AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

AI驱动软件研发
全面进入数字化时代

中国·深圳 11.24-25

AI+
software
Development
Digital
summit

程序员正在面临前所未有的机会：10X开发者的探寻与实践

马宇驰 华为云

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 深圳站

K+ 全球软件研发行业创新峰会

会议时间：2024.05.24-25

K+峰会 上海站

K+ 全球软件研发行业创新峰会

会议时间：2024.09.20-21

AiDD 峰会

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2023.11.24-25

AiDD峰会 北京站

AI+ 软件研发数字峰会

会议时间：2024.07.19-20

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2024.11.15-16

演讲嘉宾

马宇驰

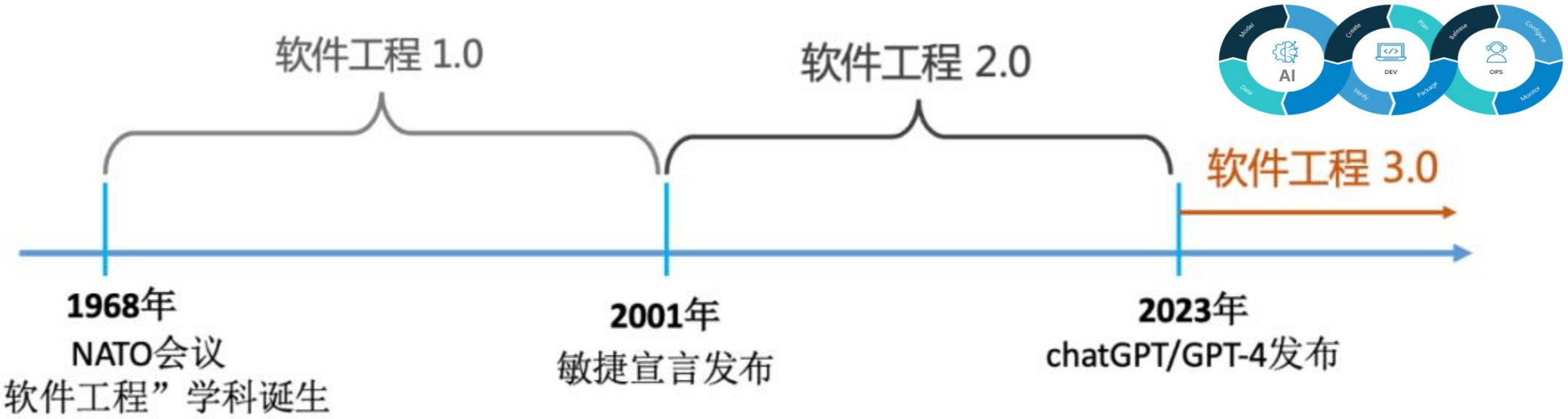
AI算法科学家 / 代码大模型高级技术专家

- 博士，华为云PaaS技术创新Lab - DevAI Lab 负责人、CodeArts Snap智能编程助手 负责人、智能化研发技术专家、代码大模型高级技术专家
- 应届加入华为，历任AI算法科学家、智能化测试技术专家、研发智能博士军团Leader 等岗位
- 17年加入华为工作至今，带领团队先后围绕智能化测试、智能化运维、智能路由与调度、智能化代码生成等方向成功孵化多项智能研发服务，完成规模化落地并外溢

目录
CONTENTS

PART 01
AIGC for SE的业界洞察

软件工程3.0：智能化软件工具生产线趋势



软件危机：进度风险、质量问题

- 大型工程实践
- CMM：过程管控，过程决定结果
- 项目化管理：角色分工、职责明确

软件爆炸：无法适应快速变化

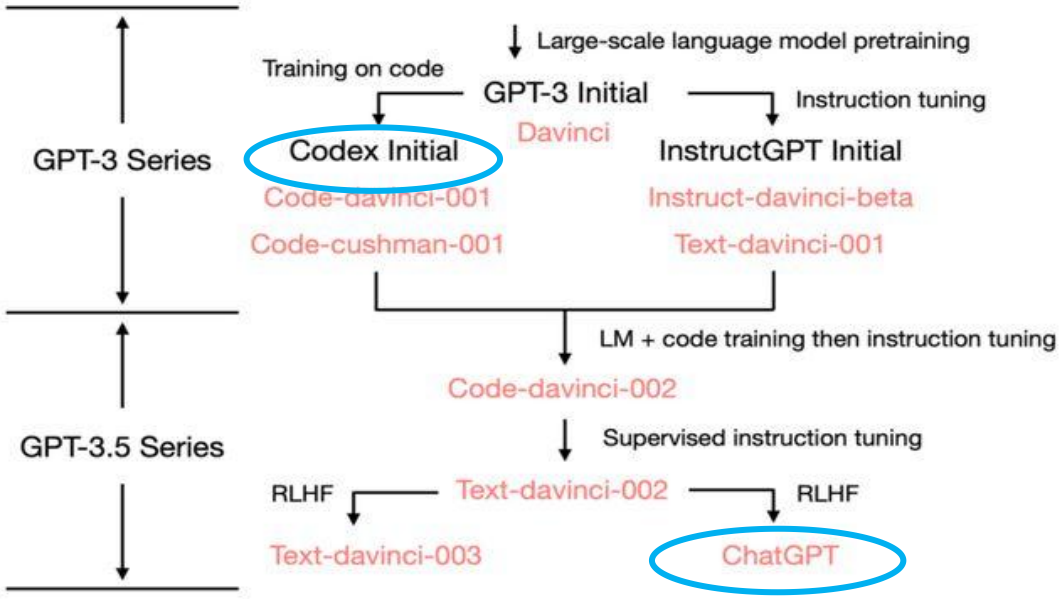
- 敏捷：强调结果、个体、拥抱变化
- CICD：持续价值交付
- DevOps：团队协作

