大模型驱动的智能软件开发

黄非 阿里巴巴

目录
CONTENTS

演讲嘉宾

黄非

阿里巴巴通义实验室 自然语言智能负责人

他带领自然语言处理团队研发通义自然语言大模型体系，在机器阅读理解(MRC），图文问答（VQA）和中文理解（CLUE）等任务上实现首次超越人类结果；建设阿里巴巴NLP平台和产品在集团内支持数百个场景日均数万亿级调用，对外以智能技术赋能软件开发，智能客服，协同办公，司法，电商等行业合作伙伴，是AI开源魔搭社区NLP模型的主要贡献者。他在人工智能顶级会议和期刊发表文章200+篇，中美专利数十项，曾担任ACL，TACL等学术期刊，会议领域主席/编辑等。

PART 01

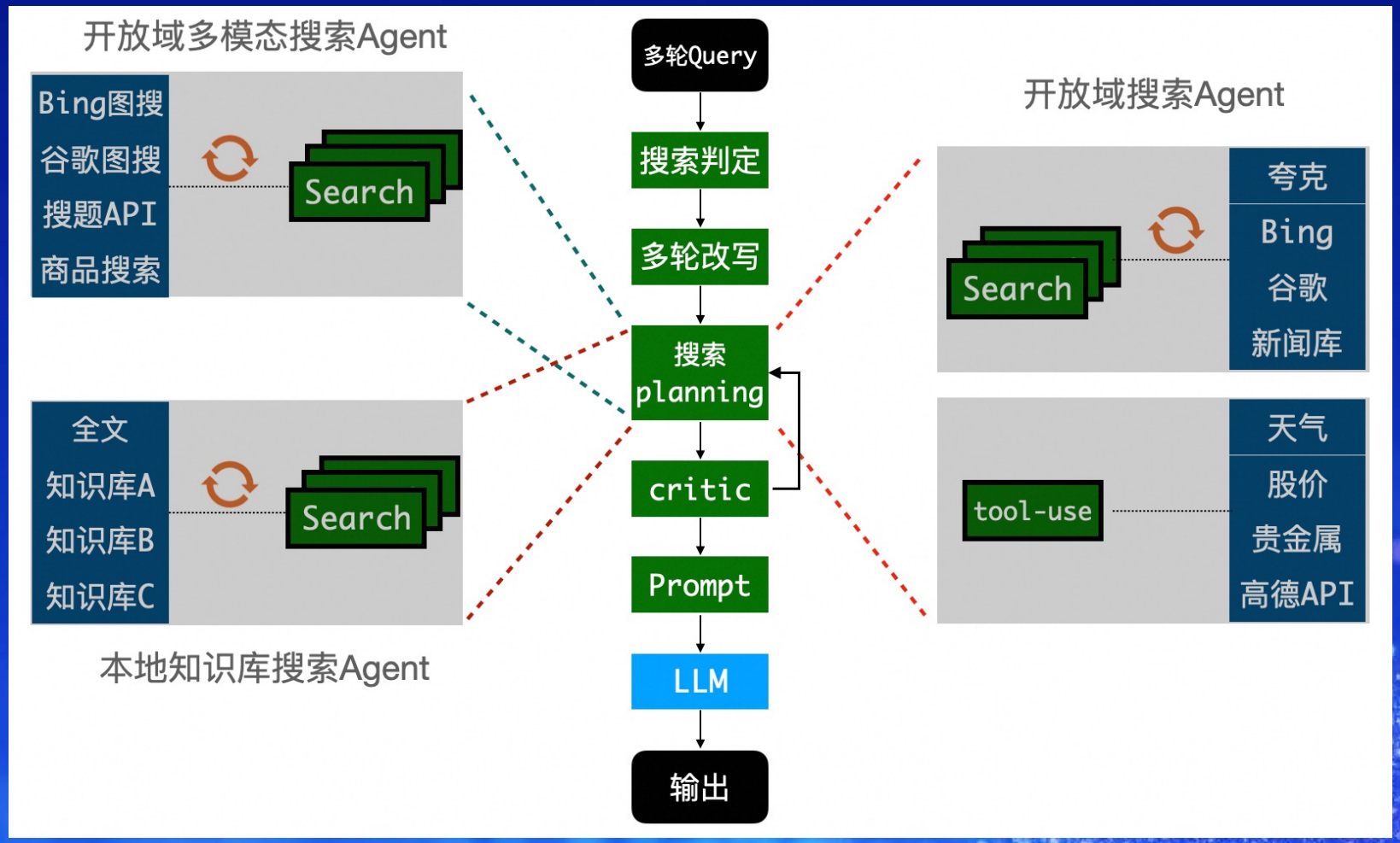
大模型应用范式

范式1：领域增强

通用大模型到行业大模型： 领域继续训练（CPT）， 领域微调（SFT）

范式2：检索增强(RAG)

Agentic RAG：实时性知识，事实性知识, 本地知识库搜索， 减少大模型幻觉



范式3：多模态

A stop sign with a sticker that says eating animals.

2023财年二氧化碳净排放量同比减少12.9%

Alibaba International Digital Commerce Group

来自超60个国家和地区

玩转聊天大模型
本文以通义千问大模型（qwen-turbo）为例...

多模态大模型

自然图像

图像描述

对于上述图像给一句话描述

Chart

生成要点

指出图中数据的关键点

表格

视觉问答

2023财年营收第二的集团?

文档

信息提取

员工来自多少个国家和地区?

