AiDD
2024 AI+研发数字峰会
AI+ Development Digital summit
AI驱动研发变革 促进企业降本增效
北京站 08/16-17

与开发者同频 —— 百度构建人机协同新范式的实践

牛万鹏 百度Comate架构师

演讲嘉宾

牛万鹏

百度Comate架构师

百度资深研发工程师，毕业于吉林大学，毕业后入职百度
长期负责DevOps工具的孵化和落地，涵盖项目管理、代码管理、流水线、制品库、应用部署、运维管理等平台建设和商业化
现负责百度研发智能化，通过构造全新智能编码工具，搭建全新的产品形态，推动百度万人研发范式的变革。

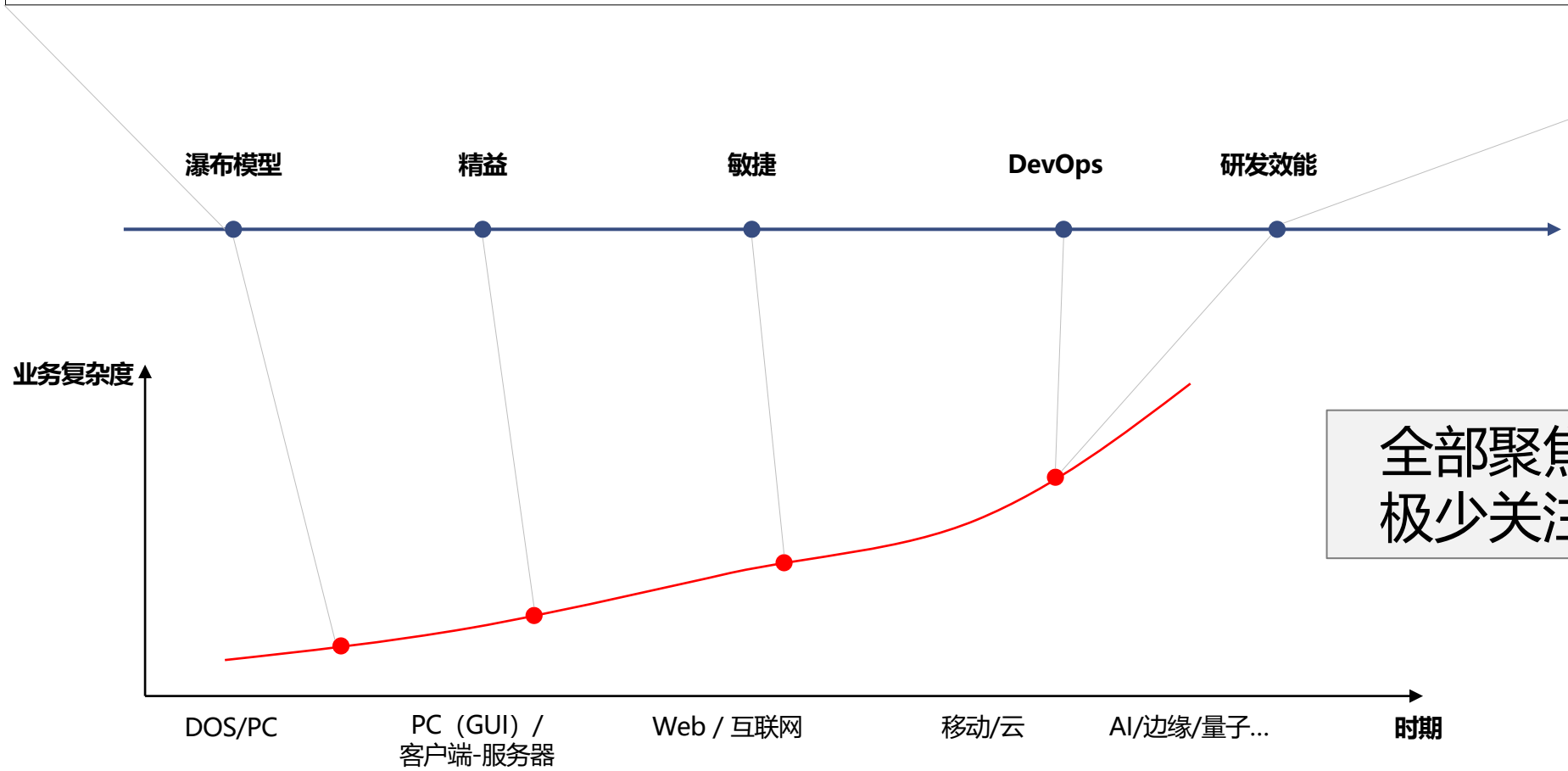
目录
CONTENTS

PART 01

在AI前夜——百度研发现状

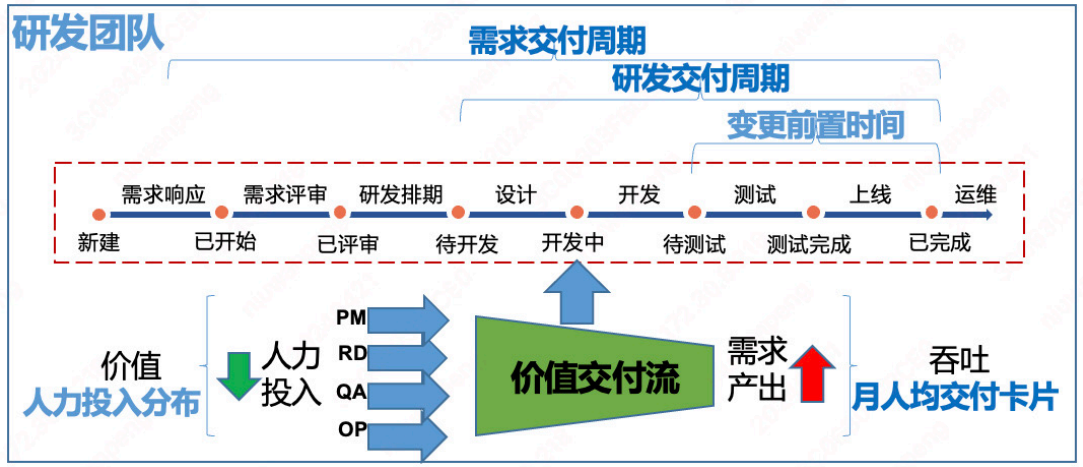
业界软件开发模式的发展轨迹

软件开发的基本性质			
沟通密集性	**协作密集性**	**持续迭代性**	**不可复制性**
需要每个角色密切沟通，保证理解一致性，降低随机性	需要从产品经理到研发、测试、运维等一系列的角色参与	代码要被不断组装在一起，不断打补丁，让软件系统能够持续工作	软件规模虽然在不断扩大，但并非重复，工程师每天都写不同的代码
研发协作的密度增高和迭代周期增长，引起效率降低。			



