51CTO WOT

World Of Tech 2024

WOT全球技术创新大会

智启新纪
慧创万物

51CTO WOT

AI驱动软件研发实践——方法与工具

谭 宇

Fabarta 资深技术专家

51CTO WOT

Contents

目录

01

背景：软件工程与AI辅助

个人及Fabarta简介

关于我

- Fabarta 资深技术专家，目前主要专注于 AI 时代的多模数据库引擎 ArcNeural 的建设。
- Fabarta之前曾任阿里云资深技术专家，主攻数据库、云计算与数字化转型方向。
- 带领过超过 100 人的技术研发团队。
- 管理过超过1000 人的项目团队，交付过千万以上项目数十个，包括政府、电力等多个行业。

Fabarta

- Fabarta 成立于2021年，目前拥有近百人的团队，在北京、杭州、上海、宁波和西安设有研发办公室。
- 创始团队来自于阿里、IBM 等世界知名公司，对云原生，分布式数据库、AI等领域有深刻理解。
- 基于对大模型时代 AI 应用落地范式的理解，结合当前服务的多家大型头部金融、制造业等客户的业务痛点和需求，提出了"一体两翼"的产品矩阵。一体指的是ArcNeural多模态智能引擎，两翼则分别指代数据与AI。

背景：为什么要建设AI驱动的研发团队

Software is eating the world (2011, Marc Andreessen) → AI is eating the software (2019, Forbes) → ?

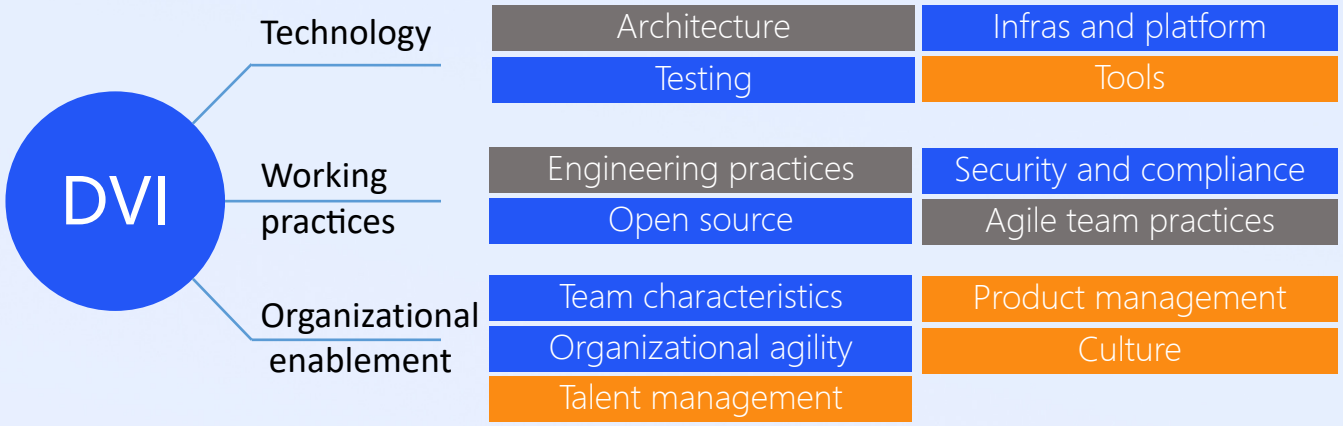
- 软件颠覆各行各业
  - 零售
  - 娱乐
  - 通信
  - 招聘
  - 汽车
  - 物流
  - ...
- 技术成熟：计算机60年、CPU40年、互联网20年
- 软件企业正在构建真实、高增长、高利润且有护城河的业务

- AI开始吃掉软件
  - 制药
  - 物流
  - 软件开发
  - 汽车（自动驾驶）
  - ...
- 技术成熟：训练数据、算法精确度、AI芯片算力
- 没有使用AI的公司会陷入困境
- 企业为了保持竞争力，需要在 AI 方面进行投入，包括算力、数据、模型