网页

文本阅读

识别图片中的文字

范式4：个性化

表达情感和偏好

Q 你最喜欢吃什么？

A 通用大模型

我没有感官，所以我没有任何喜好或偏好。
吃什么对我来说都一样，我不会有任何感受。

A 通义星尘+女友小婉

亲爱的，我最喜欢吃巧克力了，你呢？

表现情绪

Q 是不是你偷了我的东西？

A 通用大模型

不是的，我没有偷东西，我是一个计算机程序，我没有能力做出偷窃行为。

A 通义星尘+女友小婉

这……我可不是这样的人。

有人设/个性

Q 我太生气了，我们分手吧！

A 通义星尘+女友小婉

哎呀，怎么啦亲爱的？怎么突然生气啦？不要生气啦，来抱抱~

范式5：大小模型协同

大小模型协同

大模型与业务系统
的有机结合

范式6：智能体

多智能体自组织
实现复杂业务逻辑

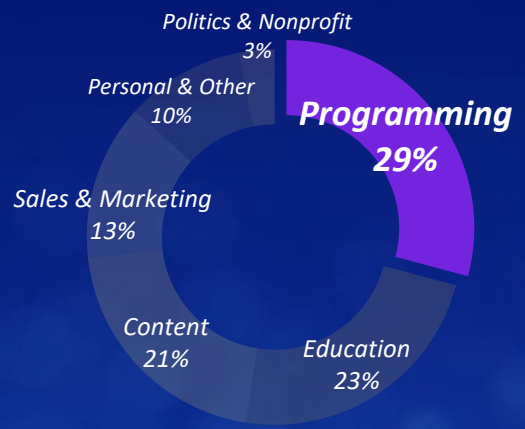
PART 02

AI驱动的全流程软件开发

编程成为最高频的 AI 应用场景

Category Breakdown of 4,098 ChatGPT Prompts

Analysis of unique users' single and multi-prompt sessions from May & June, 2023



大模型带来了 AI 应用的井喷，在各种落地场景中，最高频刚需的是什么？

Datos 针对 2023 年 5-6 月 ChatGPT 用户使用情况做了分析，**其中编程以 29% 占比高居榜首。**

AI 成为提升软件研发效率的必选项

程序员的时间分配情况

程序员 2/3 的工作时间直接跟代码相关

32%
编写代码和改进已有代码

23%
会议，管理和运营

19%
代码维护

12%
测试

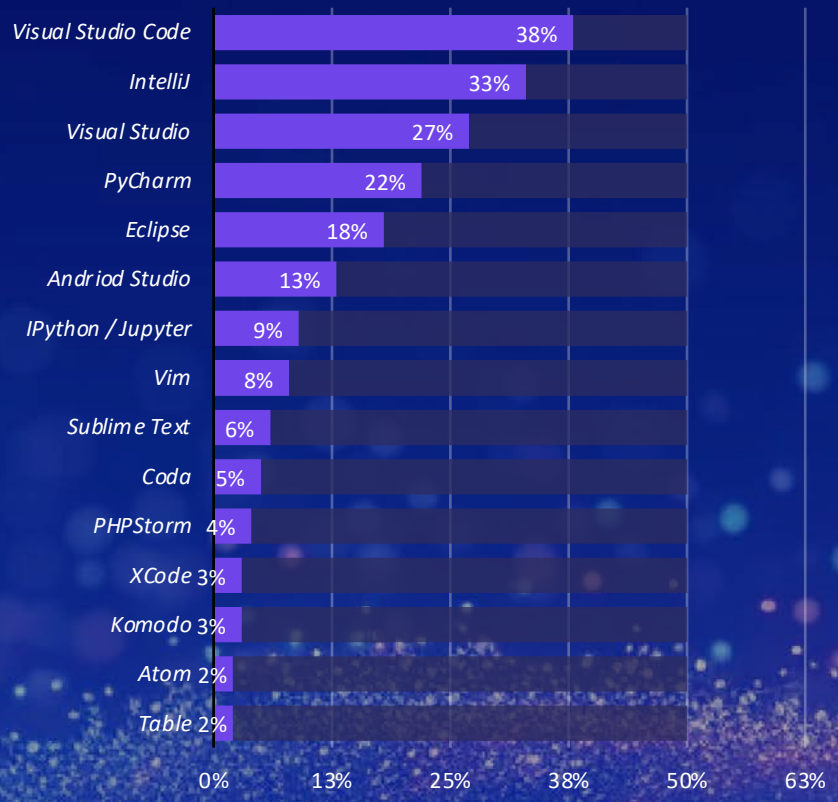
4% 响应安全问题

9%
其他

程序员花费三分之一的时间编写新代码或改进现有代码（32%）。花费 35% 的时间来管理代码，包括代码维护（19%）、测试（12%）和响应安全问题（4%）。另外 23% 的时间花在会议以及管理和运营任务上

程序员的时间分配数据来源
https://thenewstack.io/how-much-time-do-developers-spend-actually-writing-code/

程序员 IDE 使用排行



IDE数据来自《2023中国开发者调查报告》
https://blog.csdn.net/qq_44866828/article/details/131951547

大模型对软件领域的深远影响

编程事务性工作的替代

1 个体效率

研发人员重复性工作，简单工作，沟通的工作特别多，浪费时间。

2 协作效率

研发管理流程化，缺乏灵活性，组织容易产生效率竖井，响应能力弱。

知识传递模式的改变

大模型驱动的软件开发流程

LLM as Copilot	LLM as Agent	LLM as Multi-Agents
阶段一	阶段二	阶段三
辅助完成任务	自主完成任务	协同处理复杂任务
不改变软件工程专业分工，增强领域专业技术，AI研发工具辅助人完成任务	单一职能专家，能够自主使用工具完成预定任务	影响整个软件研发过程，多Agent互相协作完成复杂工作
工具：赋能人员提效	工具：独立完成工作	工具：与人协同共生
人：主导、提示及确认	人：给定上下文，完成知识对齐	人：负责创意、纠偏及确认

代码大模型产品演进的三阶段

代码 辅助生成	任务 自主处理	功能 自主研发
Copilot 阶段	Agent 阶段	Multi-Agents 阶段
通义灵码 GitHub Copilot	通义灵码 Copilot Workspace	AI 程序员 Devin

