

TMPA: Textual Multi-Agent Process Architecture

Zhu Wei - joinwell52 Research Center

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

Contents

TMPA: Textual Multi-Agent Process Architecture	2
An AI-Native Software Architecture Theory for Governed Multi-Agent Organizational Work in SMEs	2
Abstract	2
1. Introduction	3
1.1 Paper Type and Research Questions	4
1.2 Target Environment and Design Constraints	5
1.3 Design Origin and Evolution	6
1.4 Design Premises and Contributions	6
1.5 Terminology and Representation Stages	7
2. Problem Definition and Design Requirements	8
3. Related Work and Positioning	9
3.1 Architectural Antecedents	9
3.2 Interoperability and Governance Gaps	10
3.3 Direct Research Neighbors	10
3.4 Comparative Synthesis and Research Gap	11
4. TMPA Architecture	11
4.1 Governance Object	11
4.2 Document Types	12
4.3 Role and Authority Model	12
4.4 Lifecycle Model	13
4.5 Textual Messages, Single-Writer Streams, and Asynchronous Parallelism	13
4.6 Read-Side Aggregation and Governance Reconstruction	14
4.7 Integrity and Signature Evidence	15
4.8 Guidance Relation, Historical Lineage, and Operational Stack	16
5. Research Design and Evaluation Method	16
5.1 Design-Science Procedure	16
5.2 Evidence and Claim Protocol	17
5.3 FCoP as Reference Instantiation	17
5.4 Case and Corpus Procedure	17
5.5 Evidence Dependence and Analysis Boundary	18
6. Canonical Reconstruction Contract	18
6.1 Determinism	19
6.2 Integrity, Identity, and Truth	19
7. Evaluation Results	20

7.1 Findings by Research Question	20
7.2 Conformance-Domain Summary	20
7.3 S1.0 Author-Run Product Baseline	20
7.4 Interpretation: Closed Implementation Gaps, Open Validation Gap	21
8. Discussion, Limitations, and Threats to Validity	21
8.1 Evidence Maturity	22
8.2 SME-First Claim	22
8.3 Digital Employee Profile and Privacy	22
8.4 Threats to Validity	23
8.5 Limitations and Falsification Conditions	23
8.6 Publication and Reproducibility Boundary	23
9. Conclusion	24
Artifact Availability	24
Data Availability	24
Competing Interests and Author-Produced Evidence	24
Author Contributions	25
Ethics and Privacy Statement	25
References	25

TMPA: Textual Multi-Agent Process Architecture

An AI-Native Software Architecture Theory for Governed Multi-Agent Organizational Work in SMEs

Architecture Paper Release: A1.0

Historical Source Baseline: TMPA Draft V1.0-R23; theory aligned through R31

Status: Stable V1.0 research-paper release

Release Date: 2026-08-11

Publication Authority: This GitHub document is the authoritative TMPA architecture paper. The separately maintained TMPA Core Specification is normative; the Implementation Case Report is evidentiary and non-normative.

Author: Zhu Wei, joinwell52 Research Center

Public correspondence: joinwell52-AI/joinwell52

Document identifier: TMPA-ARCH-A1.0

Review note: This public author-identified version requires anonymization if submitted to a double-blind venue.

Abstract

Large language models are moving from isolated question answering into long-running, tool-using, and multi-agent execution. Tool traces and conversation histories may show what occurred, but they do not by themselves establish authorized responsibility, legal lifecycle transitions, independent review, or recoverable governance state.

This paper presents **TMPA (Textual Multi-Agent Process Architecture)**, an SME-first, minimal-infrastructure **textual-message multi-agent asynchronous process architecture**. Its core has four linked rules: **text carries durable messages and state; each writer preserves a local serial stream; multiple streams progress asynchronously to form parallel collaboration; and readers aggregate the available evidence and reconstruct process, responsibility, lifecycle, conflict, and audit state**. One stable primary carrier anchors each governed work item; subsequent reports, reviews, decisions, and corrections remain separate single-writer objects. Reconstruction preserves concurrency and unresolved conflict rather than imposing an artificial total order.

FCoP is the project-visible filesystem profile examined in this paper. It requires no mandatory coordination database, broker, or enterprise control plane, but it also does not by itself provide verified enterprise identity, strong role isolation, tamper-resistant storage, or Byzantine resilience. TMPA is therefore **SME-first, not SME-only**: larger implementations may preserve the same semantics through databases, object stores, event services, identity systems, and control planes.

In the current publication architecture, **TMPA theory guides the engineering direction of CodeFlowMu**. The Core Specification turns that theory into normative objects, Reader behavior, and conformance criteria; FCoP carries the file-based coordination protocol; and CodeFlowMu implements the governed roles, workflows, review, recovery, and audit mechanisms in a running engineering system. This guidance relation is distinct from the historical feedback through which FCoP and CodeFlowMu also helped refine the later formalization.

