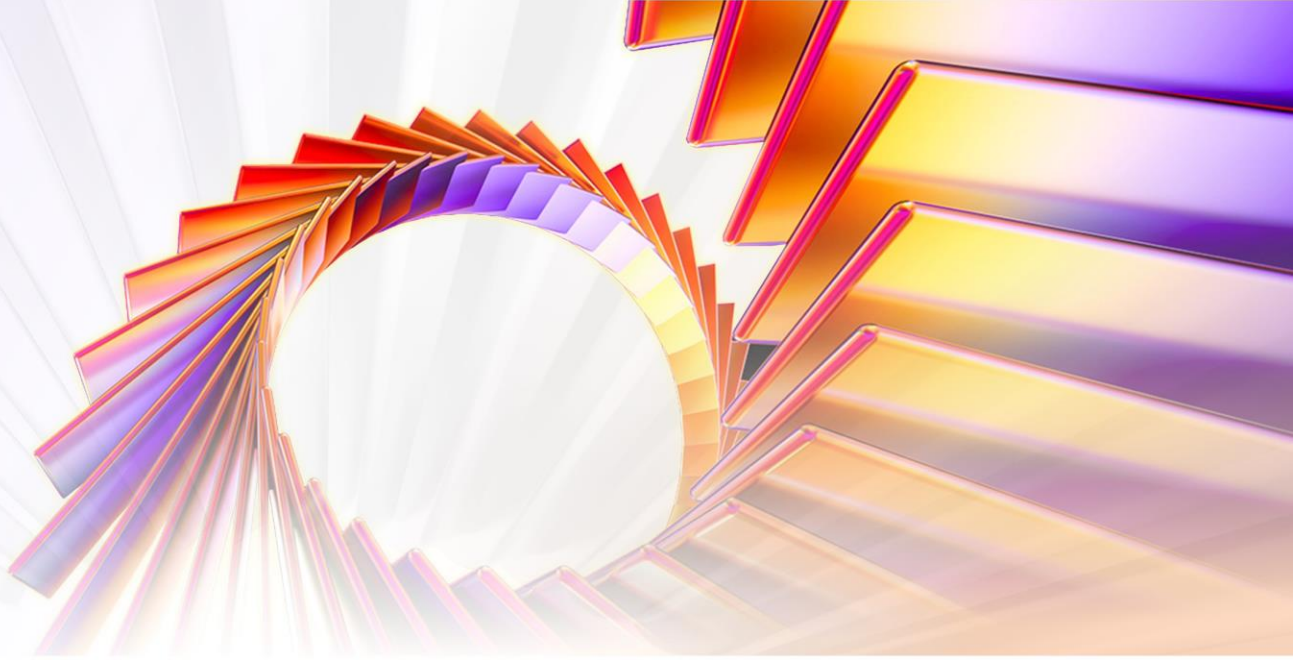
大模型推理优化

多模态 LLM 训练与应用

智能化 DevOps 流水线

上下文工程

AiDD 7th 2025 | 8月8-9日 | 北京站
第7届 AI+ Development Digital Summit
AI+研发数字峰会
拥抱AI 重塑研发



利用AI+RAG+KG挖掘代码知识宝藏

王巍巍 | 长亮科技

王巍巍

长亮科技AI研究院资深AI科学家/腾讯云名人堂专家

参编信通院团体标准《银行核心系统现代化建设水平度量模型》并被聘任为应用现代化推进中心专家，参编信通院AI团体标准《AI网关能力要求》《AI数据分类分级工具》等。曾被信通院邀请作为“MCP协议标准化研究工作线上沙龙”主题演讲嘉宾。在原生赛道，曾获得华为鲲鹏“优秀金种子开发者”，带队参加华为鲲鹏应用创新大赛金融原生开发创新赛道全国总决赛并获得银奖，对应案例也被选中进入工信部信息技术应用创新解决方案案例库。多次在各类技术大会上担任主题演讲嘉宾，在IT新媒体影响力评选中荣获“最佳影响力奖”。拥有多项发明专利，著有专著《单元化架构实践指南及AI时代思考》，有多篇AI论文发表在国内期刊。

目录
CONTENTS

PART 01

为什么基于源代码资产构建知识工程

为什么要基于源代码资产构建知识工程

知识工程解决什么问题？->解决让**计算机理解知识并且熟练运用知识（甚至用于复杂推理）**的问题。

知识工程过程：

知识获取 → 知识表示 → 知识管理 → 知识推理 → 知识应用

企业的常见痛点：

- 架构治理没有足够重视
- 关键技术文档缺失
- 关键技术文档与源代码不一致
- 关键文档非文字表述
- 企业组织结构复杂，各自为政
- 精通核心技术细节的人离职

传统基于“文本+人的经验”构建知识工程的不足：

- 知识工程的知识的精细化程度不足
- 知识有可能存在多个版本，不知道哪个版本正确
- 技术文档和代码脱节，技术文档不能反映真实情况
- 只有“显性”知识，缺乏“隐性”知识

基于“源代码资产”构建知识工程的优势：

- 源代码包含的知识 100% 准确 （反例: 技术文档）
- 源代码包含的知识最全面最完整 （反例：人脑的知识）
- 源代码包含的知识价值密度最高（反例：日志信息）

