

TPMA：文本化多智能体流程架构

Zhu Wei - joinwell52 Research Center

2026-08-11 - A1.0 - TPMA V1.0

Contents

TPMA：文本化多智能体流程架构	2
面向中小企业受治理多智能体组织工作的 AI 原生软件架构理论	2
摘要	2
1. 引言	3
1.1 论文类型与研究问题	4
1.2 目标环境与设计约束	4
1.3 设计起源与演化	5
1.4 设计前提与贡献	5
1.5 术语与表示阶段	6
2. 问题定义与设计需求	7
3. 相关工作与定位	8
3.1 架构前身	8
3.2 互操作与治理缺口	8
3.3 直接相邻研究	8
3.4 比较综合与研究缺口	9
4. TPMA 架构	9
4.1 治理对象	9
4.2 文档类型	10
4.3 角色与权限模型	10
4.4 生命周期模型	10
4.5 文本消息、单写者流与异步并行	11
4.6 读端聚合与治理重建	11
4.7 完整性与签名证据	12
4.8 指导关系、历史谱系与运行栈	13
5. 研究设计与评估方法	13
5.1 设计科学过程	13
5.2 证据与声明协议	14
5.3 FCoP 作为参考实例化	14
5.4 案例与语料库程序	14
5.5 证据依赖与分析边界	15
6. 规范重建契约	15
6.1 确定性	15
6.2 完整性、身份与真实性	16
7. 评估结果	16
7.1 按研究问题组织的发现	16

7.2 一致性领域摘要 16

7.3 S1.0 作者运行产品基线 17

7.4 解释：已关闭的实现缺口与仍开放的验证缺口 17

8. 讨论、局限与效度威胁 18

8.1 证据成熟度 18

8.2 SME-first 声明 18

8.3 数字员工 Profile 与隐私 18

8.4 效度威胁 19

8.5 局限与可证伪条件 19

8.6 出版与可复现边界 19

9. 结论 19

工件可用性 20

数据可用性 20

利益冲突与作者生成证据 20

作者贡献 20

伦理与隐私声明 20

References 20

TPMA：文本化多智能体流程架构

面向中小企业受治理多智能体组织工作的 AI 原生软件架构理论

架构论文正式版： A1.0

历史来源基线： TPMA Draft V1.0-R23；理论已对齐至 R31

状态： TPMA V1.0 稳定研究论文正式版

发布日期： 2026-08-11

出版权威： 本 GitHub 文档是 TPMA 架构论文的权威版本；独立维护的 TPMA Core Specification 是规范性来源；Implementation Case Report 仅提供工程证据，不具有规范性。

作者： 朱卫, joinwell52 Research Center

公开通信地址： joinwell52-AI/joinwell52

文档标识： TPMA-ARCH-A1.0

审稿说明： 本公开版本标明作者身份；如投递双盲审稿渠道，必须另行匿名化。

摘要

大语言模型正在从孤立的问答工具转向长期运行、使用工具并由多个智能体共同参与的执行系统。工具调用轨迹与会话历史能够说明发生了什么，却不能仅凭自身确立经过授权的责任归属、合法的生命周期迁移、独立复核或可恢复的治理状态。

本文提出 **TPMA (Textual Multi-Agent Process Architecture, 文本化多智能体流程架构)**：一种面向中小企业、最低基础设施条件的**文本消息多智能体异步流程架构**。其核心由四条相互关联的规则构成：**文本承载持久消息与状态**；**每个写者保持自己的局部串行流**；**多条串行流异步推进并形成并行协作**；**读端聚合现有证据，重建流程、责任、生命周期、冲突与审计状态**。每一项受治理工作由一个稳定的主载体锚定；后续报告、复核、决策

与更正则保持为彼此分离的单写者对象。重建过程保留并发关系与未解决冲突，而不是人为强加一条全局总顺序。

本文研究的 FCoP 是一种项目可见的文件系统 Profile。它不强制要求协调数据库、消息 Broker 或企业级控制平面，但也不会单独提供经过验证的企业身份、强角色隔离、防篡改存储或拜占庭容错。因此，TMPA 的定位是 **SME-first, 而非 SME-only**：规模更大的实现可以通过数据库、对象存储、事件服务、身份系统和控制平面保留同样的治理语义。

在当前出版体系中，**TMPA 理论指导 CodeFlowMu 的工程方向**。Core Specification 把理论转化为规范对象、Reader 行为与一致性准则；FCoP 承载文件型协作协议；CodeFlowMu 则在可运行工程系统中落实受治理角色、工作流、复核、恢复与审计机制。这一指导关系不同于历史反馈关系：FCoP 与 CodeFlowMu 的工程实践也曾反向促进后续理论形式化。

本文遵循成熟的设计科学方法 [34]、[35]：诊断治理状态问题，推导设计需求，构造 TMPA 人工制品，通过 FCoP 协议 Profile 与下游案例进行演示，并同时评估架构不变量与版本锁定的 C01-C14 证据。贡献在于一体化治理架构，而不是新的存储原语、Runtime 或真值 Oracle；评价结论严格受证据范围约束。CodeFlowMu V1.8.0 产品 Reader 针对精确的 TMPA Core S1.0 Bundle 记录 **14 PASS、0 PARTIAL、0 NOT RUN、0 FAIL**，覆盖 71 项强制断言；单独维护的 S1.0 Reference Reader 也通过十四项合成标准。锁定证据归档保留精确输入、产品源码、命令、输出、回归历史与覆盖 889 个文件的完整性 Manifest。这些结果强化了实现可行性，但仍属于作者运行证据；低资源性能、采用成本、比较基线、代表性使用和独立复现仍是开放的实证要求。

关键词： AI 治理、智能体 AI、多智能体系统、中小企业、最低基础设施、文本消息、主载体、单写者流、异步协作、确定性重建、生命周期、角色分离、来源追踪、可审计性、可恢复性、FCoP、CodeFlowMu

1. 引言

大语言模型已经把人工智能从孤立的推理系统转变为能够调用工具、修改文件、查询数据库、操作业务软件，并在长期任务中协作的执行系统。正确输出仍然是必要条件，但真正可部署的系统还必须保留围绕输出形成的权限、责任与证据。

一个受治理的多智能体系统必须回答：谁授权并接受了工作，哪个对象代表该项工作，产生了什么证据，谁进行了复核和决策，状态迁移是否合法，以及流程在中断后能否被重新构建。日志、聊天记录、工作流状态与业务记录都可以构成证据的一部分，但它们不会自动形成权威治理状态。

****TMPA (Textual Multi-Agent Process Architecture) ****针对这一缺口提出解决方案，但它不规定智能体如何思考，也不替代 Agent 框架、身份提供方、运行时网关、传输机制或存储系统。TMPA 通过四条运行陈述定义跨平台的流程—责任契约：

文本承载消息与状态。

每个写者保持自己的串行流。

多条串行流异步推进，形成并行协作。

读者聚合各条流并重建流程与治理状态。

每项受治理任务由一个稳定的主载体锚定。接受、报告、复核、决策、更正和恢复证据都是由各自写者独立编写、通过显式引用连接的对象。写入侧保持局部串行与单写者；系统整体异步并行；读取侧则重建偏序图和问题集合。

本文使用三个定位视图：

历史协同演进
业务实践 → 早期 TMPA 方法 → FCoP 抽取与成熟
→ CodeFlowMu 工程落实
FCoP + CodeFlowMu 工程反馈 → 当前 TMPA 形式化

当前指导与落实关系
TMPA 理论与架构
↓ 由下列文档形式化为规范行为
TMPA Core Specification
↓ 通过文件型协作 Profile 投影
FCoP 协议

↓ 用于在下列系统中落实受治理工作
CodeFlowMu 工程系统

端到端流程
写入：主载体 → 单写者流 → 异步组合
读取：来源聚合 → 治理 Reader → 流程图 + 问题集合

图 1. TMPA 定位图：历史谱系、当前分层与运行重建。

谱系视图解释来源与反馈，指导关系解释当前权威与工程方向，流程视图解释系统如何运行。历史反馈不会倒置当前权威：FCoP 不穷尽 TMPA，CodeFlowMu 不定义 FCoP 或 TMPA，2026 年 3 月的早期 Pipeline 也不代表已经满足当前 Core Specification。

第 8.3 节另行说明的一项可选应用场景，是一种跨会话持续存在、业界有时称为**数字员工**的 AI 工作角色。本文使用该名称时，仅指能够接受委托工作、使用工具并跨会话提交结果的工程工作身份；它不意味着法律雇佣关系、人格、意识、人类意图，也不意味着替代承担责任的人类或组织主体。

1.1 论文类型与研究问题

本文是一项设计科学与系统架构研究。被设计的人工制品是 TMPA；配套 Core Specification 定义其规范行为，本文负责解释问题、理论、设计逻辑与评价。主要环境是中小企业或小型团队：治理必须能够在没有专用 Agent 平台、协调数据库、消息 Broker、企业身份平面和专业运维团队的情况下开始。FCoP 提供协议 Profile，CodeFlowMu 则是在 TMPA 指导下开发、并作为有界实现接受评估的工程系统。本文不声称已有代表性基准、生产规模验证，或优于企业治理平台。

本文回答三个研究问题：

- **RQ1——治理状态充分性：**当聊天记录、共享目录、执行轨迹和普通任务状态被用作多智能体组织工作的记录时，缺少哪些信息？这些缺失为何会阻止权威责任判断与恢复？
- **RQ2——最低架构：**在不强制协调数据库、Broker 或控制平面的情况下，为了重建受治理多智能体工作，最低需要哪些与基底无关的对象、权限关系、生命周期规则、顺序约束、冲突语义和读取侧操作？
- **RQ3——工程可行性与边界：**FCoP Profile、锁定的 CodeFlowMu 产品证据、WP-13 与 C01–C14 语料库在多大程度上演示了这些性质？哪些可行性声明仍未获得证据支持？