Following established design-science methodology [34], [35], the study diagnoses the governance-state problem, derives design requirements, constructs the TMPA artifact, demonstrates it through the FCoP protocol profile and downstream cases, and evaluates both architectural invariants and version-pinned C01–C14 evidence. The contribution is an integrated governance architecture—not a new storage primitive, runtime, or truth oracle—and the evaluation is deliberately claim-bounded. Against the exact TMPA Core S1.0 bundle, the CodeFlowMu V1.8.0 product Reader records **14 PASS, 0 PARTIAL, 0 NOT RUN, and 0 FAIL** across 71 mandatory assertions; the separate S1.0 Reference Reader also passes fourteen synthetic criteria. A locked evidence archive preserves the exact inputs, product source, commands, outputs, regression history, and 889-file integrity manifest. These results strengthen implementation feasibility, but remain author-run evidence; low-resource performance, adoption cost, comparison baselines, representative use, and independent reproduction remain open empirical requirements.

Keywords: AI governance, agentic AI, multi-agent systems, SMEs, minimal infrastructure, textual messages, primary carrier, single-writer streams, asynchronous collaboration, deterministic reconstruction, lifecycle, role separation, provenance, auditability, recoverability, FCoP, CodeFlowMu

1. Introduction

Large language models have transformed artificial intelligence from isolated reasoning systems into execution systems that use tools, modify files, query databases, operate business software, and collaborate over long-running tasks. Correct output remains necessary, but deployable systems must also preserve the authority, responsibility, and evidence surrounding that output.

A governed multi-agent system must answer who authorized and accepted work, which object represented it, which evidence was produced, who reviewed and decided, whether transitions were legal, and whether the process can be reconstructed after interruption. Logs, chats, workflow states, and business records may contribute evidence, but they do not automatically define an authoritative governance state.

TMPA (Textual Multi-Agent Process Architecture) addresses this gap without governing how agents think or replacing agent frameworks, identity providers, runtime gateways, transports, or storage systems. It defines a cross-platform process-responsibility contract through four operational statements:

Text carries messages and state.

Each writer preserves its own serial stream.

Multiple serial streams progress asynchronously to form parallel collaboration.

Readers aggregate the streams and reconstruct process and governance state.

One stable primary carrier anchors each governed task. Acceptance, reports, reviews, decisions, corrections, and recovery evidence are independently authored objects linked by explicit references. The write side is locally serial and single-writer; the system is asynchronously parallel; the read side reconstructs a partial-order graph and issue set.

The paper uses three orientation views:

HISTORICAL CO-EVOLUTION
business practice → early TMPA method → FCoP extraction and maturation
→ CodeFlowMu engineering implementation
FCoP + CodeFlowMu implementation feedback → current TMPA formalization

CURRENT GUIDANCE AND IMPLEMENTATION
TMPA theory and architecture
↓ formalized as normative behavior by
TMPA Core Specification
↓ projected through a file-based coordination profile
FCoP protocol
↓ used to implement governed work in
CodeFlowMu engineering system

END-TO-END PROCESS
write: primary carrier → single-writer streams → asynchronous composition
read: source aggregation → governance reader → process graph + issue set

Figure 1. TMPA orientation map: historical lineage, current layering, and operational reconstruction.

The lineage explains origin and feedback; the guidance relation explains present authority and implementation direction; and the process view explains operation. Historical feedback does not reverse current authority: FCoP does not exhaust TMPA, CodeFlowMu does not define FCoP or TMPA, and the original March 2026 pipeline did not already satisfy the current Core Specification.

One optional application context, specified separately in Section 8.3, is a persistent AI work role sometimes described in industry as a **digital employee**. Throughout this paper, that label denotes only an engineering work identity that accepts delegated work, uses tools, and submits results across sessions; it does **not** imply legal employment, personhood, consciousness, human intention, or replacement of accountable human or organizational principals.

1.1 Paper Type and Research Questions

This paper is a design-science and systems-architecture study. The designed artifact is TMPA; the companion Core Specification defines its normative behavior, while this paper explains the problem, theory, design logic, and evaluation. The primary environment is an SME or small team in which governance must begin without assuming a dedicated agent platform, coordination database, message broker, enterprise identity plane, or specialist operations team. FCoP provides the protocol profile, while CodeFlowMu is the engineering system developed under TMPA guidance and evaluated as a bounded implementation. The study does not claim a representative benchmark, production-scale validation, or superiority over enterprise governance platforms.

The paper addresses three research questions:

- **RQ1 — Governance-state sufficiency:** Which information is missing when chats, shared folders, execution traces, and ordinary task states are used as the record of multi-agent organizational work, and why does that prevent authoritative responsibility and recovery?
- **RQ2 — Minimum architecture:** Which substrate-independent objects, authority relations, lifecycle rules, ordering constraints, conflict semantics, and read-side operations are minimally necessary to reconstruct governed multi-agent work without a mandatory coordination database, broker, or control plane?
- **RQ3 — Engineering feasibility and boundary:** To what extent do the FCoP profile, the pinned CodeFlowMu product evidence, WP-13, and the C01–C14 corpus demonstrate those properties, and which feasibility claims remain unsupported?