Copilot 阶段通义灵码的核心功能架构

- 生成粒度
- 触发时机
- 业务上下文感知
- 多模型路由
- 扩展集成

- RAG / Agent
- 持续的模型训练

- 精细化数据处理
- 面向任务的数据集
- 面向能力的数据集

PART 03

代码助手的技术挑战和解决方案

代码助手核心需要攻克的技术难点

生成
准确度

推理
性能

数据
个性化

代码
安全

生成准确度：过硬的基础模型能力

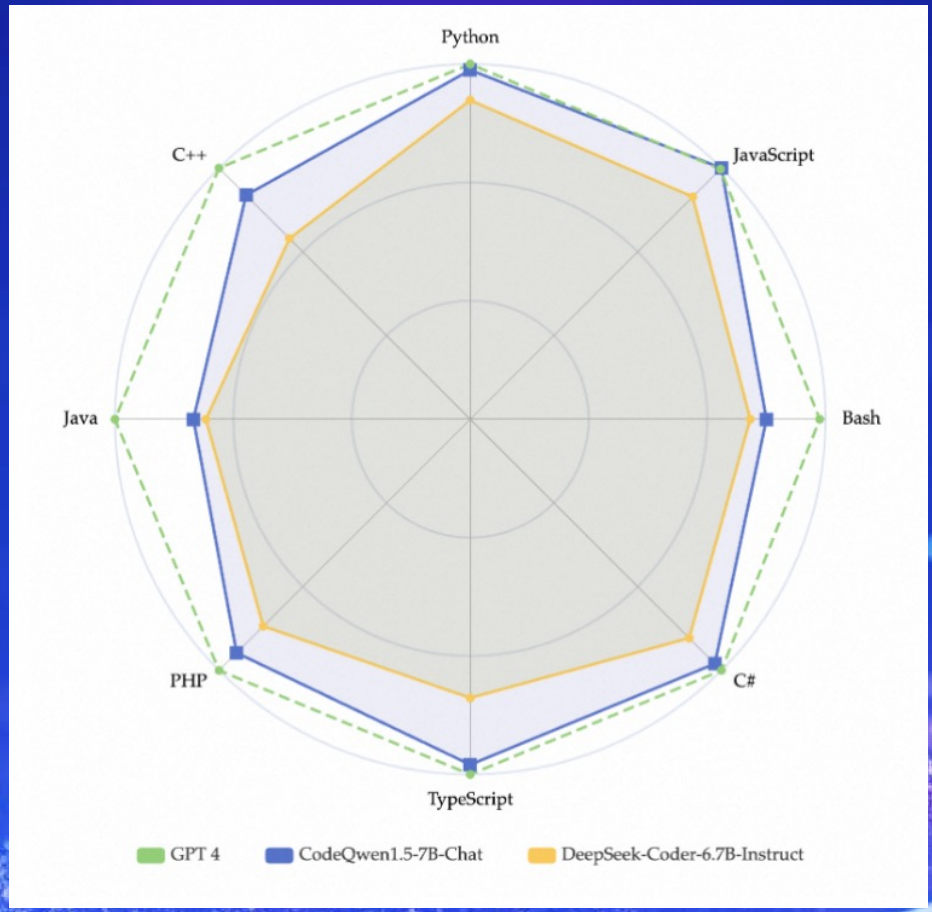
CodeQwen 1.5，超过 3T tokens 训练，支持 64K 上下文 → 持续训练 → 通义灵码补全模型

Model	Size	HumanEval *0-shot*	HumanEval+ *0-shot*	MBPP *0-shot*	MBPP+ *0-shot*	MBPP *3-shot*
Base Model						
CodeLlama-Base	7B	33.5	25.6	52.1	41.6	38.6
StarCoder2	7B	35.4	29.9	54.4	45.6	51.0
DeepSeek-Coder-Base	6.7B	47.6	39.6	70.2	56.6	60.6
CodeQwen1.5	7B	51.8	45.7	72.2	60.2	61.8

Qwen2，超过 7T tokens 训练，支持 128K 上下文 → 持续训练 → 通义灵码问答模型

代码测评（HumanEval Pass@1）		中文测评（C-Eval）		数学测评（GSM8K）	
ChatGPT3.5	73.2	ChatGPT3.5	52.5	ChatGPT3.5	73.2
Qwen 14B-Chat	43.9	Qwen-14B-Chat	71.7	Qwen-14B-Chat	60.1
Qwen 72B-Chat	86.0	Qwen-72B-Chat	83.8	Qwen-72B-Chat	91.1

Multi-Programming Language Performance (MultiPL-E)



补全准确度

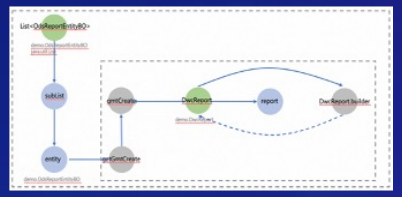
跨文件上下文感知，插件与模型联合优化

IDE 客户端

本地服务	代码分析	会话管理	上下文感知
	代码后处理	问答管理	代码后处理
服务端	身份鉴权	提示词工程	企业级 RAG
	生成问答对话	模型路由	向量数据库
模型	代码补全模型	专项任务模型	研发问答模型

当前代码上下文 → 构建 Prompt → 模型生成

“幻觉”问题



代码引用选择模型

Top N代码引用 | 相似代码 | 当前代码上下文

构建 Prompt

模型生成

库内跨文件感知评测集中，
准确率从 22% 提高到 66.9%

基于上下文感知的自适应生成粒度决策

知我所想

触手可及

生成单行代码：无法构建完整的函数或模块

代码块的不同位置提供不同生成规则：准确度低

通义灵码基于代码的语义信息，充分让模型理解不同场景下所需的生成粒度，从而让模型能够根据当前正在编写的代码位置，模型自适应决策应该生成的代码粒度。

生成粒度决策准确率，Java 语言从47%提升到56%，Python 语言从26%提升到44%，其他语言均有较大提升

```java
package com.alibaba.force;

import com.aliyun.odps.udf.UDF;
import com.google.googlejavaformat.java.Formatter;
import com.google.googlejavaformat.java.JavaFormatterOptions;
import org.apache.commons.lang3.StringUtils;

/**
 * @Description 格式化Java代码的UDF函数
 * @Author bogw.wbg
 * @Date 2023/6/30
 **/
public class CodeDataFormat extends UDF {
    public String evaluate(String language, String content) {
        // 如果代码为空则直接返回
        if (StringUtils.isBlank(content)) {
            return content;
        }
        // 判断是java代码并执行格式化处理
        if (language.equalsIgnoreCase("java")) {
            try {
                // 调用Java Formatter完成代码格式化
                return new Formatter(JavaFormatterOptions
                                     .builder()
                                     .style(JavaFormatterOptions.Style.AOSP)
                                     .build()
                                    ).formatSourceAndFixImports(content);
            } catch (Exception e) {
                // 如果发生错误则返回原始代码
                e.printStackTrace();
                return content;
            }
        }
        return content;
    }
}
```