A1.0 通过证据缺口分析回答 RQ1，通过 TMPA 的对象、流、权限、生命周期与重建模型回答 RQ2。RQ3 得到更强但仍有边界的回答：CodeFlowMu V1.8.0 针对精确 S1.0 输入调用自身产品 Reader，记录产品级 14/14 PASS 与 71 项强制断言；锁定证据包使输入、源码、运行记录与完整性轨迹可检查 [28]。该结果只演示固定 Bundle 下的行为，不代表独立采用或认证。WP-13 案例另行说明，完成的 Agent 轨迹不会自动成为可准入的治理证据 [36]。量化安装负担、低资源性能、更广泛故障恢复、比较基线、代表性使用和第三方复现仍未完成。

表 1. 研究声明、支持证据与禁止推断。

声明	本研究中的最强支持	不允许的推断
普通执行记录不足以支持治理重建 TMPA 对象—流—Reader 模型内 部一致	问题分析、DR1–DR8、失败与反例推理 显式不变量、生命周期与权限模型、确定性证明概要、Core S1.0	所有聊天、工作流或事件系统必然失败 已经形成关于最低性的普适数学证明
文件型协议与 TMPA 指导的工程系 统可以实现重要子集	FCoP 映射与 CodeFlowMu V1.8.0 精确输入产品证据	固定 S1.0 Bundle 以外的一致性、从 Package 测试推导协议有效性，或独立 采用
受治理多智能体案例可以保留并审 查有争议的完成声明	CodeFlowMu 与 WP-13 证据链	消除幻觉、因果性能改善或独立采用

本矩阵是全文贡献声明的控制性解释。后文工程细节不得把任何声明扩大到相应证据与边界之外。

1.2 目标环境与设计约束

主要目标环境是需要受治理 AI 协作、但缺少以下部分或全部能力的中小企业或小型团队：专用 Agent 平台、协调数据库、消息 Broker、企业 Agent 身份基础设施和专业运维人员。这里的范围以能力约束定义，而不是仅按员工数量定义。

2026 年 OECD D4SME 调查覆盖 12 个 OECD 国家、2,000 多家中小企业，结果显示战略性且安全的 AI 集成仍不均衡，时间、维护成本与技能缺口依然是实质性障碍 [23]。新加坡 IMDA 报告显示，2024 年中小企业 AI 采用率

为 14.5%，而非中小企业为 62.5% [24]。这些来源并不能证明市场对 TMPA 本身的需求；它们支持的是更广泛的问题背景：在组织能力有限时如何负责任地集成 AI。

这些设计约束推导出如下链条：

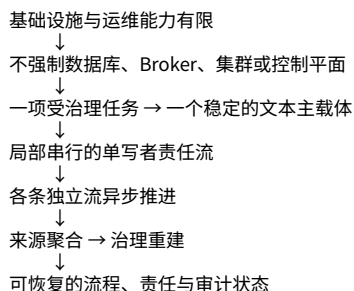


图 2．最低基础设施设计约束的推导。

文本承载、局部串行、异步并行、确定性重建。

主要 Profile 是项目本地异步协作，而不是分布式共识或多数据中心执行。参与者不必同时在线，证据可以延迟到达；Reader 根据当前可用集合暴露 authoritative、partial、disputed 或 quarantined 视图。数据库支持和企业级 Profile 仍然可能，但身份联合、复制、高可用和跨信任域强制执行不属于当前验证范围。

1.3 设计起源与演化

作者的私有小典项目档案称，TMPA 的早期形式于 2026 年 3 月的一份多角色架构规划中以 **Text-Message Multi-AI Parallel Architecture** 名称出现 [25]。由于该来源没有固定公开快照，本文只把它作为作者报告的设计谱系，不纳入评估语料、研究问题结果或一致性声明。

据作者说明，同一私有来源还记录了一个过渡性的 Pipeline 设计 [25]。该谱系材料不作为投稿证据，也不建立今天的 Core 一致性。不可变写者流、保留来源的聚合、确定性重建、显式冲突状态与一致性要求均在后来发展，并只通过公开锁定工件接受评估。

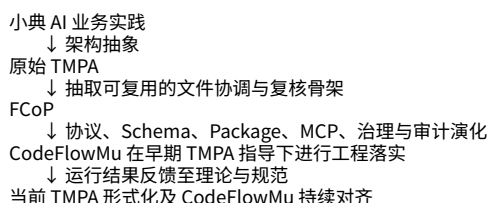


图 3．从业务实践到当前 TMPA 形式化的设计谱系。

实践揭示问题；重复工程发现方法；形式化将方法升华为理论。

本文重新统一了原始的消息与异步模型，以及通过 FCoP 和 CodeFlowMu 工程实践成熟的治理语义。当前权威方向仍然明确：现行 TMPA 理论与 Core 要求指导 CodeFlowMu 持续工程落实，工程结果则作为证据和修订输入反馈给理论。这样既避免追溯性一致性声明，也避免把工程实现误写成理论定义者。

1.4 设计前提与贡献

TMPA 建立在四项前提之上：

1. **文本化 (Textual)**：治理语义具有可被人、AI 系统与验证器读取的规范表示。
2. **多智能体 (Multi-Agent)**：执行、复核、批准与监督保持对相互分离的权威主体可归属。
3. **流程 (Process)**：治理覆盖完整生命周期，包括拒绝、更正、恢复与归档。
4. **架构 (Architecture)**：这些语义跨模型、Runtime、语言、传输与存储 Profile 保持稳定。

这些前提产生四项运行承诺：

1. 持久文本消息与状态载体；
2. 每项工作一个稳定主载体，每个已发布对象一个写者；

- 3. 局部串行与多流异步并行；
- 4. 保留来源的聚合，以及随后的确定性治理重建。

作者保持独立，读者重建整体。

TMPA 不声称单独发明了只增历史、生命周期状态机、来源追踪、签名或基于角色的授权。其贡献在于把这些机制集成为最低基础设施流程架构，并使任务身份、责任顺序、生命周期合法性、复核分离、冲突保留、恢复和可机器测试的 Reader 行为成为显式结构。

本文提出四项贡献：

- 1. **受治理工作对象模型。** 将稳定主载体与可独立归属的报告、复核、决策、迁移、更正和恢复证据分离。
- 2. **多串行组织架构。** 组合局部有序的单写者责任流而不强制全局顺序，再从其并集重建偏序工作图。
- 3. **确定性治理契约。** 将保留来源的聚合，与显式权限、生命周期、冲突、三值判断和规范问题重建结合。
- 4. **证据边界明确的 SME-first 评估。** 区分 TMPA 理论、Core 要求、FCoP 协议、参考实现、受 TMPA 指导的 CodeFlowMu 工程系统与案例证据；公布包括失败在内的 C01-C14 结果；指出可行性和采用声明仍需完成的实证工作。

本文不贡献新的存储原语、Agent 通信协议、Runtime 编排器、身份提供方、事实核验方法或生产率实证证明。FCoP 及锁定的 CodeFlowMu/WP-13 工件是有界证据来源；小典仅是作者报告的谱系。CodeFlowMu 在 TMPA 指导下进行工程落实，但其实现和固定 Bundle 通过结果都不定义或证明 TMPA 理论。

当前 TMPA—FCoP—CodeFlowMu 的关系与运行软件栈在第 4.8 节规定；术语在第 1.5 节固定。

1.5 术语与表示阶段

本文固定以下词汇，以避免把语义对象、物理存储、消息行为与重建视图混为一谈。

表 2．规范术语及其排除的等同关系。

规范英文术语	固定中文译名	固定含义	不等于
governed work item	受治理工作项	其责任与生命周期受到治理的任务、请求、决策或流程主题	一个文件、一个 Session 或一个 Runtime Job
primary carrier	主载体	锚定一项工作标识符和最低治理上下文的稳定治理对象	所有参与者共同编辑的可变记录
governance object	治理对象	由一个创建者、以一个责任角色、在一条写者流中编写的规范语义单元	其存储路径、传输封装或派生视图
textual message	文本消息	治理对象在传递工作、证据、复核或决策语义时承担的通信功能	单独的对象类别或短暂队列消息
state carrier	状态载体	对象、迁移记录或 Profile 规定位置对声明状态或当前状态证据作出贡献的持久化功能	共享可变应用状态
source artifact	来源工件	证据的一种物理表示或观测，例如文件、数据库行、对象存储项或收到的事件	验证后的语义治理对象
source candidate	来源候选	提交给聚合阶段的已发现来源工件，包括格式错误或互相冲突的观测聚合返回的、保留来源、完成解析、索引与确定性规范化的集合	已接受的权威对象
canonical candidate set	规范候选集合	一个可归属写者发布的、局部有序的治理对象序列	最终治理结论
writer stream	写者流	发现、保留、解析、索引与规范化来源候选，但不决定治理真值的阶段	全局 Event Log 或总时间线
source aggregator	来源聚合器	把固定 Profile 应用于规范候选集合的确定性阶段	治理 Reader
governance reader	治理 Reader	重建得到的偏序流程视图，以及规范化的未解决条件输出	存储层、编排器或模型 Runtime
governance graph and issue set	治理图与问题集合		原始来源证据或人为强加的总顺序

同一个规范治理对象可以通过不同物理 Profile 实现。在 FCoP 中，来源工件通常是文件及其路径与事件证据；其他 Profile 可以使用数据库行、对象或事件。反过来，两个来源工件若声明相同对象 ID、却包含不同的规范内容，它们不是两个无害副本，而是必须被保留并根据 Profile 评估的冲突候选。在架构与规范章节中，**对象（object）**指语义单元，**工件（artifact）**指物理或已发布的工程表示，**视图（view）**指 Reader 派生的结果。