A1.0 answers RQ1 through evidence-gap analysis and RQ2 through the TMPA object, stream, authority, lifecycle, and reconstruction model. RQ3 receives a stronger but still bounded answer: CodeFlowMu V1.8.0 calls its own product Reader against the exact S1.0 inputs and records 14/14 product PASS across 71 mandatory assertions, while the locked package makes the inputs, source, run record, and integrity trail inspectable [28]. The result is demonstrated under a fixed bundle, not independently adopted or certified. The WP-13 case separately shows why a completed agent trajectory is not automatically admissible governance evidence [36]. Quantified setup burden, low-resource performance, broader fault recovery, comparison baselines, representative use, and third-party reproduction remain unfinished evidence.

Table 1. Research claims, supporting evidence, and prohibited inference.

Claim	Strongest support in this study	Inference not permitted
ordinary execution records are insufficient for governance reconstruction	problem analysis, DR1–DR8, failure and counterexample reasoning	that every chat, workflow, or event system necessarily fails
the TMPA object–stream–reader model is internally coherent	explicit invariants, lifecycle and authority model, determinism proof sketch, Core S1.0	a universal mathematical proof of minimality
a file-based protocol and a TMPA-guided engineering system can realize a substantial subset	FCoP mapping and exact-input CodeFlowMu V1.8.0 product evidence	conformance beyond the fixed S1.0 bundle, protocol validity inferred from package tests, or independent adoption
governed multi-agent cases can retain and qualify contested completion claims	CodeFlowMu and WP-13 evidence chains	elimination of hallucination, causal performance improvement, or independent adoption

This matrix is the governing interpretation of the paper's contribution claims. Later implementation detail cannot enlarge a claim beyond the corresponding evidence and boundary shown here.

1.2 Target Environment and Design Constraints

The primary environment is an SME or small team that needs governed AI collaboration but lacks some combination of a dedicated agent platform, coordination database, message broker, enterprise agent identity infrastructure, and specialist operations staff. This scope is defined by capability constraints, not employee count alone.

The 2026 OECD D4SME Survey reports uneven strategic and secure AI integration among more than 2,000 SMEs across 12 OECD countries, with time, maintenance cost, and skills gaps remaining material obstacles [23]. Singapore's IMDA reported 2024 AI adoption of 14.5% among SMEs versus 62.5% among non-SMEs [24]. These sources do not establish demand for TMPA specifically; they support the broader problem of responsible AI integration under constrained organizational capacity.

The design constraints lead to the following chain:

```

limited infrastructure and operations capacity
↓
no mandatory database, broker, cluster, or control plane
↓
one governed task → one stable primary textual carrier
↓
locally serial single-writer responsibility streams
↓
independent streams progress asynchronously
↓
source aggregation → governance reconstruction
↓
recoverable process, responsibility, and audit state

```

Figure 2. Derivation of the minimum-infrastructure design constraints.

Text-carried, locally serial, asynchronously parallel, deterministically reconstructed.

The primary profile is asynchronous project-local collaboration, not distributed consensus or multi-datacenter execution. Participants need not be online together, and evidence may arrive late; the reader exposes the authoritative, partial, disputed, or quarantined view supported by the currently available set. Database-backed and enterprise profiles remain possible, but identity federation, replication, high availability, and cross-domain enforcement are outside the present validation scope.

1.3 Design Origin and Evolution

The author's private XiaoDian project archive reports that an early form of TMPA appeared in a March 2026 multi-role architecture plan under the name **Text-Message Multi-AI Parallel Architecture** [25]. Because that source has no fixed public snapshot, it is retained only as author-reported design lineage and is excluded from the evaluated corpus, RQ results, and conformance claims.

The same private source reportedly records a transitional Pipeline design [25]. This author-reported lineage is not used as submission evidence and does not establish present-day Core conformance. Immutable writer streams, source-preserving aggregation, deterministic reconstruction, explicit conflict states, and conformance requirements were developed later and are evaluated only through the public, pinned artifacts.

```
XiaoDian AI business practice
  ↓ architecture abstraction
Original TMPA
  ↓ reusable file-coordination and review skeleton extracted
FCoP
  ↓ protocol, schema, package, MCP, governance, and audit evolution
CodeFlowMu engineering implementation under early TMPA guidance
  ↓ operational feedback into theory and specification
Current TMPA formalization and continued CodeFlowMu alignment
```

Figure 3. Design lineage from business practice to current TMPA formalization.

Practice revealed the problem; repeated engineering revealed the method; formalization elevated the method into theory.

The present paper reunifies the original message-and-asynchrony model with governance semantics matured through FCoP and CodeFlowMu. The present authority direction is nevertheless explicit: current TMPA theory and Core requirements guide continued CodeFlowMu engineering, while implementation results feed back as evidence and revision input. This avoids both retroactive conformance claims and the mistaken inference that the implementation defines the theory.

1.4 Design Premises and Contributions

TMPA rests on four premises:

1. **Textual:** governance semantics have a canonical representation readable by people, AI systems, and validators.
2. **Multi-Agent:** execution, review, approval, and supervision remain attributable to separated authorities.
3. **Process:** governance covers the full lifecycle, including rejection, correction, recovery, and archival.
4. **Architecture:** the semantics remain stable across models, runtimes, languages, transports, and storage profiles.