全部聚焦『流程提效』
极少关注『个人提效』

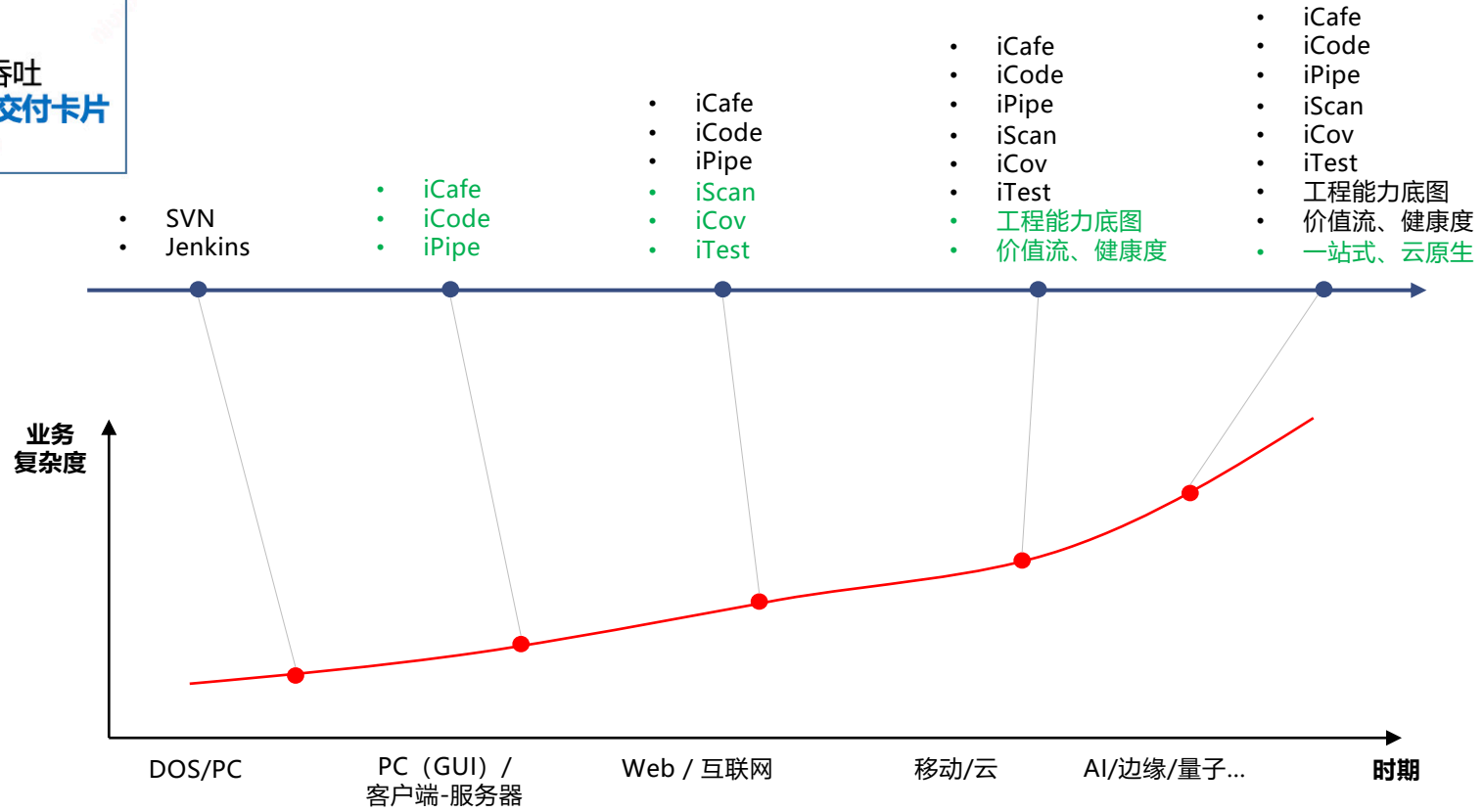
百度研效工具的发展轨迹



流程支撑

- 1w+工程师，1k+业务团队，10w+代码库
- 85%的需求一周交付，50%的研发资源云化
- 80%的应用从开发到上线全流程在线
- 每天1w次代码提交
- 每月1kw次流水线自动化任务

工具支撑



聚焦『流程』+『协作工具』

面向流程的研发提效窘境

■ 技术迭代的太快，流程的更新赶不上技术的更新，持续滞后阻碍研发效率提升

- ✓ 流程是是实践后的经验、教训等总结，一定是『滞后的』
- ✓ 如，针对Prompt、数据集等在工程团队如何管理，目前没有明确的范式

■ 多数开发者对敏捷、效能、DevOps等不感冒，也不理解其实际意义

- ✓ 面对花样繁多的项目管理流程，多数开发者更想聚焦于研发
- ✓ 过度在团队内推广研发流程，反而引起开发者的抗拒心理，大幅降低开发者的幸福感

核心原因在于整个提效的设计**不是站在开发者个人**，而是站在组织上

开发者的『iPhone』时刻

Github Copilot + ChatGPT的诞生，吹响了『开发者个人提效』的号角，几乎一夜之间所有组织都开始关注

大模型催生了开发者个人提效的『银弹』

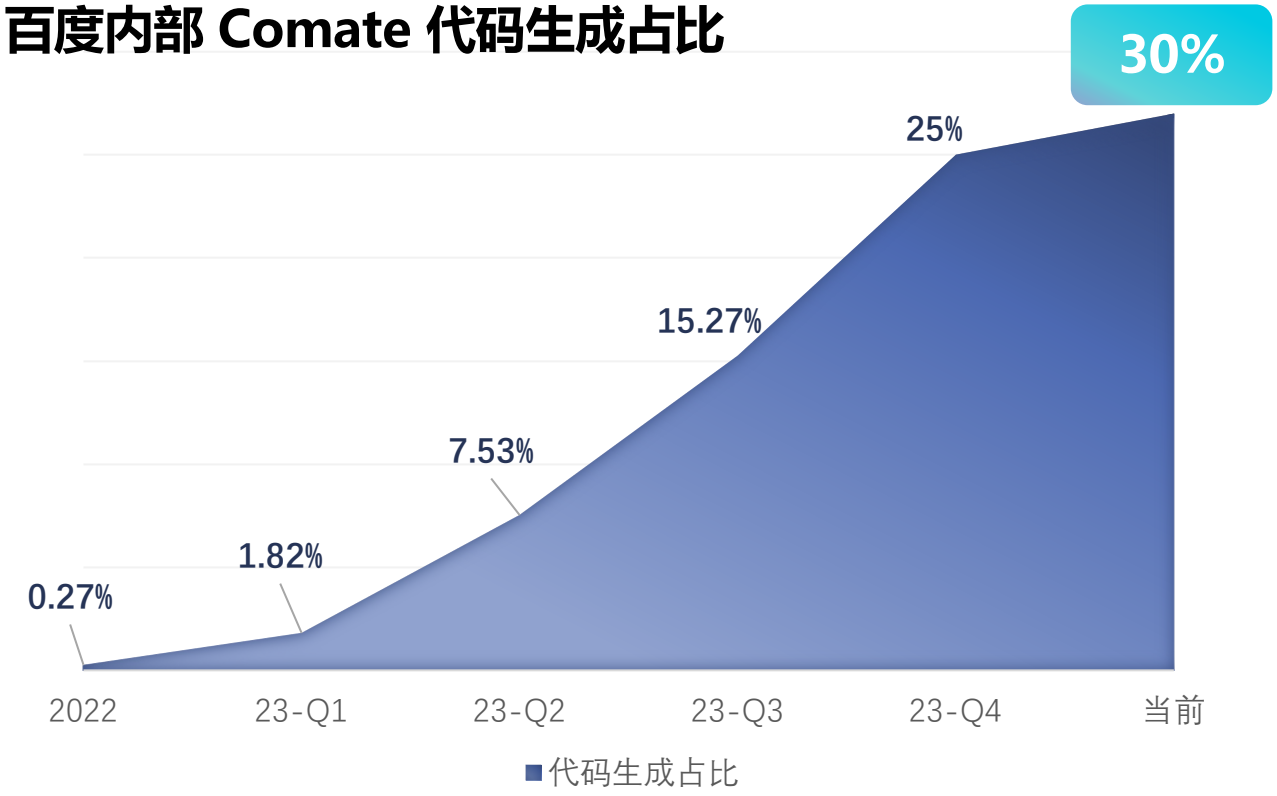
智能研发助手 = 代码自动补全（Completions） + 理解私域知识（RAG） + 独立分析需求（Agent）