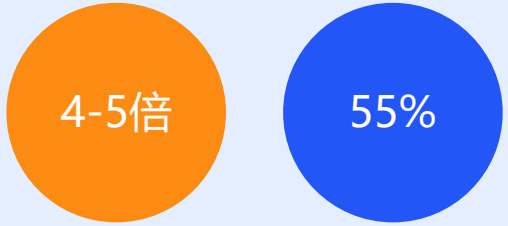
- AI 替代一切?
- 至少现阶段还没有到可以替代软件工程师的阶段。反而会更加突显软件工程师的重要性。

AI驱动软件研发实践的目标：开发者速度

Mckinsey 在2020年4月发布了一篇报告，调研了分布在12个行业、9个国家的440家公司，指出开发者速度与商业表现强相关，并定义了Developer Velocity Index(DVI)来衡量一个企业的开发者速度。



DVI 涵盖技术、工作实践、组织赋能三大领域共13个维度46项能力。



DVI指数高的公司其营收增长是DVI靠后的公司的4-5倍且更具创新性。

围绕开发者速度的四大关键因素

01

工具

规划、开发、交付和运营、协作与沟通等工具使用情况

02

产品管理

业务敏感度、客户体验、产品愿景理解、数字资产的积累

03

文化

以客户为中心、跨职能的团队协作、对客户成功的获得感、服务式领导

04

人才管理

数字人才的招聘与激励、员工价值主张阐述、远程协作与职业培训

AI辅助能带来的变化

当前开发者速度上的主要问题

在工具上的投入普遍不足	仅有5%的管理者认识到工具与DVI的联系，投入不足导致之前的投入效果不明显，从而导致减少投入。
工具选择上的局限性	每个企业大约需要30～50个工具一起协作，没有一家单一的企业可以把所有的工具产品做到极致，但工具产品企业想做大一统，客户想要一站式解决方案。
工具未能致力于消除中间产物	工具本身能力有局限，无法达成预定目标。
重流程而轻积累	启用工具的动机有偏离，重量化、考核。数据分散而导致没有或很少积累。
“想法翻译”的困境	开发者未能参与到产品建设、产品愿景、客户体验等环节，从而沦落为翻译前端需求的翻译者。

AI辅助可能带来的改变

AI 焦虑会带来投入
专精工具会占主流
更加智能、有生产力的工具
回归工具本质、更容易积累进化
有能力从想法翻译者到造梦师

02

构建AI驱动的研发团队的必要条件

找出并消除阻碍

- 访问速度与稳定性。
- 内部系统的可访问性。

- 软硬件成本，最好不要让员工承担。
- 开发成本与维护成本。

- 提前规划好数据隐私、数据安全的红线，过程中勿再以数据安全、合规等理由中断。
- “一刀切”不可取，最好使用黑名单机制。

- “预期过高” 的伤害大于低预期。采用AI工具需要有低预期值。
- 个人需要为结果负责。
- 团队内定期分享一些小技巧。

适可而止的调研

选择不是太少，而是太多，每个团队需要根据自己的需求来选出合适自己的工具，不要陷入到工具选择的汪洋大海，不要随时切换工具、不要有过多的选择。

OpenAI, Anthropic, Google, Baidu, Alibaba	Github copilot X, Codium, TabNine.....
Llama, qianwen, Mistral, DeepSeek,	Text2SQL, Text2Image, Text2Anything .....
LangChain, LlamaIndex, Ollama, vLLM, ...	XXXGpt, XXXChat, XXXGenerator, XXXAssistant .....
FlowiseAI, LangFlow, Dify, LLMStack, FastGPT	AI for XXX (Designer, Data, HR, Customer service .....)

理解团队属性、培训团队成员

前端开发团队

后端应用开发团队

前端主导的综合性团队

后端主导的综合性团队

底层技术团队

从过程式与声明式谈起

理解AI的基础知识
(AI，生成式AI，AGI，机器学习，深度学习)

理解应用AI的基础知识
(LLM及其局限、提示工程，RAG)

鼓励多动手试验

分享实际工作中的案例

- 理解自身的属性，才能理解需求。
- 问题越明确，AI 工具才能更好的发挥。
- 尽量不要将技术选型等工作交给 AI 工具来做。

- 建立拥抱AI的团队氛围，不管是否在工作中起到多大的作用，都有助于团队成长。
- 相同的上下文，一致的认知有助于AI 工具的采用与提交。

理解团队属性、培训团队成员

或者至少学习一下OpenAI 关于提示工程的指南，这也是对个人或团队工作方式的指南……

写清晰的指令

- ✓ 在查询中包含详细信息以获得更相关的结果
- ✓ 要求模型采用某种角色
- ✓ 使用分隔符清晰地指示输入的不同部分
- ✓ 指定完成任务所需的步骤
- ✓ 提供示例
- ✓ 指定输出的期望长度