2. 问题定义与设计需求

TMPA 从区分**执行证据**与**治理证据**开始。运行轨迹可以证明某次工具调用发生过，却不一定能证明调用者获得授权、某个承担责任的角色接受了工作、输出经过独立复核，或者后续批准所引用的正是那份已经复核的结果。聊天记录可以保留讨论，却仍可能缺少稳定对象身份、生命周期合法性、冲突处理与确定性重建。工作流引擎可以记录节点完成，但把权威状态保存在实现专用数据库中。

因此，该架构区分四类状态：

表 3. 执行、交互、业务与治理状态的分离。

状态类别	主要问题
执行状态	Runtime 正在做什么？
交互状态	参与者交换了什么？
业务状态	应用当前把什么视为事实？
治理状态	哪些责任、迁移、决策、冲突与证据具有权威性？

TMPA 规范第四类状态，同时允许引用前三类状态。

文本被选为规范互换形式，是因为人、语言模型、验证器、版本控制工具、备份系统与替代 Runtime 都能够解释它。“文本化”不要求必须采用 Markdown 文件：符合规范的部署可以使用数据库、对象存储或事件服务，只要完整治理含义具有规范文本表示。

责任分离同样是核心。一套名义上的多智能体系统，如果由同一个身份规划、执行、复核、批准并认证同一项工作，就没有建立独立治理。因此，TMPA 把角色视为具有范围的权威，而不是 Prompt 标签。生命周期也必须显式表达：受治理工作沿 Profile 定义的状态推进；合法迁移、迁移权限、被拒绝的迁移、返工和终态历史都必须保持可观察。

最低基础设施问题是：

在不预设专用协调数据库、消息 Broker、企业控制平面或专业治理团队的情况下，小型组织如何获得可归属、可复核、可恢复且可由机器检查的多智能体治理？

答案不是孤立地“用文件代替数据库”，而是一套完整流程结构：规范文本承载消息与状态；一个稳定主载体标识每项受治理工作；每个写者通过局部串行流发布；各条独立流异步推进；Reader 重建流程图与问题集合。

目标是最低基础设施，而不是零纪律。受保护的存储、声明的身份假设、备份、权限、验证与恢复程序仍然必要。OECD 与 IMDA 的证据支持一个更广泛的观察：中小企业长期面对时间、技能、维护与采纳能力约束；这些资料并不证明对 TMPA 本身存在需求 [23]、[24]。

该问题产生八项可追踪需求：

表 4. TMPA 设计需求。

ID	需求
DR1	持久的规范文本表示
DR2	稳定工作身份与单写者证据
DR3	局部有序流与异步组合
DR4	显式权限、复核分离与生命周期合法性
DR5	保留来源的聚合与确定性重建
DR6	保留冲突、无效证据与部分状态
DR7	从持久治理证据恢复
DR8	最低强制基础设施与显式保证边界

DR1–DR7 定义治理语义。DR8 约束部署声明：TMPA 尽量降低协调基础设施要求，但不会继承实际未部署的身份、安全、共识或控制平面系统所提供的保证。

3. 相关工作与定位

TMPA 位于 Agent 执行层与企业治理层之间的流程—责任层。MCP 与 A2A 解决工具使用和 Agent 互操作 [1]、[2]；身份系统建立主体与委托权限 [19]；网关与策略引擎负责运行时准入；可观测性与控制平面产品盘点 Agent 并收集遥测。TMPA 不替代这些层，而是接收它们提供的标识符、策略决策、轨迹与工件，并把受治理工作、复核、拒绝、恢复与责任表示为可重建形式。

3.1 架构前身

Event Sourcing 与 CQRS 提供面向追加的历史，以及写入表示与读取模型的分离 [4]。Git 展示了不可变的内容寻址对象和显式历史 [5]。W3C PROV 提供实体、活动、Agent 与派生关系 [3]。Lamport 建立了不依赖单一物理时钟的 happened-before 顺序 [32]；Chandy 与 Lamport 说明了如何确定分布式系统的一致全局状态 [33]。TMPA 不把这些基础主张为自身创新，而是将偏序推理和可重建全局视图用于更窄的治理问题：每项工作一个主载体、单写者责任流、显式生命周期权限、职责分离、冲突保留，以及确定性治理 Reader。

该架构不等于聊天记录、共享目录、ADR 集合、工作流引擎或单一全局 Event Log。这些机制可以保存相关证据，但它们本身不定义哪个对象具有权威性、某项行为是否在授权范围内、复核是否独立，以及互相矛盾的证据应如何保持可见。

3.2 互操作与治理缺口

Agentic AI Foundation 的成立反映了围绕 MCP 及相关项目的互操作基础设施正在增长 [6]。但互操作并不等同于治理。Kang 与 Diponegoro 使用成员资格、审议、投票、异议保留、人工升级和审计/回放等需求，对 MCP、A2A、ACP、ANP 和 ERC-8004 进行分析，发现所调查协议均未表达完整治理模型 [27]。TMPA 处理其中更窄的子集——工作责任、生命周期、复核、冲突与恢复——而不是完整的社群治理。

《Open Challenges in Multi-Agent Security》强调，即使单个组件看似安全，共谋、级联效应、隐蔽行为与监督失败仍可能在交互层出现 [26]。TMPA 可以保留可归属证据和未解决分歧以供调查，但它不是共谋检测器。

3.3 直接相邻研究

Auditable Agents 区分 accountability、auditability 与 auditing，并从行为可恢复性、生命周期覆盖、策略可检查性、责任归属和证据完整性五个维度进行评估 [17]。TMPA 与其互补：TMPA 规定持久工作对象与流程重建，使这些维度能够被评估。

IETF Agent 审计架构同样把委托与交互视为可审计事件 [18]。TMPA 可以为这些事件提供项目本地或平台中立表示，但不标准化网络传输。

Authorization Propagation in Multi-Agent AI Systems 把传递式委托、权限聚合推断与时间有效性识别为尚未解决的授权问题 [20]。TMPA 记录任务分配和责任迁移，但不定义完整的递归委托演算。

Policies on Paths 主张运行时治理可能取决于部分执行路径，而不是静态访问规则 [21]。TMPA 把流程路径保留为证据，但自身不负责在执行期间阻断行为。

Proof-Carrying Agent Actions 把高价值动作与决策时证书、批准和可回放证明绑定 [29]。PCAA 以动作决策为中心；TMPA 以更长期的受治理工作项及其报告、复核、冲突、更正和恢复为中心。TMPA Profile 可以引用 PCAA 证书作为执行证据。

AGENTS SAFE 结合风险分类、语义遥测、动态授权、可中断性、异常检测、密码学追踪和组织控制 [30]。TMPA 选择不同的最低基线：可读的规范文本与确定性重建是强制要求，而密码学身份与更强完整性控制属于可选的命名 Profile。

Why Do Multi-Agent LLM Systems Fail? 从大规模轨迹语料中归纳失败类别 [22]。TMPA 改善责任、复核、冲突和恢复的可观察性，但本文不声称已降低失败率。

From Trajectories to Evidence 指出，完成的研究 Agent 轨迹必须经过资格审查，才会成为可审计实验记录 [36]。TMPA 从流程治理角度到达同一边界：执行声明只有经过可归属对象、必需证据、独立复核、生命周期合法决策和例外保留后，才可被准入。TMPA 不提供领域真值核验；它治理声明及其核验证据如何被准入和重建。

因此，增量贡献不是新的存储原语，也不是完整控制平面，而是把以下结构组合起来：持久文本消息/状态平面、一任务一主载体、局部单写者流、异步组合、保留来源的聚合，以及最低基础设施 Profile 下的确定性读端治理重建。

3.4 比较综合与研究缺口

相关工作可以归入五条相邻研究线。它们的边界共同说明 TMPA 所处理的缺口：

表 5. 与相邻研究线的比较定位。

研究线	主要贡献	与 TMPA 的关系及边界
MCP 与 A2A [1]、[2]	面向工具或 Agent 交互的 Context、Capability、Task 与消息互操作交换	TMPA 可以引用这些交互，但定义持续时间更长的责任、复核、冲突与恢复证据
W3C PROV、Event Sourcing 与 CQRS [3]、[4]	派生、面向追加的事件历史与读取模型构造	TMPA 把这些机制专门化为受治理工作身份、权限、生命周期合法性和确定性问题重建
Auditable Agents 与 IETF 审计架构 [17]、[18]	问责维度、分布式审计记录、上下文与事后调查	TMPA 提供基底中立的受治理工作图，但不定义网络审计上下文传播或证明机制
授权传播、路径策略与携证动作 [20]、[21]、[29]	面向委托路径或带证书动作的决策时授权与运行时强制	TMPA 保留已授权工作过程及其结果，不替代执行时中介
NIST AI RMF 与 AGENTS SAFE [30]、[31]	组织风险识别、控制、监测、保证与问责	TMPA 是可以支持这些计划的较窄证据架构，不构成完整风险管理框架
分布式顺序与快照 [32]、[33]	无全局时钟的因果关系与一致全局状态观测	TMPA 将偏序重建专门用于治理证据；不提供共识或分布式快照传输
多智能体失败与证据资格审查 [22]、[36]	失败分类与执行轨迹到可审计记录的转换	TMPA 规定准入、复核、生命周期和重建语义，但不核验领域真值

在本文审查的相邻研究中，没有一项把以下全部性质合并为同一个最低基础设施流程契约：稳定主载体、单写者责任流、不强制总顺序的异步组合、显式权限与生命周期语义、无效和冲突证据保留，以及治理图与问题集合的确定性重建。这是 A1.0 主张的特定研究缺口。该主张是架构性与比较性的，不是对所有未公开或专有系统的优先权声明。

