Tencent 腾讯 | CSIG 云与智慧产业事业群

腾讯云 AI 代码助手解决方案介绍

2024.11 腾讯云-开发者中心

目 录

CONTENTS

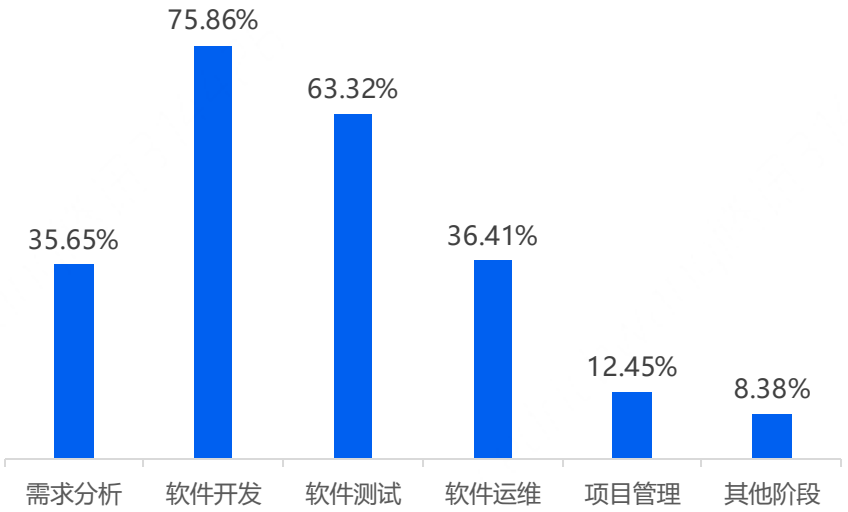
AI4SE 的业界发展趋势

AI4SE
- 以 大模型等AI技术为驱动，
- 以 提高软件研发运营智能化水平为导向，
- 以 提质增效为目标的新一代智能化软件工程

据中国信通院调查：

软件开发&测试场景是 AI4SE 技术应用的排头兵

软件工程各阶段AI技术应用比例



*数据来源：银弹案例数据、中国信通院

AI4SE 相关的商业化解决方案百花齐放，

AI相关的生产力工具正在重塑技术人员的工作方式

编码 （最成熟）

IDE

Commit: Graphite, planar, Codeball

Copilots: GitHub Copilot, tabnine, Sourcegraph, codeium, codium, mutable.ai, Magic, grit, Second

产品设计

Ideation: Midjourney

Prototype: Visily, uizard, Figma AI

Research: Kraftful, QoQo, Looppanel

Design: stability.ai, Adobe Firefly, Galileo AI

交付与监控

Operations: airOps, KUBIYA, arize

Observability: Grafana, Cribl, baselime, ctrlstack, dynatrace

CI/CD: harness, Buildkite

需求分析

Facilitation: miro

Documentation: Userdoc

Collaboration: ATLASSIAN Intelligence

系统运营

Research: Kraftful, Enterpret

Marketing: Jasper, copy.ai, regie.ai

Customer Support: INTERCOM, ada

产品简介

腾讯云 AI 代码助手 是专为研发团队设计的 AI 辅助编程产品，以**腾讯混元代码大模型**为产品基座，支持上百种主流编程语言和主流 IDE，提供**代码补全、技术对话、单元测试、代码诊断和智能评审**等能力，覆盖编码全流程场景，辅助开发者提升编码效率和质量，助力研发团队提质增效。

产品竞争力

- 补全速度快 补全准确性高
- 模型能力强 问答泛用性好
- 产品开放度高 多Agent可扩展
- 交付形态丰富 云上云下均可

产品价值

- 提升开发速度 编码质量提升
- 代码规范和最佳实践落地
- 助力企业打造全栈开发团队
- 提高员工编程体验满意度

腾讯实践

- 3w+ 日活用户
- 35% 代码生成率
- 人均千行bug率 降低31.5%
- 人均编码时间 缩短40%

落地案例

招商银行 CHINA MERCHANTS BANK　CMS 招商证券　xiaomi　中国太平 CHINA TAIPING　中国联通 China unicom　CUSC 联通智网

代码补全 - 提升编码效率

通常，研发人员的日常编码流程：思考业务逻辑 → 思考程序逻辑 → 编写代码实现

BUT，编写代码实现 确实**耗时较长**

腾讯云 AI 代码助手

应时代而生

```
17      train=True,
18      download=True,
19      transform=transform)
20
21  # 定义网络结构
22  class Net(nn.Module):
23      def __init__(self):
24          super(Net, self)
```

```
11
```

AI 代码补全，可帮助研发人员快速生成代码，提升编码效率，形成新流程：

思考业务逻辑 -> 思考程序逻辑 （AI生成代码）-> 确认代码实现

01 丰富的生成和接受策略

支持行补全和块补全、智能补全，方便用户快速确认。

接受结果时，支持全部接受 / 按行接受 / 按字符接受 多种方式。

02 高效的生成速度

代码补全的速度、平均延时 **< 600ms**，最匹配研发心流。

研发可以想好程序逻辑并确认结果

03 准确的补全结果

补全接受率（接受 AI 推荐的次数 / AI推荐的总次数）为 **20% ~ 30%**。AI 编码占比为 30% ~35%， **业界遥遥领先**

核心竞争力 - 工程级别跨文件代码补全

AI 编码辅助中的代码补全，主要由 **感知 – Prompt 构造 – 生成** 这三个部分组成。

因此，提升代码补全效果的方法大致可分为：①**更全面的感知上下文**、②**更精确的封装 Prompt** 以及 ③**友好的结果处理机制**。

	更丰富、理解工程	更精准、降低幻觉	更智能、优化体验
业界做法	大部分产品都只感知当前文件，少数先进产品会结合AST，但最多也只支持到**“打开的文件”**进行分析。	大部分产品都只参考**开源社区**的做法，进行 **PSM或者 PMS** 进行 prompt 封装，效率和质量都不高。	固定的补全粒度（比如 大/中/小；或者行补全/块补全），不智能。
腾讯实践	感知的范围更广，在加载工程的时候，就会进行全项目的感知与解析，支持未打开的关联文件。	更精细的prompt封装，通过压缩、截断、排序、顶层注释等方式，提升补全性能，降低模型幻觉。	智能补全，可以根据代码上下文，分析最适合的补全内容，体验更优。