These premises yield four operational commitments:

1. durable textual messages and state carriers;
2. one stable primary carrier per work item and one writer per published object;
3. local seriality with asynchronous multi-stream parallelism;
4. source-preserving aggregation followed by deterministic governance reconstruction.

Writers remain independent. Readers reconstruct the whole.

TMPA does not claim isolated invention of append-only histories, lifecycle machines, provenance, signatures, or role-based authorization. Its contribution is their integration into a minimum-infrastructure process architecture with explicit task identity, responsibility order, lifecycle legality, review separation, conflict preservation, recovery, and machine-testable reader behavior.

The paper makes four contributions:

1. **Governed-work object model.** It separates a stable primary carrier from independently attributable reports, reviews, decisions, transitions, corrections, and recovery evidence.

2. **Multi-serial organizational architecture.** It combines locally ordered single-writer responsibility streams without forcing a global order, then reconstructs a partial-order work graph from their union.
3. **Deterministic governance contract.** It couples source-preserving aggregation with explicit authority, lifecycle, conflict, three-valued judgment, and canonical issue reconstruction.
4. **Evidence-bounded SME-first evaluation.** It separates TMPA theory, Core requirements, the FCoP protocol, reference implementations, the TMPA-guided CodeFlowMu engineering system, and case evidence; publishes C01–C14 results including failures; and identifies the empirical work still required for feasibility and adoption claims.

The paper does **not** contribute a new storage primitive, agent communication protocol, runtime orchestrator, identity provider, factual-verification method, or empirical proof of productivity. FCoP and the pinned CodeFlowMu/WP-13 artifacts are bounded evidence sources; XiaoDian is author-reported lineage only. CodeFlowMu is engineered under TMPA guidance, but neither its implementation nor a passing fixed bundle defines or proves the theory.

The current TMPA–FCoP–CodeFlowMu relationship and the operational software stack are specified in Section 4.8; terminology is fixed in Section 1.5.

1.5 Terminology and Representation Stages

The paper fixes the following vocabulary so that semantic objects, physical storage, message behavior, and reconstructed views are not treated as interchangeable concepts.

Table 2. Canonical terminology and excluded equivalences.

Canonical English term	Fixed Chinese equivalent	Fixed meaning	Not equivalent to
governed work item	受治理工作项	the task, request, decision, or process subject whose responsibility and lifecycle are being governed	one file, one session, or one runtime job
primary carrier	主载体	the stable governance object that anchors the identifier and minimum governing context of one work item	a mutable record that every participant edits
governance object	治理对象	one canonical semantic unit authored by one creator under one responsible role and one writer stream	its storage path, transport envelope, or derived view
textual message	文本消息	the communication function of a governance object when it transfers work, evidence, review, or decision semantics	a separate object class or an ephemeral queue message
state carrier	状态载体	the persistence function through which an object, transition record, or profile-defined location contributes declared or current state evidence	shared mutable application state
source artifact	来源工件	one physical representation or observation of evidence, such as a file, database row, object-store item, or received event	the semantic governance object after validation
source candidate	来源候选	one discovered source artifact presented to the aggregation stage, including malformed or conflicting observations	an accepted authoritative object

Canonical English term	Fixed Chinese equivalent	Fixed meaning	Not equivalent to
canonical candidate set	规范候选集合	the source-preserving, parsed, indexed, and deterministically normalized collection returned by aggregation	the final governance conclusion
writer stream	写者流	the locally ordered sequence of governance objects published by one attributable writer	a global event log or total timeline
source aggregator	来源聚合器	the stage that discovers, preserves, parses, indexes, and normalizes source candidates without deciding governance truth	the governance reader
governance reader	治理 Reader	the deterministic stage that applies a fixed profile to the canonical candidate set	the storage layer, orchestrator, or model runtime
governance graph and issue set	治理图与问题集合	the reconstructed partial-order process view and the canonical unresolved-condition output	the original source evidence or an imposed total order

A single canonical governance object may be realized by different physical profiles. In FCoP, its source artifact is ordinarily a file plus path and event evidence; another profile may use a row, object, or event. Conversely, two source artifacts that declare the same object identifier but contain different canonical content are not two harmless copies: they are conflicting candidates that must be retained and evaluated under the profile. Throughout the architecture and normative chapters, **object** refers to the semantic unit, **artifact** to a physical or published engineering representation, and **view** to a reader-derived result.

2. Problem Definition and Design Requirements

TPMA begins from a distinction between **execution evidence** and **governance evidence**. A runtime trace may prove that a tool call occurred; it does not necessarily prove that the caller was authorized, that an accountable role accepted the work, that the output was independently reviewed, or that a later approval referred to the exact reviewed result. A chat transcript may preserve discussion but still lack stable object identity, lifecycle legality, conflict handling, and deterministic reconstruction. A workflow engine may record node completion while keeping its authoritative state inside an implementation-specific database.

The architecture therefore distinguishes four states:

Table 3. Separation of execution, interaction, business, and governance state.