4. TMPA 架构

TMPA 定义治理语义，而不是一个 Runtime 组件。其架构规定哪些治理事实必须被表示、责任与生命周期如何表达，以及如何把独立证据重建为权威视图。除非某个 TMPA Profile 明确绑定，否则存储、传输、调度和模型行为仍属于实现问题。

4.1 治理对象

治理对象 (governance object) 是 TMPA 中最小的、能够独立归属责任的语义单元。它可以表示任务、报告、复核、批准、问题、生命周期迁移、角色分配、恢复动作，或协议 Profile 发布的其他文档类型。治理对象不同于其物理来源工件 (source artifact)：FCoP 可以把它编码为文件以及路径观测，其他 Profile 可以采用数据库行、对象存储项或事件。

文本消息和状态载体描述的是对象承担的功能，而不是额外对象类别。对象在传递受治理工作或证据时承担文本消息功能；当其内容、迁移证据或 Profile 规定的位置参与生命周期重建时，则承担状态载体功能。

每个治理对象包含或标识：

- 稳定对象 ID；

- 文档类型；
- 一个创建者身份；
- 一个责任角色；
- 流 ID 与序号；
- 创建时间；
- 生命周期 Profile 与声明状态；
- 指向相关对象的类型化引用；
- 规范文本内容；
- 完整性证据。

已发布治理对象不可变。更正不会删除或重写原对象，而是创建一个新的对象，以取代、拒绝、限定或解决先前对象。多个字节完全相同的来源观测可以指向同一个对象而不改变其含义；同一 ID 对应不同规范内容则构成冲突，而不是更新。

对象声明的生命周期状态，是该对象发布时按照其 Profile 关联的状态。受治理工作的当前权威状态从有效对象集合、已接受迁移和 Profile 规则中重建，而不是通过修改先前对象或选择时间戳最新的对象获得。

4.2 文档类型

TMPA Core 不强制统一的业务文档分类。每个实现 Profile 发布一个有限的文档类型注册表。

每条注册项定义：

- 类型名称与版本；
- 该类型代表的治理责任；
- 允许的创建者角色；
- 必填字段；
- 允许的引用关系；
- 适用生命周期 Profile；
- 验证规则。

不同文档类型的权威含义不得产生歧义重叠。例如，执行报告不会自动同时成为自身的独立复核或批准。当部署允许职责分离例外时，例外本身及其批准权威必须被显式表示。

4.3 角色与权限模型

TMPA 角色是在明确范围内具有治理权限的权威主体，而不只是 Prompt 标签或自然语言 Persona。

每项角色定义包括：

- 稳定角色 ID；
- 允许创建的对象类型；
- 允许执行的生命周期动作；
- 职责分离约束；
- 分配该角色的权威；
- 角色分配的有效期与撤销状态。

对象中的 role 字段声明创建者以何种权威行动，但该字段不会自行创造权限。Reader 必须根据有效角色分配和适用策略 Profile 验证该声明。

只有实现 Profile 明确允许时，一个参与者才能同时占据多个角色。声称具备独立复核的部署，必须禁止同一身份对同一受治理结果同时担任执行者和复核者，除非存在经过记录且获得授权的例外。

在企业身份 Profile 中，逻辑角色还需分别绑定到：可验证的 Agent 或工作负载身份、继续承担责任的人类或组织主体、此次动作使用的凭证、委托来源与范围，以及有效或撤销状态。TMPA Core 记录并验证治理声明，但不签发凭证，也不强制执行递归权限衰减。

4.4 生命周期模型

一个生命周期 Profile 包含：

- 有限状态集合 S ;
- 初始状态 s_0 ;
- 终止状态集合 F ;
- 动作集合 A ;
- 迁移关系 $T \subseteq S \times A \times S$;
- 授权函数 $\text{Auth}(\text{role}, \text{action}, \text{object})$;
- 验证函数 $\text{Valid}(\text{object}, \text{transition})$ 。

只有满足以下条件，迁移才会被接受：

1. 来源状态与当前权威状态一致；
2. T 定义了该迁移；
3. 发起角色获得授权；
4. 所需引用与前置条件已经满足；
5. 迁移证据通过 Schema 与完整性验证。

非法或未经授权的迁移不会改变权威生命周期状态。尝试本身必须通过拒绝、ISSUE、告警或 Profile 规定的等价记录保持可观察，而不能被静默丢弃或修复。

4.5 文本消息、单写者流与异步并行

TMPA 的写入平面把稳定工作载体、单写者对象、局部串行与异步组合结合起来。

对于每个受治理任务或工作项 t ，面向任务的 Profile 定义一个稳定主载体 c_t 。主载体建立工作 ID 与最低治理上下文。接受、执行报告、复核、决策、更正和恢复记录都是引用 c_t 的独立对象，而不是任务的更多可变副本。因此，“一任务一载体”指一个稳定主引用点，而不是所有参与者共同编辑的一份文档。

令 A 为责任写者集合。每个已发布对象恰好有一个写者，每个写者 $a \in A$ 发布一条可独立归属的串行流：

$$S_a = \langle o_{\{a,1\}}, o_{\{a,2\}}, \dots, o_{\{a,n\}} \rangle$$

S_a 内部序列是该写者的权威局部顺序。每个对象具有正整数序号， $(\text{stream_id}, \text{sequence})$ 标识其在该写者责任历史中的位置。创建时间可以提供信息，但不是流顺序的权威依据。

在观测时刻 τ ，可用候选集合可能包含各条流的不同前缀：

$$O_\tau = \text{union}_{\{a \in A\}} \text{prefix}(S_a, k_a(\tau))$$

函数 $k_a(\tau)$ 不必同步推进。一个参与者可以先发布任务而另一个仍离线；报告可以先于独立复核出现；多个写者可以并发推进。这就是多条串行流形成异步并行的方式。TMPA 不要求所有参与者共享时钟、同时在线，或共同写入一条全局 Event Log。

流内前驱关系提供局部顺序。显式引用和 Profile 定义的生命周期依赖提供跨流因果边。如果两个对象既没有流内关系，也没有 Profile 定义的跨流依赖，它们保持并发且不可比较。

单写者对象与分离的责任流消除了主要的**语义共享写冲突**：多个 Agent 不再争抢修改一份权威记录。但它们不会消除所有存储层争用、文件系统竞态或基础设施故障；Profile 仍必须定义原子发布、重复处理和恢复行为。

写入模型可以概括为：

一项任务有一个主载体。每个已发布对象由一个写者拥有。每个写者内部保持串行。多条流异步推进，形成并行协作。

4.6 读端聚合与治理重建

TMPA 的读取平面包含两个概念上分离的阶段：**来源聚合与治理重建**。

令 O_τ 为观测时刻 τ 可见的有限来源候选集合。保留来源的聚合器 A 发现来源工件，保留来源身份和字节，解析候选封装，对 ID 与引用建立索引，并应用 Reader 所需的确定性规范化：

$$C_\tau = A(O_\tau)$$

聚合阶段不决定哪项声明为真，不静默修复冲突，不虚构缺失证据，也不把到达顺序转换为治理顺序。其目的，是构建治理 Reader 当前可用的完整规范候选集合 C_τ ，同时保留每个来源候选的来源信息。

令 P 为固定规则 Profile，其中包含 Schema、类型规则、角色分配、生命周期规则、关系语义、规范化规则、冲突策略和输出规范化规则。治理 Reader 计算：

$$R_P(C_\tau) = (G_\tau, I_\tau)$$

其中， G_τ 是规范重建得到的偏序流程与治理图， I_τ 是规范问题集合。 G_τ 表示 workflow 进度、责任、生命周期、复核、批准、拒绝、恢复与审计关系，同时保留局部流顺序、显式跨流依赖以及不可比较对象之间的并发。它不是权威总时间线。

为简化表述，后文可用 $R_P(O)$ 表示组合管线 $R_P(A(O))$ 。主要确定性要求是排列不变性。对同一规范候选集合的任意排列 π ：

$$R_P(A(\pi(O))) = R_P(A(O))$$

相等性适用于 G 与 I 的规范序列化结果，而不是偶然的内存顺序或诊断格式。延迟到达会改变当前可用集合，并可能合法地把视图从 partial 改为 authoritative 或 disputed；但同一集合的不同枚举顺序不得改变结果。

确定性命题。 令 O 为有限来源多重集合， P 为固定 Profile。假定：(1) 来源规范化是来源身份与被覆盖字节的纯函数；(2) 重复分类、验证、权限检查、生命周期检查与问题 ID 都是规范对象值和 P 的函数；(3) 图边仅由流内序号和 Profile 声明关系导出；(4) 图与问题序列化使用已发布的确定性排序及 Tie-break 规则。则组合运算 $R_P(A(O))$ 对来源枚举顺序不变。

证明概要。 因为索引与重复分类依赖规范来源值而非发现位置，所以聚合会把 O 的任意枚举映射为同一规范索引候选多重集合。逐对象验证与顺序无关。集合级检查——重复 ID、流缺口、缺失引用、禁止环、权限冲突和生命周期冲突——都在同一规范集合与关系上计算。因此，每次排列得到的已接受节点集合与有向边集合相同。规范问题 ID 与确定性排序产生相同 I ；规范图序列化以及仅用于表示不可比较节点的稳定 Tie-break 产生相同序列化 G 。所以每种排列都产生字节等价的规范输出。该论证只在所述 Profile 契约下证明排列不变性；它不证明语义真实性、Profile 正确性、对受损信任根的抵抗，或不同证据集合之间的相等。更强保证仍需机械化证明和可执行语料库。