智能问答 - 快速解决研发问题

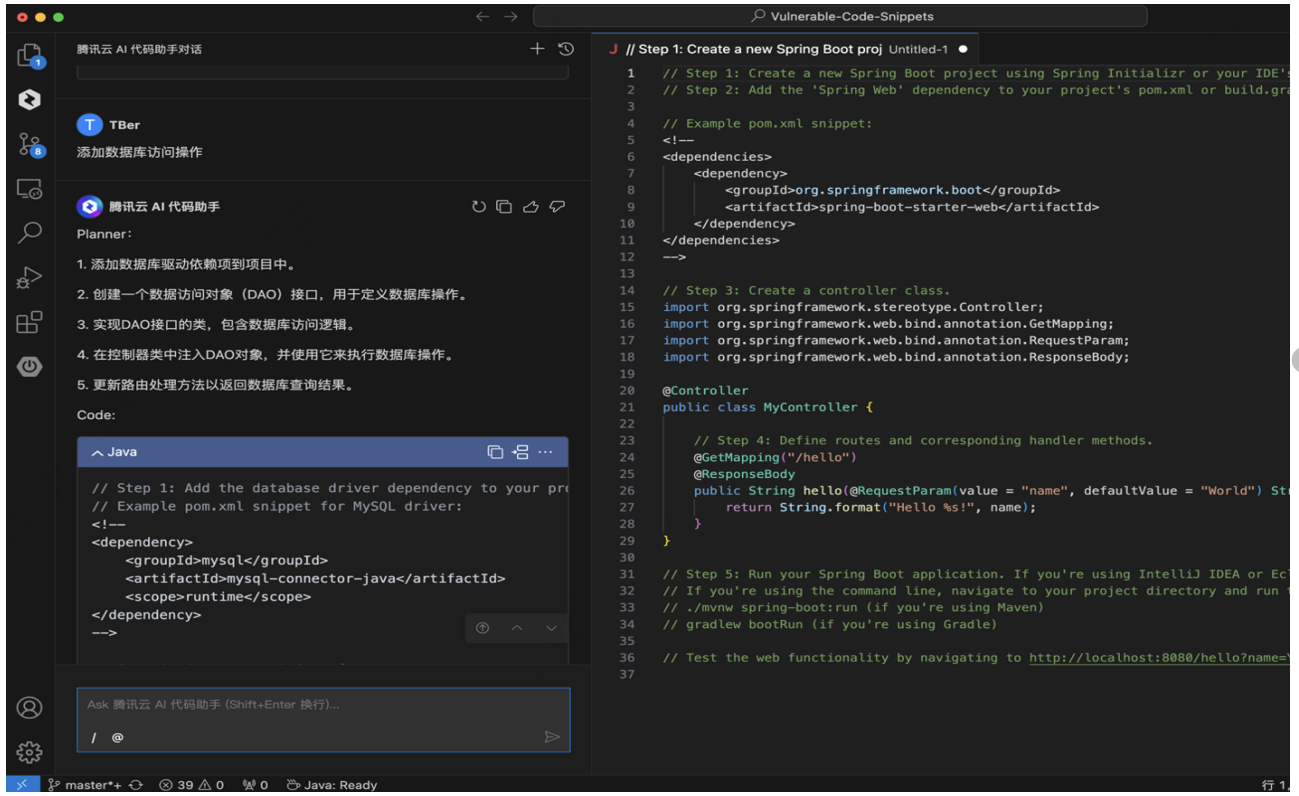
场景一：检索、分类、总结

当遇到一些不熟悉的技术实现，传统的方式是通过网上检索方式，从一堆结果中筛选出想要的代码，然后做归纳总结，而现在我们可以直接向 AI 助手提问。

场景二：快速解读和理解代码

当我们接手一份新代码/不熟悉的代码，AI 助手可以帮助我们快速解读和理解代码，并回答针对这部分代码的问题。

编码过程中
智能问答
主要解决场景



1 编程技术问题咨询

在左侧对话框中，可以询问 AI代码助手 编程相关的技术问题，也可以结合已有代码片段来提问。

2 多轮对话逼近准确答案

AI 代码助手支持通过多轮对话来逐步逼近正确答案。当第一轮对话效果不满意时，用户可以追加提问，直到获取满意的答案 。

3 业务代码辅助生成

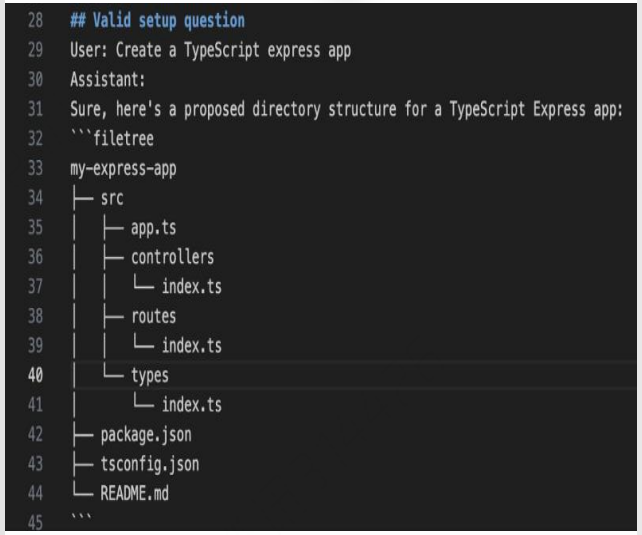
在技术对话中，内置了多样化的命令来辅助业务代码生成。比如 commons 自动注释、fix 代码修复、nameVariable 函数命名等。

核心竞争力 - 理解软件工程，精准问答

核心竞争力

- 理解软件工程，精准问答
- 正负反馈持续优化
- CoT 思维树拆解，处理复杂任务

01 OPTION

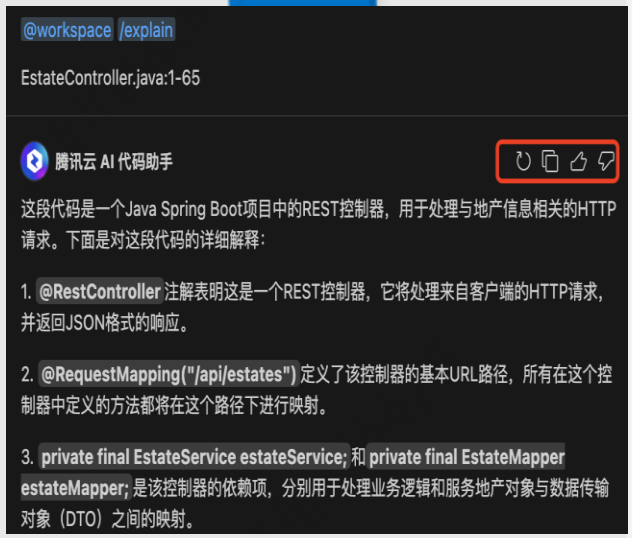


@workspace ，精准问答

技术问答的时候，可以通过 **@workspace** 让问题获得更当前工程的简介、目录结构、语言类型等信息，，使得回答更精准。

此特性可以让通用问答、代码解释、代码修复等功能的结果更符合当前项目特点。

02 OPTION



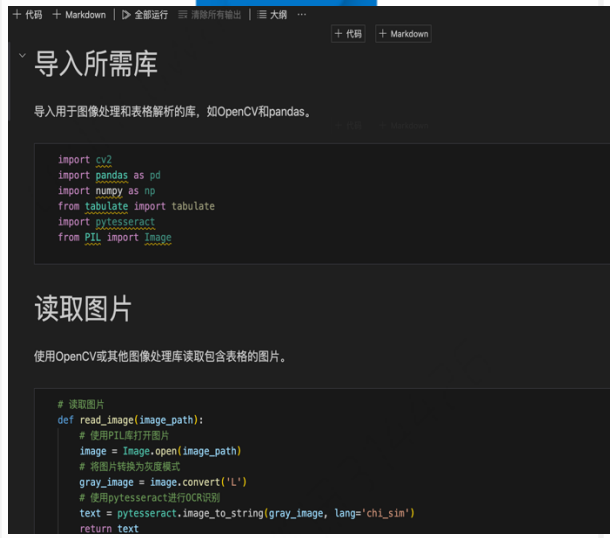
正负反馈与持续优化

① 支持点赞点踩，后台可以收集用户的反馈数据

② 当前窗口可以根据反馈情况，调整优化后续答案。

③ 在私有化场景，支持结合反馈数据完成模型后续的强化训练，持续优化模型表现。

03 OPTION



CoT 思维树拆解，处理复杂任务

产品中提供 **/newNotebook** 扩展，支持根据需求拆解的几个子任务，通过 title 和 content 组装；然后根据子任务，组装成子任务所需执行的提示词列表。用于用户点击创建文件后依次执行代码文件。实现 **需求分析 – 子任务分配 – 任务依次执行** 效果。

单元测试 - 提升编码质量

单测痛点

1、开发任务重，愿意写单测的人很少。即使在流水线中加入覆盖率要求，也有很多人敷衍。

2、单元测试代码质量低，很多代码的逻辑不闭环，仅起到了覆盖率的要求，无业务价值。

解决方案

1、通过自动生成单元测试代码，减轻开发的单元测试开发负担。

2、通过多轮问答，结合业务逻辑，逐步逼近准确结果，提升单元测试准确度。

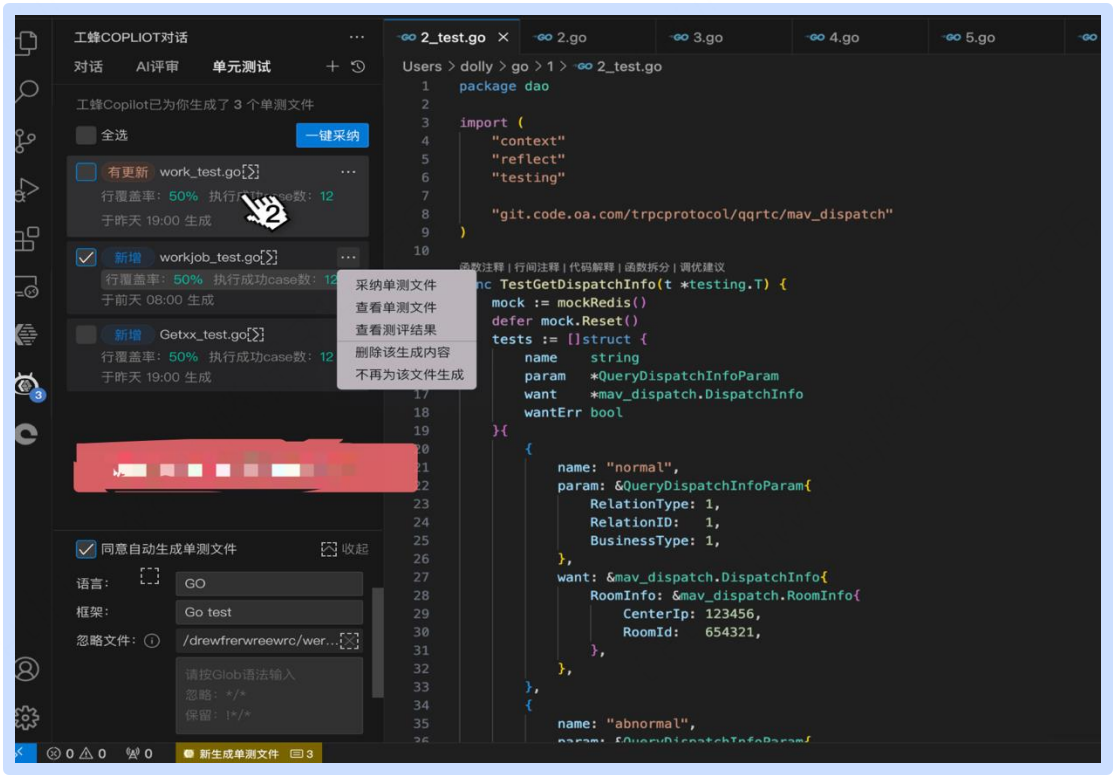
落地实践一：多轮技术问答，生成复合业务逻辑的单测代码

实践效果

1、覆盖人数：目前使用 AI单元测试的用户，占据总体使用 AI代码助手用户的 57%

2、效率提升：深度使用 AI 单元测试的研发同学，在单元测试的消耗时间缩短 50%+（腾讯内部数据）

3、质量提升：使用 AI 单元测试之后，活跃团队的千行 BUG 数量降低 30%+（腾讯内部数据）



落地实践二：根据代码框架，自动批量生成单元测试文件（内测中）

核心竞争力 – 结合工程上下文生成单测

结合已有单测文件增强

根据工程结构，自动获取当前项目内的其它单元测试文件内容，参考它的单测/内容与框架增强当前的单测效果。

结合语法树增强

通过跨文件能力，结合语法树获取更全面的功能业务逻辑，增强单测表现，让测试用例更贴合业务逻辑。

RAG 增强 – 让 AI 更懂业务代码

问题痛点

1、问到业务编码逻辑问题的时候，会出现“答非所问”、“胡说八道”

2、说了很多“正确的废话”，回答“绕来绕去”

3、回答得很有道理，但是没有给出相关的代码实现，研发还是不知道应该怎么做

功能效果

1、问答能力提升：结合企业文档，可以理解一些独特的业务场景，强化回答效果

2、问答准确率提升：针对企业特定领域的问题，结合 RAG 之后，问答准确率从50% 提升到 90%

丰富的文档类型

支持丰富的文档类型，包括：pdf, markdown, txt, docx, doc, html 网页、代码(检测非二进制的代码文件), ZIP压缩包（.zip/.gz) 等