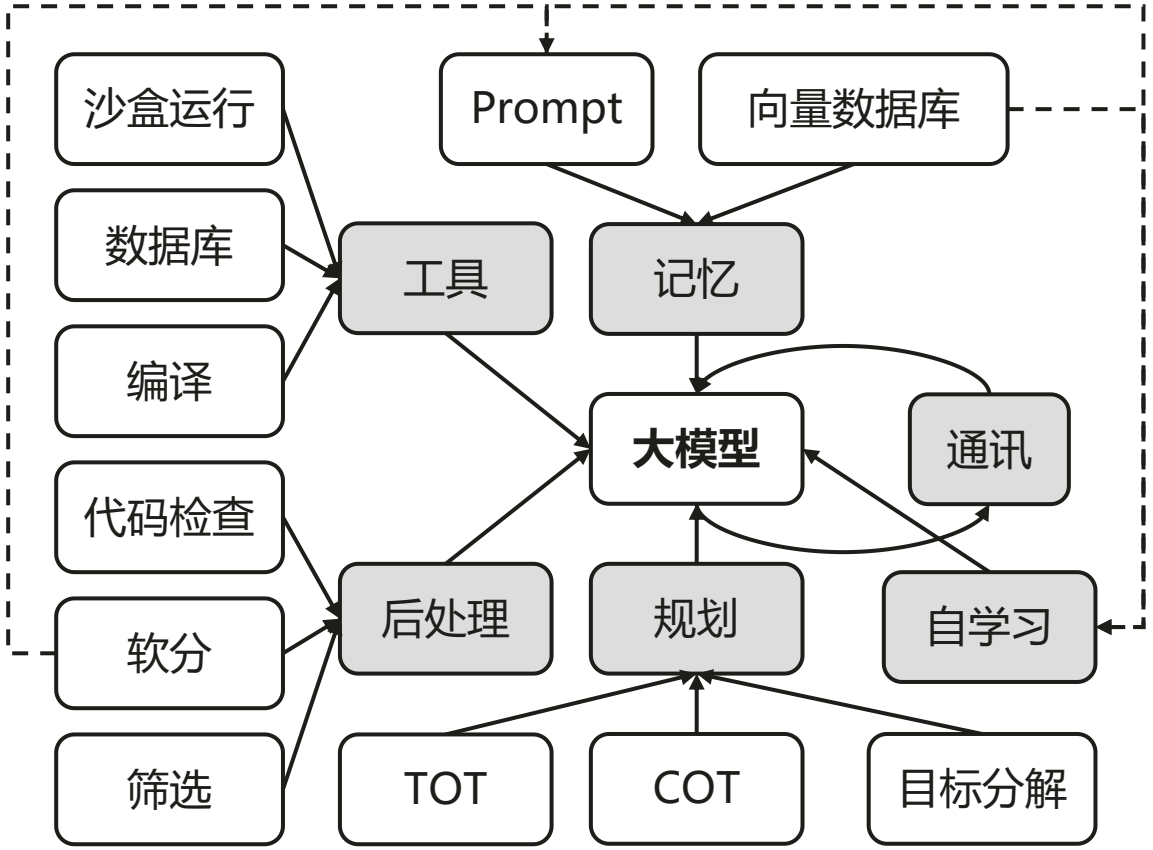
- **AIGC**：验收标准、测试用例、UI、代码、测试脚本等
- **人机交互智能**：软件研发过程就是人与计算机的交互过程
- **数字化**：软件研发平台开始理解需求、设计、代码等
- **以模型和数据为本**：研发人员服务于大模型和大数据平台

LLM的演进

2021年9月:
Sam Altman, the CEO of OpenAI, said that GPT-4 would focus more on coding
代码给了大模型推理(思维链)的能力?