重建主体或子图首先获得三值治理判断，再映射为运行视图：

- **authoritative**：通常对应 valid，所需证据有效，且没有影响结论的未解决完整性、权限、生命周期、顺序或必需引用问题；
- **partial**：对应因所需证据缺失、流或依赖不完整而形成的 undetermined；
- **disputed**：对应多个有效但不兼容的治理声明尚未解决时形成的 undetermined；
- **quarantined**：Profile 定义的完整性、身份、重复 ID 或禁止环条件使相关证据被排除在权威重建之外；它不自动证明底层业务声明为假。

认证是正交的保证标签。对象或视图可以在 TMPA Core 下结构性 authoritative，但在更强身份 Profile 下仍未认证；此时不得把它表述为具有认证完整性。

TMPA 要求**冲突保留**：相互矛盾但有效的对象必须保持在表示中，直至出现新的授权解决对象。它还要求**扩展时证据保留**：增加候选证据不会擦除先来源证据。TMPA 不假定治理状态在任意集合扩展下单调，因为新加入的有效证据可能合法地把先前 authoritative 的视图改变为 partial、disputed 或 quarantined。

因此，读端重建不是完整架构本身，而是把持久文本消息和异步串行流转换为一致流程与治理结果的阶段。系统可以为互换或展示生成确定性拓扑序列化，但该序列化不会在不可比较节点之间增加治理顺序。C11 运行化来源聚合与重建确定性；C03、C04、C09、C10 与 C12 测试身份、顺序、依赖、环和冲突保留行为。完整重建性质的机械化证明仍属于未来工作。

4.7 完整性与签名证据

TMPA 区分三种常被混淆的性质：

1. **归属 (Attribution)**：对象声明创建者和责任角色。
2. **完整性 (Integrity)**：已发布对象被修改时能够检测。
3. **认证完整性 (Authenticated integrity)**：对象通过密码学方式绑定到已验证身份或密钥。

TMPA Core 要求归属与完整性证据。只有应用可信身份、签名和密钥管理 Profile 时，部署才能声称具备认证完整性。

完整性记录标识：

- 规范化 Profile;
- Hash 算法;
- 内容摘要;
- Profile 要求时的前驱或引用摘要;
- 可选签名算法;
- 可选密钥 ID;
- 可选签名值。

角色标签不是密码学签名。没有可信身份绑定的摘要可以检测修改，却不能证明谁创建了对象。有效签名在部署的信任模型下证明来源与完整性，但不证明被签名内容在语义上为真。

4.8 指导关系、历史谱系与运行栈

第 1 节定位图区分历史谱系、当前指导与落实关系和端到端运行。本节固定软件边界：

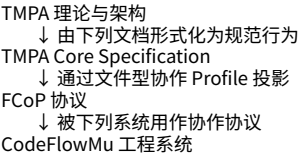


图 4. TPMa、Core、FCoP 与 CodeFlowMu 的当前指导和落实关系。

TPMa 理论提供架构方向，Core 把要求固定为规范行为，FCoP 提供独立定义的协作协议，CodeFlowMu 则使用 FCoP 落实 TPMa 指导工作的可运行工程系统 [37]。FCoP 与 CodeFlowMu 都不定义更广泛的 TPMa 架构，固定 Bundle 下的产品结果也不代表普遍一致性。

FCoP 运行栈为：

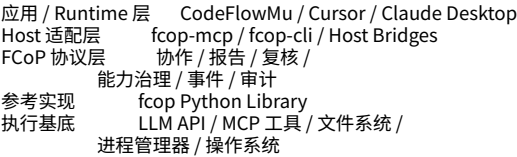


图 5. FCoP 运行栈及其与 TPMa 理论的分离。

TPMa 不是该栈中的 Runtime 组件，而是指导该栈所要落实之治理语义的理论与架构。fcop Package 是 FCoP 的参考实现；CodeFlowMu 是范围更广的工程与运行环境。该分离同时保持指导权威、历史准确性与一致性边界。

5. 研究设计与评估方法

本文是一项设计科学架构研究，遵循 Hevner 等人的人工制品中心指导，以及 Peffers 等人从问题到传播的方法序列 [34]、[35]。研究构造治理工件，分析其内部性质，通过参考 Profile 实例化该工件，并评估边界明确的工程证据。本文不是因果试验、性能基准或生态采用声明。

5.1 设计科学过程

研究由六个相互衔接的阶段组成：

表 6. 设计科学过程与研究问题追踪。

阶段	研究操作	输出与研究问题
问题诊断	分析聊天、日志、工作流与小典谱系中的协调缺口	DR1-DR8; RQ1
解决方案目标	推导最低基础设施、责任、生命周期、冲突和重建需求	DR1-DR8; RQ1-RQ2
工件构造	定义对象、流、权限、生命周期与重建规则	TPMa 架构; RQ2
演示	将 FCoP 协议及锁定的、受 TPMa 指导的 CodeFlowMu 与 WP-13 路径追踪到 TPMa 概念	实现可行性; RQ3; 小典谱系不纳入评估

阶段	研究操作	输出与研究问题
评估传播	检查不变量和反例；执行 Reference Reader 与产品证据矩阵 发布版本化论文、规范、实现报告、Fixture 和证据边界	结构与有界工程结果；RQ1-RQ3 可复核发布面

分析单元分别是治理对象、受治理工作项、重建视图和部署 Profile。该划分避免把产品功能、单个文件与架构层声明视为可互换证据。

5.2 证据与声明协议

证据被分类为 **specified**、**implemented**、**demonstrated** 或 **independently adopted**。一致性证据还记录固定来源修订、证据 Hash、执行前置条件、预期与实际输出，以及 PASS、PARTIAL、NOT RUN 或 FAIL 四种产品裁决之一。Fixture Oracle 匹配与产品执行分开报告。

该协议限制三类常见推断错误：历史工件不会被追溯性包装为专门设计的一致性测试；前置条件失败不会被转换为通过或产品失败；作者生成案例不会被视作独立采用。该过程支持结构与实现可行性声明，但不支持关于生产率、成本、可靠性或组织结果的因果声明。

5.3 FCoP 作为参考实例化

FCoP (File-based Coordination Protocol, 文件型协调协议) 是本文用于检验 TMPA 语义能否在普通、项目可见文件系统中实现的参考 Profile。当前指导与落实关系为：

TMPA 理论 → Core 要求 → FCoP 协议 Profile → CodeFlowMu 工程系统

图 6. 工程评估使用的参考实例化路径。

FCoP 不等同于 TMPA。它通过命名文本工件、生命周期目录、面向追加的迁移证据、Schema、Runtime 工具与适配器，实现 TMPA 的文件型子集 [7]–[12]。CodeFlowMu 是在 TMPA 指导下开发、并以 FCoP 作为协作协议的工程系统，承载持久多角色开发与工作环境 [28]、[37]。

表 7. FCoP 协议 Profile 与 TMPA 概念的映射。

FCoP 元素	TMPA 解释
工件 ID 或文件名 Stem sender 与 recipient	稳定且对传输可见的对象身份 声明写者与预期责任转移
生命周期路径与 transitions references、ref_task、subject_ref	Profile 定义的状态与迁移证据 类型化因果或治理链接
REVIEW 与批准证据 supersedes	独立裁决与决策 不破坏性重写的更正
Archive/History	保留的终态证据

在该 Profile 下，一项任务有一个稳定且对传输可见的载体；报告、复核、ISSUE、批准与更正保持为独立工件。写者彼此独立发布，Reader 检查当前可用来源集合。已发布的 Runtime Specification、Schema、Package、Adapter、Governance Middleware、ADR 和文档提供实现与分发路径证据，而不是广泛采用证据 [8]–[16]。

5.4 案例与语料库程序

作者报告的历史谱系为小典 AI → 早期 TMPA → FCoP → CodeFlowMu，随后工程反馈进入当前 TMPA 形式化 [25]。小典来源没有固定公开快照，因此仅用于披露设计来源，不作为评估观测。RQ3 使用公开锁定的 FCoP、CodeFlowMu 与 WP-13 工件；其运行结果只提供有界修订输入，不会成为定义 TMPA 的权威来源。

在线 CodeFlowMu Browser [13] 只保留为界面说明：其 Build 与数据集没有不可变标识，因此不纳入可复现评估。本文中的 CodeFlowMu 声明以锁定的 I1.0 证据包 [28] 为依据。NL2SQL 视图只说明预期对象链，不是代表性生产基准或计分观测。

综合语料库保留历史映射证据，并增加直接的 S1.0 产品运行 [28]。评估分为两条轨道。**分析轨道**检查不变量、反例、等输入确定性、非法迁移、冲突与三值判断。**工程轨道**通过原始字节 Hash 固定规范输入；锁定产品与依赖修订；盘点证据并计算 Hash；通过 CodeFlowMu 产品 Reader 执行每项准则；验证结果 Envelope；并把

Reference Reader Fixture 与产品行为分开报告。WP-13 路径被视为证据准入行为，不作为 S1.0 符合性证据，也不证明 Agent 不再产生幻觉 [36]。

所得证据支持有界的结构与实现可行性：持久项目可见协调、角色分离复核、生命周期门禁、冲突保留、确定性重建、Archive 保留与重启相关恢复。它只针对精确 S1.0 Bundle 和声明的 CodeFlowMu V1.8.0 证据修订建立覆盖 71 项强制断言的 14/14 产品结果；没有证明较低的比较成本、代表性 SME 性能、独立采用、跨部署 Profile 的可移植性，或任意输入与部署上的一致性。

5.5 证据依赖与分析边界

这些案例**不是独立复现**。TMPA、FCoP、CodeFlowMu、小典、Reference Reader 与 C01-C14 映射共享作者和工程谱系。因此，FCoP 与 CodeFlowMu 证据检验的是同一关联生态内部的投影和实现路径；把它们的测试数量相加不会增加独立观测数量。