源代码知识是宝藏

挖掘“代码宝藏”之旅

矿山挖宝

矿山挖宝的三大步骤和原则：

- 定坐标-解决宝藏在哪里的问题。
- 定方案-解决怎么挖的问题。
- 施工队-挖宝的具体实施过程。

代码挖宝

定坐标 ⇨ 明确基于哪一个代码工程来构建

定方案

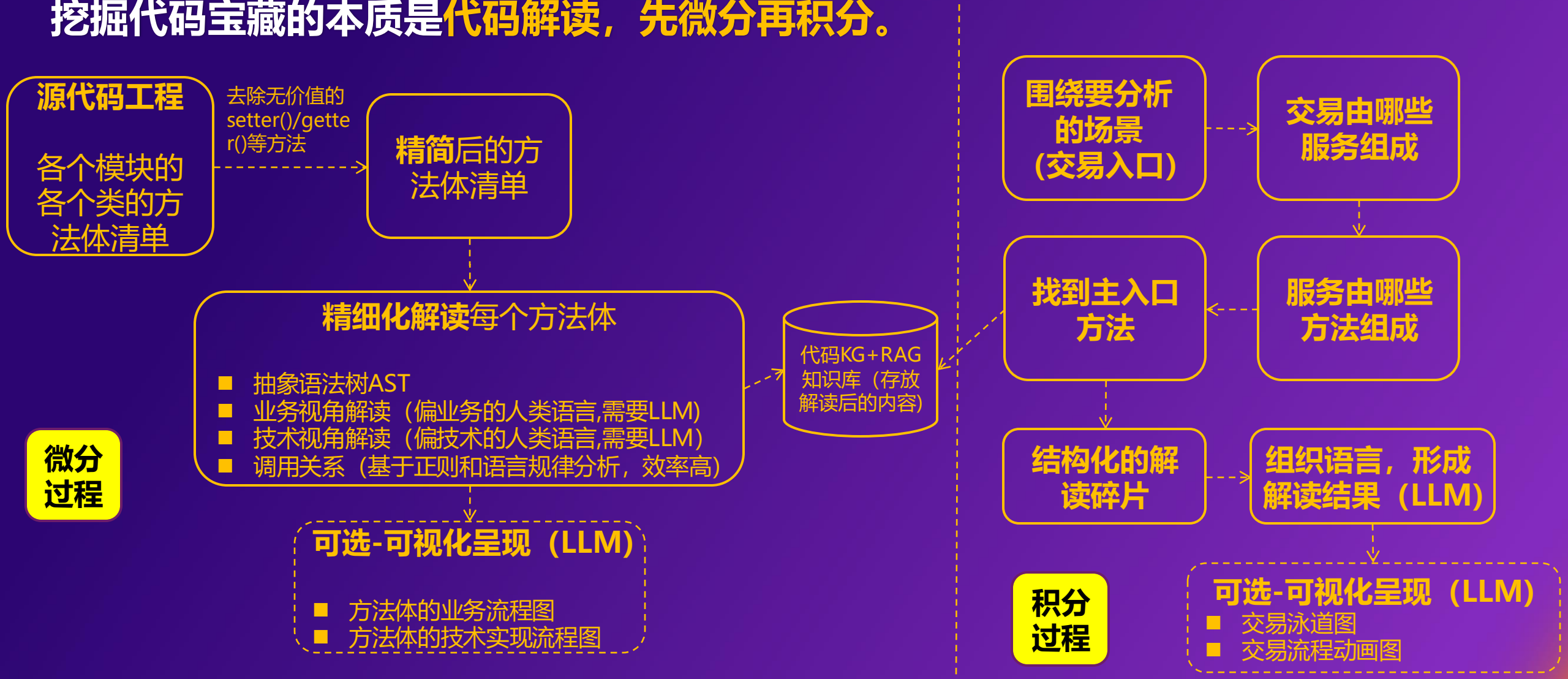
- 明确使用场景
- 明确使用对象
- 明确使用频次
- 明确更新频率

- 关键难点识别
- 关键决策点识别

施工队 ⇨ 具体构建过程

挖掘“代码宝藏“的本质是代码解读

挖掘代码宝藏的本质是代码解读，先微分再积分。



等一下，你说的是Cursor或者通义灵码类似这样的AI辅助代码工具吗？它们也有代码解读能力

以Cursor为例		我们的代码解读
Cursor适用于“指哪打哪”的解读，要么给定具体代码片段，要么给定要解读的代码位置	VS	不是“指哪打哪”的解读，而是“自动瞄准”。因为要基于具体的场景（比如用户询问交易码、用户询问具体某个业务细节）来自动找对应的代码并解读。
Cursor解读一组与业务场景相关的源代码，需要用户自行把这些源代码（或者位置）给到Cursor上下文进行分析。	VS	只要有主入口方法，则引擎会自动的识别调用关系，组件/方法的血缘关系，然后按照逻辑顺序自行找到对应的解读片段，并且串联起整个解读后的逻辑。甚至这些血缘关系不一定是通过方法调用识别的，而是通过编排文件的编排逻辑定义的。
Cursor解读更多是从技术层面的解读，很难从业务角度进行解读。而往往代码解读的受益者是业务人员。	VS	不仅有技术解读，还支持业务解读。
Cursor的解读成果一般是以人类可读文本形式输出。	VS	支持多元化输出，如业务流程图、业务泳道图、动画。