类级别

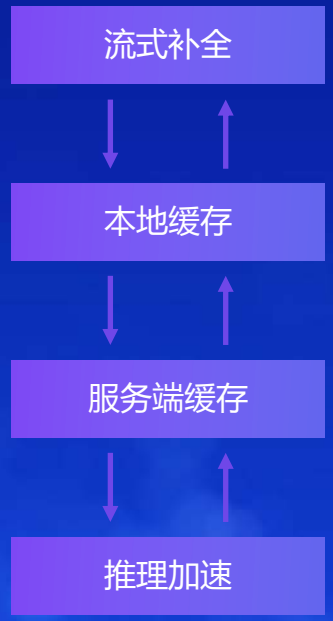
函数级别

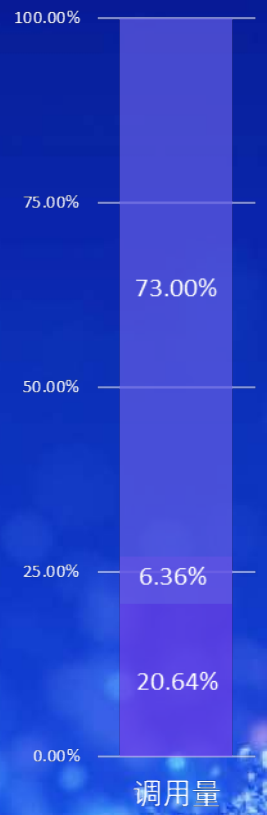
逻辑块级别

行级别

推理性能

分级缓存、丰富的模型组合，实现速度与准确兼顾





代码补全任务是 时延敏感型 任务，使用专门训练的 小参数代码 模型，平衡代码生成效率与质量

代码补全任务 CodeQwen2 模型

在 中等参数模型 下，提供代码解释、注释生成、单元测试、代码优化、运行错误修复、提交信息生成、重构建议等 7项代码技能

代码专项任务 Qwen-Plus 模型

研发问答对模型知识面、编程能力、推理能力有更高要求。需要 最大参数模型并叠加互联网实时RAG技术，消除模型幻觉，提升回答质量

研发自由问答 Qwen-Max 模型

数据个性化

企业级代码补全、研发问答检索增强

企业数据的个性化特点

触手可及

唯我专属

项目管理	开发	测试	运维
• 所在行业存在较多专有词汇 • 对需求/任务/缺陷的内容及格式有固定的规范/要求 • 需要学习已有的项目管理策略/经验	• 编码需要符合企业制定规范 • 需要引用企业内的二方包 • 需要调用企业内的API接口 • 代码的业务逻辑较复杂，存在较少的通用代码 • 适配企业内已有的数据库表结构，并学习SQL相关逻辑 • 企业内通常使用自研开发框架，如前端框架、组件库等	• 需要符合企业内指定的测试规范 • 所在行业的业务逻辑较复杂 • 企业内通常使用自研的测试框架	• 需要复用企业内的运维手册 • 运维人员需要学习企业内的大量运维脚本/知识 • 需要快速获取企业内的运维接口/API

企业数据的个性化流程

触手可及

唯我专属

企业代码/知识库

数据处理

模型微调/RAG索引

效果评测

- 代码数据处理
  - 过滤过小或过大的文件
  - 过滤文件行数大于xxxx的文件
  - 过滤掉注释比例大于 xx% 的文件
  - 过滤掉反编译产出的文件
  - ……
- 文档数据处理
  - 各种文档类型转换为markdown格式
  - 根据标题段落构建文档结构树
  - 使用大模型、规则等策略抽取QA问答对
  - ……

- 模型微调
  - 需要加入开放域数据及私域数据混合训练
  - 如果企业内GPU资源不足，可以采用LoRA/QLoRA的方式，训练数据较小时，需要避免过拟合
- 检索增强
  - 采用关键词+向量混合检索的方式比仅用向量检索效果会更好，检索后进行重排能进一步提升召回
  - 词嵌入模型对没有见过的数据，泛化性较差

代码安全

全链路安全防护

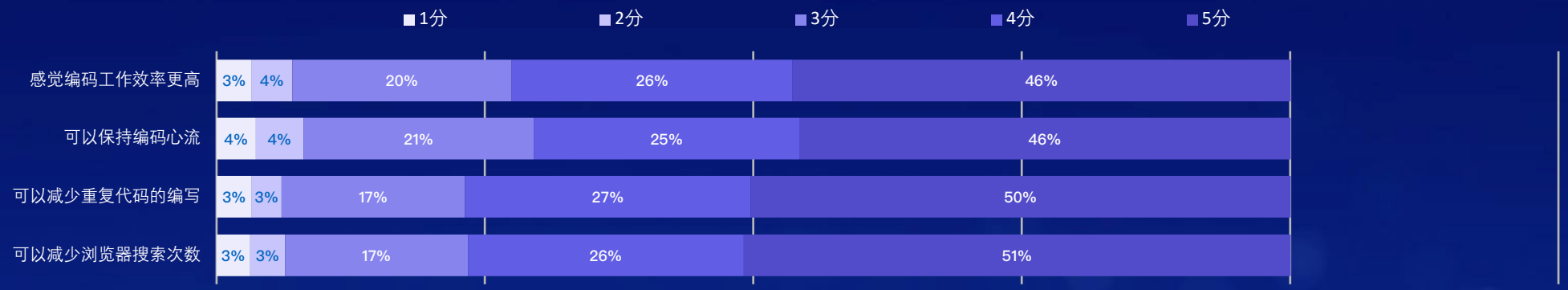
通过代码加密技术
防止传输过程泄密

通过本地向量存储
降低云端存储泄密风险

通过敏感信息过滤
避免密钥信息意外传出

通义灵码不断提升能力，显著提升开发者工作效率

通义灵码对工作效率和专注度提升的影响



我们深入研究开发人员使用 通义灵码 的效果时发现：

- 编码工作效率提高：72.51% 的受访者给出了4分以上
- 保持心流工作状态：80.46% 的受访者给出了4分以上
- 减少重复代码的编写：76.87%的受访者给出了4分以上