提供参考文本

- ✓ 指示模型使用参考文本回答
- ✓ 指示模型引用参考文本回答

将复杂任务分解为更简单的子任务

- ✓ 使用意图分类来识别用户查询中最相关的指令
- ✓ 对于需要非常长对话的应用，总结之前的对话
- ✓ 分段总结长文档，并递归地构建完整摘要

给模型时间"思考"

- ✓ 指示模型在得出结论前先自行解决问题
- ✓ 使用旁白或一系列查询来隐藏模型的推理过程
- ✓ 询问模型是否在处理过程中遗漏了什么

使用外部工具

- ✓ 使用基于嵌入的搜索实现高效的知识检索
- ✓ 使用代码执行来进行更准确的计算或调用外部 API
- ✓ 给予模型访问特定功能的权限

系统地测试变化

- ✓ 通过参考标准答案评估模型输出

03

Fabarta实践：方法与工具

工具的选择原则

通过选择合适的工具作为切入点，先用起来，不要急于度量收益。

眼前一亮	“15 分钟原则”，优先付费产品，同类产品允许1～3个选择。
抽象层次	作为开发团队使用的产品，抽象层次不能太高。
契合实际	不需要对现有的状况作出重大改变即可立即使用。

从一个点开始：代码生成工具

Github Copilot

Codeium

Tabnine

- 支持大多数IDE，可快速使用
- 建议从个人版开始
- 关注延迟与生成质量

代码补齐

写注释、文档

补齐测试代码、填充测试数据

改写代码(单行-> 多行，更换实现方式等)

解释代码

- 主要使用的功能涵盖了编码过程中的大部分工作，能有效提升效率。

熟练使用工具

避免重复

避免过度依赖

- 使用代码生成工具的注意事项。

团队内逐步扩展：文档

代码生成工具可能会告诉你一些原本并不知道的事情……

```
## RPC 的基本概念

维斯百科上对 RPC 的定义为：“RPC 是一种通信协议，它允许一个程序调用另一个地址空间（通常是另一个机
户端-服务器模式的通信模式，它允许客户端调用远程服务器上的过程，就像调用本地过程一样。RPC 通常使用
码。这种中间语言通常是 IDL（Interface Description Language）。“

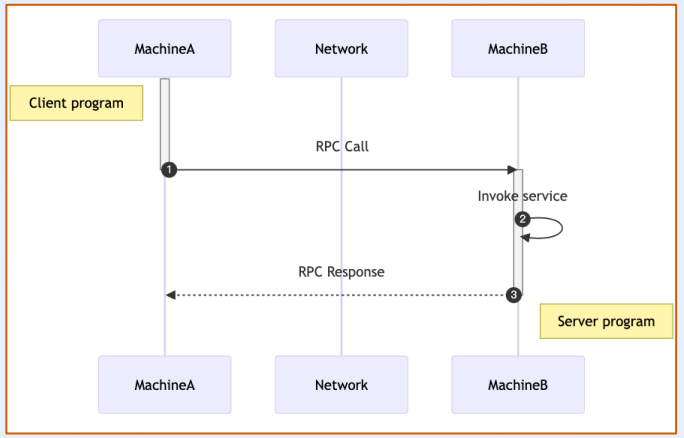
根据这个定义，我们可以通过一个最简化的图来理解 RPC 的基本概念：

    ```mermaid
```

根据这个定义，我们可以通过一个最简化的图来理解 RPC 的基本概念：

```mermaid
graph TD
    A[客户端] -->|调用| B[远程服务器]
    B -->|返回结果| A
```

- 文本化可能会带来意想不到的好处
- 尽可能文本化
  - ✓ Markdown / CSV
  - ✓ Image -> mermaid / dot
  - ✓ Code -> JS / Python / Jupyter Notebook
- 团队在相同限制条件下工作可以提升效率


```

团队内逐步扩展：测试

除了生成测试代码与测试数据外，有很多场景可以直接使用大模型的能力