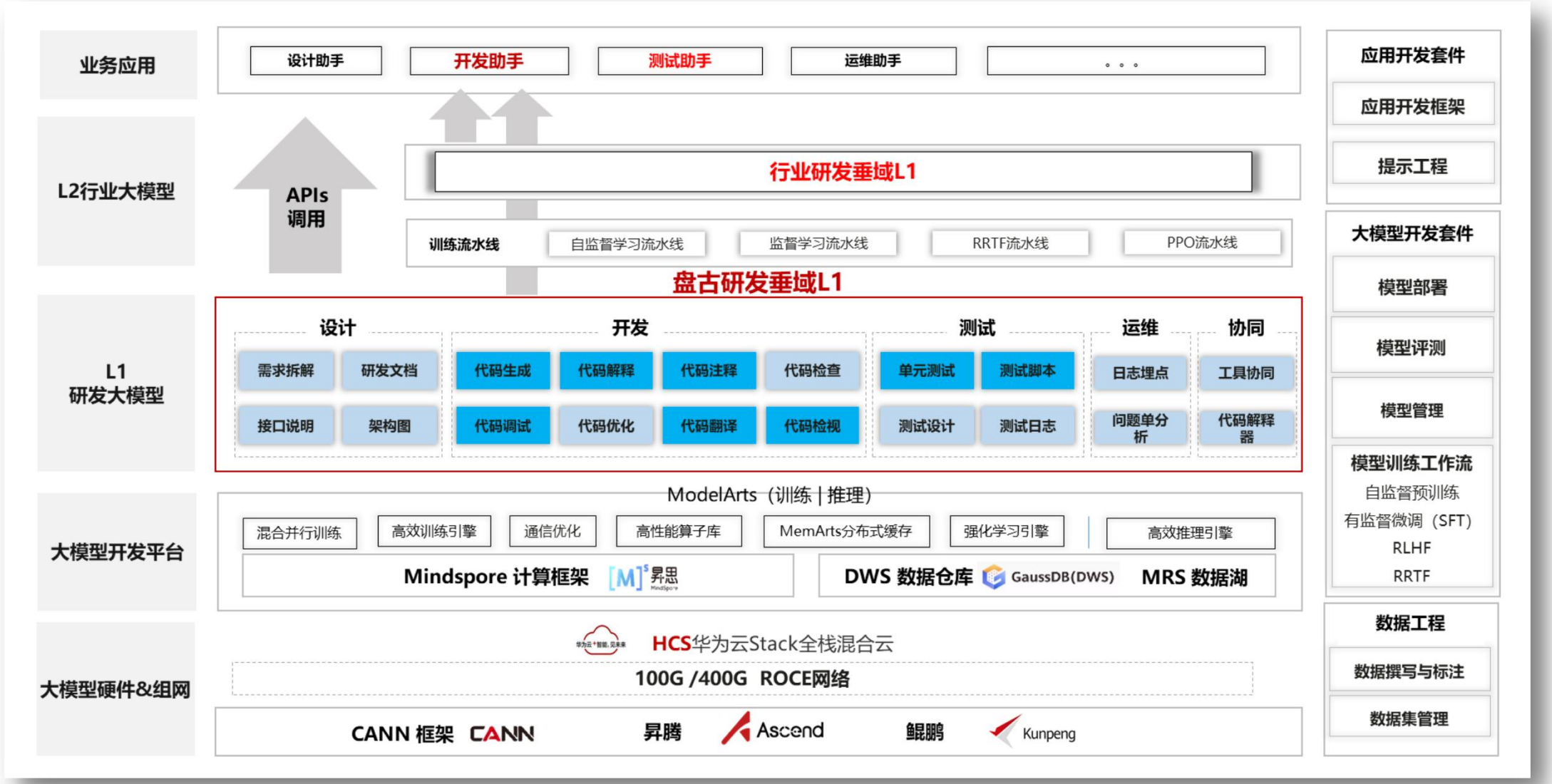
2023年:
GPT4元年，AI Agent全面爆发，GPT4 Turbo出世





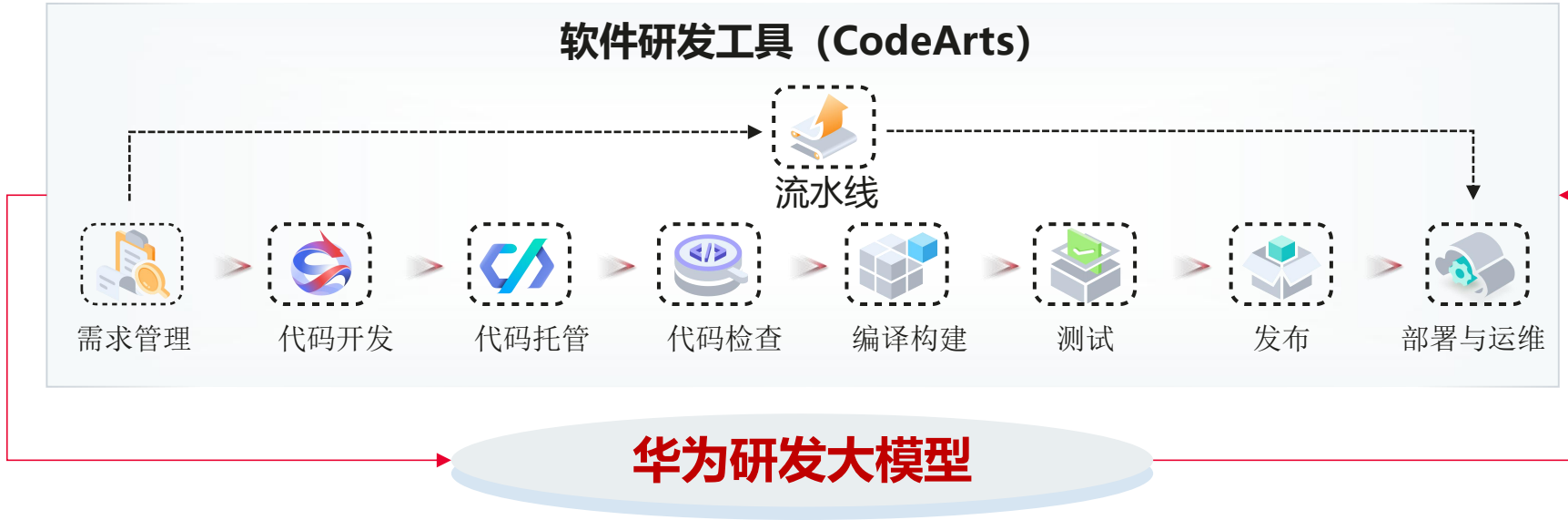
PART 02
AI邂逅一站式软件开发

华为研发大模型全景


业务应用
设计助手
开发助手
测试助手
运维助手
。。。
L2行业大模型
APIs
调用
行业研发垂域L1
训练流水线
自监督学习流水线
监督学习流水线
RRTF流水线
PPO流水线
盘古研发垂域L1
L1
研发大模型
设计
需求拆解
研发文档
接口说明
架构图
开发
代码生成
代码解释
代码注释
代码检查
代码调试
代码优化
代码翻译
代码检视
测试
单元测试
测试脚本
测试设计
测试日志
运维
日志埋点
问题单分析
协同
工具协同
代码解释器
大模型开发平台
ModelArts（训练 | 推理）
混合并行训练
高效训练引擎
通信优化
高性能算子库
MemArts分布式缓存
强化学习引擎
高效推理引擎
Mindspore 计算框架
昇思
DWS 数据仓库
GaussDB(DWS)
MRS 数据湖
大模型硬件&组网
HCS华为云Stack全栈混合云
100G /400G ROCE网络
CANN 框架
CANN
昇腾
Ascend
鲲鹏
Kunpeng
应用开发套件
应用开发框架
提示工程
大模型开发套件
模型部署
模型评测
模型管理
模型训练工作流
自监督预训练
有监督微调（SFT）
RLHF
RRTF
数据工程
数据撰写与标注
数据集管理

华为研发大模型定位：辅助研发效率提升

知识获取 (时间占比: 14%)	需求管理	系统设计	编码 (时间占比: 17%)	构建	测试 (时间占比: 19%)	发布
研发知识推荐			代码自动生成		测试用例自动生成	



2)研发作业数据**清洗与评价**，提升大模型能力

1)大模型提升**工具智能化**，助力研发能力提升

华为研发大模型的构建


利用软件研发工具
对数据数据进行加工/合成/筛选/清洗
华为内部的研发数据
(X万个代码仓，X亿行 代码，
X万篇技术博客)
外部开放的研发数据
(8600万个代码仓，1500亿行
代码，1300万篇技术博客)
需求管理
代码检查
编译构建
测试
软件算法
代码检查
智能化特性
代码仓维度
代码仓
准入门禁
华为可信质量评级
文件维度
代码开发工具
华为Clean Code标准
函数级维度
代码开发工具
数十项 原生分析能力
数十项 原生后处理能力
百亿
华为内部数据
千亿级
精品数据
万亿级
高质量海量数据
华为研发大模型
（华为内部业务）
通用研发大模型
（内外部通用）
基础模型

构建高质量的训练数据，让模型“天生”就有高质量

“脏代码”训模型，模型出“脏代码”