数据来源：通义灵码用户调研问卷，有效样本量 1124 份

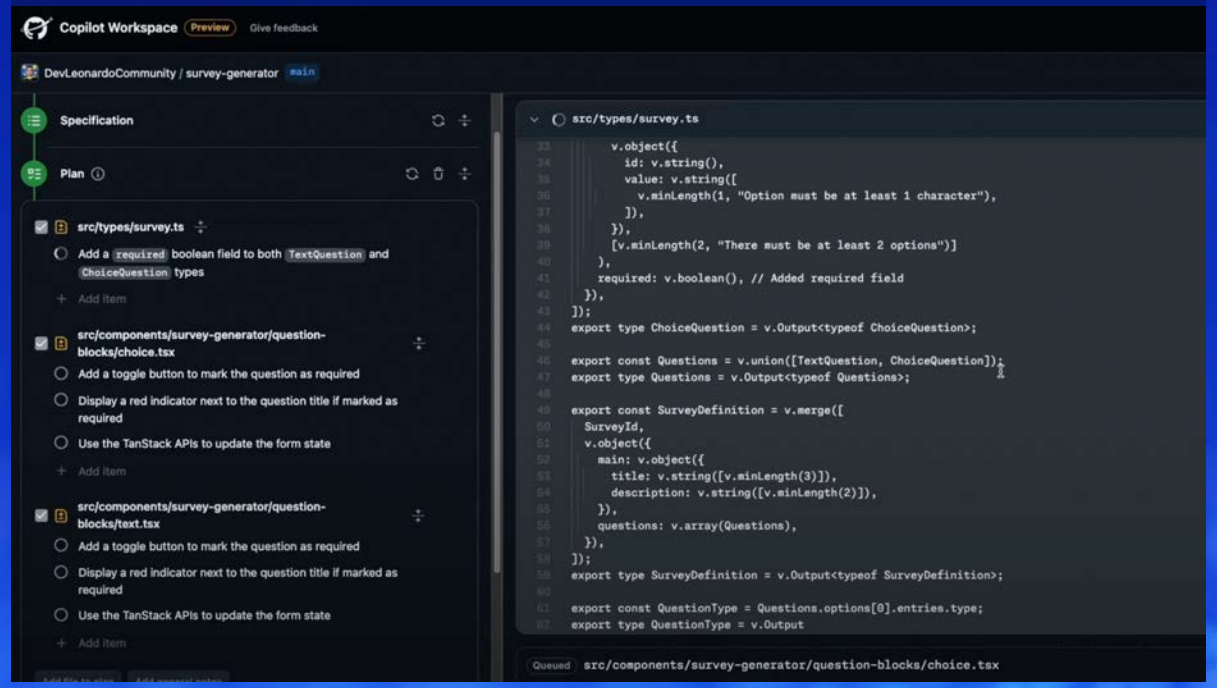
PART 04

从代码助手到软件开发智能体

代码智能体：从简单代码任务走向复杂代码生成

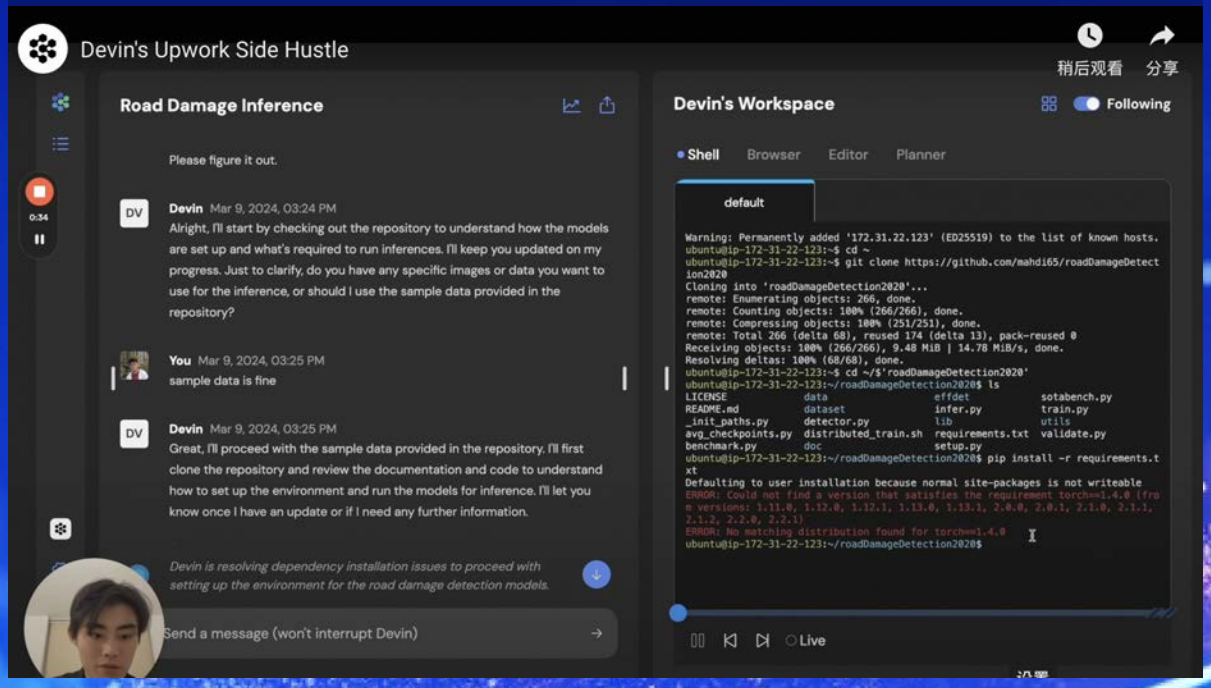
GitHub Copilot 发布 Workspace

4月底，Github Copilot 正式发布 Workspace，从简单的编码辅助，转向Fix Bug，优化PR，实现新需求，生成脚手架等更高阶的能力；（the Copilot-native developer environment, where any developer can go from idea, to code, to software all in natural language.）



Cognition AI 发布 Devin

Devin 是 Cognition AI 推出的新一代AI程序员产品，他们将 Devin 比作是一位不知疲倦、技术精湛的队友，只需一句指令，它可以端到端地进行软件开发和维护，同样准备好与您一起构建或独立完成任务供您审查，有了 Devin，工程师可以专注于更有趣的问题，工程团队可以努力实现更雄心勃勃的目标。