结论：Cursor在非指向性解读、复杂结构解读、业务解读、多元化输出解读结果方面并不符合要求。

PART 02

代码知识宝藏挖掘的难点和关键决策点

代码知识宝藏挖掘的常见难点

隐性知识难以挖掘

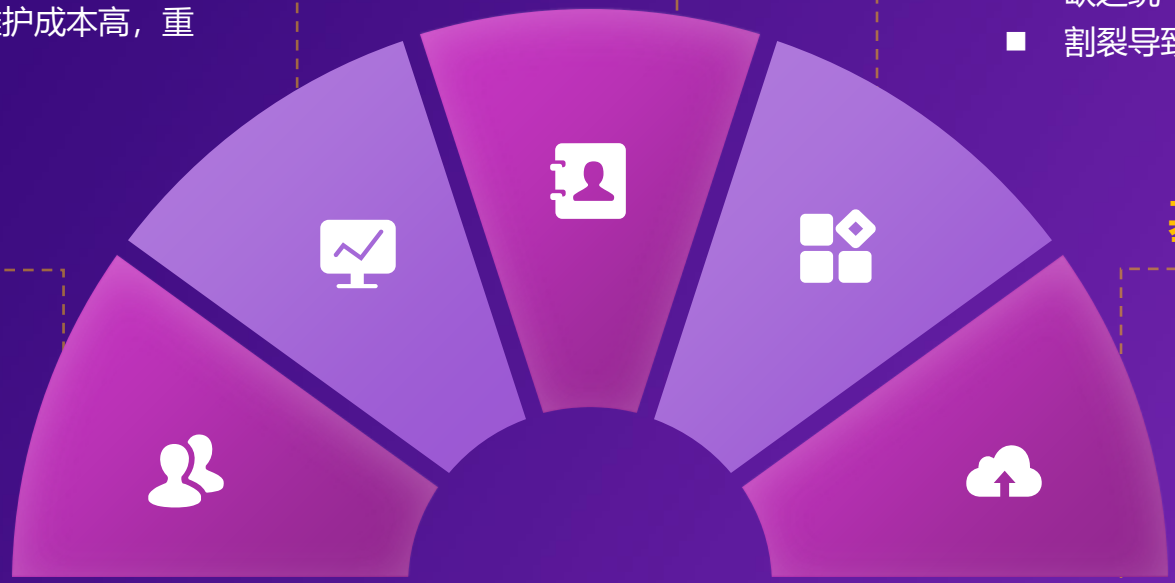
- 业务逻辑和实现细节深藏于代码中，难以通过表面分析获取

代码规模和历史遗留问题

- 总是做加法，遗留的“屎山”代码难以整合
- 旧代码缺乏文档和注释，理解和维护成本高，重构难度大，原始开发者早已离职

代码资产与其他资产关联难度大

- 代码与需求、设计、测试等文档类资产割裂，缺乏统一的知识映射和追踪机制
- 割裂导致的知识可追溯性差



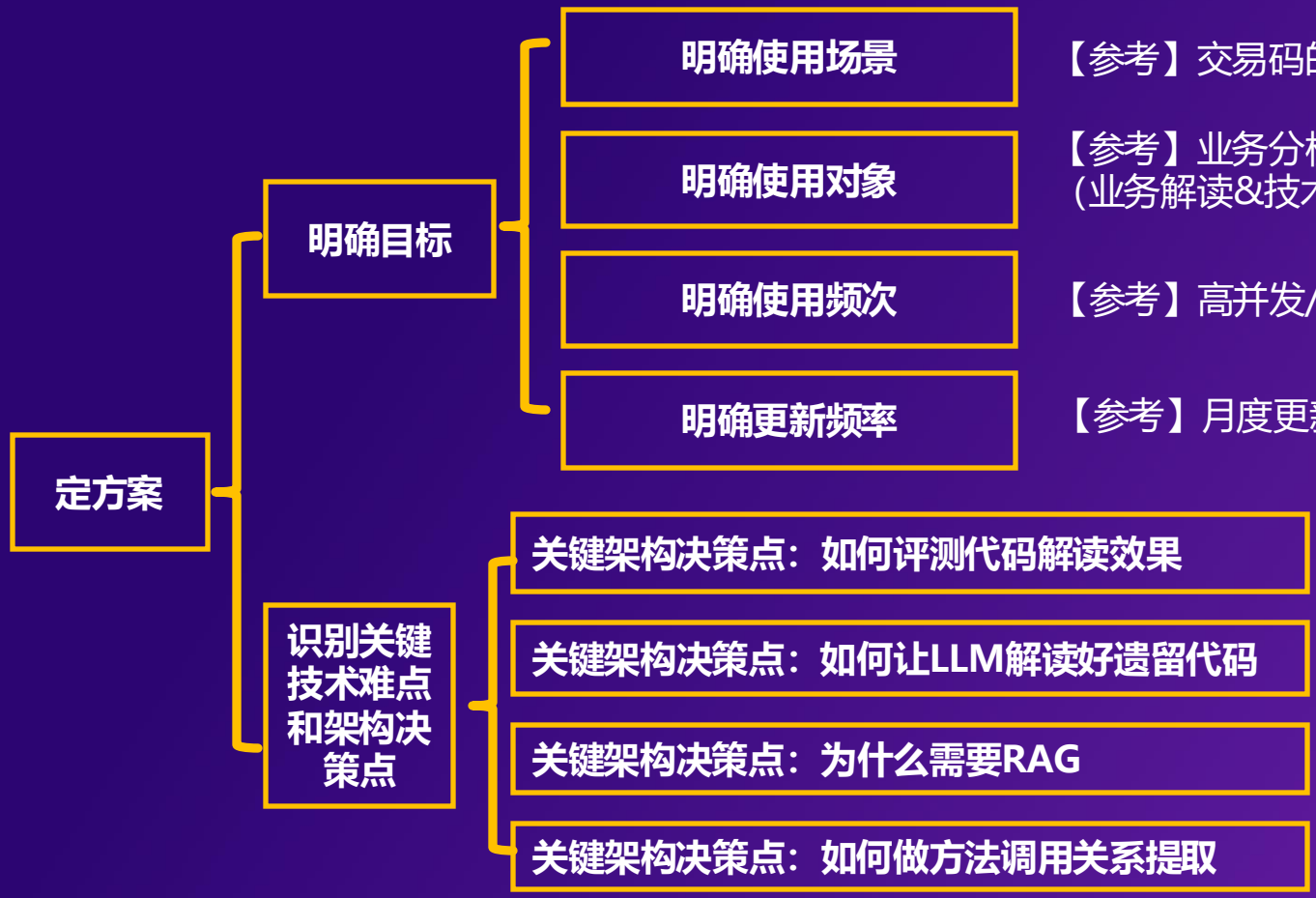
代码异构性

- 各种编程语言、开发框架、技术栈
- 代码结构、代码风格迥异

基准大模型解读能力差

- 代码是语言，但是是“编程语言”，不是“人类自然语言”
- 如果对企业特有的编程模型没有对齐，则LLM解读效果很差

定方案过程中引出的架构决策



【参考】交易码的解读/基于代码的知识问答

【参考】业务分析师（业务解读）/系统架构师（业务解读&技术解读）/开发人员（技术解读）

【参考】高并发/高频

【参考】月度更新或者大版本更新/增量更新

架构决策：应用侧是Copilot形态

架构决策：基于问题的意图识别

架构决策：私有化部署采用吞吐量最大的vllm，而不是推理速率最快的Sglang

架构决策：增量构建RAG知识库和KG知识图谱的方式

关键架构决策点：如何评测代码解读效果

非常遗憾：市面上的主要的代码相关的评测工具都是针对代码生成的，而非代码解读的。

评测工具1：CodeXGLUE （不满足）

- 基于BLEU打分算法没有充分的考虑抽象语法树Ast。
- BLEU算法针对长答案（比如“正确的废话”）的打分会明显分值偏高。
- CodeXGLUE的jsonl数据集中的正文只针对javadoc的注释。