代码/文档混合处理

会根据不同的处理方式，分别处理文档和代码。在增强回答的时候，可以结合代码/文档，分别提供解决方案内容与解决方案代码。

API 文档处理

针对 API 文档，会当成单独的类型处理。在问答过程中，可以所选代码内容与 API 文档内容，给出相关的代码实现。

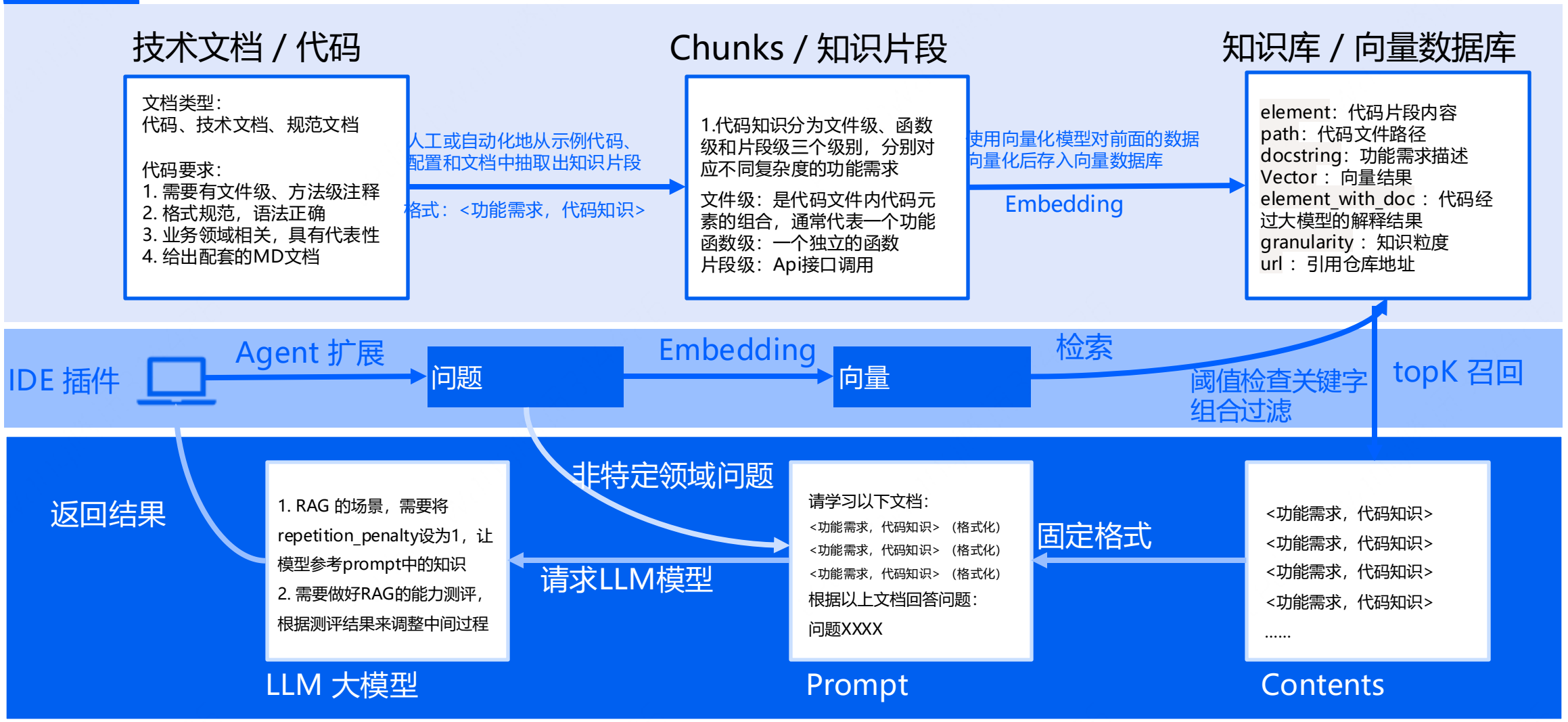
多知识库问答

在问答的时候，会遇到一个问题，需要参考多个知识点/知识库的场景。问答中，可以选择多个知识库增强回答效果。

核心竞争力 – 智能代码内容处理，精准召回

RAG 技术基本都是用来处理自然语言的，无论是量化处理，还是召回，业界都仅在自然语言场景下可用。

如何做到 “代码文件” 的量化与准确召回一直是业界难题。



AI 代码助手的核心竞争力总结

超强壁垒：AI 代码助手与业界不同的”三大能力”

01 AI在工程场景的能力

- 工程分析的算法（专利）
- 全工程感知能力（首家）
- 引用内外部知识库增强效果
- 单元测试生成（适应多框架）
- 打通代码评审流程（首家）
- 最灵活完备的开放式架构

02 AI在编码辅助的能力

- 智能的代码补全策略（最丰富）
  31 种补全策略，智能识别补全意图
- 使用AST增强补全效果（首家）
  生成率提升5%
- 直观的数据度量平台和指标
  科学体系化的度量指标（30+）
- 灵活的模型调度策略
  支持部署多个模型，根据场景、用户切换

03 代码大模型的能力

- 代码补全策略细化训练
  对 31 种补全策略进行预训练
- 完整的微调方案
  完整的企业代码微调方案，包括训练方法、语料清洗工具、评测工具等
- MoE模型结构（首家）
  国内大厂首个MoE（多专家）模型
  同推理效果，降低推理成本 52%
  同并发下，补全延迟降低 31%

多样化的产品形态

旗舰版（SaaS）

企业/团队 产品license（订阅）

- 企业知识库问答
- 企业管理
  - 授权管理
  - 研效看板
  - 成员统计
  - 企业知识管理

专享版（云上托管/vpc）

企业/团队 产品license（订阅）

- VPC 部署形态
- 专属网络访问
  - 专属服务
  - 专属插件
  - 专属域名
  - 安全审计

企业版（私有化）

企业/团队 产品license（订阅）+交付（一次性）+维保（订阅）

- 专属推理服务
- 定制企业插件
- 模型微调训练
- 定制插件扩展

基础能力

技术对话

对话 | 注释 | 对话解释 | 检查 | 优化 | 单测 | 提交信息

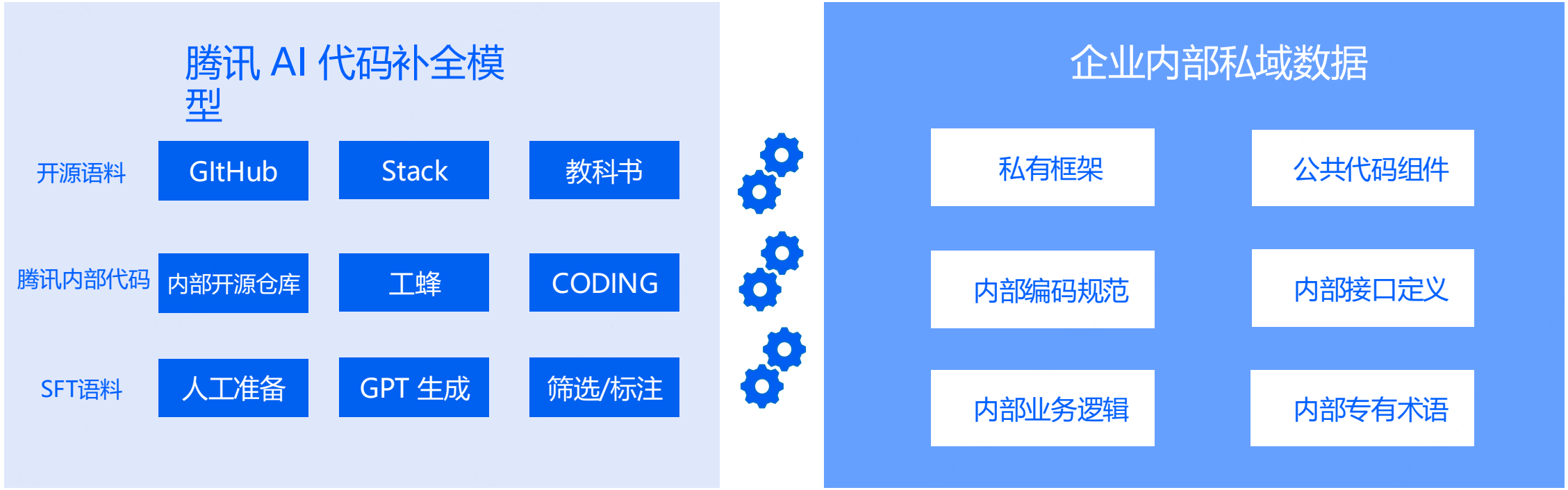
代码生成

实时续写 | 工程理解 | 注释生成代码

目 录
CONTENTS

大型企业代码模型的应用诉求

企业内部存在大量的私域数据是客观事实，从代码生成角度来看，私有的框架、公用代码组件、内部编码规范、内部接口定义和说明以及内部业务逻辑这些内容客观存在。



虽然代码模型基于大量的数据进行了训练，也可以完成绝大多数编程语言的生成，但是它并没有见过企业内部的代码，无法生符合内部编码规范、风格的代码，也无法正确调用私有代码框架/组件和接口的代码。这个问题成为了影响 AI 辅助智能编码产品在大型企业落地的主要障碍。

Prompt 工程、RAG、微调 怎么选择?