```python
def search(lst):
    """
    给你一个非空的正整数列表。返回大于零的最大整数，其频率大于或等于整数值本身。整数出现的频率是它在列表中出现的次数。如果不存在这样值，返回-1。
    示例:
    search([4, 1, 2, 2, 3, 1]) == 2
    search([1, 2, 2, 3, 3, 3, 4, 4, 4]) == 3
    search([5, 5, 4, 4, 4]) == -1
    """
    if not lst:
        return -1
    max_num = max(lst)
    if max_num <= 0:
        return -1
    max_num_count = lst.count(max_num)
    if max_num_count >= max_num:
        return max_num
    else:
        return -1
```

逻辑不清，代码难懂

初级程序员的水平

“Clean Code”进模型，模型出“Clean Code”

```python
def search(lst):
    """
    给你一个非空的正整数列表。返回大于零的最大整数，其频率大于或等于整数值本身。整数出现的频率是它在列表中出现的次数。如果不存在这样值，返回-1。
    示例:
    search([4, 1, 2, 2, 3, 1]) == 2
    search([1, 2, 2, 3, 3, 3, 4, 4, 4]) == 3
    search([5, 5, 4, 4, 4]) == -1
    """
    # 创建一个字典存储每个整数的频率
    freq_dict = {}
    for num in lst:
        if num in freq_dict:
            freq_dict[num] += 1
        else:
            freq_dict[num] = 1
    # 按key倒序将字典进行排序
    sorted_dict = dict(sorted(freq_dict.items(), key=lambda item: item[0], reverse=True))
    # 遍历排序后的字典
    for num, freq in sorted_dict.items():
        # 如果数字频率大于或等于本身值则返回
        if freq >= num:
            return num
    # 如果不存在这样的整数，返回-1
    return -1
```