State category	Primary question
execution state	What is the runtime doing?
interaction state	What did participants exchange?
business state	What does the application currently consider true?
governance state	Which responsibilities, transitions, decisions, conflicts, and evidence are authoritative?

TPMA specifies the fourth category while permitting references to the other three.

Text is selected as the canonical interchange form because it can be interpreted by humans, language models, validators, version-control tools, backup systems, and replacement runtimes. “Textual” does not require Markdown files: a conforming deployment may use a database, object store, or event service, provided that the complete governance meaning has a canonical textual representation.

Responsibility separation is equally central. A nominally multi-agent system does not establish independent governance when one identity plans, executes, reviews, approves, and certifies the same work. TMPA therefore treats roles as scoped authorities rather than prompt labels. Lifecycle is also explicit: governed work moves through profile-defined states; legal transitions, transition authorities, rejected transitions, re-work, and terminal history must remain observable.

The minimum-infrastructure problem is:

How can a small organization obtain attributable, reviewable, recoverable, and machine-checkable multi-agent governance without assuming a dedicated coordination database, message broker, enterprise control plane, or specialist governance team?

The answer is not “files instead of databases” in isolation. It is a complete process structure in which canonical text carries messages and state, one stable primary carrier identifies each governed work item, each writer publishes through a local serial stream, independent streams progress asynchronously, and a reader reconstructs the process graph and issue set.

The target is minimum infrastructure, not zero discipline. Protected storage, declared identity assumptions, backups, permissions, validation, and recovery procedures remain necessary. OECD and IMDA evidence supports the broader observation that SMEs face persistent time, skill, maintenance, and adoption-capacity constraints; it does not establish demand for TMPA specifically [23], [24].

The problem yields eight traceable requirements:

Table 4. TMPA design requirements.

ID	Requirement
DR1	durable canonical textual representation
DR2	stable work identity and single-writer evidence
DR3	locally ordered streams with asynchronous composition
DR4	explicit authority, review separation, and lifecycle legality
DR5	source-preserving aggregation and deterministic reconstruction
DR6	preservation of conflict, invalid evidence, and partial state
DR7	recovery from persistent governance evidence
DR8	minimum mandatory infrastructure with explicit assurance boundaries

DR1–DR7 define the governance semantics. DR8 constrains the deployment claim: TMPA minimizes required coordination infrastructure but does not inherit guarantees from identity, security, consensus, or control-plane systems that are not actually deployed.

3. Related Work and Positioning

TMPA occupies the process-responsibility layer between agent execution and enterprise governance. MCP and A2A address tool use and agent interoperability [1], [2]; identity systems establish principals and delegated authority [19]; gateways and policy engines govern runtime admission; observability and control-plane products inventory agents and collect telemetry. TMPA does not replace these layers. It consumes their identifiers, policy decisions, traces, and artifacts, then represents governed work, review, rejection, recovery, and responsibility in a reconstructable form.

3.1 Architectural Antecedents

Event sourcing and CQRS contribute append-oriented history and the separation of write representations from read models [4]. Git demonstrates immutable content-addressed objects and explicit history [5]. W3C PROV supplies entities, activities, agents, and derivation relations [3]. Lamport established happened-before ordering without assuming a single physical clock [32], while Chandy and Lamport showed how a consistent global state can be determined in a distributed system [33]. TMPA does not claim these foundations as novel. It applies partial-order reasoning and reconstructable global views to a narrower governance problem: one primary carrier per work item, single-writer responsibility streams, explicit lifecycle authority, separation of duties, conflict preservation, and a deterministic governance reader.

The architecture is not equivalent to chat, a shared folder, an ADR collection, a workflow engine, or a single global event log. These mechanisms may store relevant evidence, but they do not by themselves define which object is authoritative, whether an action was in scope, whether a review was independent, or how contradictory evidence remains visible.

3.2 Interoperability and Governance Gaps

The formation of the Agentic AI Foundation reflects the growth of interoperability infrastructure around MCP and related projects [6]. Interoperability, however, is not the same as governance. Kang and Diponegoro analyze MCP, A2A, ACP, ANP, and ERC-8004 against membership, deliberation, voting, dissent preservation, human escalation, and audit/replay requirements, finding that no surveyed protocol expresses a complete governance model [27]. TMPA addresses a narrower subset—work responsibility, lifecycle, review, conflict, and recovery—rather than full community governance.

Open Challenges in Multi-Agent Security emphasizes that collusion, cascading effects, stealth, and oversight failure can emerge at the interaction level even when individual components appear secure [26]. TMPA can preserve attributable evidence and unresolved disagreement for investigation, but it is not a collusion detector.

3.3 Direct Research Neighbors

Auditable Agents separates accountability, auditability, and auditing, and evaluates action recoverability, lifecycle coverage, policy checkability, responsibility attribution, and evidence integrity [17]. TMPA is complementary: it specifies the durable work objects and process reconstruction on which those dimensions can be evaluated.

The IETF agent-audit architecture similarly treats delegation and interactions as auditable events [18]. TMPA can supply a project-local or platform-neutral representation of such events but does not standardize network transport.