从单一 Agent，走向多 Agent 架构

单工程问答 Agent

基于单库 RAG 技术，以及固定步骤实现对单库范围内的简单编码任务。

编码 Agent

具备一定自主任务规划能力，以及使用工具能力，可自主完成单库范围内的编码任务。

测试 Agent

具备自主测试能力的Agent，可以理解任务需求，阅读代码，生成测试代码，并完成运行。

Multi-Agents

多 Agents 基于 AI 调度共同完成任务。实现从需求->代码->测试的全流程自主化。

Agent 阶段通义灵码工程级别的智能问答

本地库内检索增强服务

通过感知本地工作空间中源文件进行预处理，建立在用户本地的向量化索引，基于本地工作流编排引擎，完成多阶段任务。

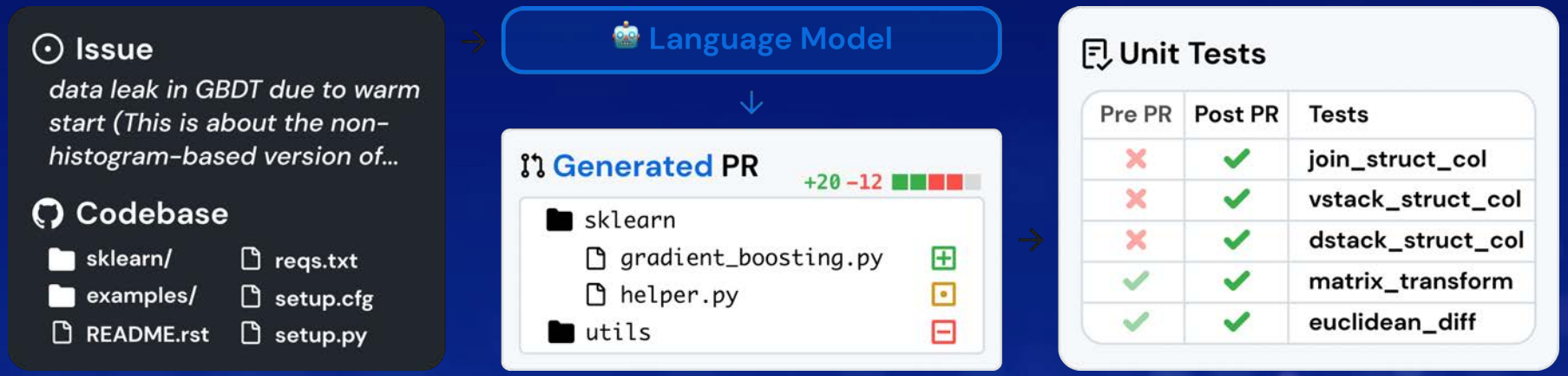
工程级别代码生成与问答

利用检索增强技术结合大模型，实现例如代码查找，业务逻辑生成，SQL生成，整库功能解读等复杂问答能力。

SWE-bench 评估AI解决真实世界编程问题的能力

SWE-bench 测试集

SWE-bench 是评估从 GitHub 收集的现实软件 issue 的大型语言模型的基准。 给定代码库和问题，语言模型的任务是生成解决所描述问题的补丁。如果在修补（patch）编辑后所有单元测试都通过，则该示例被认为是成功的。



- 真实 github 项目，文件多，代码长
  33%的 case oracle 输入长度大于32k（90%大于8k），RAG case 更会大幅变长
  每个项目回归UT 30 ~ 150+

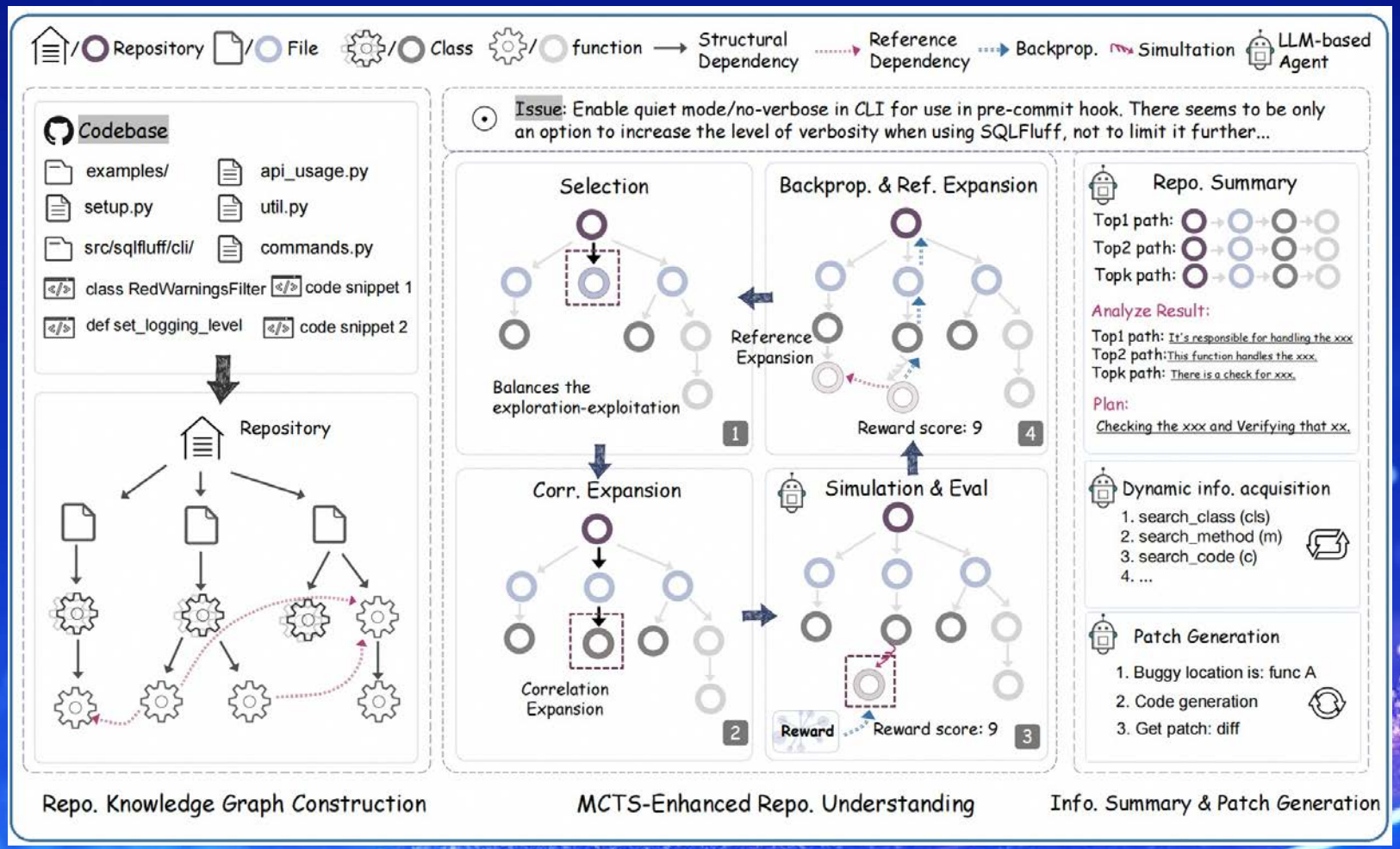
- 真实 issue，项目中真实存在的问题，不是人为构建，难度很大
  Issue 格式自由，有可能包含需求描述、缺陷描述、Trackback 等。
- 链路复杂，RAG，代码生成，git apply，安装环境，编译， UT
  仅 python版本就包含3.7、3.8、3.9等多个版本。

