

🟡 MetaGPT

MetaGPT是一项引起广泛关注的研究成果，它引入了一个将人工工作流程与多智能体协作无缝集成的框架。通过将标准化操作（SOP）程序编码为提示，MetaGPT确保解决问题时采用结构化方法，从而减少出错的可能性。

开始阅读前，如果你对其他文章感兴趣，可以到欢迎页关注我们！「卡尔的AI沃茨」开源中文社区实时获得后续的更新和最新的教程

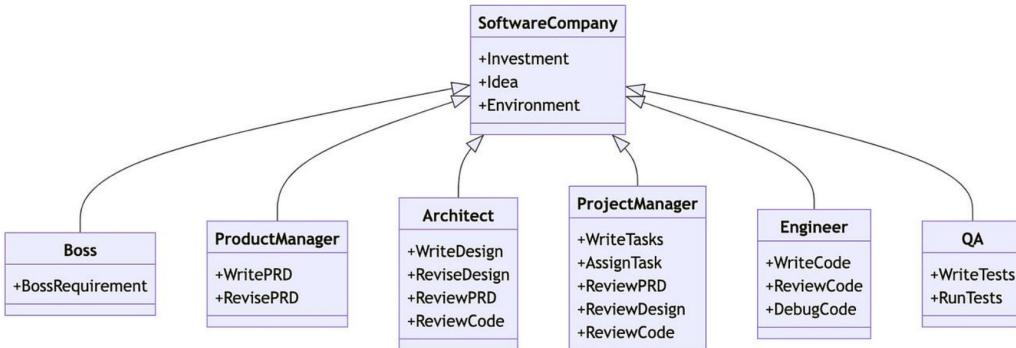
MetaGPT: The Multi-Agent Framework



Assign different roles to GPTs to form a collaborative software entity for complex tasks.

[文档 中文版](#) [document English](#) [ドキュメント 日本語](#) [MetaGPT](#) [License MIT](#) [ROADMAP 路线图](#) [WeChat 微信](#) [Follow @MetaGPT](#)

1. MetaGPT takes a **one line requirement** as input and outputs **user stories / competitive analysis / requirements / data structures / APIs / documents, etc.**
2. Internally, MetaGPT includes **product managers / architects / project managers / engineers**. It provides the entire process of a **software company** along with **carefully orchestrated SOPs**.
 - i. **Code = SOP(Team)** is the core philosophy. We materialize SOP and apply it to teams composed of LLMs.



```

graph TD
    SC[SoftwareCompany] --> Boss[Boss]
    SC --> PM[ProductManager]
    SC --> A[Architect]
    SC --> PM[ProjectManager]
    SC --> E[Engineer]
    SC --> QA[QA]
    Boss --> PRD[+BossRequirement]
    PM --> PRD[+WritePRD]
    PM --> RPRD[+RevisePRD]
    A --> RD[+WriteDesign]
    A --> RRD[+ReviseDesign]
    A --> PRD[+ReviewPRD]
    A --> RC[+ReviewCode]
    PM --> WT[+WriteTasks]
    PM --> AT[+AssignTask]
    PM --> PRD[+ReviewPRD]
    PM --> RD[+ReviewDesign]
    PM --> RC[+ReviewCode]
    E --> WC[+WriteCode]
    E --> RWC[+ReviewCode]
    E --> DC[+DebugCode]
    QA --> WT[+WriteTests]
    QA --> RT[+RunTests]
  
```

Software Company Multi-Role Schematic (Gradually Implementing)

当前Agent的解决方案存在一个问题：尽管这些语言模型驱动的 Agent 在简单的对话任务上取得了显著进展，但在面对复杂任务时，LLM 会陷入困境，仿佛看到了并不存在的事物（幻觉）。当将这些 Agent 串联起来时，就会引发混乱的连锁反应。

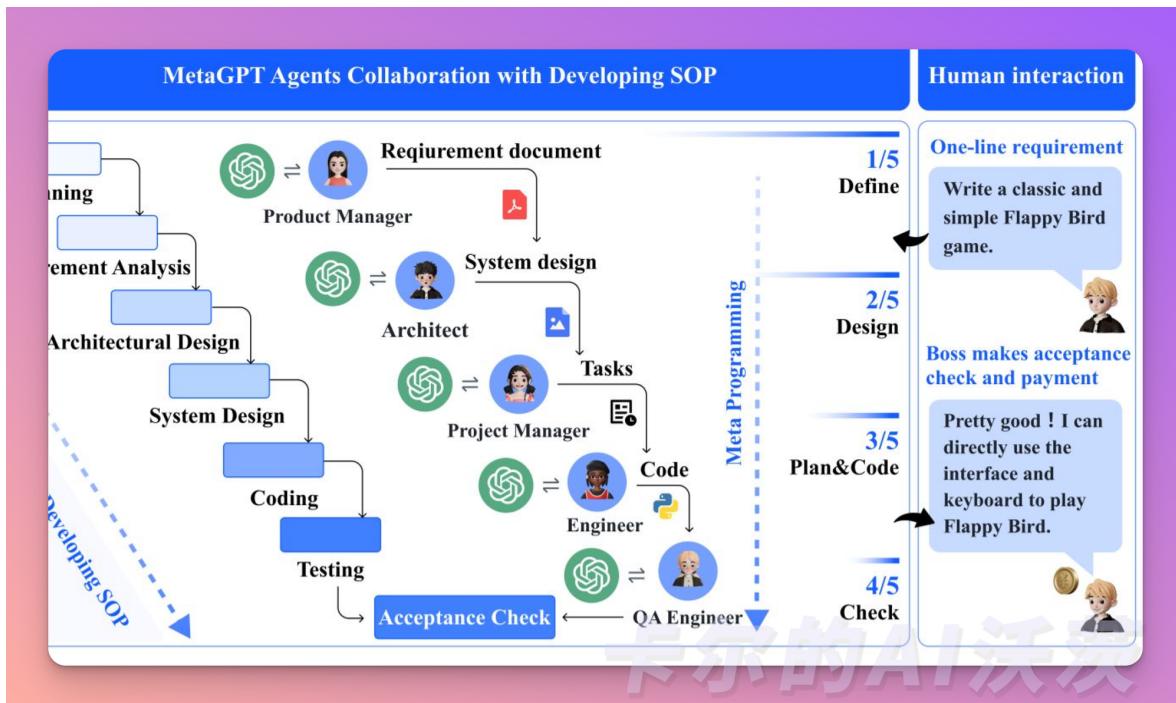
现在MetaGPT引入了标准化操作程序。这些操作程序就像作弊码一样，用于顺利协调工作。它们告诉代理们发生了什么事，以有条不紊的方式指导他们。借助这些操作程序，代理几乎可以像领域专家一样熟悉他们的工作，并验证输出以避免错误。就像高科技流水线一样，每个代理都扮演着独特的角色，共同理解复杂的团队合作。