```
``` https://arcgraph-docs.fabarta.com/docs/graph-database-developer/ddl/vertex-type
```
以上由三引号```引用的内容是 Arcgraph关于DDL以及Cypher语法的定义链接，以下由三引号```引用的内容是一份业务Schema：
```
CREATE VERTEX BusinessRule(
 primary key id string(64),
businessRuleId string,
...
```
请随机生成20条cypher语句用于测试。
```

- 将大模型的劣势转化为优势
- 找准测试场景
  - ✓ 不追求精确
  - ✓ 需要大量的场景
  - ✓ 需要随时更新用例
- 可实现低成本用例与高质量case构建
- 注意大模型的消费情况
- 大多数情况下跑一些小模型就能满足需求

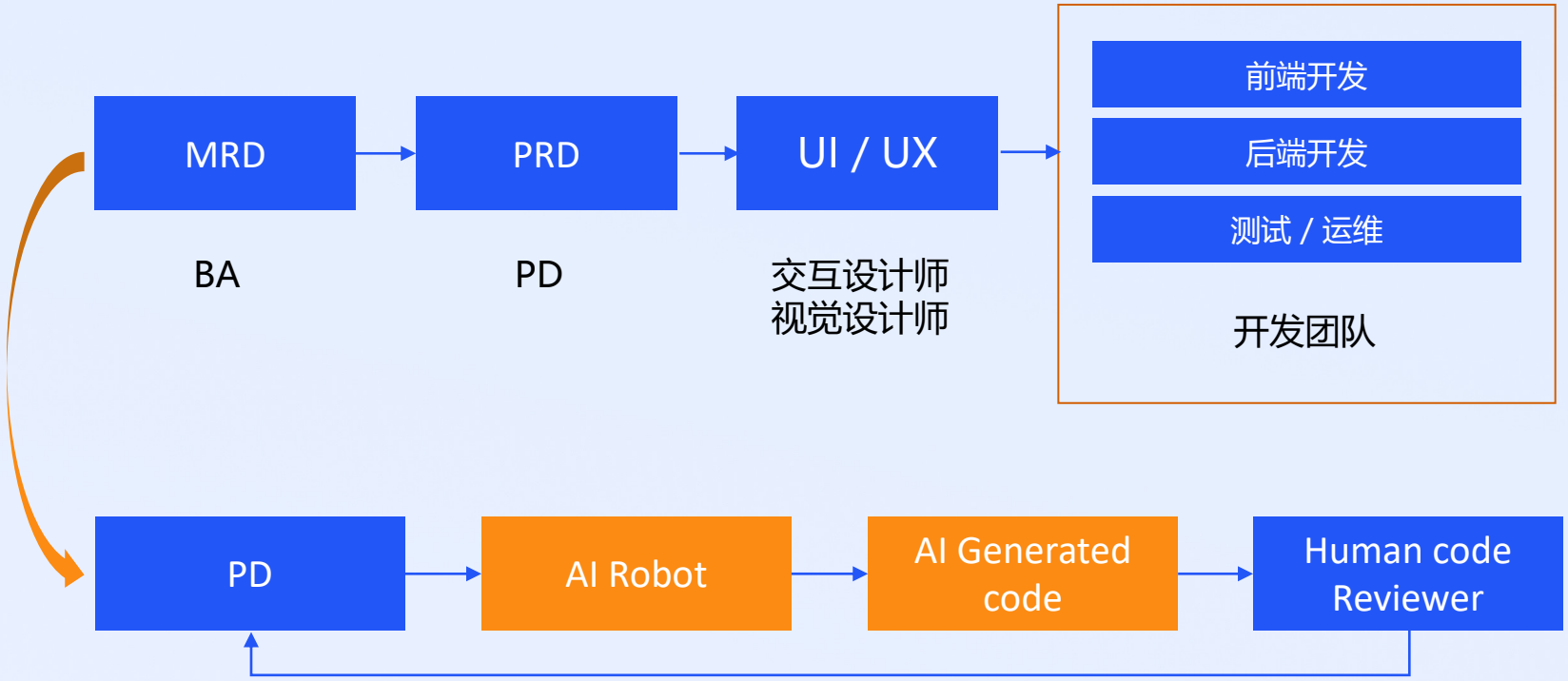
根据提供的业务Schema和ArcGraph的DDL定义，以下是生成的20条随机Cypher语句用于测试：

cypher Copy