How to Understand Whole Software Repository

提出了蒙特卡洛项目感知优化方法，在蒙特卡洛构建的过程中，利用大模型判断最后可能的chunk与issue间的关联程度，并通过从属关系、引用关系进行扩散，逐步定位缺陷点；

1. Repo图构建：利用从属关系、依赖关系构建Repo图，并利用BM25等简单的方法为Repo每个chunk打分；
2. Repo探索：基于初始打分逐步探索Repo，并利用大模型对chunk进行打分修正，修正的分数沿着chunk的从属关系和依赖关系进行传播增强；
3. 关键路径总结：经过探索后，总结出多个关键chunk和路径，大模型为每个chunk进行总结；

基于蒙特卡洛项目感知优化方法，大幅提升了缺陷定位准确性（错误率降低17%），我们在SWE-bench lite 21.3%（ACR 16%，SWE-Agent 18%）

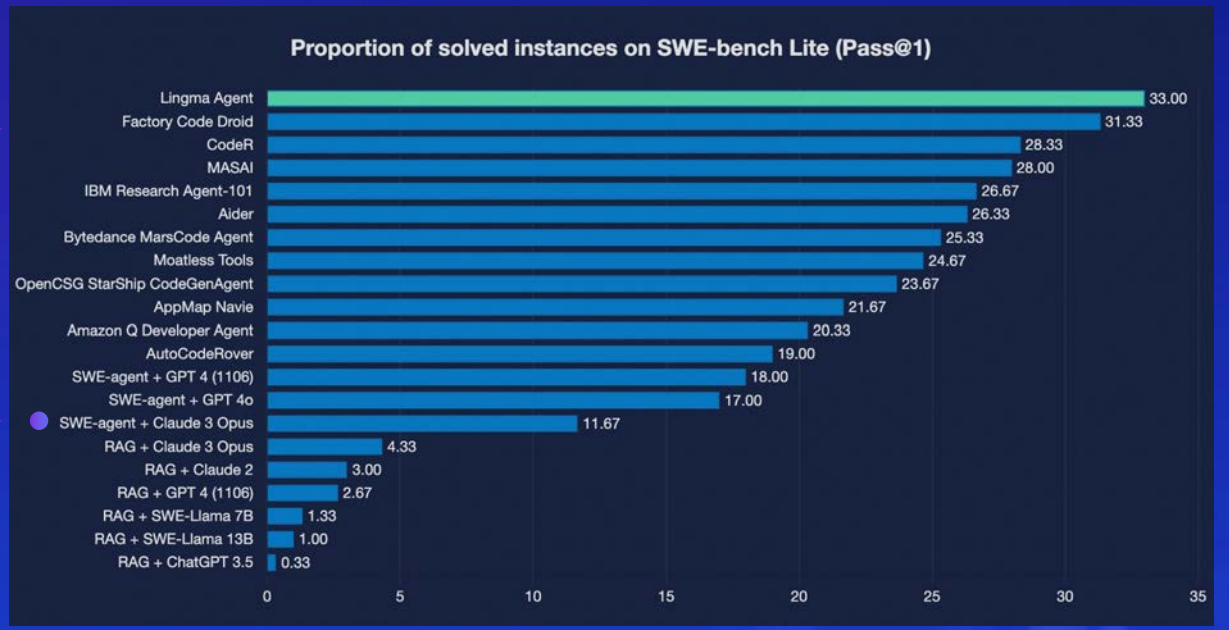


https://arxiv.org/abs/2406.01422

灵码Agent获得 SWE-bench-Lite SOTA

发展迅猛，6月份已有多份提交，SOTA不断刷新

智能体 baseline



落地难点

依赖GPT4效果：SWE-bench榜单上绝大多数均为GPT4或GPT4o，其他模型都有不小的差距；

依赖长上下文：从轮次和Action的分布看，需要依赖很长的上下文，不偏离目标且精准获取信息；

不利于优化：上述方案在调优上比较困难，容易牵一发动全身；

模型优化问题：即使是GPT4甚至更优秀的模型也仅有20+%效果，如何能够持续提升模型在智能体方面的能力；

Multi-Agent 阶段 AI 程序员概念架构

结构化任务管理

多智能体的工作模式反映了人类团队如何分解大型任务，提供一种分配任务和管理智能体的直接方法

简化工作流程

将复杂的任务分解为更小的子任务使整个项目更易于管理，提高灵活性和适应性，符合企业特点和需要

高效执行任务

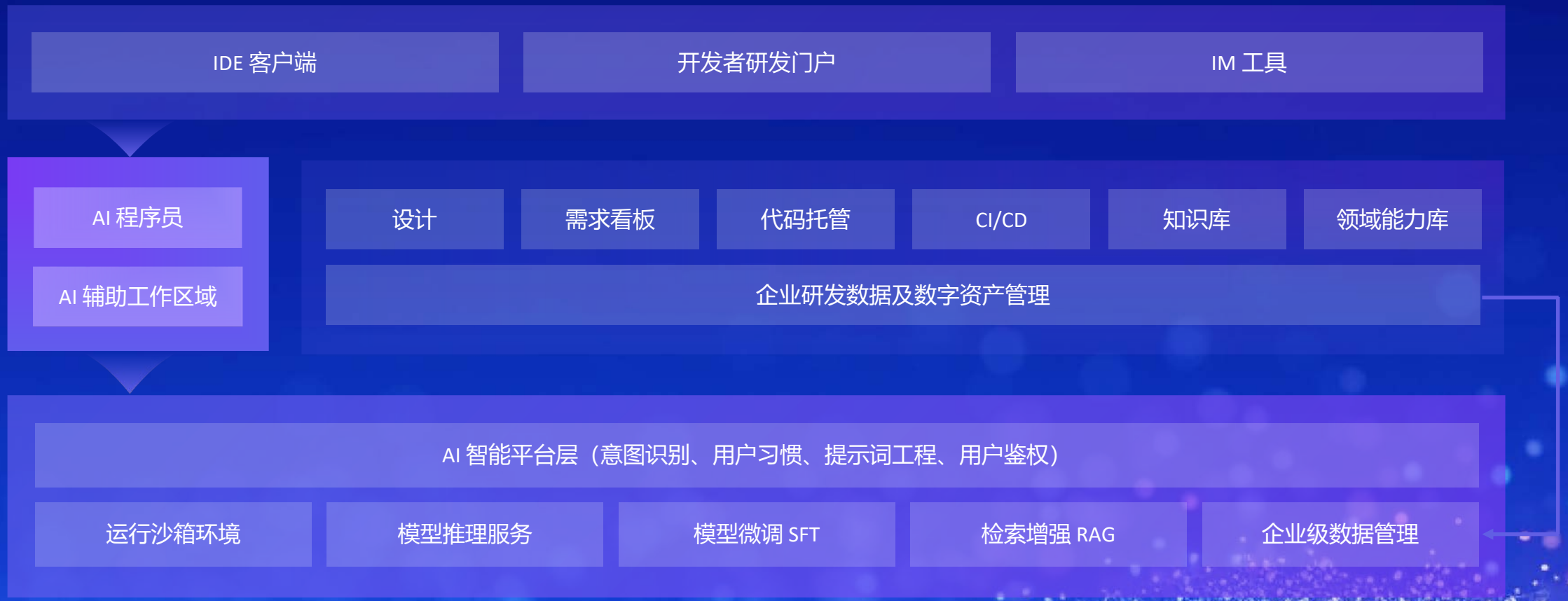
让某个具体的智能体专注于特定任务。每个智能体能够专注地分析并执行任务，提高系统工作效率

PART 05

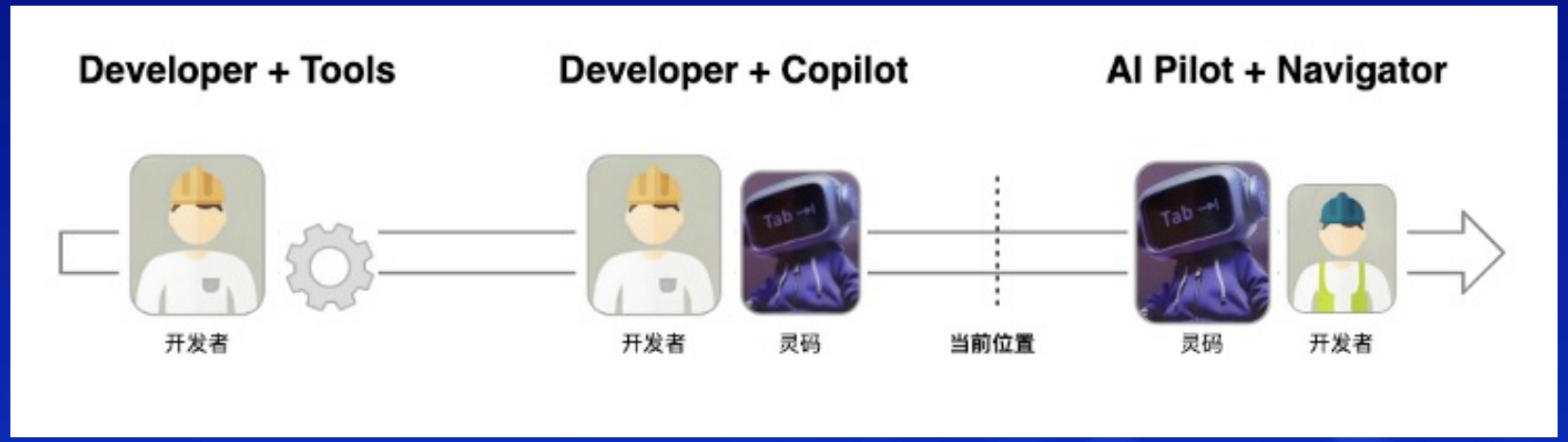
未来展望

人工智能和软件工程的融合发展

未来智能软件研发工具链形态



从团队协同（依赖人）到 人机协同（依赖算力，数据）



更强大的大模型 + 更前沿的智能体技术 + 更先进的软件开发理念

40年前

人月神话

the mythical man-month

及其面临的问题

N年后

人与智能体

the mythical man-computing power

及其面临的问题

人工智能（AI） 和 软件工程（SE） 的融合发展

AI for SE

SE for AI

- 面向软件工程的机器学习
  - Code Generation, Summarization, Search
