# FlowCabal 技术报告

田照涛. Author*
贵州轻工职业大学, 贵州, 中国

杨森. CoAuthor#
辽宁轨道交通职业学院, 辽宁, 中国

(Drafted 06 February 2026)

FlowCabal 是一款专注于 AI 辅助写作的软件，专门面向高质量长篇写作的场景。软件提供蓝图式工作流编排、高级查询函数、多种工作流控件、内置 agent 助手和官方 agent skill。本文将从产品愿景、关键功能、技术选型、软件架构、交互设计和具体用例等方面介绍这款软件。软件源代码在 Github 开源

## 一、 引言

受限于如今的模型性能，AI 辅助写作很难找到一站式的解决方案，为此，FlowCabal 希望提供一种高度定制化且足够现代的范式来回答这个问题。

当以 ChatGPT 为首的，基于 LLM 的 chat-ai-app 刚刚进入人们视野的时候，第一个被广泛关注的概念是**提示词工程**[1]；而在 2025 年初到 2026 年初的 AI 编程大爆发和 agent 大爆发中[2]，另一个在幕后扮演重要角色的概念是**上下文工程**。这两个概念几乎涵盖了当下 LLM 应用的所有注意事项，而对于 AI 辅助写作这个领域，FlowCabal 提供的范式同样围绕这两个概念展开。

## 二、 关键功能

本节以渐进式的视角审视了 FlowCabal 的功能设计，省略了很多软件实现上的细节，力求以更通用的视角解释清楚 FlowCabal 的原理和优点。

### A. 节点工作流

假如将 LLM 抽象为 `Model(query) = Answer`，那么常见的 chat-ai-app 多是以如下的形式组织 `query` 的：

```
"query": [
    { "role": "user", "content": "你好" },
    { "role": "assistant", "content": "你好! 有什么我
可以帮助你的吗? " },
    { "role": "user", "content": "你是谁" },
  ......
]
```

而节点工作流则是这种形式的超集，Unreal Engine 的蓝图、ComfyUI 或是如今已经成为 agent 领域经典实践的 LangGraph 都采用了这种组织方式。

FlowCabal 的一个节点工作流可以描述为 `[Model]_n`，其中 `Model_k.query = fuc_k(m.Answer | m in M_k)`，`M_k` 是 `[Model]_n` 的一个子集，`fuc_k` 是自定义函数，比如 `[fuc_add(a, b, c) → a + "+" + b + "+" + c]["1", "2", "3"] = "1+2+3"`。

这样的工作流一般来说会有满足 DAG 形式的依赖关系，使用 Kahn 算法排序后即可逐个解析，如果有环状依赖出现，在工程上则可以设置停机收敛条件。

FlowCabal 还提供了不少有趣且实用的功能，这些功能使 FlowCabal 成为节点式工作节点式工作流的超集。在 section II B 和 section II C 中有进一步的描述。

另一个值得一提的现象是节点工作流的调用方式天然带有上下文压缩的色彩，`M_k.query` 完全不会出现在其余任何 `M_p.query` 中。section II D 详细介绍了 FlowCabal 的上下文策略。

### B. 高级查询函数与高级功能

有非常多的研究如 RSA[1] 和 RLM[2] 表明，递归和复杂迭代可以作为 LLM 推理的强力可选项，这些推理过程显然是 section II A 中提到的节点工作流无法涵盖的。FlowCabal 提供了一组高级查询函数 `[adv_fuc]_n`，可以在工作流中使用。

由于 LLM 的不稳定性，工作流的每个节点 `Model_k` 可能都会有多次重试，所以 FlowCabal 会默认保留每个节点的 `Model_k.Answer` 的历史记录，并提供 “冻结” 功能使一个节点在本次工作流运行中永不重试。基于这两点设计，一个 FlowCabal 工作流的运行依然按照 Kahn 序一个一个解析节点——只是对人类介入的需求过于高了。

另一个值得一提的设计是子流程，假如工作流的运行只有人类来介入，这个设计的价值将大打折扣。请看 section II C。

* Contact author: isirin1131@outlook.com
# Contact author: billhoyou@163.com

[1] reddit 的 PromptEngineering 板块创建于 2021 年 2 月 26 日
[2] Anthropic 于 2024 年 11 月 25 日开源了他们的 MCP 协议，可以看作是 agent 时代开始的里程碑事件

### C. Agent 赋能

FlowCabal 的 agent 功能不只是用于编写工作流，还用于监控工作流的运行。这样的 agent 几乎必定需要在整个小说内探索需要的上下文，也因此必然需要引入成熟的上下文管理器从而一站式解决长短期记忆问题。FlowCabal 的选型是字节跳动开源的 OpenViking[3]，其基于虚拟文件系统带来的检索可追踪性和多级摘要的上下文披露无疑是 FlowCabal 所需要的，OpenViking 还内置递归式的上下文检索，这也与 section II B 的主张有所呼应。

### D. 摘要与上下文艺术

“上下文腐化”[3]

## 三、 技术选型

## 四、 软件架构

## 五、 前端交互设计

## 六、 用例和效果评估

## 七、 总结

## 致谢

感谢我的家人、老师和朋友

[1] S. Venkatraman 等, *Recursive Self-Aggregation Unlocks Deep Thinking in Large Language Models*, https://arxiv.org/abs/2509.26626.

[2] A. L. Zhang, T. Kraska, 和 O. Khattab, *Recursive Language Models*, https://arxiv.org/abs/2512.24601.

[3] K. Hong, A. Troynikov, 和 J. Huber, Context Rot: How Increasing Input Tokens Impacts LLM Performance, technical report, 2025, https://research.trychroma.com/context-rot.

[3] https://github.com/volcengine/OpenViking