步骤清晰，带有注释，代码易读

高级程序员的水平

典型场景1：代码自动生成

代码生成

测评数据集（Java）：
业界通用的HumanEval数据集（HE）

Benchmark	SNAP（华为）	WizardCoder	Code Llama
HumanEval (Python)	82.3%	68.29%	48.17%
HumanEval (Java)	67.1%	56.71%	41.46%
HumanEval (C/C++)	58.5%	49.39%	44.51%

在所对比的百亿级模型中，**SNAP**在代码生成上准确率最佳，一次生成通过测试的概率在业界通用的**HE数据集**上达到**同等规模参数模型业界Top1**。

过去

开发者需要手动编写每一行代码，这需要投入大量的时间和精力；对于复杂的算法和数据结构，尤其痛苦和耗时

现在

通过理解**自然语言和关联上下文**自动生成代码，大大减少了开发者的工作量，提高了开发效率

```
import utils.Tools;
1 related problem
public class Main {
    public static void main(String[] args) {
        Tools helper = new Tools();
        int day = 2;
        int hour = 8;
        int minute = 2;

        // use helper get second through hour and minute


        helper.get_second_by_hour(hour);
    }

}
```

跨文件关联“utils.tools”上下文的代码生成示例

典型场景2：单元测试用例生成

单元测试用例生成

测评数据集：无业界公认的单元测试用例评测集，人工构建了一个包含10个Java类，14个Java方法的评测集。

数据集组成	基础类型	控制流	面向对象	微服务
Java类	3	2	3	2
Java方法	7	2	3	2

测评方法：采用业界通用的，通过对模型推理的代码进行修改，对完整的函数测试行覆盖率。

SNAP生成的代码相对于开源 SOTA生成的代码，**代码更简洁**，需要**人工修改**的代码行数**更少**，**行覆盖率更高**。

指标	SNAP（华为）	开源 SOTA
平均测试代码行数	**52**	156
平均修改代码行数	**10**	31
平均行覆盖率	**77%**	70%

过去

手动编写每个单元测试用例的详细步骤和预期结果，在有限的开发时间保证代码测试覆盖率困难

现在

自动创建单元测试用例，提高测试覆盖率，确保每个功能和场景都被测试到

```java
public void testReadLines4(String pathname0) throws Exception {
    // setup
    File file = new File(pathname0);

    // run the test
    List result = FileUtil.readLines(file);

    // verify the results
    Assertions.assertEquals(null, result);
}

public String startsWithLiteral(String str) {
    if (str.starsWith("1234567890")) {
        str = str.replace("3", "A");
    } else {
        str = str.trim();
    }

    if (str.charAt(0) == 'x'){
        return str;
    } else {
        return str.toLowerCase();
    }
}
}
```

String类型含两个if-else的单元测试用例生成

典型场景3：代码调试

代码调试

测评数据集：参考了公开数据集，构建了常见异常的Java代码和运行时报错评测数据集。

测评方法：将模型推理的代码，直接在测试环境中进行单元测试，采用Pass@3通过率作为评价指标。

测评功能：
- 代码纠错：采用错误代码作为prompt，模型纠正代码错误。
- 代码Debug：采用错误代码+报错信息作为prompt，模型纠正代码错误。

SNAP不仅具备**代码纠错**能力，也提供了**代码Debug**能力。在输入代码+报错信息和只输入代码的**两种代码调试场景**下，华为**SNAP准确率更高**。且在只输入代码进行纠错的场景下，**误报率更低**。

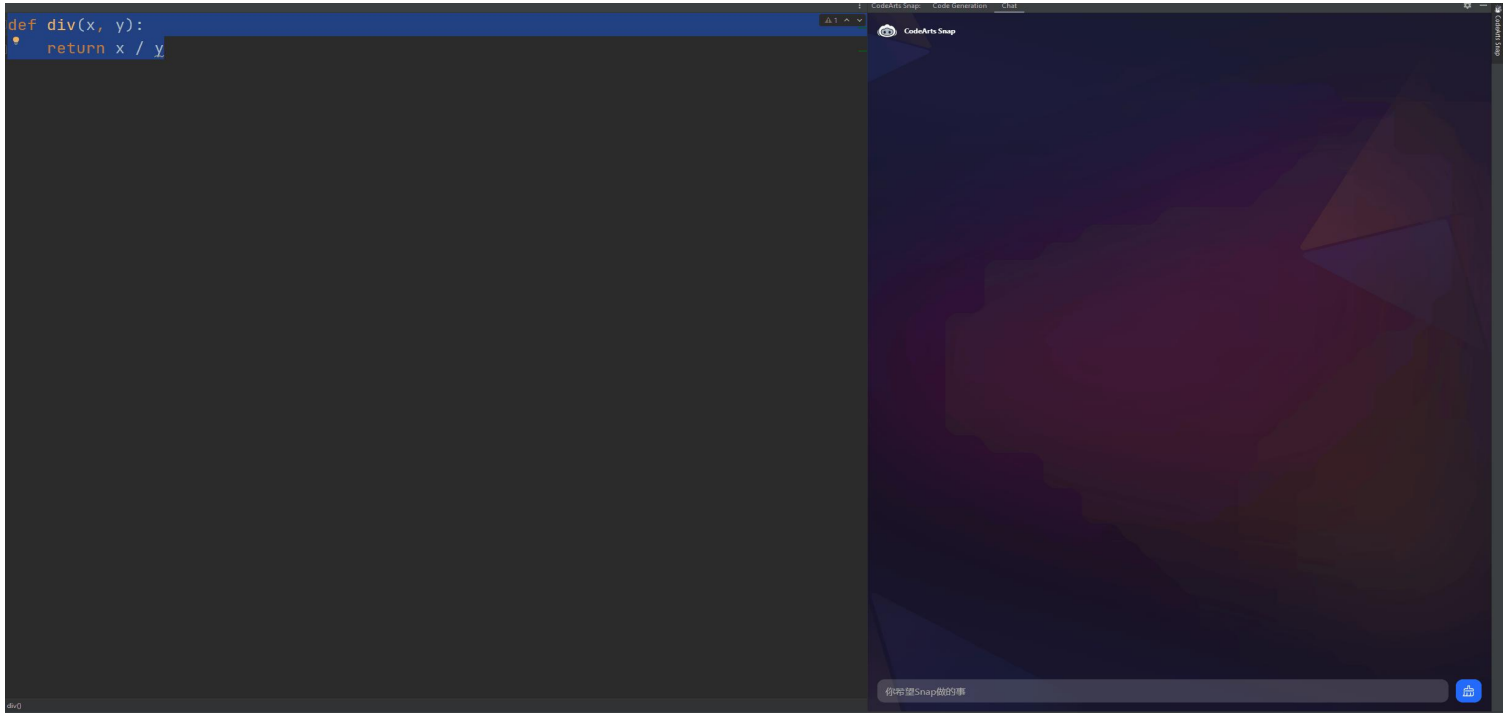
调试细分功能	输入内容	SNAP（华为）	开源SOTA
		Pass@3	Pass@3
代码纠错	仅代码	**65%**	55%
代码Debug	代码+报错信息	**65%**	60%