PART 02

在AI时代——智能研发助手

Baidu Comate的从无到有

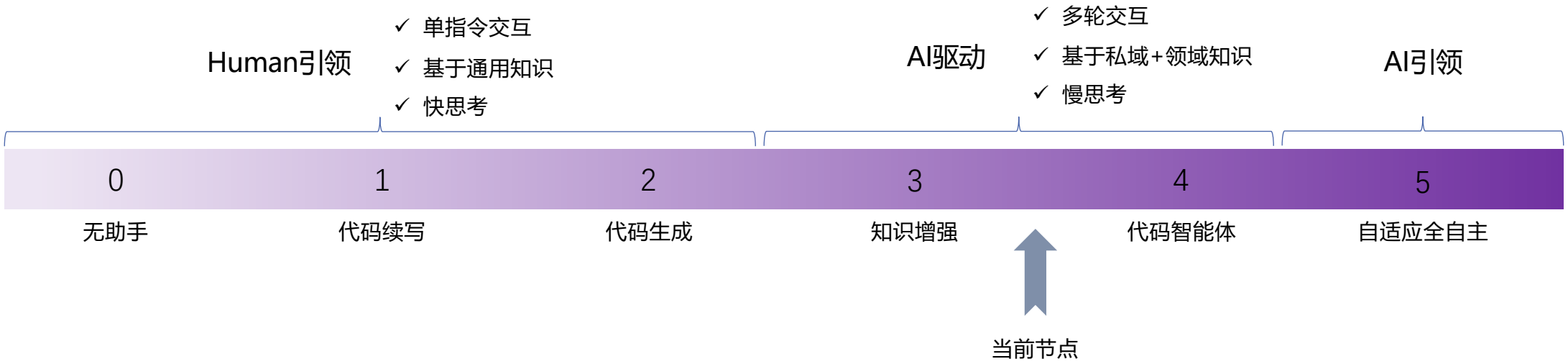
百度内部 Comate 代码生成占比



百度全局提效 12%

工程师使用	85%+
用户满意度	90%+
代码生成占比	30%
采纳率	46%

智能研发助手发展路径


Human引领
✓ 单指令交互
✓ 基于通用知识
✓ 快思考
AI驱动
✓ 多轮交互
✓ 基于私域+领域知识
✓ 慢思考
AI引领
0
1
2
3
4
5
无助手
代码续写
代码生成
知识增强
代码智能体
自适应全自主
当前节点