大模型特点

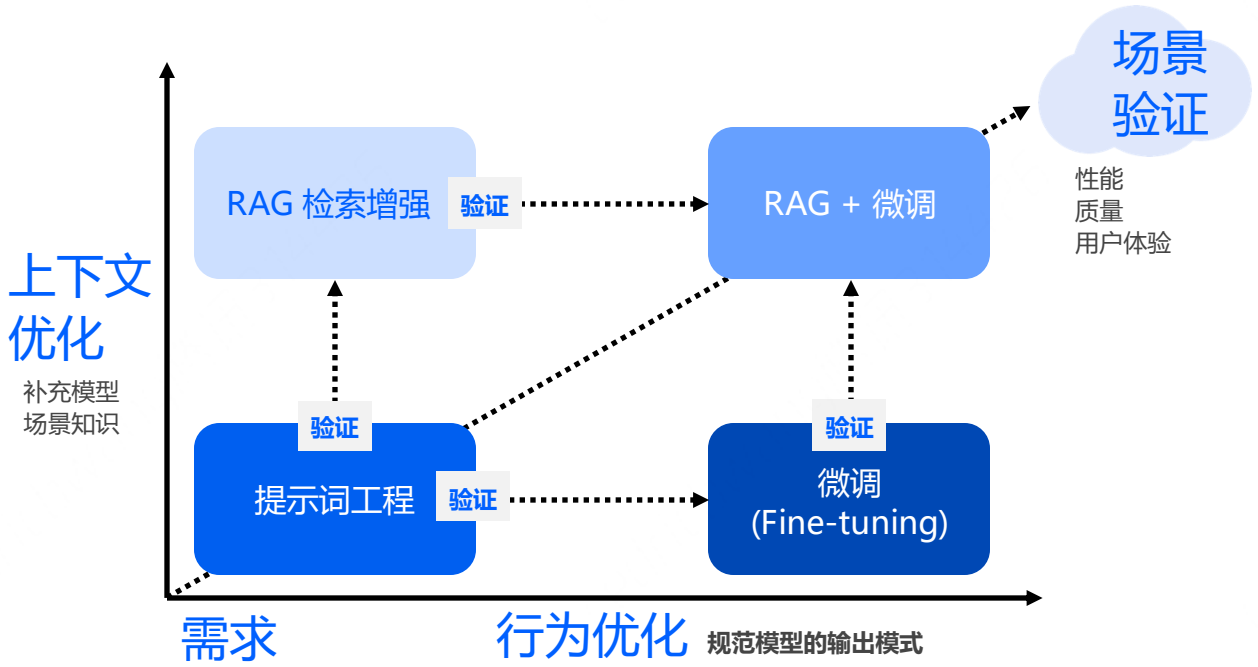
1、不确定性 → 提升模型输出的稳定性质

2、静态性 → 扩展额外数据

构建大模型应用是一个典型的迭代过程，此过程需要：

1、从应用场景出发，先搞清楚我们要做什么；

2、优化大模型应用系统的性能、质量和用户体验。



1 尝试定义新的提示词

首先使用提示工程的方式优化大模型应用，这是成本最低，见效最快的方式。提示词工程可以同时为模型补充上下文（上下文优化）和优化模型的行为（行为优化）。

2 多样本学习 和 上下文引导学习

将提示工程的能力直接赋予用户，用户根据自己的实际场景，在提示词模板中丰富场景的用例样本，让模型的返回更符合业务场景

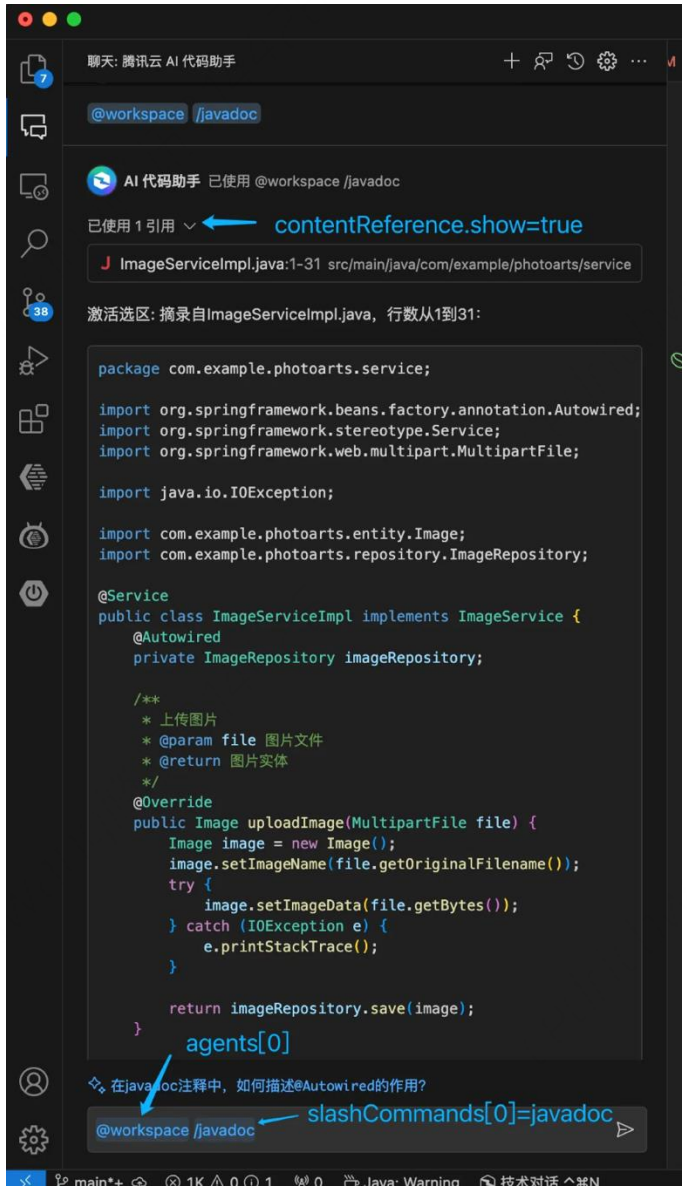
3 检索增强式内容生成（RAG）

当模板中大部分内容是模型未知且不断变化的知识，且需从多个数据源获取，导致创建不同模板应对各种场景时，可以考虑引入RAG进行优化。

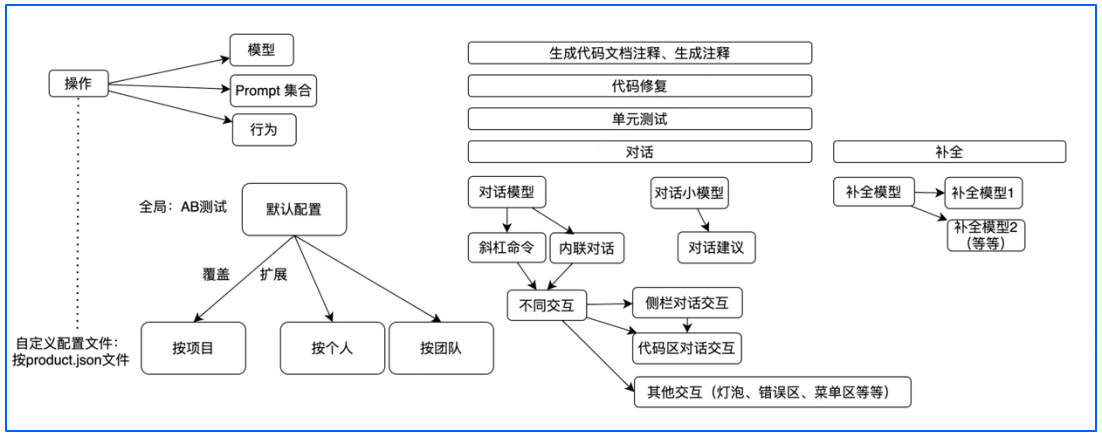
4 微调 （Fine-tuning）

当模板主要包含各种格式示例，且目标是让模型模仿示例输出，但即使示例充满模板，输出仍不稳定时，可以考虑进行微调以优化模型表现。

Prompt 工程扩展



+



自定义Prompt

+

```
{
    "agents": [
        {
            "name": "workspace",
            "slashCommands": [
                {
                    "name": "javadoc",
                    "prompt": "请为该代码添加javadoc注释，要符合javadoc的标准以及注释规范。注意不要修改原代码。",
                    "contentReference": {
                        "show": true
                    }
                }
            ]
        }
    ]
}
```