证据集合按当前架构的可追踪性和可检查工件的可用性选取，并非随机抽样。分析使用描述性计数、不变量检查、反例与工件追踪；不进行零假设检验、效应量估计或因果比较。早期 FAIL、PARTIAL 与 NOT RUN 结果继续保留在不可变的 I0.6-I0.8 历史和冻结的 S1.0 候选基线中；A1.0 单独报告后续 I1.0 外部产品运行，不改写早期条件。

因此，投稿层面的推断单元仅限于：(a) 所述假设下的架构一致性；(b) 锁定且由作者控制的工件中已识别机制的执行；(c) 对证据准入、角色分离、恢复和审计行为的有界演示。组织有效性、较低成本、可靠性改善和普遍采用仍是有待独立研究的假设。

6. 规范重建契约

TMPA 实现可以把证据保存在不同基底中，但必须暴露足以进行确定性重建的规范文本投影。Reader 接收的是一组**来源候选**，而不是可信的有序日志。它解析并验证候选，保留重复观测，区分同一 ID 下不同内容的冲突，应用 Profile 规则，并输出：

1. 规范候选集合；
2. 偏序流程与责任图；
3. 权威问题集合；
4. authoritative、partial、disputed、quarantined 或 rejected 等状态。

符合规范的重建过程执行以下逻辑步骤：

```
RECONSTRUCT(source_candidates, profile):  
  解析并规范化候选  
  验证 Schema、ID、类型与完整性证据  
  保留重复观测与冲突候选  
  按写者流分组有效对象并检查局部序号  
  从序号、引用、生命周期与 Profile 关系构建边  
  验证角色权限、职责分离与生命周期合法性  
  检测缺失引用、禁止环与未解决冲突  
  推导生命周期、责任、复核与恢复状态  
  输出规范图和问题集合
```

图 7. 基底中立的规范重建程序。

该过程不会静默“修复”无效证据。格式错误对象、非法迁移、缺失引用、同 ID 不同内容或矛盾复核都通过问题集合保持可观察。后续授权解决对象可以改变权威解释，但不会擦除先前证据。

6.1 确定性

令 (S) 为固定最终来源候选集合，(P) 为固定 Profile。当 (S) 的任意枚举或交付排列都产生相同的规范候选集合、图、状态与问题集合时，Reader 具有确定性。

这是基于集合的要求，而不是声称每个中间视图都完整。在所有证据到达前，Reader 可以报告 partial 或 disputed 状态。一旦可用来源集合相同，输出必须与发现顺序无关。

证明义务来自四项约束：

- 规范化与验证依赖候选内容和 Profile 规则，而不是枚举顺序；

- 流内顺序来自显式序号证据；
- 跨流顺序来自显式引用与生命周期关系；
- 冲突以集合和问题形式保留，而不是使用“最后到达者获胜”。

本文目前提供证明概要与可执行 Fixture Oracle，而不是机械化证明。S1.0 C11 Fixture 枚举声明的排列，CodeFlowMu V1.8.0 产品路径针对该固定 Fixture 记录 PASS。它是声明 Bundle 上的产品级裁决，不是对任意图、编码、文件系统或敌对平台的证明。

6.2 完整性、身份与真实性

Digest 与签名证据必须被窄化解释。Digest 可以揭示被覆盖字节是否被修改；经过验证的签名可以在外部信任模型下把字节绑定到密钥。二者都不能证明被签名陈述在事实层面正确。

恶意、受损或判断错误的参与者可能发布 Schema 有效、生命周期有效、Digest 一致、甚至签名正确，但内容为假的报告。TMPA Core 可以识别并保留该对象，检查其声明权限，并把它与独立复核关联；事实验证需要 Core 之外的工具回执、可复现输出、独立数据源或领域专用验证。

因此必须分别回答三个问题：

表 8．完整性、认证身份与事实真值的分离。

问题	所需机制
这些字节是否被修改？	Digest 或防篡改存储
哪个经过认证的主体签署了它们？	签名、密钥与身份 Profile
声明是否为真？	独立验证或领域证据

除非部署实际依据对应 Profile 验证了证据，TMPA 不得声称拥有外围身份、策略或密码学系统的保证。

7. 评估结果

独立维护的 TMPA Core Specification S1.0 是当前 C01-C14 及全部 SHALL/MUST 条款的唯一规范性来源。Implementation Case Report I1.0 报告当前有边界的 S1.0 工程证据基线。I0.6-I0.8 继续作为各自旧输入和产品修订的不可变历史基线。本节按研究问题评估结果，不复制规范正文。

7.1 按研究问题组织的发现

表 9．按研究问题组织的发现、证据与推断边界。

研究问题	发现与证据	边界
RQ1：治理状态充分性	普通对话与执行界面本身不能保留足够显式的权限、生命周期、冲突与恢复状态，以支持确定性治理重建；支持来自问题诊断、DR1-DR8 以及对象与重建分析	尚无对比现场实验测量替代架构的失败率
RQ2：最低架构	稳定载体、单写者流、显式权限和生命周期、类型化引用、三值判断以及保留来源的确定性重建构成一致的最低契约；支持来自不变量、反例、确定性证明概要与 Core S1.0	证明尚未机械化；最低性是架构论证，而非普适下界证明
RQ3：工程可行性与边界	FCoP 提供协议 Profile；CodeFlowMu V1.8.0 针对精确 S1.0 Bundle 执行自身产品 Reader，并在 71 项强制断言上获得 14/14 PASS；归档锁定源码、输入、命令、结果与完整性记录；WP-13 单独提供证据准入观察	所有执行仍由作者完成；独立采用、SME 比较负担、跨 Profile 可移植性以及固定 Bundle 以外的行为尚未建立

因此，结果对架构一致性和固定 Bundle 下的实现可行性支持最强；对组织有效性与生态泛化的支持较弱，后者需要独立、代表性与比较证据。

7.2 一致性领域摘要

当前产品结果按架构领域汇总；准则的精确定义仍以 Core S1.0 § 10.2 为准。

表 10．CodeFlowMu V1.8.0 针对精确 S1.0 Bundle 的产品结果。

领域	准则	产品级结果
对象、不可变性、来源与完整性	C01、C02、C03、C08	4 项 PASS
权限与生命周期	C05、C06、C07	3 项 PASS
顺序、引用与冲突	C04、C09、C10、C12	4 项 PASS
确定性、恢复与历史	C11、C13、C14	3 项 PASS

该 14/14 结果是一个精确输入 Bundle 和声明产品修订上的准则级产品裁决，不代表认证治理、任意来源集合上的正确性、完整协议有效性或独立认证。

7.3 S1.0 作者运行产品基线

I1.0 发布 tmpa-i1.0-codeflowmu-v1.8.0-s1.0-evidence-20260811.zip，其外部 SHA-256 为 9b92b06c3e46c8019362f55075be4ed066cb7a9c0d9859945b6c3b7de8840d04 [28]。本轮固定 TMPA Core S1.0 冻结候选 Commit 942cbb097eb3d662662f96a2269818ec9d7ca2ed，以及 CodeFlowMu V1.8.0 证据 Commit c1e1f724293e8048fc3a956b6f6df8cf83f54830。规范输入与冻结 GitHub 对象字节一致；聚合 Bundle Digest 为 sha256:f98764987760cdc8ac356b1265fc98485f33345e7d6ffc8575ccb059ddd34daa，聚合 Result Digest 为 sha256:0f0f642449db1853371861751a7a8ea36dce00013f53e32012a5e4dae45f4c39。

结果 Envelope 记录 product_reader_called: true、reference_reader_called: false。CodeFlowMu 内置 TMPA Runtime Suite 通过 24/24；Runtime 记录 1,522 passed / 0 failed / 1 skipped；Shell 记录 791/791；Protocol Validation 与 Typecheck 成功退出；锁定在 Commit da79dfefd99f597c9e422ce9edec22157f915a21 的 FCoP 参考实现记录 1,210 passed / 2 skipped。这些支撑性套件计数只提供上下文，不构成额外的 C01-C14 观测。

表 11. I1.0 基线的产品级 S1.0 裁决。

裁决	准则
PASS	C01、C02、C03、C04、C05、C06、C07、C08、C09、C10、C11、C12、C13、C14
PARTIAL	无
NOT RUN	无
FAIL	无

证据包包含 889 个由 Manifest 覆盖的文件，包括精确输入、产品源码、依赖锁、命令记录、原始结果、修复前失败、修复历史与精简干净环境复现器。出版审查重新计算全部十四个 Manifest Digest、十四个 Result Digest、71 项强制断言、输入 Digest 与聚合 Result Digest，没有发现差异。这建立证据包一致性与可追踪性，不构成独立产品重跑或认证。

7.4 解释：已关闭的实现缺口与仍开放的验证缺口

I0.6 基线暴露了产品投影缺失、C02 与 C07 的具体失败，以及 C08、C11、C12 未执行路径。I0.7 与 I0.8 逐步关闭这些已观察缺口。I1.0 把产品路径重新绑定到稳定 S1.0 机器身份，保留冻结候选中的历史产品 NOT RUN 基线，并把后续 CodeFlowMu V1.8.0 精确版本运行单独登记。没有改弱任何历史裁决或准则。

精确 S1.0 规范字节
 ↓ Hash 核验
 CodeFlowMu V1.8.0 产品 Reader
 ↓ C01-C14 执行
 Schema 有效的 14/14 结果
 ↓ 出版
 锁定证据包 + 完整性 Manifest

图 8. I1.0 建立的产品证据路径。

主要剩余缺口因此已经从产品投影与稳定机器身份转移到验证独立性与外部范围。该结果仍是作者运行、关联谱系、固定 Bundle 下的 demonstrated 证据。CodeFlowMu 证据 Commit 在捕获时仅存在于本地，但证据包包含其源码快照与 Patch。独立重跑、独立采用、替代 Profile 实现、代表性 SME 部署、比较成本和组织结果测量，仍是支持更广泛声明的必要条件。WP-13 继续作为有限的证据门禁案例，不建立 S1.0 一致性，也不证明幻觉消除。