整体建设思路

能力层

IDE端（目录区、编辑区、Console区等）

知识层

代码类（代码片段、代码依赖等）+ 文本类（技术文档、需求文档等）

框架层

Agent + RAG + P-RAG

模型层

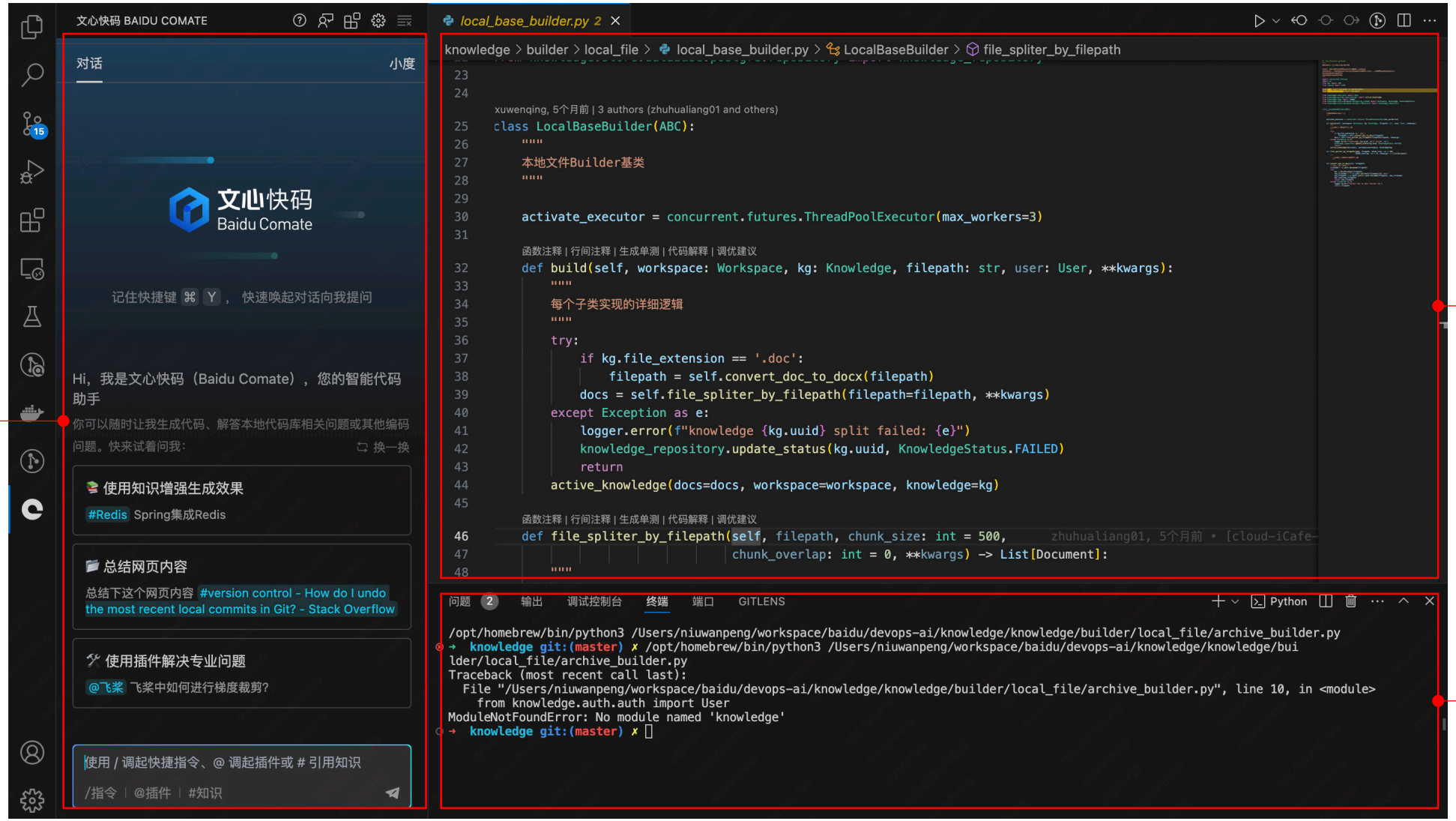
推理调度 + 代码生成 + 代码续写 + Embedding + 意图识别

能力层

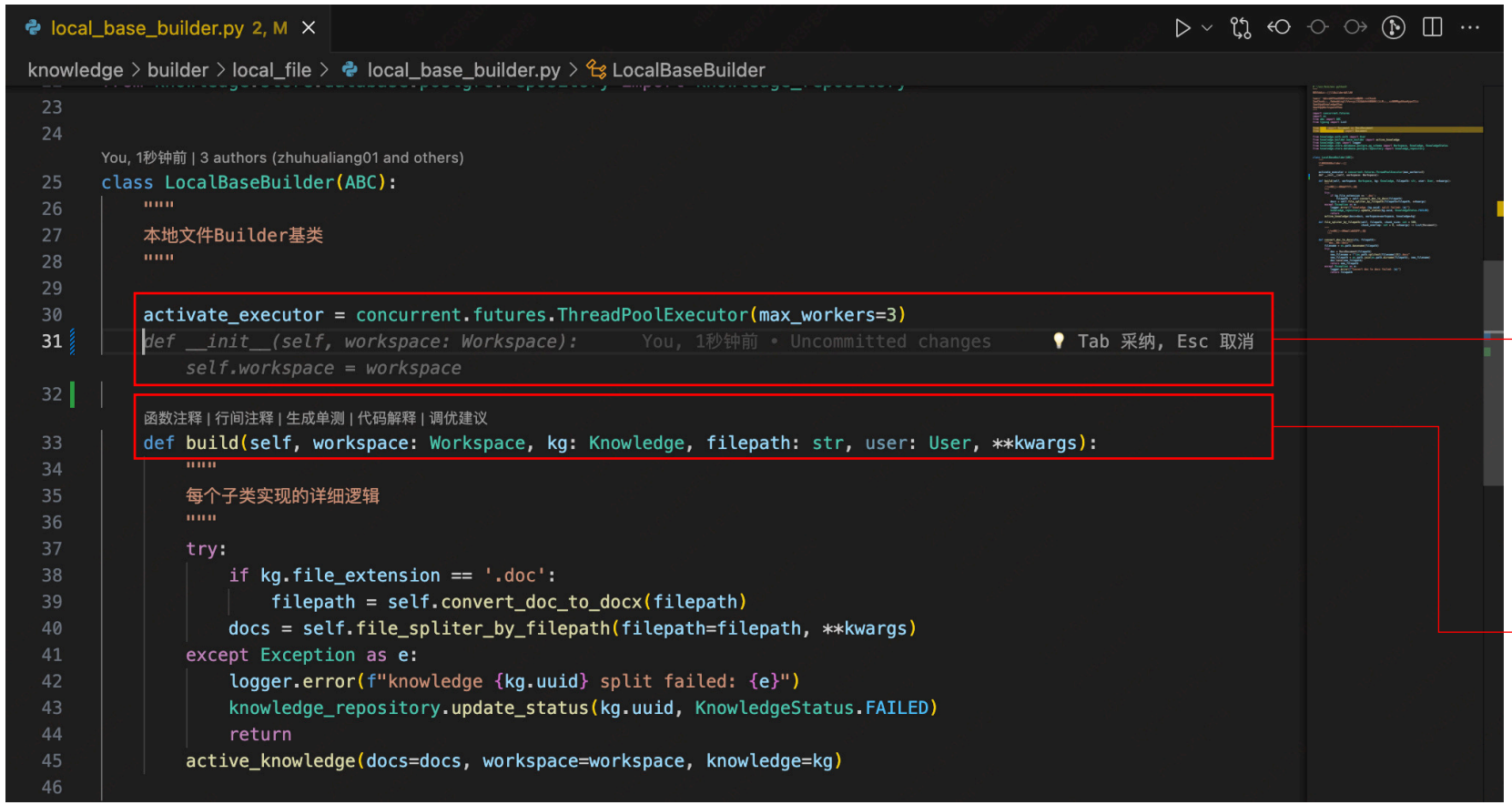
问答区

编辑区

Console区



能力层 —— 编辑区



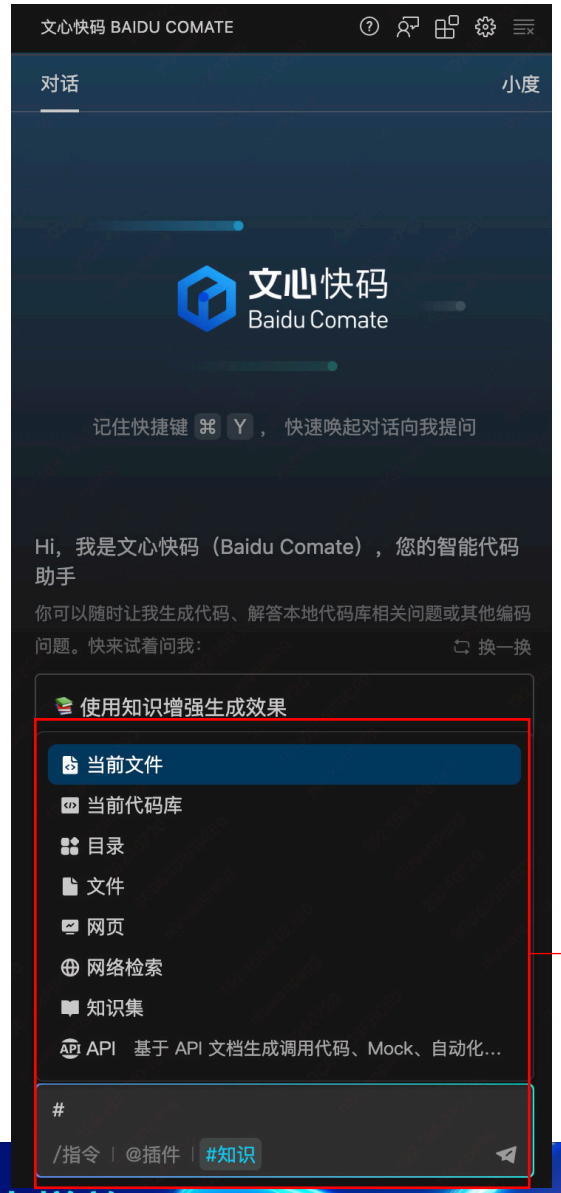
1. 代码续写，根据代码上下文自动触发
   - 单行推荐
   - 多行推荐
   - 基于注释推荐
   - 基于上下文依赖推荐