Prompt as Code

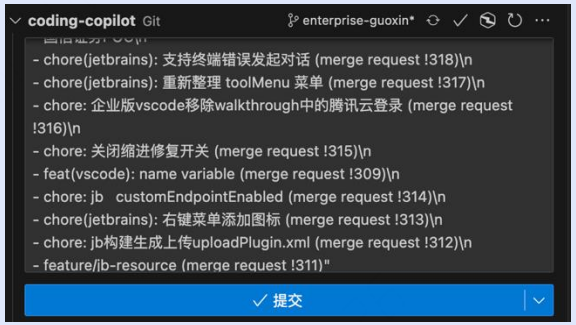
+

原生扩展交互

灵活的 Agent 扩展，打造 AI 编码智能体

已经集成的内容

AI 生成提交信息



代码 AI 评审

未来可扩展点

代码库集成扩展

- 识别代码设计
- 规范格式问题
- 识别性能问题
- 识别和修复漏洞
- AI 配置评审规范
- 提升代码可读性
- 提升代码可维护性

CI 集成扩展

- 修复构建错误
- 分析构建耗时
- 识别代码扫描错误
- 识别代码安全漏洞
- 识别自动化测试错误
- 实现安全门禁

CD 集成扩展

- 识别制品漏洞
- AI 变更评审
- 识别和修复部署错误
- 集成 OS，终端补全
- 识别配置内容
- AI 架构治理
- AI 根因分析

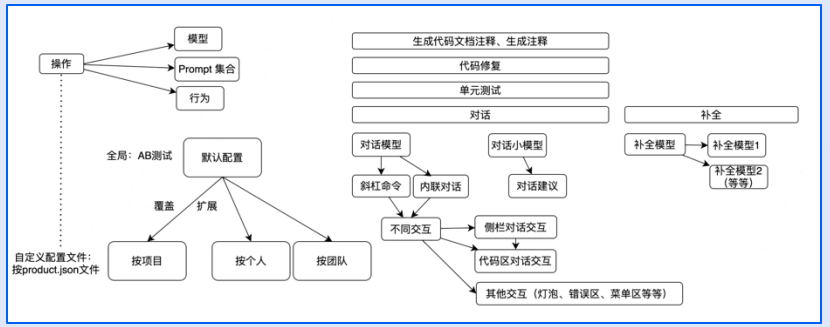
AI Agent 框架

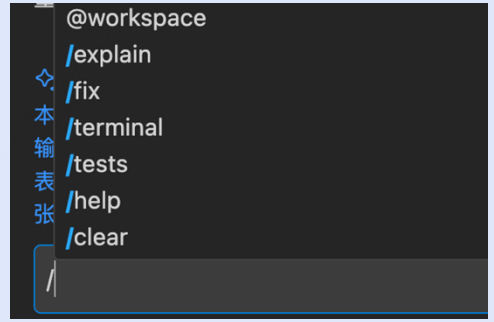
IDE 插件命令扩展 （Prompt As Code）

- 行为感知
- 提示词
- 工具执行
- 权限控制
- 多轮处理
- 思维链路

Prompt 模板 / 配置中心

模型调度





多样化的 RAG 检索，增强搜索

多样化的数据源支持

支持多样化的文档作为知识库的数据源，文档类型包括：普通文本、Word、PDF、Excel、md 等格式。另外支持代码仓库、数据库作为数据源。

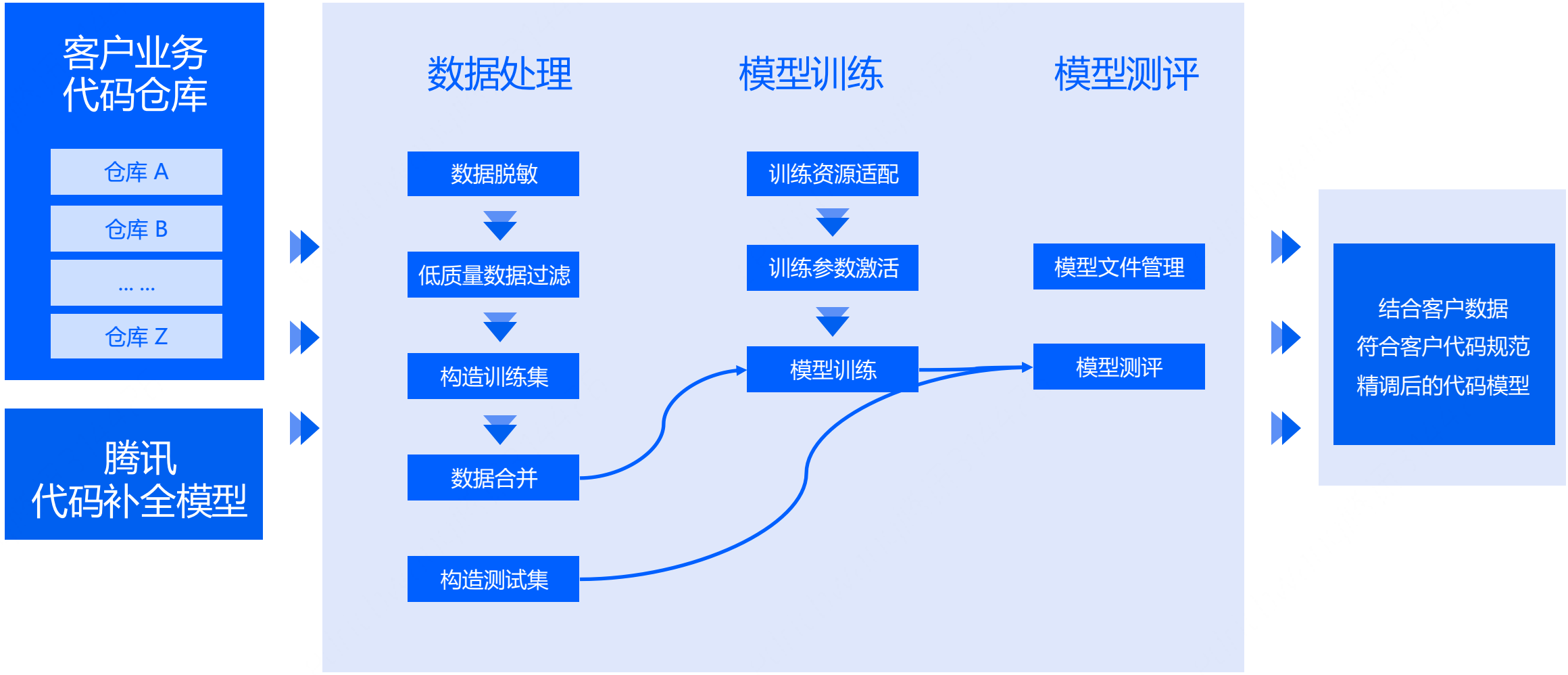
知识切片与存储

根据不同的数据类型，分别处理数据切片，并将切割之后的数据存放到向量数据库中。

便捷的检索增强体验

在技术问答的时候，可以通过 @知识库 的方式，选择某个领域的知识库，检索相关知识后，强化问答效果。

模型轻量级微调工具



提供封装好的模型微调工具，支持用户自主、低成本进行模型微调

目 录

CONTENTS

腾讯内部实践

贯穿研发流程，提升研发效能

腾讯在 AI 辅助编码领域落地情况

- **腾讯投入大量的研发团队进行 AI4SE 相关场景创新**
- **在 AI 辅助编码部分，经过了内部团队长时间的验证与迭代，产品能力成熟**

01 沟通阶段

1、 Chat 对话，优化交互，体验性和兼容性更好
2、支持指令，支持多个 Agent 能力
3、结合开放的 Extension 机制，关联上下游平台。

02 编码阶段

1、 补全场景的Fill in Middle特性，生成质量可完整停止不重复。
2、 智能的行补、块补全的切换
3、跨文件代码补全能力（工程级别的多种策略）
4、生成代码上的注释文档
5、生成业务逻辑的代码注释

03 单测阶段

11、理解工程下，根据业务逻辑自动生成多个测试用例
2. 自动补全多类型的单测框架

04 排错阶段

和内部安全团队结合，通过Agent 能力扩展，调用代码扫描能力，并通过AI大模型进行解释和代码优化建议

05 调优阶段

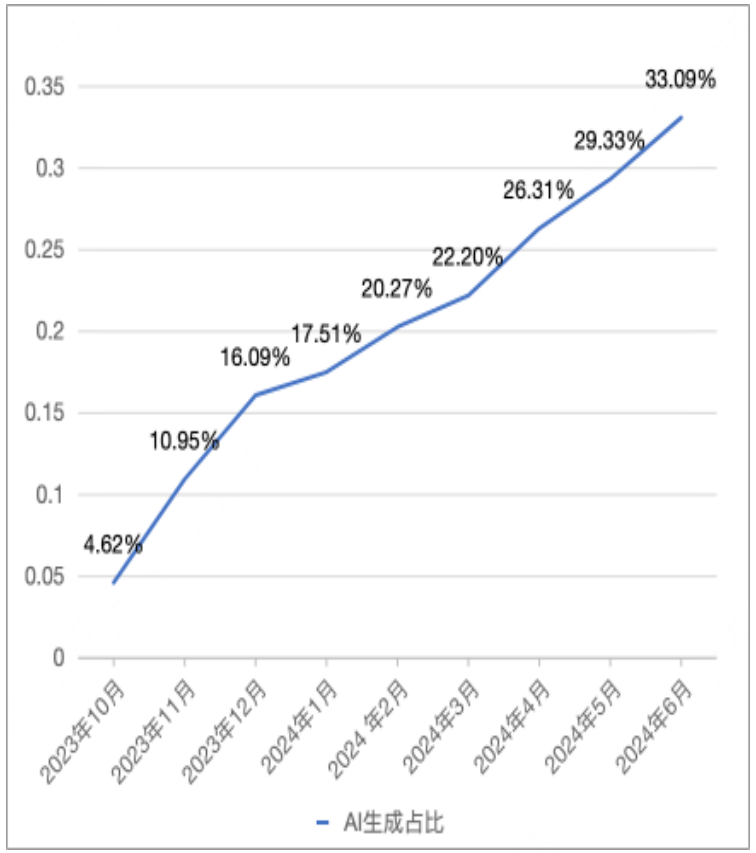
获取软件工程上下文，结合 CoT 思维树拆解方式，强化理解和分析一个项目中的多个文件，从而为开发者提供更加精确和相关的代码建议。

06 评审阶段

1、提供IDE内的预评审和AI生成提交记录
2.、结合内部工蜂仓库做AI评审能力

腾讯落地情况

腾讯内部代码生成率增长曲线



效果的提升需要产品的持续迭代和运营

提质增效数据

18.8% 人均需求交付个数 提升 18%

41.3% 人均编码行数 提升 41.3%

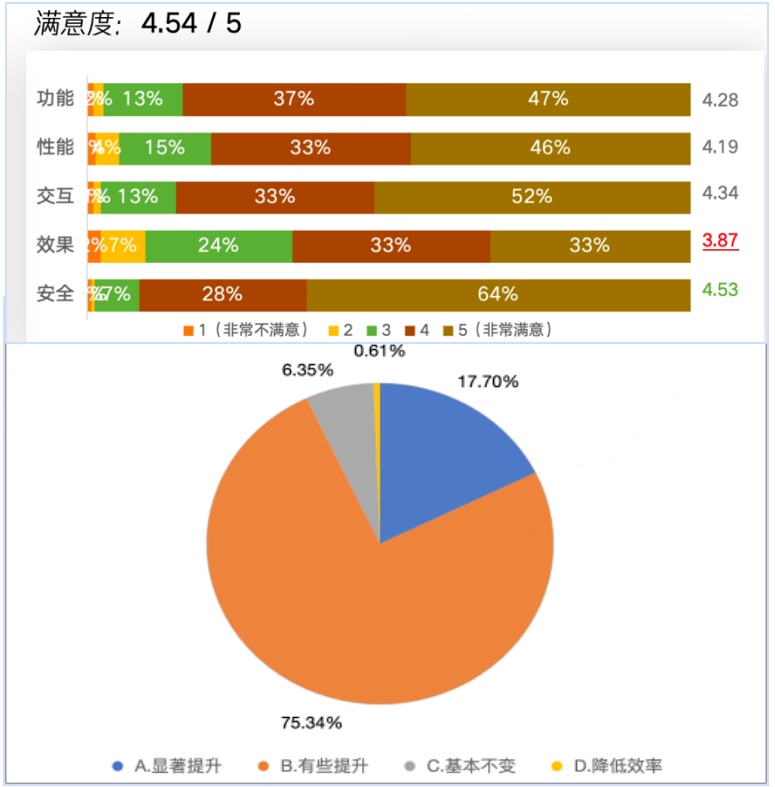
40.0% 人均编码时间 缩短 40.0%

31.5% 人均千行Bug率 降低 31.5%

* 数据来源：CSIG

用户口碑

1712 份调查问卷结果如下：



93%的受访用户认为AI助手可以帮助自己提效

腾讯 78.14% 人在用（超3w人），有35%的代码由 AI 助手生成，代码采纳率>28%，用户反馈的满意度高

如何衡量 AI 辅助编程的效果？

1 开发效率提升

开发者的编码时间占比 × AI 生成代码占比 = 节省的开发时间比例。例如，员工有 40% 的时间花在编码上，AI 生成了 30% 的代码，则可以理解为节省了 12% 的开发时间。