评测工具2：bigcode-evaluation-harness（不满足）

- 其中的HumanEvalExplain中的代码解读评测任务是穿透到CodeXGLUE来完成的，所以也不满足。

评测工具3：MBPP（不满足）

- 只支持Python语言，对于老系统用的较多的Java不支持。
- 只有编程能力评测，没有代码解读评测。

评测工具4：LiveCodeBench（不满足）

- 只能评测编程能力，不能评测代码解读能力。

LLM评估普遍存在的“冗长偏差”现象

架构决策点：自研AI考核点打分工具

代码解读效果的考核点举例

- 模型回复与用户提问相关性
- 格式（比如是否有层级关系）
- 是否简洁(对抗LLM固有的“冗余偏差）
- 是否存在事实性错误
- 解读表述连贯性

基本原理：

- 任何打分一定可以拆分为若干的相互独立的考核点的组合，每个考核点命中得1分否则得0分。
- AI擅于判断考核点命中/未命中
- 但是AI不擅于打数值类分数（分数数值的高低缺乏可解释性）
- 在训练时，可以让AI解释为什么考核点命中以及考核点未命中。
- 通过人工校准，可以补充AI没有识别出的新考核点。最终考核点组合趋向于完备。

关键架构决策点：如何让LLM解读好遗留代码

选择适合“代码解读”的基座大模型

[大模型分类]
- 理科LLM（推理类LLM)
- 文科LLM(事实作答LLM)

[不要相信主观直觉！]
代码解读LLM，从评测效果看，文科LLM>理科LLM

为什么需要SFT微调+DPO偏好优化?