2. 函数头上快捷键，开发者主动触发
   - 函数注释
   - 行间注释
   - 生成单测
   - 代码解释
   - 调优建议
   - 函数拆分

能力层 —— 问答区

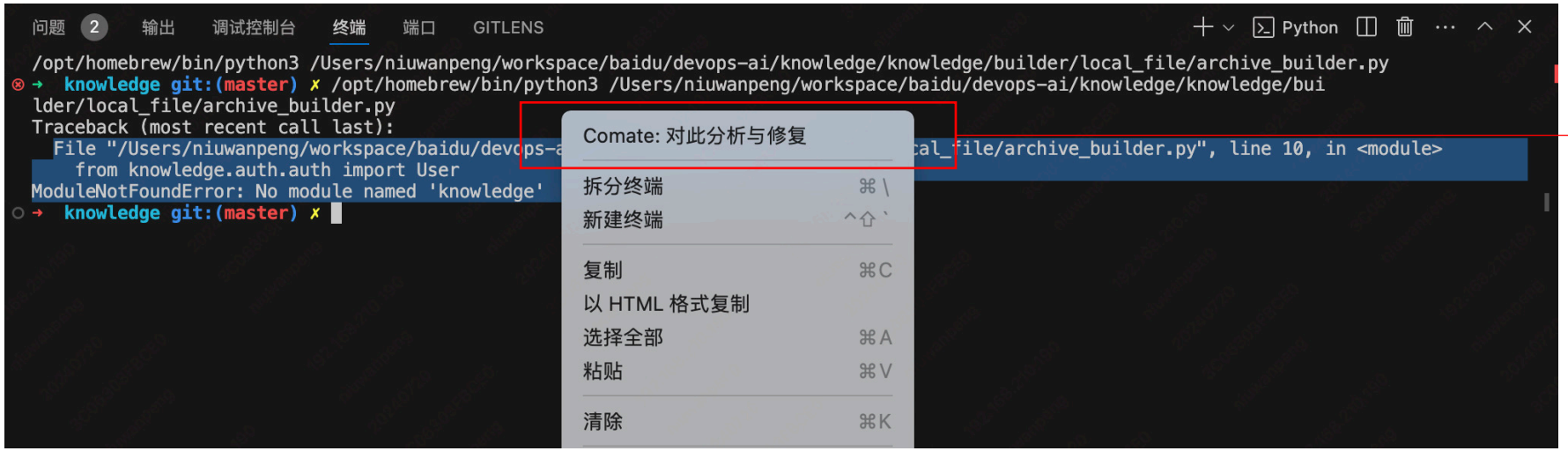
1. 点击唤起Comate问答区

2. 向Comate提问，咨询任何通用问题、生成通用代码等

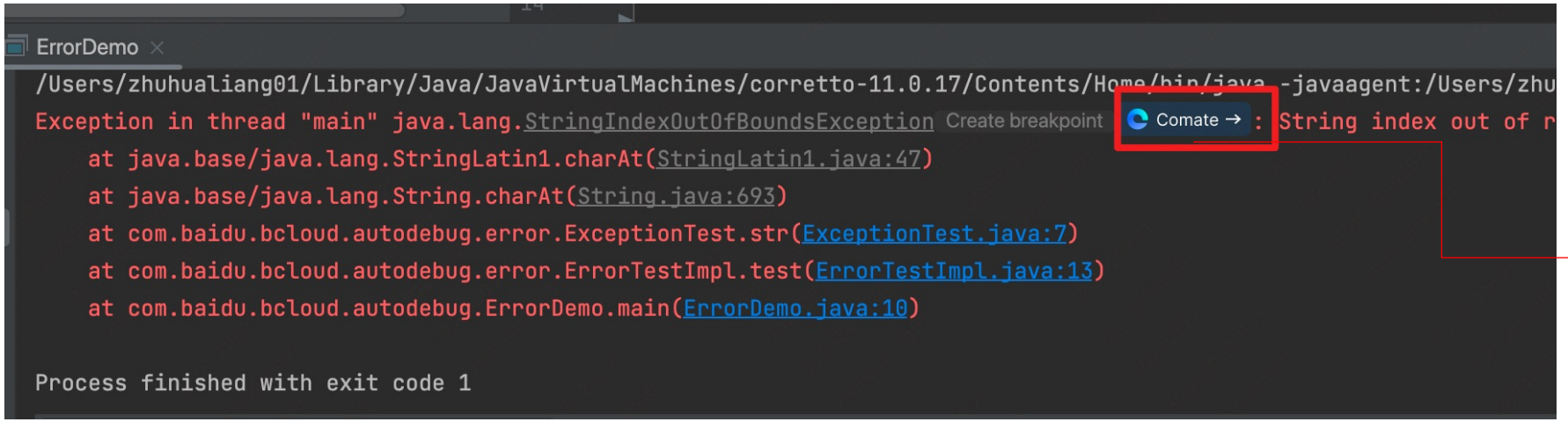


3. 通过『#』命令符唤起知识增强选项，如
   - 基于网络检索，查询最新的技术方案
   - 基于当前代码库，检索代码库内的相关代码
   - 基于知识集，查询团队内的技术方案

能力层 —— Console区



1 在VSCode通过选中错误内容后，右键打开Comate快速修复。



2 在JetBrains自动捕捉错误，开发者可一键点击修复。

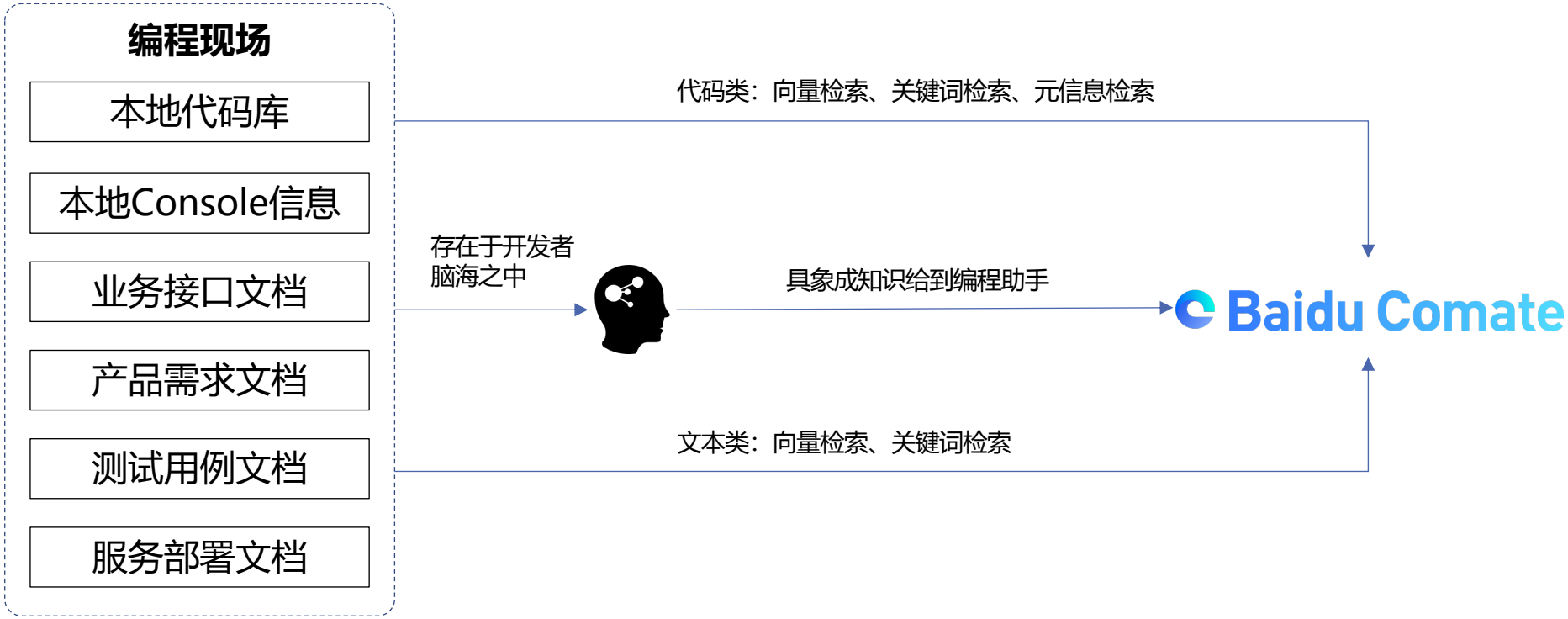
知识层 —— 为什么需要知识增强

- **通用模型存在瓶颈**：GitHub采纳率停留在46%，长期没有显著增长。主要在于依靠模型内部压缩的知识（以及少量的Neighborsource）没有办法再给出更符合『当前代码库』、更符合『当前业务逻辑』的代码，达到模型瓶颈。

- **人类程序员在开发时也需要掌握额外的知识**：编程现场会存在大量『私域知识』，如本地代码库、业务接口文档等。这些私域知识组成了每个『业务/项目/服务』的全链路开发指南，当我们对这些知识掌握的越完整、越熟练，在开发新的代码时速度越快，编写的代码质量越高。

- **知识增强是Comate进化成智能体的必要前提**：对『编程现场』的理解是所有编码助手的下一步方向，是能够突破现有瓶颈，大幅提高开发者效率，构造真正的人机协同的必经之路。

知识层 —— 与开发者同频



依靠对开发者编程现场的理解，保持和开发者同频，帮助开发者解决繁琐、重复的问题。对『编程现场』的理解：

- **是所有编码助手的下一步方向**
- **是从『简单续写』到『复杂生成』的必经之路**
- **是能够大幅提高开发者效率，构造真正的人机协同的必经之路**

知识层 —— 如何完全理解文本

文本

- 业务接口文档
- 产品需求文档
- 测试用例文档
- 服务部署文档

PaddlePaddle某篇技术说明文档

```
# Linux上GPU预测部署示例

## 1 C++预测部署示例

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`和`编译运行示例`两方面介绍。

### 1.1 流程解析

#### 1.1.1 准备预测库

请参考[推理库下载文档](https://www.paddlepaddle.org.cn/documentation/docs/zh/develop/guides/05_inference_deployment/inference/build_and_install_lib_cn.html)下载Paddle C++预测库，名称中带有 `cuda` 的为用于GPU的预测库。

....省略部分内容

#### 1.1.4 设置Config

根据预测部署的实际情况，设置Config，用于后续创建Predictor。

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。使用Paddle-TensorRT相关说明和示例可以参考[文档](https://www.paddlepaddle.org.cn/inference/mas

```cpp
paddle_infer::Config config;
if (FLAGS_model_dir == "") {
config.SetModel(FLAGS_model_file, FLAGS_params_file); // Load combined model
```

1 明确的主标题和子标题，分层清晰

2 段落描述清晰，有总述、有样例、有更详细的参考文档

文字描述丰富

文档中尽可能增加详实的文字描述而不仅仅是图片，将会显著提高学习质量。

文档逻辑清晰

文档内容有层次、语言描述言简意赅无歧义、每个段落有标题等结构，会显著提高检索质量。

目录结构清晰

如果有大量的文档分散在不同的目录中，建议每个层级目录的命名言简意赅，能够充分表达这一层级目录的文档类别。

AI阅读文档的逻辑和人一样，文档结构越清晰，AI学习的越好。
```

知识层 —— 人类如何完全理解代码

宏观
- 项目职责
- 目录层级
- 技术栈
- 关键业务词汇

微观
- 核心逻辑
- 实体关系
- 启动/构建
- 对外API

脑海中『直接』沉淀出如下总结 **显性代码知识**

这是一个用来*xxxxx*的项目，它的使用了*xxxx*、*xxxx*等框架，分为*xxxx*、*xxxx*等模块。主要逻辑包括*xxxx*、*xxxx*、*xxxx*。对外暴露了*xxxx*、*xxxx*等关键API。它使用*xxxx*方式进行部署，启动入口在*xxxx*。