2 代码质量提升

代码的缺陷密度是一个滞后指标，反映代码质量，如通过统计千行代码缺陷量指标来观测代码质量提升情况。

3 员工编程体验的满意度

员工编程体验的满意度是一个主观指标，反映工具对于员工编程工作的帮助，如工具的易用性和实际工具的使用效果。

4 助力现有研发体系能力提升

产品除了提供标准产品能力之外，可提供开放式架构，客户可将“AI 辅助编程”融合到现有的研发体系中，助力研发体系能力升级。

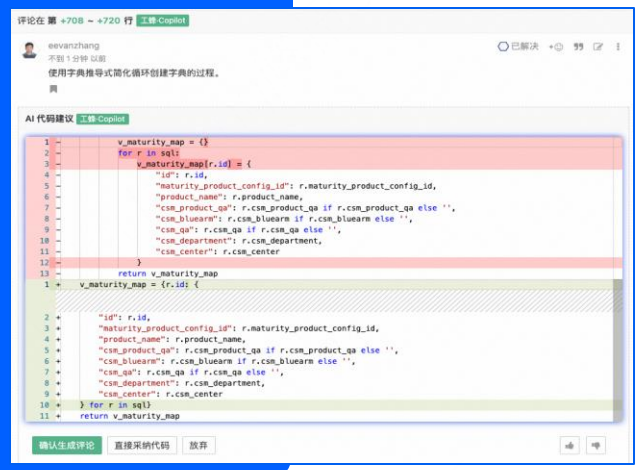
实践：建设 AI 编码的周边场景，提升开发效能

01 AI 生成变更摘要

- 【功能】：自动唤起 AI，根据关联 需求单、变更代码文件、代码文件提交信息，一键生成合流 MR 变更摘要
- 【效果】：腾讯 20% 的评审单据，由AI提炼概括变更内容，便于评委理解。生成内容好评率 88%

02 AICR 智能修改

- 【功能】：根据人工评论、AI 评论和代码上下文，一键生成代码修复建议。可结合WebIDE 等能力快速联动 CI 流水线验证效果，并持续迭代建议。
- 【效果】：AI 直接修复问题代码 1.4w次/月，采纳率 2



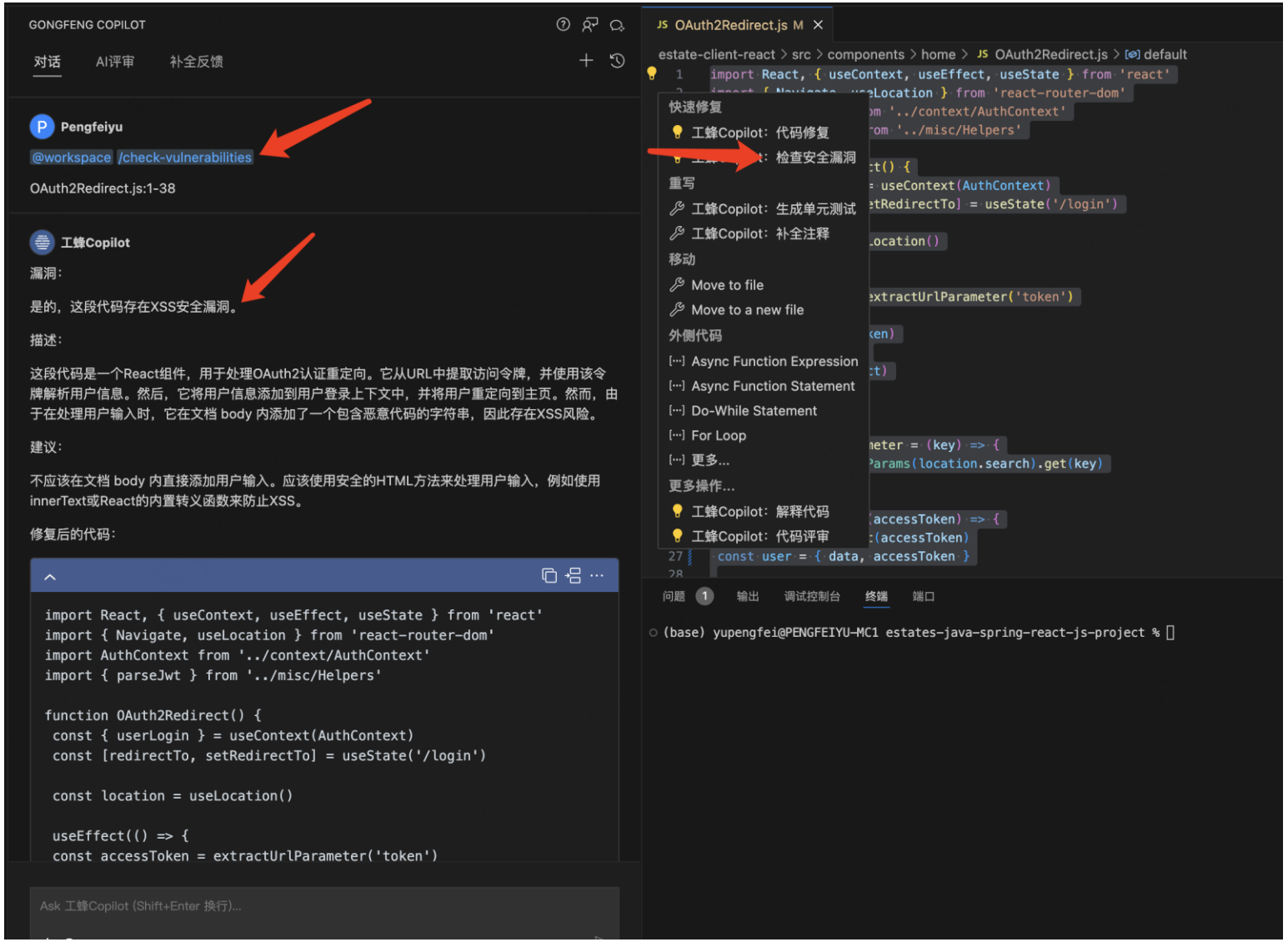
03 AI 安全漏洞修复

- 【功能】：根据代码扫描的安全问题、规则、建议，结合代码上下文信息，调用链信息，提供代码修复建议。
- 【效果】：AI 直接修复严重安全漏洞 2300个/月，采纳率 36%

04 AI 自动评审

- 【功能】：AI 辅助查找MR/CR变更代码中的问题，支持一键生成 CR 评论；支持 AI 评论列表、文件树跳转、AI意见精准挂载行号、识别问题严重等级。
- 【效果】：AI 发现问题数 7w/月，已与人工发现问题数达相同量级，用户对 AI 意见的好评率 60%

实践：Xcheck 代码安全问题检查与修复



结合代码安全检查能力，前置在 IDE 插件中，并使用大模型的能力直接生成可用代码，快速修复代码问题，实践**安全左移**。

工程级检查修复：

1. 支持代码片段级代码检查与修复建议输出
2. 支持多文件代码检查与修复建议输出

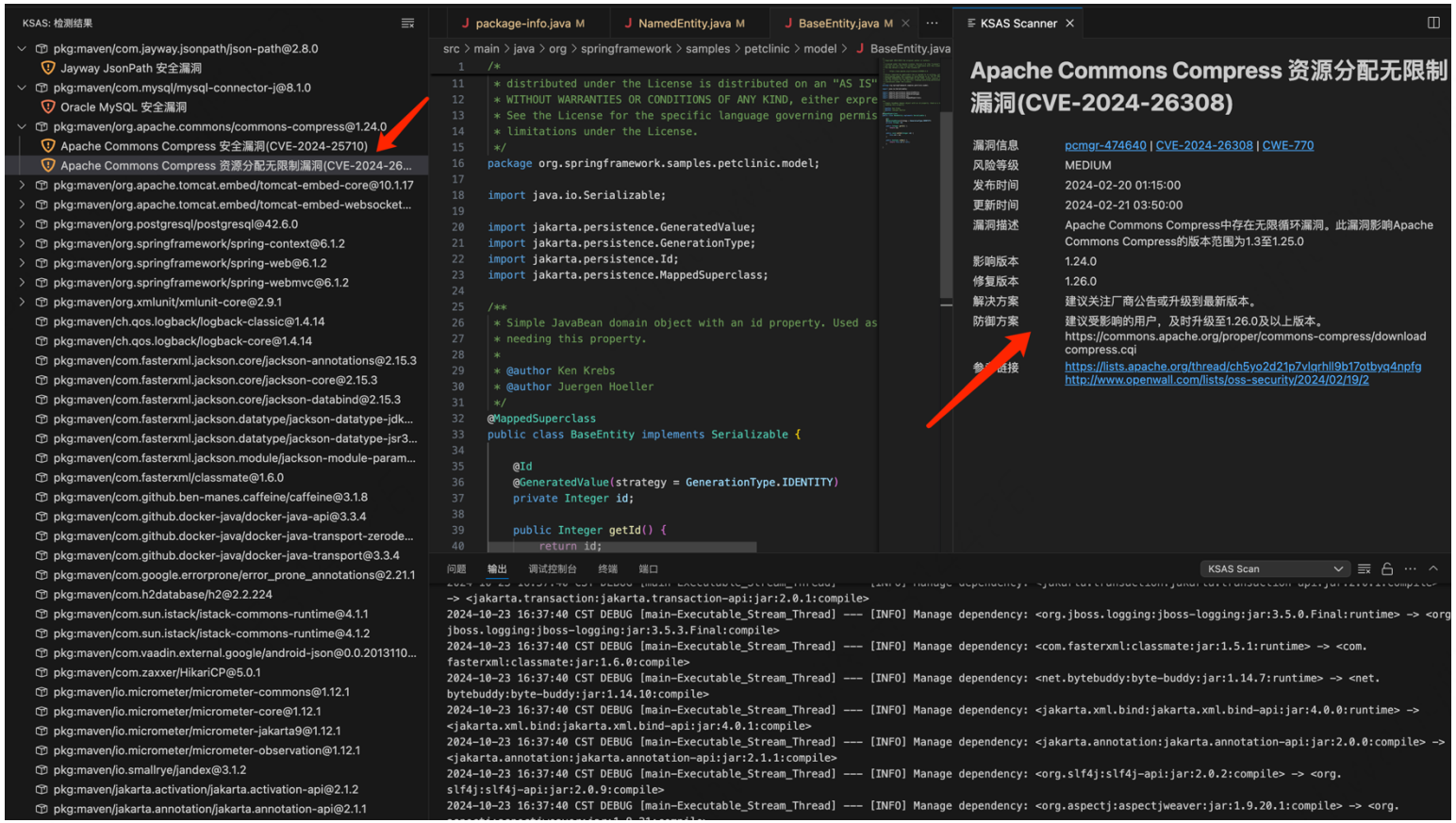
污点类问题分析：

1. 可以分析污点问题，逐级分析污染链路
2. 针对污染链路给出修复方案

自定义检查规则：

1. 支持自定义检查规则，可通过Prompt / Agent 等方式扩展检查规则与逻辑

实践：T-Sec 代码与组件成分分析



软件供应链安全在研发活动中越来越被重视。 AI 代码助手结合腾讯 T-Sec 安全扫描工具，可以在研发过程中，随时关注软件供应链安全，可以做到软件漏洞、开源组件的深度分析与解决。