[特有编程模型和编程规范的解读能力差]
- 自定义注解
- 自定义编程模型
- 自定义编码习惯

[直译VS 业务话术]
- 擅长直译
- 不擅长业务话术

关键架构决策点：为什么需要RAG

- **代码解读的结果，往往只是回答最终问题的中间过程。**

- **回答效率的提升**

a. 事先解读好的结果->问答场景的回答效率。
b. 场景适应性更广的知识图谱RAG。

关键架构决策点：如何做方法调用关系提取

效果不佳的尝试

方式1：基于LLM提取

缺点：
- 效率极其低下
- 平均方法调用关系提取要7.5秒左右

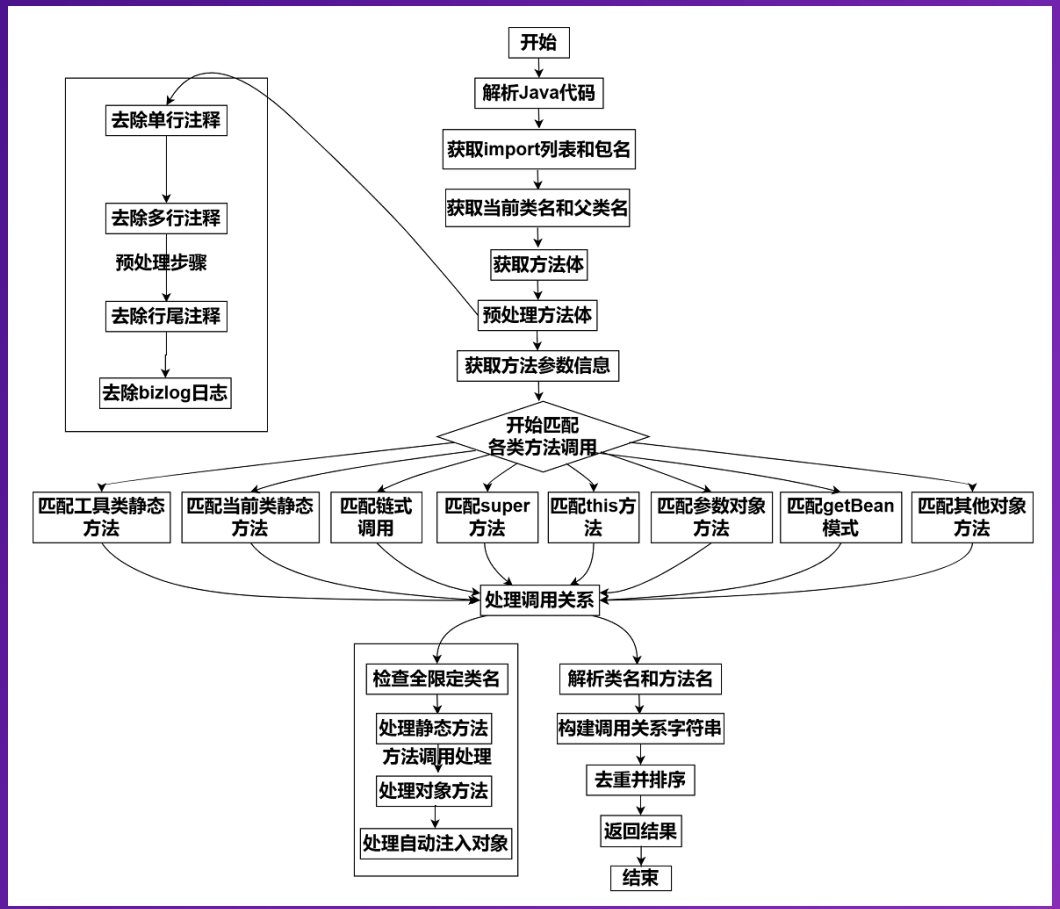
方式2：基于Ast抽象语法树

缺点：
- 解析过程耗时长
- 平均方法调用关系提取要在4-5秒左右

效果不错的尝试

基于正则表达式做方法调用关系提取

优点：
- 效率极高
- 平均方法调用关系提取只需要0.1秒。



AiDD 7th 2025

PART 03

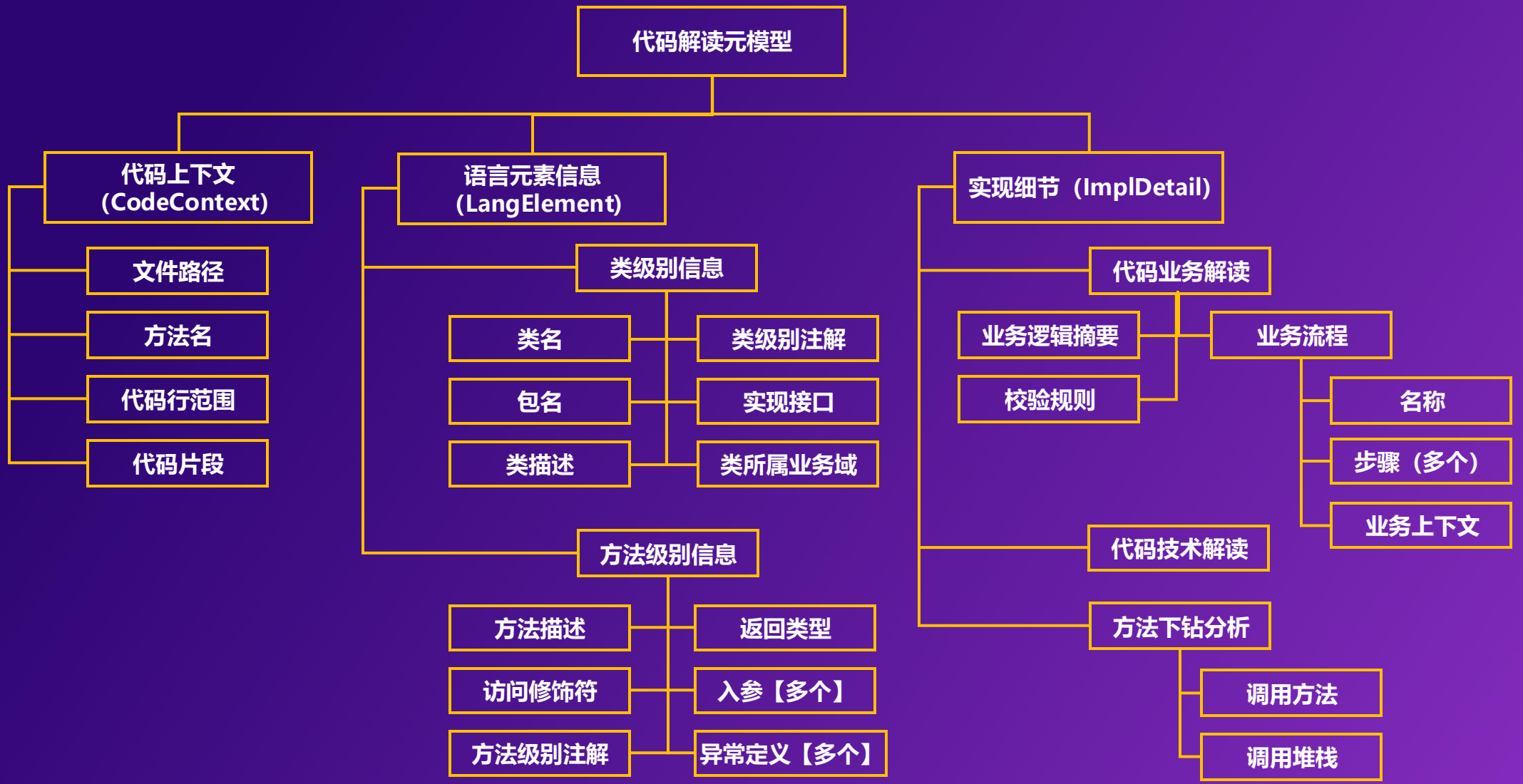
案例分享：代码解读KG+RAG知识库

代码解读知识库构建过程-整体架构

核心要点：

- 除了源代码解读需要用LLM以外，其他的尽量不用LLM，更多依托传统分析方式（比如正则）解决。
- 代码解读的原子颗粒度为Method，并且去除getter/setter等无意义的Method。
- 代码解读效果更多是偏业务解读，而基模能力主要是偏重技术解读，因此需要对基模做SFT微调，使其成为能从业务视角解读的“代码解读大模型”
- 代码解读大模型建议部署在vllm上，因为vllm的吞吐量是所有开源推理平台中最强的，可并行解读。sglang在这个场景没有vllm有优势。
- 支持增量解读，以大版本更新为迭代。

代码解读元模型 （以Java源代码为例）



代码解读知识库构建方式

- 代码解读知识库的呈现方式是图数据库。因此用NebulaGraph来构建。
- 不选择Neo4j主要原因是其社区版不支持集群化部署。

四步走策略：：

基础数据铺底：

- 基于AST抽象语法树分析出主要实体（类、接口、方法等）以及关联关系，基于代码解读元模型的数据结构定义来格式化这些实体。
- 未知字段（如业务逻辑摘要、业务流程等）先空缺

交易/服务实体构建：

- 利用识别器识别以下关系：
a. 交易与服务的关系
b. 服务与业务表的关系
c. 服务与错误码的关系
d. 交易与交易流程的关系
- 将交易、服务、业务表、错误码、交易流程都作为知识图谱的新实体写入图数据库。并关联对应关系。

方法调用关系提取：

- 基于方法调用关系提取引擎提取方法之间的调用关系
- 将方法调用关系写入图数据库，作为方法实体之间的关系

代码解读并完善方法实体的”业务解读”属性和“技术属性”：

- 利用已经微调过的代码解读大模型，对于有效方法（非getter/setter)方法进行业务视角的解读，解读结果填入对应方法实体的”业务解读“属性。
- 利用强力的基模对有效方法进行技术解读，解读结果填入对应方法实体的“技术解读”属性。