脑海中『间接』沉淀出如下总结 **隐性代码知识**

这个项目中，新增xxx相关的开发任务需要在*yyy*、*zzz*等目录下，命名格式是*xxxx*，如果新增一个*xxxx*，需要配套生成一个*xxxx*。如果要调用外部API，可以使用已经封装好的*xxxx*工具。

知识层 —— AI如何完全理解代码

显性代码知识

文心快码 Baidu Comate

- 定时更新
- 手动更新
- 打开新项目时更新
- Git Commit事件更新

遍历本地文件

- 代码片段向量化
- 函数调用链分析
- 公共函数签名
- 目录层级树
- 元信息

元信息包含：

启动信息	实体关系
构建信息	核心逻辑
配置信息	高频业务关键词
依赖框架	语言分布
API定义	README.md

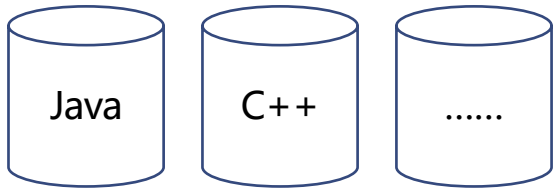
将元信息给到模型生成项目描述

项目描述

遍历Git Log

Commit Message向量化

全部基于各语言专家整理的规则池提取



知识层 —— AI如何完全理解代码

隐性代码知识

代码开发规则

什么是代码开发规则

每个代码库独有的开发规则，和开发语言、应用框架强相关，和代码库建库之初定义的规则强相关。

代码开发规则有什么用

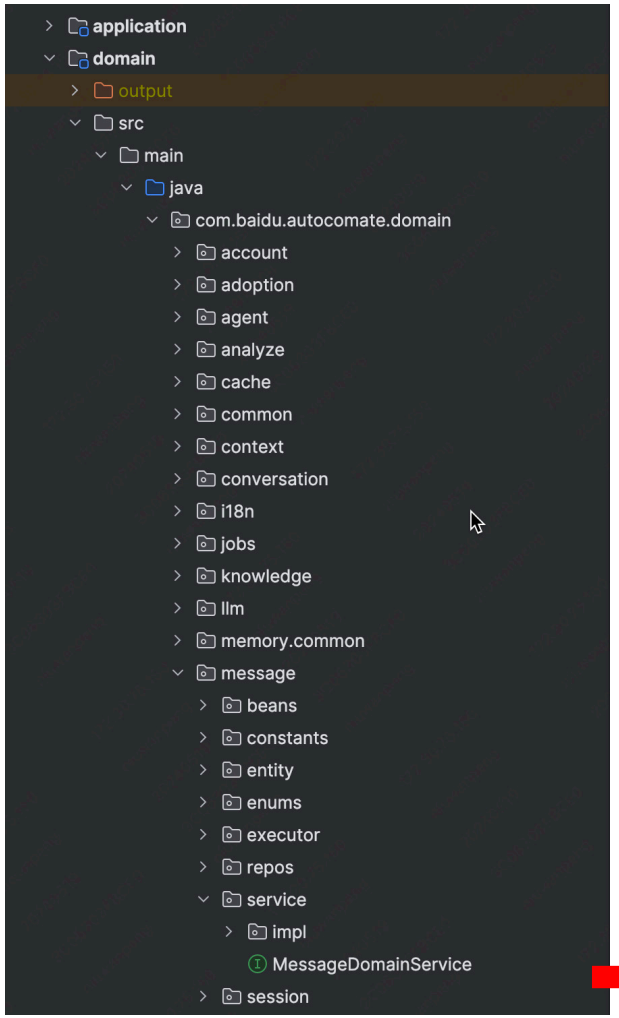
每位开发者接手代码库时首先尝试掌握的就是代码开发范式，这决定了新增一个目录、文件、函数等需要放在哪里、如何命名、结构如何定义。

Comate如何利用

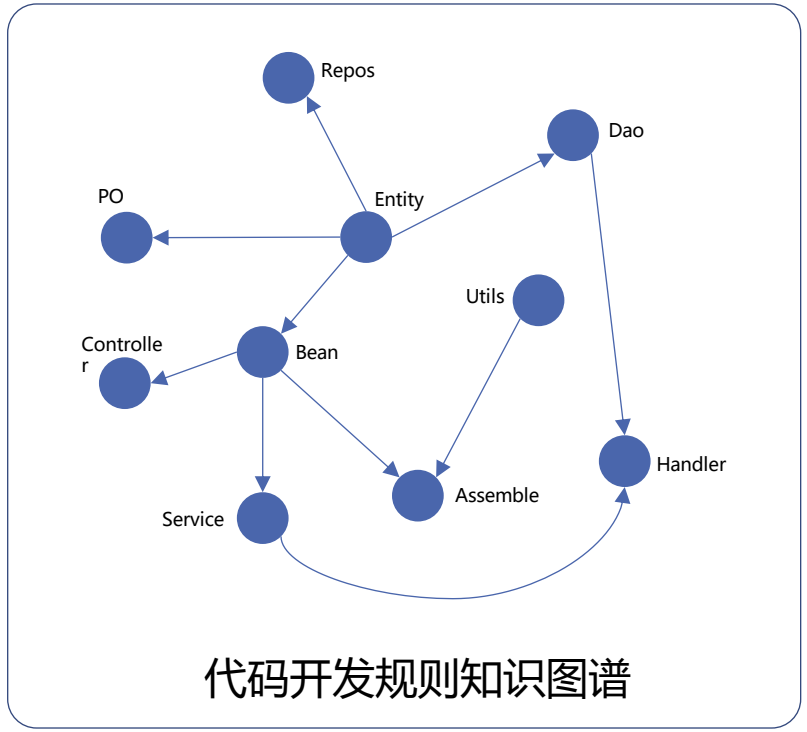
当开发范式可提取、可量化，编码助手在用户新建文件、新增函数时可以预测更大范围的代码，如新建一个Entity文件，同步将Dao、Service、Controller创建。