过去

查找代码中的错误需要大量的时间和耐心，尤其是对于复杂的代码和大型项目来说，定位问题可能会非常困难

现在

快速定位问题，并提供错误发生的详细信息和修复建议，大大缩短了定位错误的时间



检出“除数为0”和“类型错误”并给出修复代码示例

典型场景3：代码调试

代码调试

测评数据集：参考了公开数据集，构建了常见异常的Java代码和运行时报错评测数据集。

测评方法：将模型推理的代码，直接在测试环境中进行单元测试，采用Pass@3通过率作为评价指标。

测评功能：
- 代码纠错：采用错误代码作为prompt，模型纠正代码错误。
- 代码Debug：采用错误代码+报错信息作为prompt，模型纠正代码错误。

SNAP不仅具备**代码纠错**能力，也提供了**代码Debug**能力。在输入代码+报错信息和只输入代码的**两种代码调试场景**下，华为**SNAP准确率更高**。且在只输入代码进行纠错的场景下，**误报率更低**。

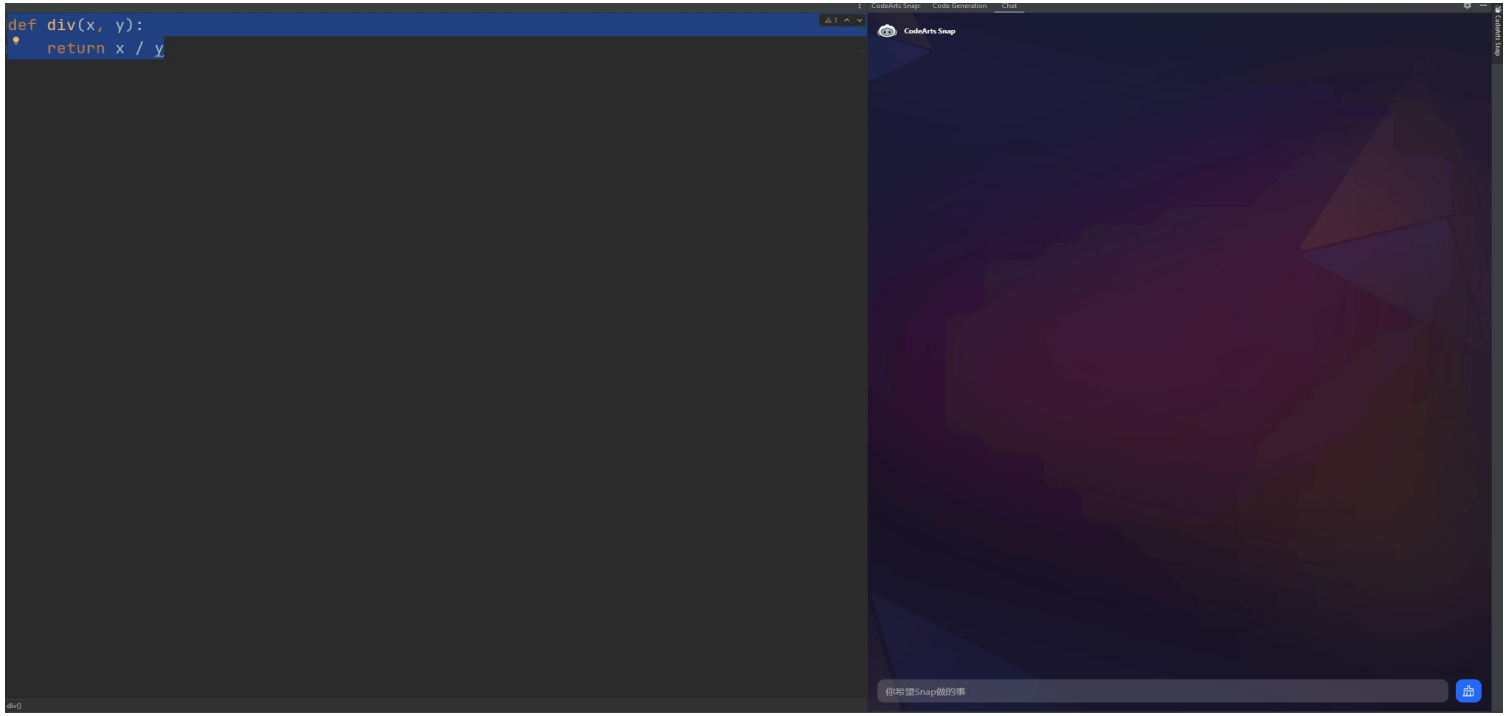
调试细分功能	输入内容	SNAP（华为）	开源SOTA
		Pass@3	Pass@3
代码纠错	仅代码	**65%**	55%
代码Debug	代码+报错信息	**65%**	60%