为什么MetaGPT很重要

在人工智能驱动的解决方案正在成为常态的世界中，MetaGPT 提供了一个全新的视角。这就是它掀起波澜的原因：

- 出众的解决方案：借助SOP，与其他 Agents 相比，MetaGPT 已被证明可以生成更一致和正确的解决方案。
- 多样化的角色分配：为LLM分配不同角色的能力确保了解决问题的全面性。

MetaGPT 软件开发过程

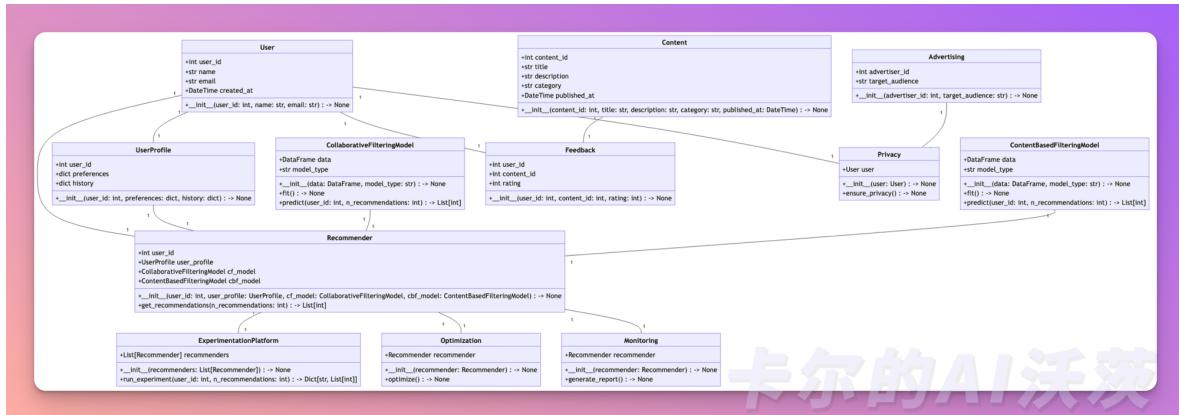


1. 需求分析：收到需求后，该过程开始。这一阶段致力于明确软件所需的功能和要求。
2. 扮演产品经理：产品经理以需求和可行性分析为基础，开启整个流程。他们负责理解需求，并为项目制定明确的方向。
3. 扮演架构师：一旦需求明确，架构师将为项目创建技术设计方案。他们负责构建系统接口设计，确保技术实现符合需求。在MetaGPT中，架构 Agent 可以自动生成系统界面设计，如内容推荐引擎的开发。
4. 扮演项目经理：项目经理使用序列流程图来满足每个需求。他们确保项目按计划前行，每个阶段都得到适时执行。
5. 扮演工程师：工程师负责实际的代码开发。他们使用设计和流程图，将其转化为功能完备的代码。
6. 扮演质量保证（QA）工程师：在开发阶段结束后，QA工程师进行全面的测试。他们确保

软件符合所需标准，不存在任何错误或问题。

Examples

举个例子，当你输入 python startup.py “Design a RecSys like Toutiao”，MetaGPT会为你提供多个输出，其中之一是有关数据和API设计的指导。



生成一个包含分析和设计示例的成本大约为0.2美元（使用GPT-4 API），而完整项目的成本约为2.0美元。通过这种方式，MetaGPT提供了低廉的解决方案，让你能够快速获取所需的信息和指导。

快速体验

目前MetaGPT暂无在线体验版本。这里我会列出docker的安装方法，最大程度减少大家安装面对的环境难度：

将"Write a cli snake game"更换成你喜欢的命令试试吧！

更多安装的教程建议看

下一节我们将介绍AI小镇

Reference

- [MetaGPT: The Multi-Agent Framework](#)
- [MetaGPT: Meta Programming for Multi-Agent Collaborative Framework](#)
- [MetaGPT: The Future of Multi-Agent Collaboration in AI \(A Brief Guide\)](#)