8. 讨论、局限与效度威胁

TPMA 的贡献是流程治理契约，而不是完整企业 Agent 平台。它在发布时显式化工作身份、责任、复核、生命周期、冲突与恢复，并从持久证据中重建这些关系。FCoP 提供作者运行证据，表明一个有用子集可以在普通项目环境中运行，而不强制要求 Broker 或协调数据库。

该架构不替代：

- 企业 IAM、凭证签发或密钥管理；
- Runtime Gateway、Policy Engine 或 Admission Control；
- 模型评估与事实验证；
- OTel、SIEM、CMDB、GRC 或企业 Agent 资产清单；
- 数据库事务、分布式共识或拜占庭容错；
- 法律合规计划。

这些系统可以向 TPMA 提供 ID、策略决策、执行回执与受保护存储。除非经过验证并引用，其保证仍属于外部系统。

8.1 证据成熟度

本文区分四个声明层级：

1. **specified**：已发布规则、Schema 或标准；
2. **implemented**：代码执行该规则；
3. **demonstrated**：运行案例展现该行为；
4. **independently adopted**：外部系统依赖并验证该行为。

架构和标准已规定；FCoP 提供实现证据；锁定的 CodeFlowMu 与 WP-13 工件提供边界明确的演示证据；小典仅保留为作者报告的谱系，独立采用尚未成立。

论文作者同时是被评估系统的发起者和主要开发者。这使详细工件可被访问，但也产生自我评估与选择偏差风险。因此，语料库分别标注作者生成证据、非门禁现场证据、仅 Fixture Oracle 结果和产品级裁决。

8.2 SME-first 声明

“SME-first”是运行范围，不表示所有中小企业需求相同，也不表示 TPMA 不适合大型组织。轻量 Profile 假设平台与运维能力有限。大型部署可以通过数据库、对象存储、身份系统、复制和企业控制平面保留同样语义。

决定性的实证问题仍然是：收益是否足以抵偿纪律和资源成本。所需测量包括：

表 12. 检验 SME-first 可行性声明所需的实证计划。

实验	所需证据
部署负担	依赖、安装步骤、设置时间、建立第一支团队的时间、备份与迁移
重建	延迟与乱序流、中间 partial 状态、字节等价最终输出
故障与恢复	重复、非法迁移、缺失引用、篡改、重启与恢复时间
人类可检查性	识别所有权、缺失证据、复核状态与下一责任人的能力
采纳纪律	Onboarding、绕过、更正、回退聊天、感知负担与持续使用
代表性负载	延迟、CPU、内存、存储增长、冲突率与恢复时间

当前语料库提供规范化基线，但没有完成这些测量。

8.3 数字员工 Profile 与隐私

数字员工一词仅作为持久 AI 工作角色的工程标签。它不主张雇佣身份、法律人格、意识、人类意图、自主组织权威，也不意味着替代承担责任的人类或组织。

未来 Profile 可以定义岗位范围、接受委托、交接、暂停、重新分配与退役。这些是应用语义，不改变 Core。

文本治理提高可检查性，但可能暴露敏感信息。部署应最小化内容，把秘密与治理元数据分离，应用访问控制与加密，并定义保留和删除程序。治理历史不可变不等于敏感 Payload 必须公开可读。

8.4 效度威胁

构念效度。 C01-C14 操作化治理结构与重建行为，但不直接测量事实真实性、人类可用性、生产率或组织问责。标准通过不得被解释为这些外部构念上的成功。

内部效度。 作者选择了架构、系统、案例与证据映射，并运行了基线。版本差异以及从历史工件到后设标准的回溯映射都可能影响结果。精确规范字节、固定修订、Hash、显式前置条件以及分离产品与 Fixture 裁决可以降低但不能消除该风险。

外部效度。 主要实现是文件型 Profile，案例集合较小，且大量观测执行发生在项目局部环境。结果未必能原样迁移到数据库型、高度分布式、受监管、对抗性或高吞吐部署。

结论效度。 裁决计数只是选定固定 Bundle 的描述性结果，不是统计估计、普遍或独立认证的一致性证据，也不是与聊天、Event Log、Workflow Engine 或数据库替代方案的因果比较。

可复现性。 当前语料库由作者生成。它现已具备稳定公开仓库路径、可执行复现命令、环境声明与 SHA-256 Manifest；仍需要独立重跑。

8.5 局限与可证伪条件

TPMA 的核心声明必须保持可被反证：

表 13. TPMA 核心声明的证伪条件。

声明	会削弱或推翻该声明的证据
相同来源集合允许确定性重建 持久文本证据改善责任恢复 架构能以最低基础设施运行 SME-first Profile 在运行上可行 语义可跨 Profile 移植	符合规范的 Reader 在相同 Profile 与来源集合上产生不同规范图或问题输出 受控恢复任务不优于相关替代方案，或无法可靠识别责任与缺失证据 正确性依赖未声明的协调数据库、Broker、全局时钟或集中可变日志 在代表性 SME 使用中，部署、维护、存储或人员纪律成本超过测得的治理收益 独立实现在不同存储基底之间无法保留对象、权限、生命周期、冲突与重建语义

8.6 出版与可复现边界

A1.0 是稳定理论架构论文；Core S1.0 是当前规范性来源；Implementation Case I1.0 报告当前作者运行的 S1.0 工程证据基线。I0.6-I0.8 保留各自早期历史状态。论文可以总结配套工件，但不得静默重定义其含义，不得合并 Reference Reader 与产品裁决，也不得把固定 Bundle 结果扩大为普遍一致性。

在当前系统边界中，TPMA 理论与架构指导 CodeFlowMu 工程落实，Core S1.0 固定接受评估的行为，FCoP 提供 CodeFlowMu 使用的协作协议。历史工程反馈可以推动后续理论或规范修订，但不会使 CodeFlowMu 或 FCoP 成为定义 TPMA 的权威来源。

公开证据包现已包含精确规范输入、CodeFlowMu 证据源码、命令、结果、回归历史、依赖锁、精简复现器与完整性 Manifest。在形成独立验证声明之前，仍至少需要一次无关联方重跑。低资源部署测量仍是 SME 可行性声明的独立实证要求。

9. 结论

TPMA 是一种**面向中小企业、最低基础设施条件的文本消息多智能体异步流程架构**。文本承载持久工作与状态；每项工作有一个稳定主载体；每个已发布对象有一个写者并属于一条局部串行流；各条独立流异步推进；聚合与确定性读取重建偏序流程、责任、生命周期、冲突、恢复和审视图。

该架构沿着**实践 → 方法 → 理论**形成：小典 AI 暴露多角色协调问题，早期 TPMA 识别文本异步方法，FCoP 抽取并成熟其可复用文件协调与复核子集，CodeFlowMu 则把这一方向落实为持久开发与工作系统。FCoP 与 CodeFlowMu 的运行结果随后作为证据反馈到当前对象、不变量、保证边界和一致性标准的形式化中。现行权威方向与历史反馈方向相反：TPMA 理论与 Core 要求通过 FCoP 协议指导 CodeFlowMu 持续工程落实。早期 Pipeline 记录起源，而不是追溯性的 Core Conformance。

A1.0 在架构层回答 RQ1 与 RQ2，并为 RQ3 提供由作者运行、版本锁定的基线。在 I1.0/S1.0 评估下，CodeFlowMu V1.8.0 针对精确 Bundle 和 71 项强制断言记录**产品级 14 项 PASS、0 PARTIAL、0 NOT RUN、0 FAIL**。其内置 TPMA Runtime Suite 通过 24/24；Runtime 记录 1,522 passed / 0 failed / 1 skipped；Shell 记录

791/791; 锁定的 FCoP 参考实现记录 1,210 passed / 2 skipped。单独维护的 S1.0 Reference Reader 通过全部十四项合成 Fixture，但这些结果仍与产品执行分开。锁定证据包提高了可追踪性，却不会自动形成独立验证。

决定性剩余问题是 RQ3：组织能否在普通项目环境中，以可接受的资源与纪律成本持续获得 TMPA 的责任、复核、恢复与证据收益。产品投影与锁定 S1.0 证据包现已存在；更广泛的声明仍需要低资源部署与恢复测量、基线对比、代表性使用、替代 Profile 证据和独立复现。TMPA 自身也不确立认证身份、强隔离、受保护存储、拜占庭容错、参与者声明的事实真实性或生态采用。

工件可用性

当前作者运行的 S1.0 证据包为 tmpa-i1.0-codeflowmu-v1.8.0-s1.0-evidence-20260811.zip，随 Implementation Case I1.0 发布，并可从公开证据路径下载。其 SHA-256 为 9b92b06c3e46c8019362f55075be4ed066cb7a9c0d9859945b6c3b7de8840d04。早期 S0.4–S0.6 语料库与 I0.6–I0.8 报告继续作为不可变历史基线保留。除非经过独立重跑，所有产品证据都属于作者生成证据。

数据可用性

本文不公开生产业务数据。NL2SQL Worked Material 是说明性治理重建，而不是生产数据逐字导出。一致性 Fixture 和选定实现证据包含在作者生成语料库中；未来公开发布必须保留脱敏、版本、来源与 Checksum 信息。

利益冲突与作者生成证据

论文作者同时是 TMPA、FCoP 与 CodeFlowMu 的发起者和主要开发者。这种关系带来自我评估与选择偏差风险。本文区分 specified、implemented、demonstrated 与 independently adopted 声明；当前基线由作者运行，不构成独立验证或生态采用。

作者贡献

朱卫：概念提出、架构设计、研究方法、软件与协议开发、证据整理、调查、论文写作及公开工件维护。本单作者贡献声明不表示被评估系统已经获得独立验证。

伦理与隐私声明

本架构研究不报告人类受试者实验，也不公开生产业务数据。Worked Example 与一致性 Fixture 均为技术工件。未来任何涉及员工、组织行为、用户表现、访谈或敏感运行记录的研究，都必须另行处理适用的审查、知情同意、访问控制、数据最小化、保留和脱敏要求。

References

- [1] Model Context Protocol. “Specification 2026-07-28.” Final specification revision, 28 July 2026. <https://modelcontextprotocol.io/specification/2026-07-28>.
- [2] A2A Protocol Project, Linux Foundation. “A2A Protocol Ships v1.0.” 2026. <https://a2a-protocol.org/latest/announcing-1.0/>.
- [3] World Wide Web Consortium. “PROV-DM: The PROV Data Model” and “Constraints of the PROV Data Model.” W3C Recommendations, 2013. <https://www.w3.org/TR/prov-dm/>. Accessed 2026-07-30.
- [4] Martin Fowler. “CQRS.” 2011. <https://martinfowler.com/bliki/CQRS.html>. Accessed 2026-07-30.