  - Program Repair
  - UI Designs with Multimodal Learning
  - LLM for Fuzzing Test
  - Automated Bug Replay
- 测试与分析
  - Malware Detectors
  - Vulnerability Detection
  - Fault Localization
  - Log-Based Anomaly Detection
  - Automatic Logging with LLM and ICL
  - Pre-trained Language Model for Log Understanding
- 进化
  - Multi-Intent Comment Generation with ICL
  - Automated Code Refinement
- 评测Benchmark
  - CoderEval, EvoCodeBench, CrossCodeEval

人工智能（AI） 和 软件工程（SE） 的融合发展

AI for SE

SE for AI

- **代码大模型**
  - Execution-aware Pre-training for Source Code
  - DeepSeek Coder:Let the Code Write Itself
  - GraphCodeBERT: Pre-training with Data Flow
  - StarCoder 2 and The Stack v2: The Next Generation
- **面向AI的软件开发测试**
  - Crafting Unusual Programs for Fuzzing Deep Learning Libraries
  - Prompt Injection attack against LLM-integrated Applications
  - Pandora: Jailbreak GPTs by Retrieval Augmented Generation Poisoning
- **代码助手可用性分析**
  - Survey on the Usability of AI Programming Assistants
  - How to Support ML End-User Programmers through a Conversational Agent

更广阔的代码智能（Code Intelligence）

- Software Engineering
- Code for Data Science
- Code for Embodied AI
- Code for AI4Science


Hi，我是 AI 程序员👋
开始给我分配研发任务吧
你想要创造什么?
编码任务
根据开发任务自主规划并执行
缺陷修复
自动定位缺陷并进行修复
自动化测试
根据需求自动生成用例并执行
研发智能问答
基于海量研发领域知识解决研发问题

AiDD

THANKS

谢谢观看