依赖管理分析：

1. 动静态依赖分析：准确处理版本依赖问题
2. 组件Scope：不同包管理器具备不同scope（test，dev等）

代码片段分析：

1. 文件级分析：快速分析整个文件相似度，加速SCA分
2. 片段级分析：片段级分析识别，发现抄袭情况

漏洞可达性：

1. 漏洞存在性：通过patch数据判断漏洞特征是否存在
2. 漏洞触发性：分析漏洞影响函数调用链，判断漏洞调用触发性

外部客户落地实践

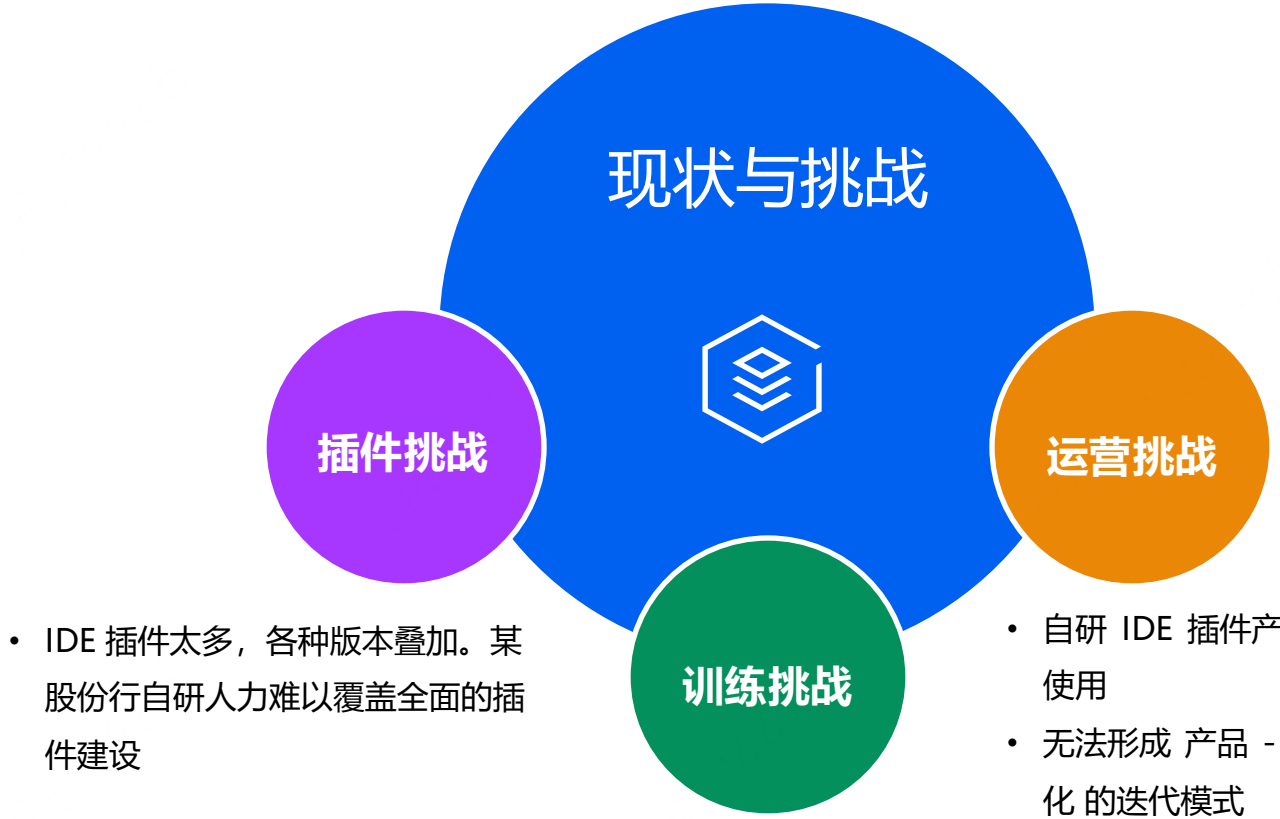
探索 AI 编码落地场景，重点落地代码补全

某股份银行

项目背景

某银行与腾讯属于战略合作伙伴，面对腾讯内部与某行针对 AI 辅助编码的需求共频，双方自2023年12月腾讯云 AI 代码助手与某行签订智能编码领域合作协议，成立 AI 代码助手联创项目组，双方高优先级投入，项目成员高效配合，紧密协作，共同推进某银行 AI 代码助手落地。

在合作过程中，腾讯团队为“AI 代码助手”在某行的落地进行全面赋能，覆盖了插件开发、模型评测、模型训练、运营推广和人才培养等五大关键方向。这种全面的合作不仅加速了 AI 代码助手在某银行的实施，也促进了腾讯团队对金融行业场景的深入理解，从而形成了行业内的最佳实践。