知识层 —— AI如何完全理解代码

隐性代码知识



1、遍历文件
2、依赖解析
3、权重计算
4、填充节点



一个典型的Spring应用，有bean、entity、po、dao、service等层级关系。

知识层 —— AI如何完全理解代码

隐性代码知识

文心快码 Baidu Comate

- 定时更新
- 手动更新
- 打开新项目时更新
- Git Commit事件更新

遍历本地文件 → 开发规则提取

代码文件的命名、函数命名、代码文件所在目录定义都有显著的『业务特征 + 规则特征』。

表明这个文件要做什么具体的事，如

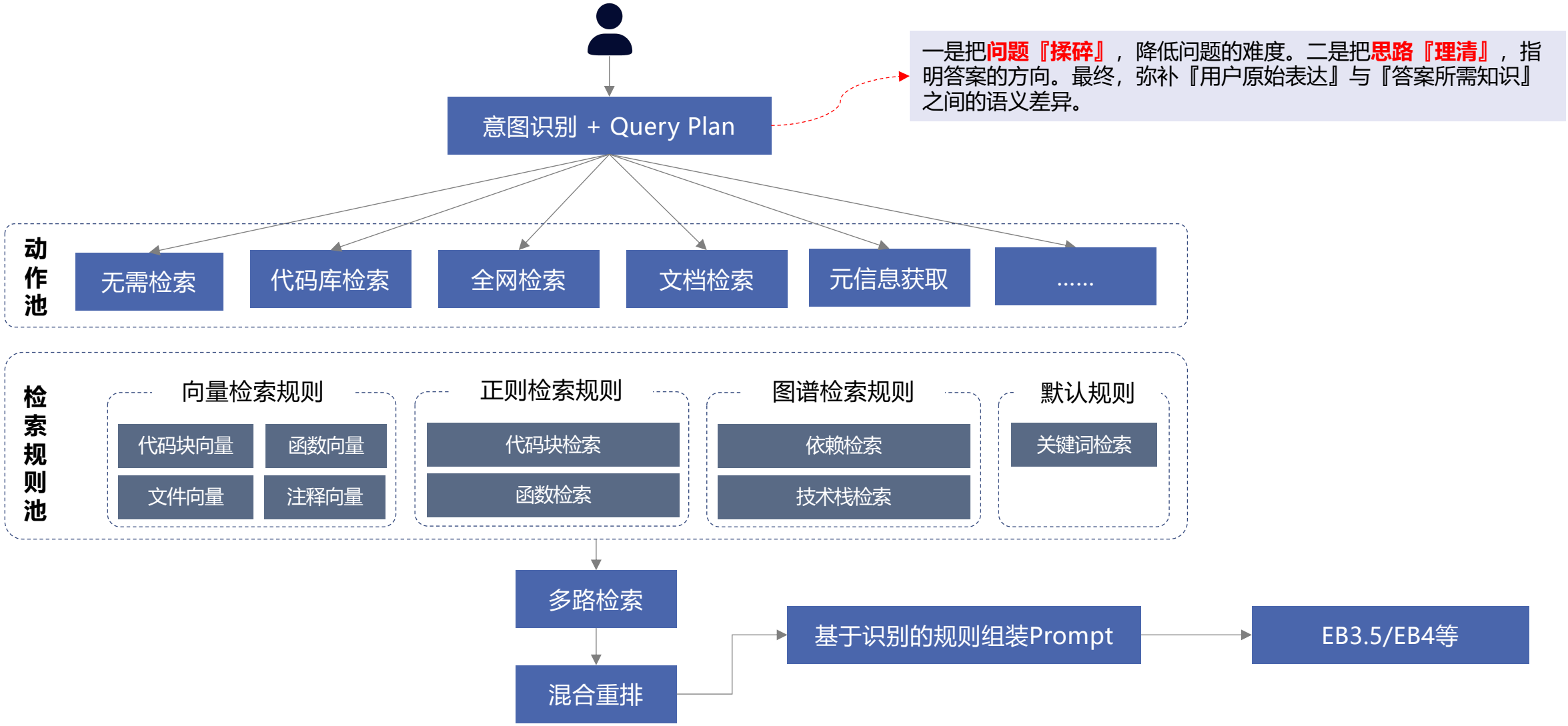
- router_api.py中router是业务特征表明要做路由
- AgentController.java中Agent是业务特征表明要做Agent的API定义

表明这个文件的属性，如

- router_api.py中api是规则特征表明这个属于api
- AgentController.java中Controller是规则特征表明要做API

框架层—— RAG + P-RAG

意图识别+RAG = 更能激发检索质量



框架层 —— Agent

人机协同新模式：『人』专注创造性的、复杂性高的逻辑设计并进行决策，『机』负责具体的实现，帮助『人』解决繁琐、重复的问题。

Comate = 理解私域知识（RAG） + 独立分析需求（Agent） + 人机结对编程（Pair）

人机结对编程

#代码库 当前代码架构怎样的？同时帮我梳理核心逻辑。

和代码库直接『对话』，快速掌握逻辑

#UserController.java #user 参考user目录下的内容，为Controllller添加增删改查逻辑。

#代码库 #文心一言开发文档 参考当前代码库的编写风格，生成调用Ernie Bot 4.0的代码。

引用相关文件、挂载私域知识生成代码

经过分析，当前开发的代码需要增加API调用异常捕捉逻辑，并增加必要日志，代码已经为您生成完毕：查看代码Diff >>>

自主巡航提供建议，与开发者并肩编程

独立分析需求

1. 意图理解
2. 检索知识
3. 制定计划
4. 任务执行

Comate

自主巡航

工具包

- 单测生成
- 百度搜索
- 代码生成
- 脚本执行
- 需求澄清
- ……

Agent框架驱动

理解私域知识（与开发者同频）

Retrieval

代码：语言、框架、依赖等

RAG
（Splitter、Retrieval、Re-rank）

文本：需求文档、技术文档等

框架层 —— Agent

■ 基于Tree of Thought思想，模拟『人』的自我反思行为，提高Agent输出质量

- ✓ Agent的思考过程中是一个连续的、多步骤的过程，每一个步骤执行完毕之后，都会等待Teacher Agent对执行结果做出评估，评估策略的核心是判断该执行结果对全局执行过程是『正向』还是『负向』的影响，并给出『整改策略』
- ✓ Agent根据『整改策略』判断是继续执行，还是回溯到上游某个步骤重新执行，重复这个过程，直到Teacher Agent认为输出结果达到标准

模型层

PaddlePaddle

- 高性能推理优化
- 多平台支持
- 搞笑模型压缩

文心大模型

- 强大的语言理解能力
- 高效的推理生成能力
- 自我学习和进化能力

代码大模型套件

- 低延迟的**续写模型**
- 检索增强**问答模型**
- 强推理的**计划模型**
- 代码Embedding模型
- 意图识别模型

PART 03

在AI未来——人机协同新范式

如何实现和开发者同频 —— 典型场景

代码续写

当我在某文件编写一段代码时，Comate怎么才能基于对代码库的理解，预测出当前文件不存在且需要导入的文件、类、函数?

代码解释

当我要求Comate分别解释代码块、函数、文件、目录、代码库时，它怎么才能从业务角度给出解释？如基于调用链分析给出流程图、ER图、出架构图等?

代码检索

当我新接手一个代码库，需要了解代码架构、具体的业务逻辑，如何才能给出像Mentor一样的讲解?

代码生成

当我希望将整个代码库的Hibernate组件替换成Mybaties组件，期间涉及到实体定义、SQL语句、事务等方方面面的分析，Comate怎么做才能把这件事做好

如何实现和开发者同频 —— 代码续写

当我在某文件编写一段代码时，Comate怎么才能基于对代码库的理解，预测出当前文件不存在且需要导入的文件、类、函数?

以 List<WebSearchResult> 为key，检索哪些函数Return了它，得到这些函数的签名

```
private static String generateWebChunk(CodeSearchResult codeSearchResu
    if (CollectionUtils.isEmpty(codeSearchResult.getWeb())) {
        return StringUtils.EMPTY;
    }
    List<WebSearchResult> webSearchResults =
```

1

```
private static String generateWebChunk(CodeSearchResult codeSearchResult,
    if (CollectionUtils.isEmpty(codeSearchResult.getWeb())) {
        return StringUtils.EMPTY;
    }
    List<WebSearchResult> webSearchResults = codeSearchResult.getWeb();
```

2

以 List<WebSearchResult> 为key，检索哪些函数把它作为入参，得到这些函数的签名

增强续写效果的知识

- 公共函数签名
- 开发规则提取

正在编辑：*WeChatPayController.java*

提取规则和业务特征：*.*Controller.java*、*WeChatPay*

寻找具有相似规则特征的文件：*ALiPayController.java*

提取业务特征：*ALiPay*

寻找具有相似业务特征的文件：*ALiPayService.java*

提取规则特征：*.* Service.java*

组合业务和规则特征：*WeChatPayService.java*

找到以上文件并提取函数签名

如何实现和开发者同频 —— 代码解释

当我要求Comate分别解释代码块、函数、文件、目录、代码库时，它怎么才能从业务角度给出解释？如基于调用链分析给出流程图、ER图、架构图等？

- API类函数解释：
  - 基于调用链找到向下找到此函数的调用关系，以及对应的Auth、Interceptor、Filter等
- 持久化类函数解释：
  - 基于调用链向上找到此函数的调用关系，以及对应的SQL语句、实体定义等
- 通用函数解释：
  - 同时向上、向下获取N层调用关系