过去

查找代码中的错误需要大量的时间和耐心，尤其是对于复杂的代码和大型项目来说，定位问题可能会非常困难

现在

快速定位问题，并提供错误发生的详细信息和修复建议，大大缩短了定位错误的时间



检出“除数为0”和“类型错误”并给出修复代码示例

PART 03
研发大模型的关键问题与技术挑战

关键问题与技术挑战

模型优化

关键技术1：中文友好的代码生成

- 在保障模型性能的条件下，增强中文语义的理解能力
- 利用**大模型能力持续补齐数据**上中文社区短板

关键技术2：Prompt组装

- 判断用户输入的任务描述完整性和合理性，并通过交互明确意图，提高代码生成准确率
- Prompt**结合精准上下文**，扩大模型生成感知野，

关键技术5：模型在线学习

- 基于用户的**显式和隐式反馈**对在线的大模型进行微调，实现在线模型实时更新

关键技术6：低成本SFT

- 如何实现各种研发场景的训练/验证数据集快速低成本建设，以及模型的训练以及自动验证部署
- 模型参数外挂等技术，实现低成本训练与数据隔离

工程优化

关键技术3：AI Agent探索

- 结合预训练模型，在Agent逻辑下，完成复杂任务的精准生成与推理
- 结合记忆模块，进行搜索增强，**减少模型幻觉**，提升回答准确率

关键技术4：体验评估与优化

- 对于越来越多的下游任务构建成体系的**评测方法论**构建客观且贴近真实工程的**Benchmark**

关键技术7：后处理

- 根据项目上下文检查和修复所生成代码的编译运行错误
- 利用沙盘，通过编译与运行**直接反馈并迭代完成修改**
- 结合单元测试，修复生成程序中的逻辑性错误

关键技术8：模型量化

- 在保证精度不过多下降的前提下，量化模型，支撑端测算力实现模型推断

关键问题与技术挑战

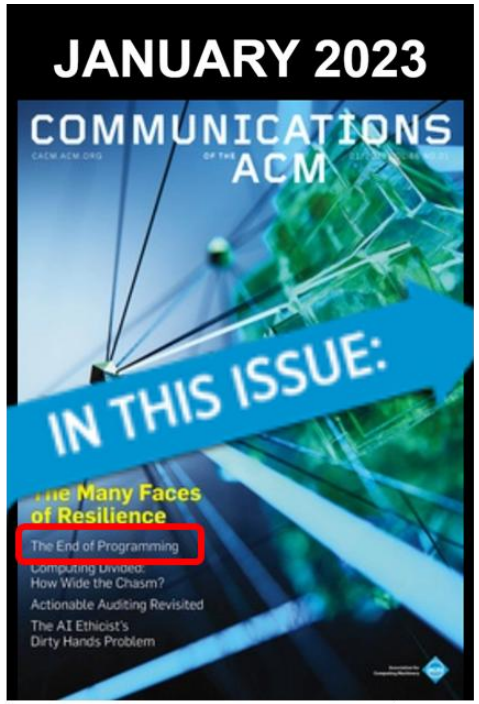

HUAWEI
CCF-华为胡杨林基金-软件工程
2023年度课题宣讲会
宣讲介绍
CCF-华为胡杨林基金是华为面向泛计算领域的综合科研基金，旨在通过搭建产学合作平台，连接产业实践问题与学术科研问题，支持海内外优秀青年学者开展与产业结合的前沿科研工作。CCF-华为胡杨林基金-软件工程专项的主要目标是持续提升相关软件工程技术能力，针对业界典型问题和痛点，解决关键技术瓶颈或形成技术断裂点，大幅提升产品/平台/解决方案竞争力，创造产业价值。