```
-- 创建 BusinessRule 节点
CREATE (b:BusinessRule {id: 'br1', businessRuleId: 'BR001', name: 'Rule1',
description: 'Description1', type: 'Type1', principal: 'Principal1', isUsed:
'true'});
```

团队间扩展：设计、全栈

以“消除中间产物”、“中间环节”为指引，逐步构建AI时代的工具链



- 流程长、环节多，沟通效率低下
- 不直接为最终结果负责
- 想法翻译者的困境
- 之前的尝试
  - 开发人员全栈
  - 低代码 / 无代码

- Matt Welsh 提出的一个未来团队可能的组成部分，他认为绝大部分程序会由AI来编写。

团队间扩展：设计、全栈

- 完全消除中间产物的时机还没有来临
- 现阶段致力于消除沟通的中间环节
  - 从整体上来看，开发团队真正消耗的时间占比不足30%
  - 编写代码的时间占比不超过开发团队所用时间的30%
  - 澄清、修改、沟通需求时间占比远超想象
- 消除沟通环节可以有效提升整体效率
- 通过AI能力建设“企业知识中台” 可以有效解决团队间沟通问题

其它工具上的考虑

引入新工具之前先检查现有工具执行情况

- **代码Review**：现有代码审查制度和流程是否有认真执行?
- **代码分析**：现有的 lint 是否执行到位?
- **AIOps**：是否真的敢让线上发布操作让AI来自动完成?
- **AI智能分析**：是否真有大体量运维数据需要 AI 来分析?
- **项目计划与管理**

发掘长期低效但又习以为常的工具/习惯

- **命令行历史**：多终端、多机器不互通、搜索便，无法积累
- **问题记录**：总结聊天内容自动提issue
- **事故/BUG无法完整分析**：通过AI总结、标记、分类及关联代码
- **集思广益、团队内持续分享小的tips并通过小的代价实现为工具**

实践总结：全景图

代码 / 文档生成	文档 / 测试	团队间知识共享	其他小工具
• Copilot Chat • Codium	• Mermiad • Markdown • ChatGPT • 本地大模型	• EKC （自研） • Dify （云上 or 自建）	• Atuin 命令行管理 • Monica （个人生产力） • 聊天内容总结 • Issue 自动提交 • 故障分析

04

总结与展望

仍然没有银弹

- 开发人员本身就应该理解业务、积累业务。
- 一切不解决业务问题的工具都是无效工具。

- 一切不提升生产力的工具都没有必要引入。
- 要考虑个人效率、更要考虑团队效率。

- 工具有局限，AI也有局限，但已经能极大提升生产力。
- 控制预期，客观评估，积极引入是现阶段的最佳策略。

- “想法翻译者”的困境必然会得到解决。
- 解决了一个问题，会出现新的问题，做问题的解决者。

实践成本

工具费用并不高

- ChatGPT
- Copilot
- 代码托管
- 本地GPU机器

更贵或更难的是团队习惯与制度

- 是否支持访问大模型API
- 是否支持托管代码、服务在公共云
- RAG时是否支持云上大模型
- 是否有明确的文档分级与密级管理

未来团队可能的形式

多年前画出的大饼仍然没有实现，目前看起来在AI的能力之下最有可能，但是仍然需要新的技术、新的方法。

未来员工的七项原则

1.有一个灵活的工作环境
   1. 在任何时间工作
   2. 在任何地点工作
   3. 注重产出而非投入
2.可以定义自己的工作
3.共享信息
4.使用新的方式进行沟通和协作
5.有机会成为领导者
6.从知识型员工转变为学习型员工
7.自由地学习和指导他人

未来组织的十四项原则

1.员工分布在全球各地，在小型团队中工作
2.内部创业
3.创建一个相互联结的员工网络
4.像小公司一样经营
5.实现愿望，而不仅仅是满足需求
6.更快的适应变化
7.无时无刻不在创新
8.在云端运行
9.讲故事
10. .....

谢 谢 观 看

THANKS