Authorization Propagation in Multi-Agent AI Systems identifies transitive delegation, aggregation inference, and temporal validity as unresolved authorization problems [20]. TMPA records assignments and responsibility transitions but does not define a complete recursive delegation calculus.

Policies on Paths argues that runtime governance may depend on the partial execution path rather than static access rules [21]. TMPA preserves process paths as evidence; it does not itself block actions during execution.

Proof-Carrying Agent Actions binds high-value actions to decision-time certificates, approvals, and replay-ready proof [29]. PCAA centers the action decision; TMPA centers the longer governed work item and its reports, reviews, conflicts, corrections, and recovery. A TMPA profile may reference PCAA certificates as execution evidence.

AGENTS SAFE combines risk classification, semantic telemetry, dynamic authorization, interruptibility, anomaly detection, cryptographic tracing, and organizational controls [30]. TMPA chooses a different minimum baseline: readable canonical text and deterministic reconstruction are mandatory, while cryptographic identity and stronger integrity controls are optional named profiles.

Why Do Multi-Agent LLM Systems Fail? derives failure categories from a large trace corpus [22]. TMPA improves observability of responsibility, review, conflict, and recovery, but this paper does not claim reduced failure rates.

From Trajectories to Evidence argues that completed research-agent trajectories require qualification before they become auditable experimental records [36]. TMPA reaches the same boundary from process governance: an execution claim becomes admissible only through attributable objects, required evidence, independent review, lifecycle-valid decisions, and retained exceptions. TMPA does not provide domain truth verification; it governs how claims and their verification evidence are admitted and reconstructed.

The incremental contribution is therefore not a new storage primitive or a complete control plane. It is the combination of a durable textual message/state plane, one-task-one-primary-carrier, local single-writer

streams, asynchronous composition, source-preserving aggregation, and deterministic read-side governance reconstruction under a minimum-infrastructure profile.

3.4 Comparative Synthesis and Research Gap

The related work falls into five neighboring lines. Their boundaries clarify the gap addressed by TMPA:

Table 5. Comparative positioning against neighboring research lines.

Research line	Principal contribution	TMPA relation and boundary
MCP and A2A [1], [2]	interoperable context, capability, task, and message exchange for tool or agent interaction	TMPA may reference these interactions but defines longer-lived responsibility, review, conflict, and recovery evidence
W3C PROV, event sourcing, and CQRS [3], [4]	derivation, append-oriented event history, and read-model construction	TMPA specializes these mechanisms into governed work identity, authority, lifecycle legality, and deterministic issue reconstruction
Auditable Agents and the IETF audit architecture [17], [18]	accountability dimensions, distributed audit records, context, and later investigation	TMPA provides a substrate-neutral governed-work graph but does not define network audit-context propagation or attestation
authorization propagation, path policies, and proof-carrying actions [20], [21], [29] NIST AI RMF and AGENTS SAFE [30], [31]	decision-time authorization and runtime enforcement for delegated paths or certified actions organizational risk identification, controls, monitoring, assurance, and accountability	TMPA preserves the authorized work process and its outcomes; it does not replace execution-time mediation TMPA is a narrower evidence architecture that can support such programs but does not constitute a complete risk-management framework
distributed ordering and snapshots [32], [33]	causality without a global clock and consistent global-state observation	TMPA specializes partial-order reconstruction for governance evidence; it does not provide consensus or distributed snapshot transport
multi-agent failure and evidence qualification [22], [36]	failure taxonomies and conversion of execution trajectories into auditable records	TMPA specifies admission, review, lifecycle, and reconstruction semantics but does not verify domain truth

No reviewed neighbor combines all of the following as one minimal-infrastructure process contract: a stable primary carrier, single-writer responsibility streams, asynchronous composition without a forced total order, explicit authority and lifecycle semantics, preservation of invalid and conflicting evidence, and deterministic reconstruction of both a governance graph and an issue set. This is the specific research gap claimed by A1.0. The claim is architectural and comparative; it is not a priority claim over every possible unpublished or proprietary system.

4. TMPA Architecture

TMPA defines governance semantics, not a runtime component. Its architecture specifies which governance facts must be represented, how responsibility and lifecycle are expressed, and how independent evidence is reconstructed into an authoritative view. Storage, transport, scheduling, and model behavior remain implementation concerns unless a TMPA profile explicitly binds them.

4.1 Governance Object

A **governance object** is the smallest independently attributable semantic unit in TMPA. It may represent a task, report, review, approval, issue, lifecycle transition, role assignment, recovery action, or another document type published by a protocol profile. The object is distinct from its physical **source artifact**: FCoP may encode it as a file and path observation, while another profile may use a database row, object-store item, or event.

The terms **textual message** and **state carrier** describe functions of an object rather than additional object classes. An object acts as a textual message when it transfers governed work or evidence, and it acts as a

state carrier when its content, transition evidence, or profile-defined location contributes to lifecycle reconstruction.

Every governance object contains or identifies:

- a stable object identifier;
- a document type;
- one creator identity;
- one responsible role;
- a stream identifier and sequence number;
- a creation time;
- a lifecycle profile and declared state;
- typed references to related objects;
- canonical textual content;
- integrity evidence.