反例-经验教训分享

一开始的策略是利用微软GraphRAG：

- 基于源代码的内容，以源文件为单位，用程序生成语料包。
- 用GraphRAG对语料包进行E-R提取，来生成GraphRAG的知识图谱RAG。

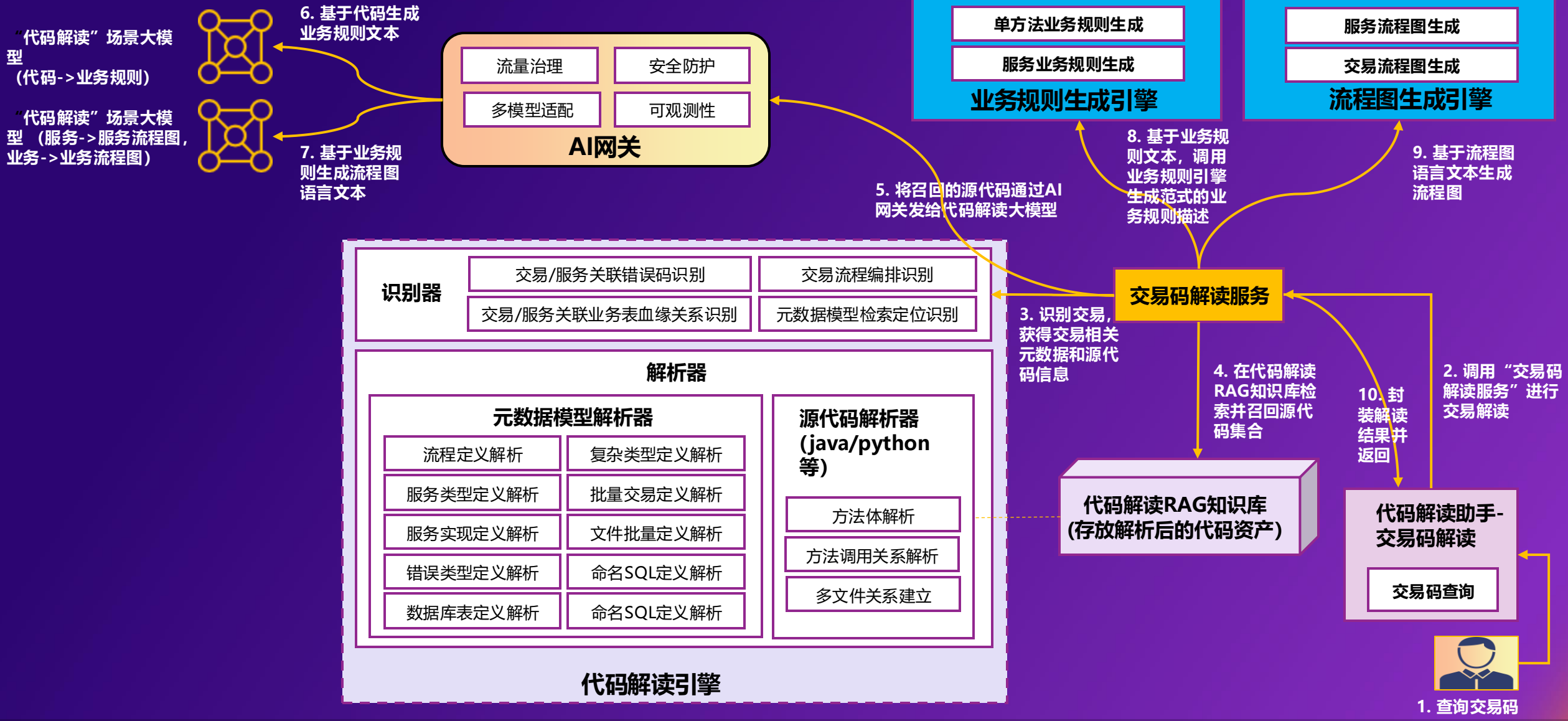
这种方式的问题：

- **工作量巨大。**有多少源文件，就对应有多少语料包，而且需要用人类连接词（例如“是”，“属于”等)来连接。
- 语料包采用GraphRAG基于LLM的实体-关系提取方式，非常**消耗Token和时间。**仅以存款模块为例，需要2000多万个token和几十小时。后续社区聚类又需要接近的工作量。
- 即便采用了提取类型为**NLTK**的策略，节约了一半左右的耗时，但是质量比基于LLM的要差。
- 构建完有不少需要**人工优化的工作**，比如2个不同的类都有name字段，会被建模为相同的实体，导致关系错乱，需要人工调整，利用类名作为前缀将其区分为不同的实体。

因此不采用基于GraphRAG的知识图谱RAG构建方式，而是基于上述方式构建。

基于代码解读知识库的应用-代码解读助手

【场景举例】代码解读助手之交易码解读



【场景举例】代码解读助手之代码知识问答

技术要点

- 识别器

  a.基于正则识别$R_{\{交易流程,被编排的交易\}}$、 $R_{\{交易,错误码\}}$、 $R_{\{交易,参与交易的实体\}}$

  b.基于O/R Mapping定义找到$R_{\{实体，表\}}$

  c. 基于表架构分析、如外键约束、字段命名、视图、存储过程等来展示业务表血缘关系。
- 元数据模型解析器-基于元数据模型解析，如交易、组合服务、原子服务、领域对象等。
- 源代码解析器-基于AST抽象语法树进行源代码解析。
- 源代码资产检索和生成-将源代码资产基于代码解读元模型转为基于KG的知识库。
- 术语检索和生成-将用户的口语化表达（自然语言、随意说法）与标准术语（如数据字典中的字段、表名、业务术语等）建立映射关系，并形成RAG知识库。