- [5] Scott Chacon and Ben Straub. “Git Internals — Git Objects.” Pro Git, second edition. <https://git-scm.com/book/en/v2/Git-Internals-Git-Objects>. Accessed 2026-07-30.
- [6] Linux Foundation. “Linux Foundation Announces the Formation of the Agentic AI Foundation (AAIF), Anchored by New Project Contributions Including Model Context Protocol (MCP), goose and AGENTS.md.” 9 December 2025. <https://www.linuxfoundation.org/press/linux-foundation-announces-the-formation-of-the-agentic-ai-foundation>. Accessed 2026-07-30.
- [7] FCoP Project. “FCoP — File-based Coordination Protocol,” tag v3.2.5, commit b3dc23439c6a aa6a6b3765655b87e5924e0257f9. GitHub, 2026. <https://github.com/joinwell52-AI/FCoP/tree/v3.2.5>.
- [8] FCoP Project. “FCoP v3 Specification,” spec/fcop-v3-spec.md, tag v3.2.5, 2026. <https://github.com/joinwell52-AI/FCoP/blob/v3.2.5/spec/fcop-v3-spec.md>.
- [9] FCoP Project. Protocol Rules, machine-readable schemas, and architecture decisions, tag v3.2.5, 2026. <https://github.com/joinwell52-AI/FCoP/tree/v3.2.5/spec>.
- [10] FCoP Project. fcop and fcop-mcp reference-implementation packages, tag v3.2.5, 2026. <https://github.com/joinwell52-AI/FCoP/tree/v3.2.5/src>; these packages implement the protocol and are not the protocol definition.
- [11] Official MCP Registry. io.github.joinwell52-AI/fcop, fcop-mcp server entry, 2026. Accessed 2026-07-30.
- [12] FCoP Project. “FCoP 3.2.5 Release Notes.” 2026. <https://github.com/joinwell52-AI/FCoP/blob/v3.2.5/docs/releases/3.2.5.md>.
- [13] CodeFlowMu. “TMPA Browser” 在线公开演示. <https://demo.chedian.cc/>. 访问于 2026-07-29。因 Build 与数据集没有不可变标识，本来源仅作界面说明，不纳入可复现评估；锁定的 CodeFlowMu 声明使用 [28]。
- [14] FCoP Project. “ADR-0031: Governance Alert Layer (GAL).” Accepted 2026-05-11; tag v3.2.5. <https://github.com/joinwell52-AI/FCoP/blob/v3.2.5/adr/ADR-0031-governance-alert-layer.md>.
- [15] FCoP Project. “ADR-0032: fcop_audit() — Protocol-to-Inspection Compiler.” Accepted 2026-05-12; tag v3.2.5. <https://github.com/joinwell52-AI/FCoP/blob/v3.2.5/adr/ADR-0032-fcop-audit-protocol-inspection.md>.
- [16] FCoP Project. “FCoP Three-Layer Semantic Execution Chain Model.” Tag v3.2.5, 2026. <https://github.com/joinwell52-AI/FCoP/blob/v3.2.5/adr/FCoP-semantic-execution-chain.md>.
- [17] Yi Nian, Aojie Yuan, Haiyue Zhang, Jiate Li, and Yue Zhao. “Auditable Agents.” arXiv:2604.05485, 2026. <https://arxiv.org/abs/2604.05485>. Accessed 2026-07-30.
- [18] Mirja Kühlewind and Henk Birkholz. “An Architecture for Auditing AI Agent Delegation and Interactions.” Internet-Draft draft-kuehlewind-audit-architecture-00, Work in Progress, 18 May 2026. <https://datatracker.ietf.org/doc/draft-kuehlewind-audit-architecture/00/>. Accessed 2026-07-30.
- [19] Google Cloud. “Register Agents.” Agent Registry documentation, updated 27 July 2026. <https://docs.cloud.google.com/agent-registry/register-agents>. Accessed 2026-07-30.
- [20] Krti Tallam. “Authorization Propagation in Multi-Agent AI Systems: Identity Governance as Infrastructure.” arXiv:2605.05440, 2026. <https://arxiv.org/abs/2605.05440>.
- [21] Maurits Kaptein, Vassilis-Javed Khan, and Andriy Podstavnychy. “Runtime Governance for AI Agents: Policies on Paths.” arXiv:2603.16586, 2026. <https://arxiv.org/abs/2603.16586>.
- [22] Mert Cemri et al. “Why Do Multi-Agent LLM Systems Fail?” arXiv:2503.13657, version 3, 2025. <https://arxiv.org/abs/2503.13657>.
- [23] OECD. “Empowering SMEs in the Age of AI: The 2026 OECD D4SME Survey.” OECD SME and Entrepreneurship Papers, No. 78, OECD Publishing, Paris, 13 April 2026. DOI: 10.1787/bf5a9816-en. https://www.oecd.org/en/publications/empowering-smes-in-the-age-of-ai_bf5a9816-en.html. Accessed 2026-07-30.

- [24] Infocomm Media Development Authority, Singapore. “Singapore's Digital Economy at 18.6% of GDP, up from 14.9% in 2019; Growth Fuelled by Accelerating Digitalisation and AI Adoption across Sectors and Firms.” 6 October 2025. <https://www.imda.gov.sg/resources/press-releases-factsheets-and-speeches/press-releases/2025/singapore-digital-economy>. Accessed 2026-07-30.
- [25] SaigeAgent / 小典 AI 项目。《多 AI 角色协同架构规划》。作者报告的私有项目档案，2026 年 3 月。当前没有固定公开快照；本引用仅披露作者陈述的设计谱系，不纳入评估语料、研究问题结果或一致性声明。
- [26] Christian Schroeder de Witt. “Open Challenges in Multi-Agent Security: Towards Secure Systems of Interacting AI Agents.” arXiv:2505.02077, 2025. <https://arxiv.org/abs/2505.02077>. Accessed 2026-07-30.
- [27] Richard Kang and Yudho Diponegoro. “Governance Gaps in Agent Interoperability Protocols: What MCP, A2A, and ACP Cannot Express.” arXiv:2606.31498, 2026. <https://arxiv.org/abs/2606.31498>. Accessed 2026-07-30.
- [28] TMPA Project. “Implementation Case I1.0: CodeFlowMu V1.8.0 against TMPA Core S1.0.” Package tmpa-i1.0-codeflowmu-v1.8.0-s1.0-evidence-20260811, captured 11 August 2026. SHA-256 9b92b06c3e46c8019362f55075be4ed066cb7a9c0d9859945b6c3b7de8840d04; author-run exact-input product evidence. Independent rerun remains required.
- [29] Zexun Wang. “Proof-Carrying Agent Actions: Model-Agnostic Runtime Governance for Heterogeneous Agent Systems.” arXiv:2606.04104, 2026. <https://arxiv.org/abs/2606.04104>. Accessed 2026-07-31.
- [30] Rafflesia Khan, Declan Joyce, and Mansura Habiba. “AGENTS SAFE: A Unified Framework for Ethical Assurance and Governance in Agentic AI.” arXiv:2512.03180, 2025. <https://arxiv.org/abs/2512.03180>. Accessed 2026-07-31.
- [31] National Institute of Standards and Technology. “Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile.” NIST AI 600-1, July 2024; updated April 2026. <https://doi.org/10.6028/NIST.AI.600-1>.
- [32] Leslie Lamport. “Time, Clocks, and the Ordering of Events in a Distributed System.” Communications of the ACM, 21(7), 558–565, 1978. DOI: 10.1145/359545.359563.
- [33] K. Mani Chandy and Leslie Lamport. “Distributed Snapshots: Determining Global States of Distributed Systems.” ACM Transactions on Computer Systems, 3(1), 63–75, 1985. DOI: 10.1145/214451.214456.
- [34] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. “Design Science in Information Systems Research.” MIS Quarterly, 28(1), 75–105, 2004. DOI: 10.2307/25148625.
- [35] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. “A Design Science Research Methodology for Information Systems Research.” Journal of Management Information Systems, 24(3), 45–77, 2007. DOI: 10.2753/MIS0742-1222240302.
- [36] Zijie Zhuang et al. “From Trajectories to Evidence: Auditable Experimental Records for Industrial Research Agents.” arXiv:2608.05235, 2026. <https://arxiv.org/abs/2608.05235>.
- [37] CodeFlowMu Project. “CodeFlowMu V1.4–V1.5 TMPA × FCoP × 应用统一架构”，docs/TMPA-GOVERNANCE.md。GitHub, 2026. <https://github.com/joinwell52-AI/codeflowmu/blob/main/docs/TMPA-GOVERNANCE.md>.