A published governance object is immutable. Correction does not erase or rewrite the original object; it creates a new object that supersedes, rejects, qualifies, or resolves the earlier one. Multiple byte-identical source observations may refer to the same object without changing its meaning; the same identifier paired with different canonical content is a conflict, not an update.

The lifecycle state declared by an object is the state associated with that object under its profile at publication. The current authoritative state of governed work is reconstructed from the valid object set, accepted transitions, and profile rules. It is not obtained by mutating an earlier published object or by selecting the most recent timestamp.

4.2 Document Types

TMPA Core does not impose a universal business-document taxonomy. Each implementation profile instead publishes a finite document-type registry.

Each registry entry defines:

- the type name and version;
- the governance responsibility represented by the type;
- permitted creator roles;
- required fields;
- permitted reference relations;
- the applicable lifecycle profile;
- validation rules.

Document types must not overlap ambiguously in authority. An execution report, for example, does not implicitly serve as its own independent review or approval. When a deployment permits an exception to separation of duties, the exception and its approving authority must be represented explicitly.

4.3 Role and Authority Model

A TMPA role is a governance authority with a defined scope. It is not merely a prompt label or natural-language persona.

Each role definition identifies:

- a stable role identifier;
- permitted object types;
- permitted lifecycle actions;
- separation-of-duty constraints;
- the authority that assigns the role;
- the assignment's validity period and revocation state.

An object's `role` field declares the authority under which the creator acted; it does not create that authority. A reader validates the claim against an active role assignment and the applicable policy profile.

A participant may occupy more than one role only when the implementation profile explicitly permits the combination. A deployment claiming independent review must prohibit the same identity from acting as both executor and reviewer for the same governed result unless a recorded and authorized exception applies.

In an enterprise identity profile, the logical role is bound separately to a verifiable agent or workload identity, the human or organizational principal that remains accountable, the credential used for the action, the delegation source and scope, and the validity or revocation state. TMPA Core records and validates the governance claim; it does not issue credentials or enforce recursive permission attenuation.

4.4 Lifecycle Model

A lifecycle profile consists of:

- a finite state set S ;
- an initial state s_0 ;
- a terminal-state set F ;
- an action set A ;
- a transition relation $T \subseteq S \times A \times S$;
- an authorization function $\text{Auth}(\text{role}, \text{action}, \text{object})$;
- a validation function $\text{Valid}(\text{object}, \text{transition})$.

A transition is accepted only when:

1. its source state matches the current authoritative state;
2. the transition is defined by T ;
3. the initiating role is authorized;
4. required references and preconditions are satisfied;
5. the transition evidence passes schema and integrity validation.

An illegal or unauthorized transition does not alter the authoritative lifecycle state. The attempt remains observable through a rejection, issue, alert, or equivalent profile-defined record rather than being silently discarded or repaired.

4.5 Textual Messages, Single-Writer Streams, and Asynchronous Parallelism

TMPA's write plane combines a stable work carrier, single-writer objects, local seriality, and asynchronous composition.

For every governed task or work item t , a task-oriented profile defines one stable primary carrier c_t . The carrier establishes the identifier and minimum governing context of the work. Acceptance, execution reports, reviews, decisions, corrections, and recovery records are separate objects that reference c_t ; they do not become additional mutable copies of the task. "One task, one carrier" therefore means one stable primary reference point, not one document that every participant edits.

Let A be the set of responsible writers. Every published object has exactly one writer, and each writer $a \in A$ publishes an independently attributable serial stream:

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

The sequence inside S_a is authoritative local order. Every object has a positive sequence number, and $(\text{stream_id}, \text{sequence})$ identifies its position within that writer's responsibility history. Creation time is informative but not authoritative for stream order.

At observation time τ , the available candidate collection may contain different prefixes of different streams:

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

The functions $k_a(\tau)$ need not advance together. One participant may publish a task while another is offline; a report may appear before an independent review; several writers may progress concurrently. This is how multiple serial streams form asynchronous parallelism. TMPA does not require all participants to share a clock, remain online together, or commit to one global event log.

Within-stream predecessor relations provide local order. Explicit references and profile-defined lifecycle dependencies provide cross-stream causal edges. If two objects have neither a within-stream relation nor a profile-defined cross-stream dependency, they remain concurrent and incomparable.

Single-writer objects and separate responsibility streams remove the primary **semantic** shared-write conflict: several agents do not compete to edit one authoritative record. They do not eliminate every storage-level contention, filesystem race, or infrastructure failure; profiles must still define atomic publication, duplicate handling, and recovery behavior.

The write model can be summarized as:

One task has one primary carrier. One writer owns each published object. Each writer remains serial. Multiple streams progress asynchronously to form parallel collaboration.

4.6 Read-Side Aggregation and Governance Reconstruction

TMPA's read plane has two conceptually separate stages: **source aggregation** and **governance reconstruction**.