直接获取已经初始化好的项目描述，结构如下：

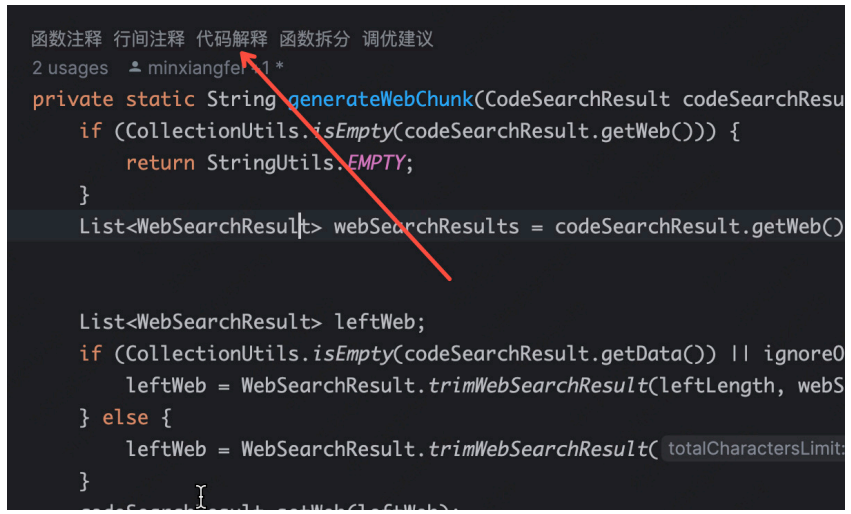
这是一个用来xxxxx的项目，它的使用了xxxx、xxxx等框架，分为xxxx、xxxx等模块。主要逻辑包括xxxx、xxxx、xxxx。对外暴露了xxxx、xxxx等关键API。该项目有如下重点业务关键词：

- *xxxxxx*
- *xxxxxx*

增强解释效果的知识

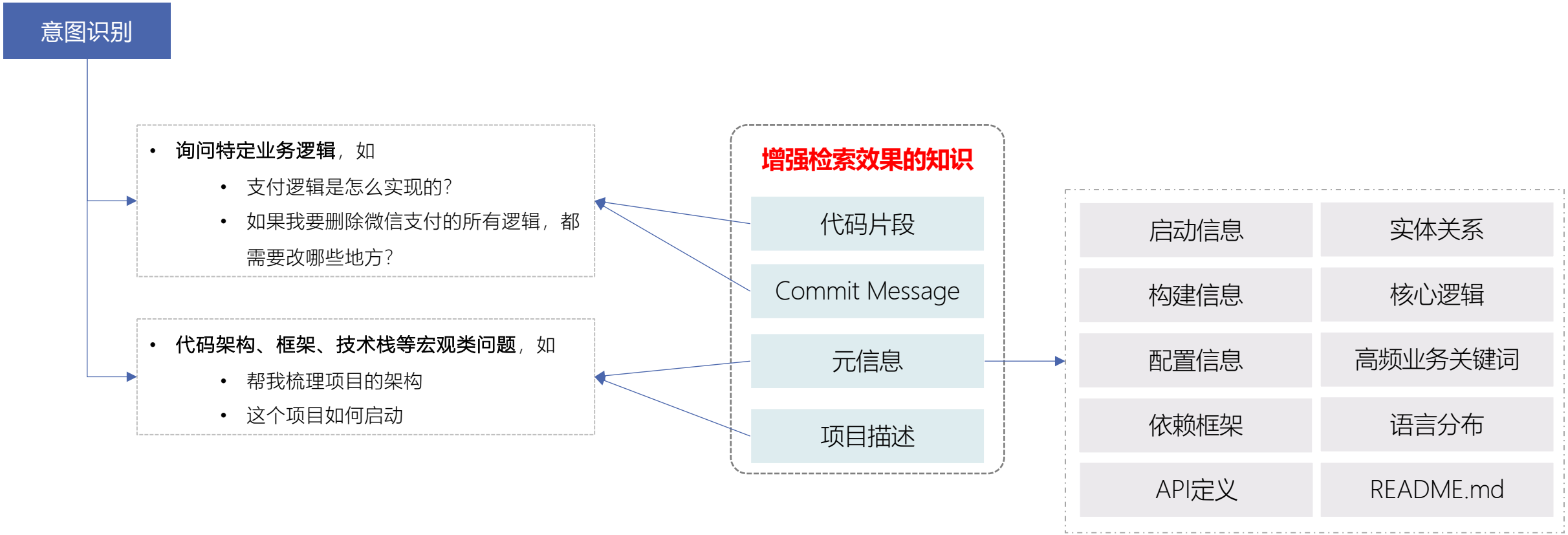
函数调用链

项目描述



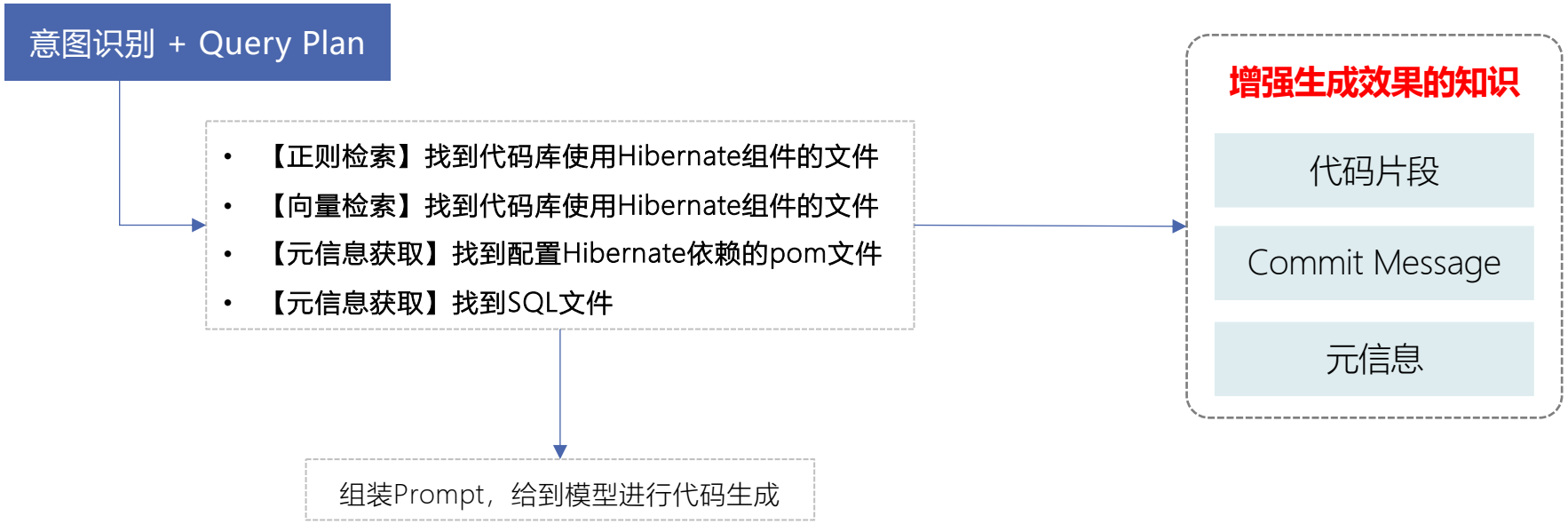
如何实现和开发者同频 —— 代码检索

当我新接手一个代码库，需要了解代码架构、具体的业务逻辑，如何才能给出像Mentor一样的讲解?



如何实现和开发者同频 —— 代码生成

当我希望将整个代码库的Hibernate组件替换成Mybaties组件，期间涉及到实体定义、SQL语句、事务等方方面面的分析，Comate怎么做才能把这件事做好?



但这远远不够，只靠一次知识增强没有办法获取全部知识，对于复杂问题的解决没有办法做到立竿见影。需要获取哪些知识大部分是在在解决问题的过程逐步中分析、明确出来的——Agent。

人机协同新范式——效能提升银弹

AI 对效能的提升是替代Task而不是Job! AI 对效能的提升绝不仅限于编码领域！ AI 可以赋能整个软件工程！

人机协同新范式——人机协同宣言

用好AI将是工程师的基本功

掌握并善用AI将成为工程师的必备能力，率先用好的组织和个人将建立竞争优势。

You are in control

AI能极大的提升研发效率，但需要人确保方向的正确性，最终为业务结果负责。

人机协同，共同进化

繁琐、重复的事情AI来做，创造性的事情人来主导，多使用、勤反馈，实现人机共同成长。

THANKS