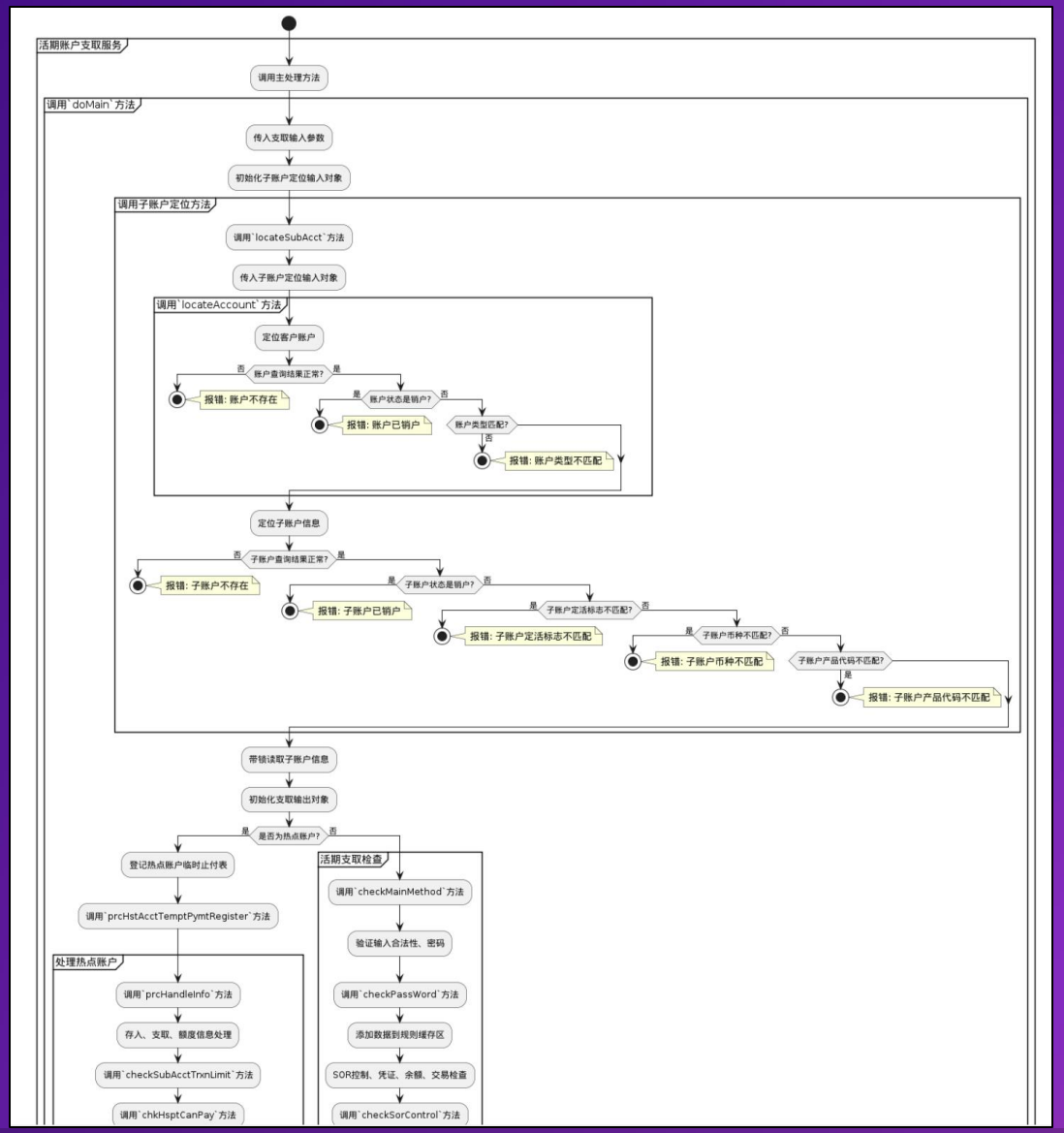
展示-用业务规则描述+业务流程图表征复杂业务规则

提问：核心交易dp3180的业务规则是什么？

2.2.1 业务规则

该方法主要用于外汇买卖中间记账，业务规则如下：

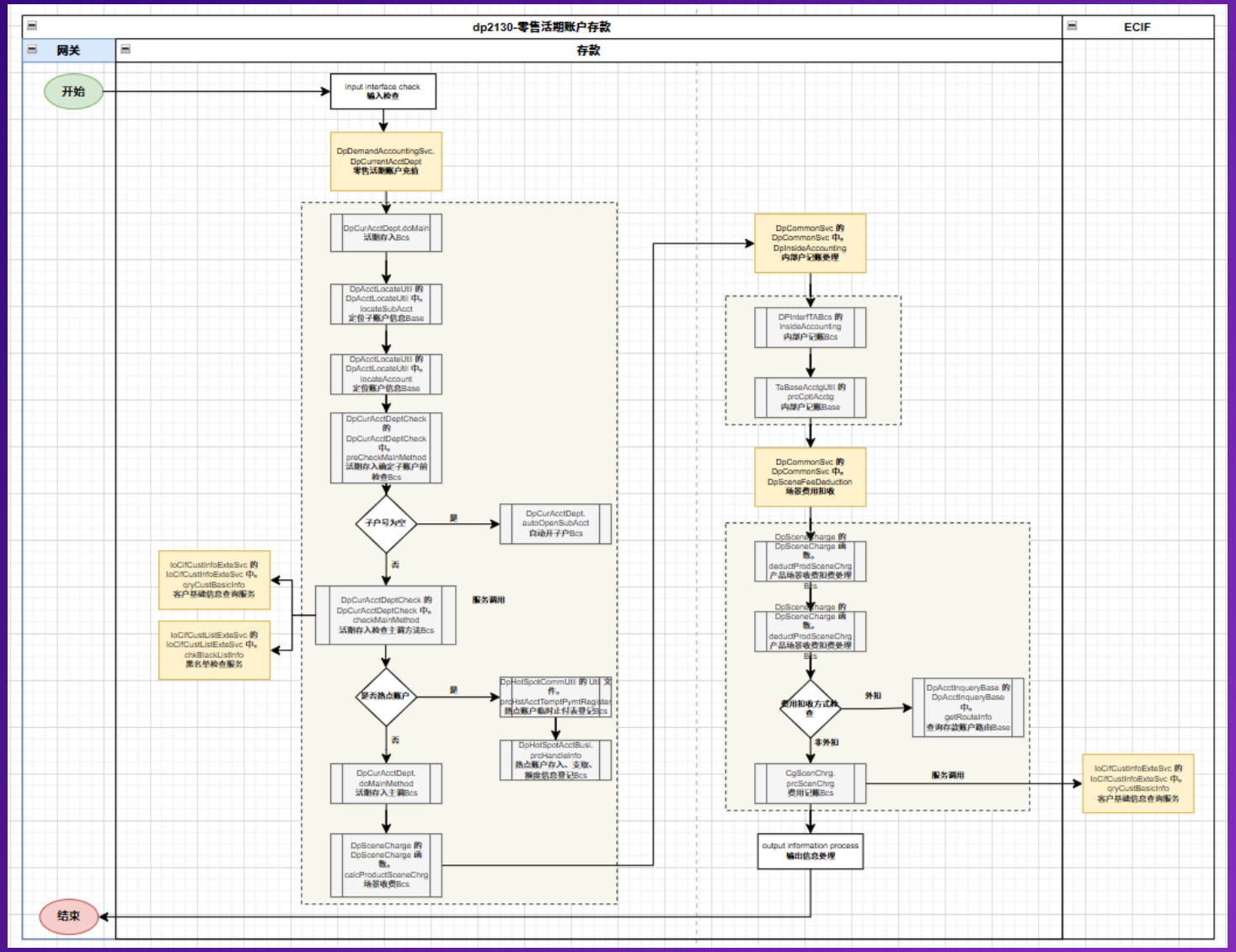
1. 调用外汇买卖中间记账服务
  1.1 调用“外汇买卖中间记账服务”方法，传入外汇买卖中间记账服务输入对象
    1.1.1 调用“外汇买卖通用记账”方法，传入外汇买卖中间记账服务输入对象
      1.1.1.1 判断买入币种和卖出币种是否相同，如果相同则返回null
      1.1.1.2 根据买入币种和卖出币种确定外汇交易类型
      1.1.1.3 设置外汇交易的相关信息
      1.1.1.4 调用“外汇买卖通用记账主方法”，传入外汇交易信息对象
        1.1.1.4.1 计算入账金额
          1.1.1.4.1.1 调用“计算外汇买卖金额”方法，传入计算外汇买卖金额输入对象
          1.1.1.4.1.2 如果计算结果为空，报错"计算外汇买卖金额失败"
          1.1.1.4.1.3 如果计算结果的买入金额和卖出金额与输入不一致，报错"外汇买卖金额不一致"
        1.1.1.4.2 获取记账账号
          1.1.1.4.2.1 调用“获取外汇买卖记账账号”方法，传入获取外汇买卖记账账号输入对象
          1.1.1.4.2.2 如果获取结果为空，报错"取记账账号失败"
        1.1.1.4.3 登记外汇买卖账务
          1.1.1.4.3.1 贷记买入方外汇买卖账户，借记卖出方外汇买卖账户