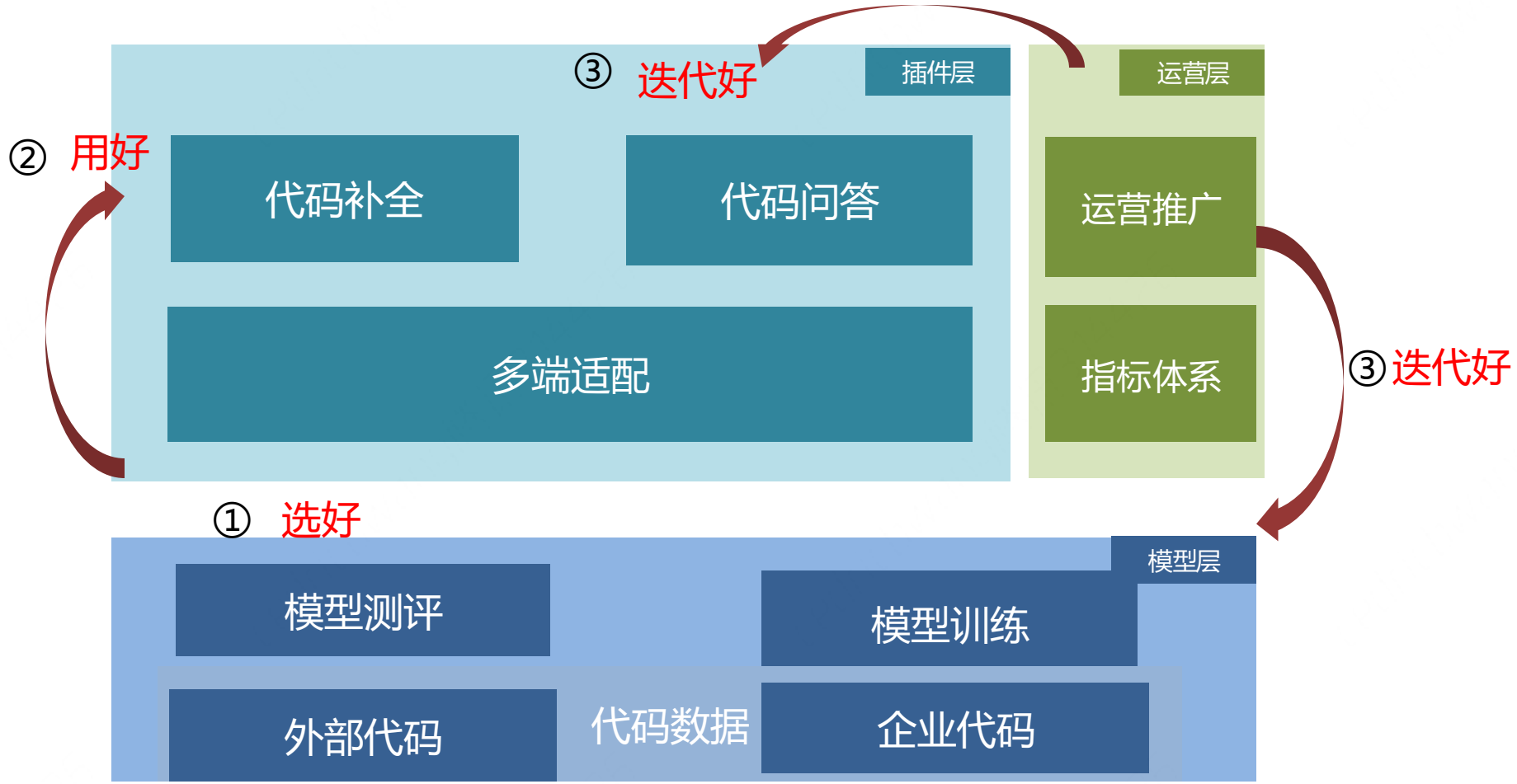
- IDE 插件太多，各种版本叠加。某股份行自研人力难以覆盖全面的插件建设

- 自研 IDE 插件产品体验不佳，难以推广使用
- 无法形成 产品 -> 推广 -> 反馈 -> 优化 的迭代模式

- 模型训练需要依赖大量数据，并需要不断在实践中积累拼接方式、比例等经验，模型训练效果难以保证
- 模型训练失败率较高，需要依赖训练经验对训练参数进行合理调整

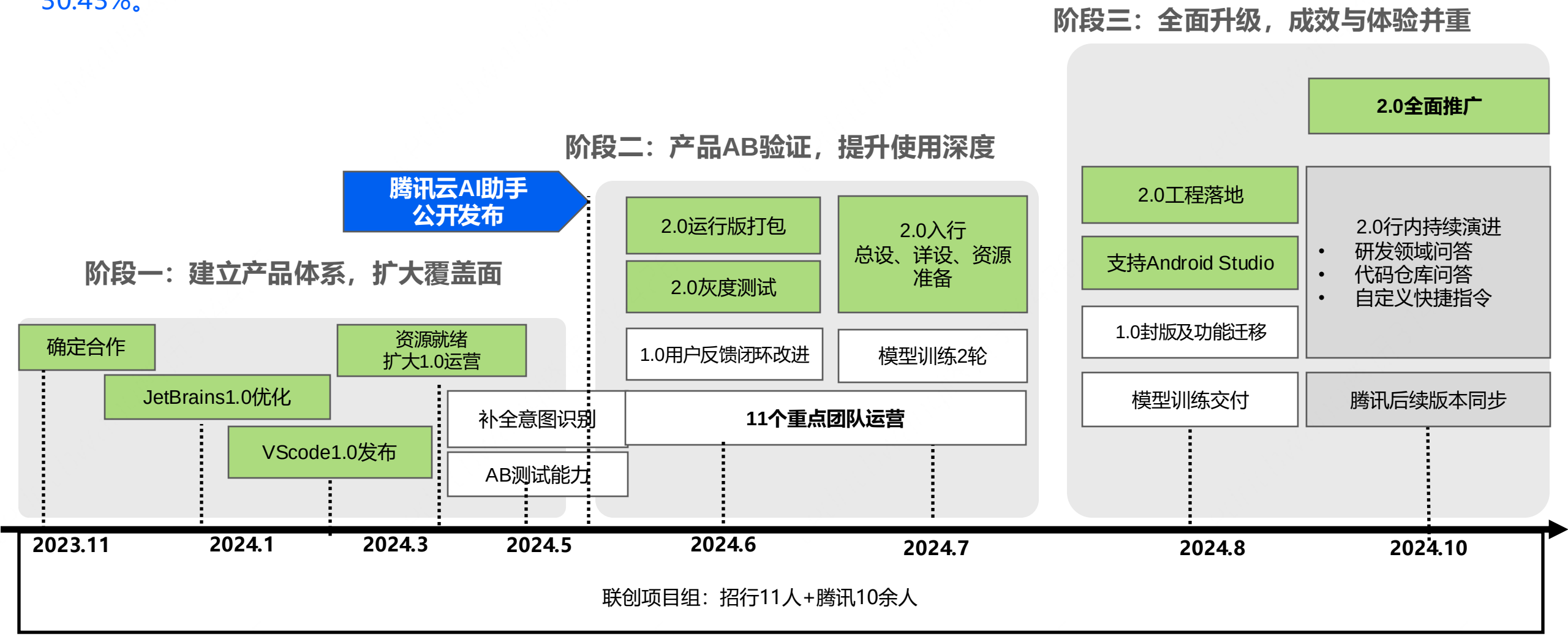
项目解决方案

- 通过三层清晰的结构，逐渐形成模型选好、插件用好、运营迭代好的产品发展闭环。
- 模型是辅助编码产品的基石，决定产品的上限，合作重点为行内数据的扩展
- 插件是用户交互的入口，体验决定产品的接受度，合作重点为行内功能的扩展
- 运营是产品演进的关键枢纽，相对非大模型产品，通过运营数据及badcase提升模型能力是产品持续演进的关键



项目建设过程

联创项目组发布 AI 编码助手 两个大版本共12次迭代，覆盖行内主要 IDE（JetBrains、VSCode、Android Studio。目前“腾讯版本 AI 编码助手”处于全面推广阶段，覆盖用户数超 6000+人，活跃率超 70%，代码采纳率达28.12%，AI生成字符占比达30.43%。



项目建设效果

1 沉淀 AISE 建设能力

以腾讯插件为基础，结合某股份行实际业务需求，沉淀了 Agent 智能体、知识库扩展、模型调度、度量指标等能力

2 具备模型持续训练能力

完成训练了10+轮模型训练/微调，结合内部4W+仓库积累了 2**60G+ 优质语料**，并借助腾讯的经验（数据配比、数据拼接等），将训练成功率提升到 60% （原30%）

3 插件融合行内场景

结合腾讯最佳实践与行内用户编码习惯，总结 31 种补全策略，智能识别补全意图，精准的补全时机，将代码采纳率提升 8%（20% -> 28%）

4 运营推广落地

持续优化产品体验，并基于腾讯内部推广经验、研效度量指标体系赋能招行。**覆盖用户数超6000人**，活跃率超70%，代码行采纳率达28.12%，代码生成率达30.43%，且用户认可，满意度高

小米集团

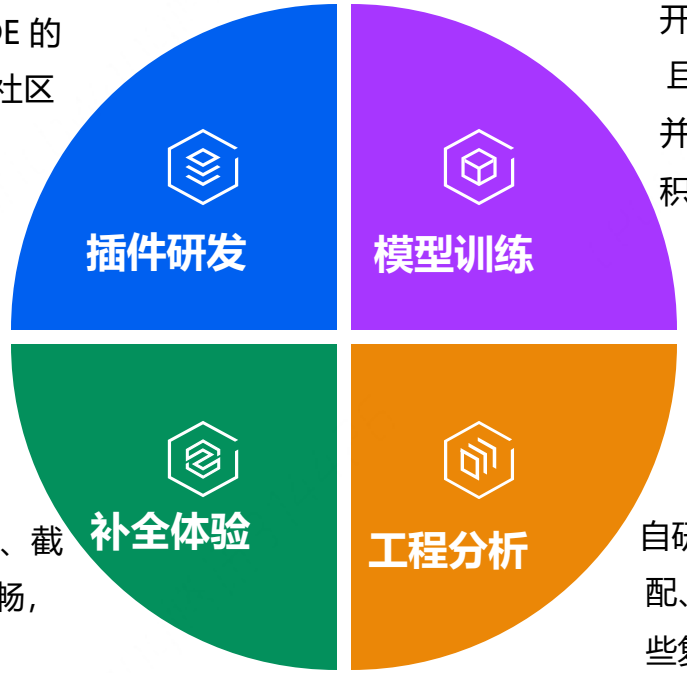
项目背景

大型模型技术为研发流程注入了新活力，小米一直在追求更高效的研发效能和更优质的研发成果，因此始终密切关注AI技术在研发领域的最新进展。随着 AI 辅助编码工具在全球范围内获得越来越多开发者的认可，小米于2023年开始着手探索AI辅助编码的相关技术和实践。在进行了一段时间的自研建设后，小米发现项目进展并未达到预期效果。

因此，小米决定引入更为成熟的商业化解决方案。经过一番严格细致的技术评估，小米最终选择与腾讯云进行合作，以期借助其在 AI 领域的专业能力和丰富经验，推动小米在 AI 辅助编码领域的进一步发展。

自研 AI 编码工具的挑战

人力投入较大，建设效果慢，各类 IDE 的兼容，市场上插件功能的追赶，开源社区活跃度低都成为了插件研发的挑战



开源模型微调困难，结果不收敛，并且存在大量的 bad case 无法解决。并且训练语料、测评方法需要大量的积累，模型训练之后的验证与数据分析，也有较高的门槛

自研 IDE 插件体验较差，补全响应慢、截断不准确。补全内容的后置处理不顺畅，无法和上下文结合，影响使用

自研的 IDE 插件，在跨文件检索、匹配、推理方面存在技术瓶颈，对于一些复杂的业务代码，补全效果不理想

建设符合小米场景的“AI 辅助编程平台”

腾讯 AI 代码助手提供产品基础能力（含代码大模型、各 IDE 插件、插件服务后台）及开放式架构，客户可结合实际需求，建设具备自身特色的“AI 辅助编程平台”

项目建设效果

截止2024年9月底，小米 DAU > 2300

代码补全生成率 >35%

计算方式为：AI 生成的字符数/所有新增代码字符。主要为了评估 AI 编码带来的价值/作用

代码补全接受率率 > 25%

计算方式为：接受的次数/ai推荐的次数。主要为了度量 AI 的准确度。同时还有另一个配套的度量数据（**补全接受后有效率：接受的代码最终保留的比例**），同步评估代码生成的准确度

人均日补全次数 > 100

深度用户人均日补全次数大于100次，每日生成代码行数大于 100 行

代码有效率 > 75%

代码补全接受后的有效率超过 75%，即大部分接受的代码都最终保留了下来，为业务提供价值。

目 录

CONTENTS

腾讯在 AISE 场景的布局

腾讯在AI4SE各个阶段均有布局，重点发力在开发和测试领域，其它领域在做积极探索

产品设计

- 需求
  - 用户调研
  - 产品创新
  - 产品定义
  - 功能拆解
- 规划
  - 验收条件
  - 优先级
  - 迭代计划
  - 发布计划
  - 里程碑计划
- 设计
  - 原型设计
  - 交互设计
  - 视觉设计

架构设计

- 领域建模
  - 领域分析
  - 事件建模
  - 架构分层与上下文定义
- 架构设计
  - 时序图设计
  - 组件图设计
  - 类图设计
  - 设计文档
- API设计
  - API 文档

开发

- 详细设计
  - 代码设计
  - 代码相关问答
- 编码
  - 代码生成
  - 单元测试生成
- 评审
  - 代码评审
  - 提交信息
- 安全
  - 代码安全扫描
  - 解决修复安全问题

测试

- 缺陷分析
  - 根因分析
  - 缺陷分析
  - 缺陷修复
- 自动化测试
  - 自动化测试代码生成
- 测试用例设计
  - 测试用例生成

交付

- 发布
  - 发布文档
  - 更新文档
- 部署
  - 部署脚本生成

运维

- 监控
  - 日志分析
  - 客户支持
  - 客户情绪分析

感谢倾听