CCF-TSE&华为软件工具战略研讨会暨胡杨林基金指南评审


宣讲议程
14:10-14:15 华为云PaaS技术创新Lab主任 开场 王千祥(华为)
课题一：基于大模型的软件研发技术项目群
14:15-14:30 子方向1-代码大模型的提示工程技术 张芮恺(华为)
14:30-14:45 子方向2-代码表征与生成场景的预训练模型微调技术 李钟麒(华为)
14:45-15:00 子方向3-工具增强的代码大模型后处理技术 申 博(华为)
15:00-15:15 子方向4-面向可信代码生成的人机协同自然交互编程语言 申 博(华为)
15:15-15:30 子方向5-代码分析与大模型结合的单元测试用例生成和演化技术 吴添勇(华为)
15:30-15:45 子方向6-基于大模型的系统集成测试生成和优化技术 万锐媛(华为)
15:45-16:00 子方向7-基于大模型的研发生产线智能支持技术 郝月婵(华为)
16:00-16:15 子方向8-基于大模型的代码精准重构技术项目 郭肇强(华为)

PART 04
10X开发者展望

编程的终结，言之过早



top 5 articles

- The End of Programming
- AI's Jurassic Park Moment
- The Many Faces of Resilience
- Donald Knuth's 2022 'Christmas Tree' Lecture Is About Trees
- Democratizing Domain-Specific Computing

Viewpoint

The End of Programming

The end of classical computer science is coming, and most of us are dinosaurs waiting for the meteor to hit.

I CAME OF AGE in the 1980s, programming personal computers such as the Commodore VIC-20 and Apple][e at home. Going on to study computer science (CS) in college and ultimately getting a Ph.D. at Berkeley, the bulk of my professional training was rooted in what I will call "classical" CS: programming, algorithms, data structures, systems, programming languages. In Classical Computer Science, the ultimate goal is to reduce an idea to a program written by a human—source code in a language like Java or C++ or Python. Every idea in Classical CS—no matter how complex or sophisticated, from a database join algorithm to the mind-bogglingly obtuse Paxos consensus protocol—can be expressed as a human-readable, human-

AI辅助能有效提升研发人员的单兵作战能力

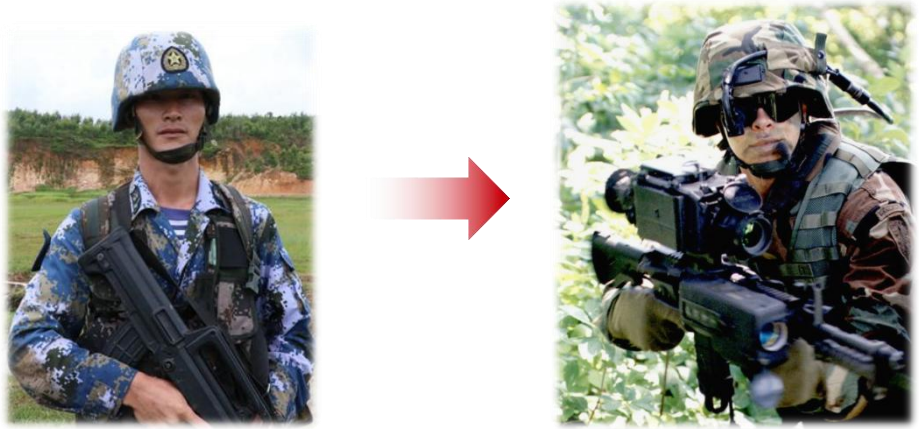
软件工程1.0（质量）	软件工程2.0（效率）	软件工程3.0（智能）
瀑布模型	敏捷、精益	人机协同

一场大赛，研发智能化已悄然到来：

- 上届冠亚军选手没有使用大模型，在今年大赛中，冠军排在170名，亚军排在450名。
- 今年8位满分选手全都使用大模型。

AI辅助研发

支撑研发人员从 **普通士兵** **到** **特种兵**的升级



新时代软件研发

由三个关键角色即可组成未来完整的软件研发全栈团队：**产品专家、架构专家、QA专家**，大量软件研发工作将协同AI完成，极速开发迭代，智能高效测试，无人化运维运营

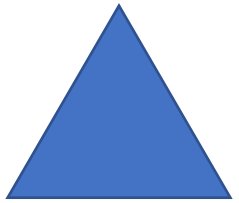
研发流程	开发				测试			运维		运营	
作业场景	需求&协同	设计&编码	检查&合入	构建	测试设计	执行	测试分析	问题发现	问题定位	智能配置	技术支持

Scaling Law的终结与模型的自我进化

扩展法则 Scaling Law的关键变量：计算量、数据集和参数量

- AI算力的新摩尔定律，引导一线大厂开始算力军备竞赛
- 24年国内将以万片为单位计算集群，意味着万亿级模型的算力**底座已初步具备**

计算量



数据集

- 全球高质量数据集是有限的，目前统计有XX T Token
- 基于当前版本GPT4的标注数据，在部分领域**已超越人的标注数据**
- 随着国内主流厂商专业用户量的提升，也会带来更为精准的数据反馈

参数量

- 23年开源模型年初与年尾参数规模对比，整体可用版本的规模大了约20~50倍（约X B ~ XXX B）
- 24年预计开源模型也会**有同等规模的提升**
- 24年闭源模型将加快追赶GPT的脚步

AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

THANKS