技术解密：业务流程图的生成方式是把业务规则的文本内容基于LLM生成plantuml语法，再渲染为业务流程图

展示-用业务泳道图表征复杂业务流程

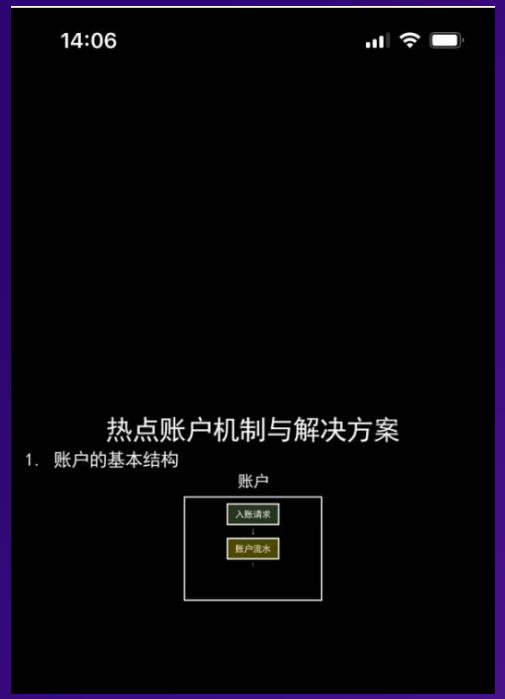
提问：把交易dp2130的整体流程描述下，并绘制业务泳道图。

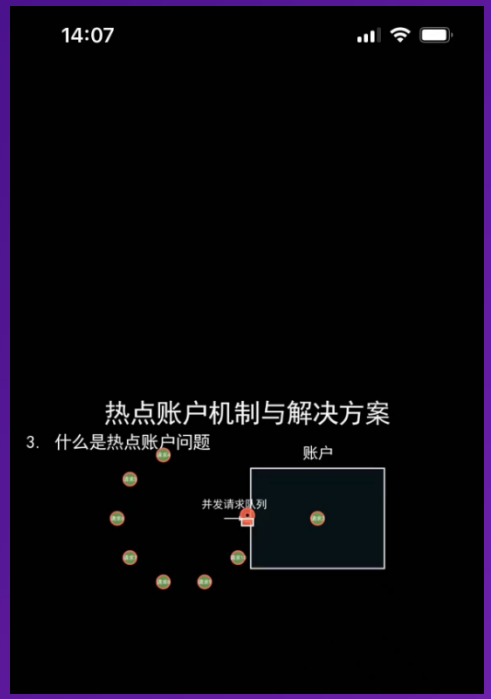


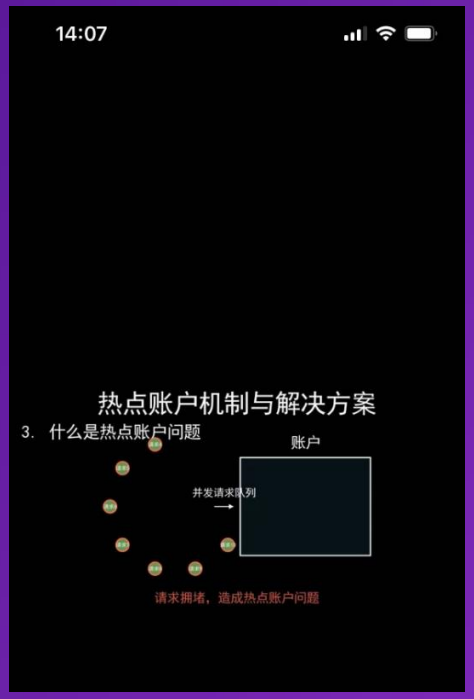
技术解密：业务泳道图的生成方式是把业务规则的文本内容基于LLM生成mermaid格式的语法，再渲染为drawio泳道图。

展示-用动画的来呈现复杂的业务知识问答

提问：请以尽可能清晰的方式描述热点账户机制和解决方案







技术解密：动画是基于manim来实现的，manim是开源的动画引擎，用于制作解释性的动画视频。因此通过让LLM把将问答结果转为manim的python脚本，并进行动画渲染。

PART 04

总结和展望

总结和展望

新兴挖掘代码知识宝藏的方式：

- 目标：智能化挖掘+自动化挖掘
- 实现手段：RAG高效检索（定位知识）+KG结构化知识表达（串联知识）+AI的理解和生成能力（理解和表达知识）。

未来展望：

- 更智能的代码助手
- 自动化代码审查和优化
- 代码更新的解读以及测试用例更新的联动
- 跨领域融合

AiDD 8th 2025

第8届AI+研发数字峰会

拥抱AI 重塑研发 AI+ Development Digital Summit

查看会议详情

下一站预告 11/14-15 | 深圳站 12/19-20 | 上海站

深圳站论坛设置

智能装备与机器人

超越“编程 Copilot”

下一代知识工程

智能网联与汽车智能化

AI 测试工具开发与应用

AI 基础设施和运维

数据智能及其行业应用

可信 AI 安全工程

大模型和 AI 应用评测

多 Agent 协同框架

从智能测试到自主测试

大模型推理优化

多模态 LLM 训练与应用

智能化 DevOps 流水线

上下文工程

AiDD

「深行·浅智」

Walk Deep, Think Light.

2025.11.16

AiDD首届麦理浩径徒步

KEYLINK ing

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

AiDD峰会详情

上海站

AI+研发数字峰会

AiDD 峰会

时间：

2025.05.23-24

北京站

AI+研发数字峰会

AiDD 峰会

时间：

2025.08.08-09

深圳站

AI+研发数字峰会

AiDD 峰会

时间：

2025.11.14-15

香港站

「深行·浅智」

AiDD 首届麦理浩径徒步

时间：

2025.11.16

上海站

AI+产品创新峰会

AiDD × 联合峰会

时间：

2025.12.19-20

第7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