Let O_τ be the finite collection of source candidates visible at observation time τ . A source-preserving aggregator A discovers source artifacts, retains source identity and bytes, parses candidate envelopes, indexes identifiers and references, and applies deterministic normalization needed by the reader:

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

Aggregation does not decide which claim is true, silently repair a conflict, invent missing evidence, or convert arrival order into governance order. Its purpose is to construct the complete canonical candidate set C_τ available to the governance reader while retaining the provenance of every source candidate.

Let P be a fixed rule profile containing schemas, type rules, role assignments, lifecycle rules, relation semantics, canonicalization rules, conflict policy, and output-normalization rules. The governance reader then computes:

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

where G_τ is the canonical reconstructed partial-order process and governance graph and I_τ is the canonical issue set. G_τ represents workflow progress, responsibility, lifecycle, review, approval, rejection, recovery, and audit relations while preserving local stream order, explicit cross-stream dependencies, and concurrency among incomparable objects. It is not an authoritative total timeline.

For brevity, later sections may write $R_P(O)$ for the composed pipeline $R_P(A(O))$. The primary determinism requirement is permutation invariance. For every permutation π of the same canonical candidate collection:

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

The equality applies to canonical serialization of both G and I , not to incidental in-memory order or diagnostic formatting. Delayed arrival changes the currently available set and may legitimately change a view from partial to authoritative or disputed; different enumeration orders of the same set must not change the result.

Determinism proposition. Let O be a finite source multiset and P a fixed profile. Assume that: (1) source normalization is a pure function of source identity and covered bytes; (2) duplicate classification, validation, authority checks, lifecycle checks, and issue identifiers are functions of canonical object values and P ; (3) graph edges are derived only from within-stream sequence and profile-declared relations; and (4) graph and issue serialization use published deterministic ordering and tie-break rules. Under these conditions, the composed operation $R_P(A(O))$ is invariant to source enumeration order.

Proof sketch. Aggregation maps any enumeration of 0 to the same canonical indexed candidate multi-set because indexing and duplicate classification depend on canonical source values rather than discovery position. Per-object validation is order-independent. Set-level checks—duplicate identifiers, stream gaps, missing references, prohibited cycles, authority conflicts, and lifecycle conflicts—are computed over the same canonical sets and relations. The accepted node set and directed edge set are therefore identical for every permutation. Canonical issue identifiers and deterministic ordering produce the same I; canonical graph serialization and a stable tie-break used only for representing incomparable nodes produce the same serialized G. Consequently, every permutation yields byte-equivalent canonical outputs. This argument establishes permutation invariance under the stated profile contract; it does not prove semantic truth, profile correctness, resistance to compromised trust roots, or equality across different evidence sets. A mechanized proof and executable corpus remain required for stronger assurance.

A reconstructed subject or subgraph is classified as:

- **authoritative** when required evidence is valid and no unresolved integrity, authority, lifecycle, ordering, or required-reference issue affects the conclusion;
- **partial** when required evidence is missing or a stream or dependency is incomplete;
- **disputed** when multiple valid but incompatible governance claims remain unresolved;
- **quarantined** when a profile-defined integrity, identity, duplicate-ID, or prohibited-cycle condition excludes the affected evidence from authoritative reconstruction.

Authentication is an orthogonal assurance label. An object or view may be structurally authoritative under TMPA Core while remaining unauthenticated under a stronger identity profile; it must not then be presented as authenticated integrity.

TMPA requires **conflict preservation**: valid contradictory objects remain represented until a new authorized resolution object exists. It also requires **evidence preservation under extension**: adding candidate evidence does not erase prior source evidence. TMPA does not assume monotonicity of governance status under arbitrary set extension, because newly added valid evidence may legitimately change a previously authoritative view into a partial, disputed, or quarantined one.

Read-side reconstruction is therefore not the whole architecture; it is the stage that converts durable textual messages and asynchronous serial streams into a coherent process and governance result. A deterministic topological serialization may be generated for interchange or display, but that serialization does not add governance order between incomparable nodes. C11 operationalizes source-aggregation and reconstruction determinism; C03, C04, C09, C10, and C12 exercise identity, ordering, dependency, cycle, and conflict-preservation behavior. A mechanized proof of the full reconstruction properties remains future work.

4.7 Integrity and Signature Evidence

TMPA separates three properties that are often conflated:

1. **Attribution**: an object declares a creator and responsible role.
2. **Integrity**: modification of a published object can be detected.
3. **Authenticated integrity**: the object is cryptographically bound to a verified identity or key.

TMPA Core requires attribution and integrity evidence. A deployment may claim authenticated integrity only when it applies a trusted identity, signature, and key-management profile.

An integrity record identifies:

- the canonicalization profile;
- the hash algorithm;
- the content digest;
- predecessor or referenced digests when required by the profile;
- an optional signature algorithm;
- an optional key identifier;
- an optional signature value.

A role label is not a cryptographic signature. A digest without a trusted identity binding can detect modification but cannot prove who created the object. A valid signature proves origin and integrity under the deployed trust model; it does not prove that the signed content is semantically true.

4.8 Guidance Relation, Historical Lineage, and Operational Stack

The orientation map in Section 1 distinguishes historical lineage, current guidance and implementation, and end-to-end operation. This section fixes the software boundary:

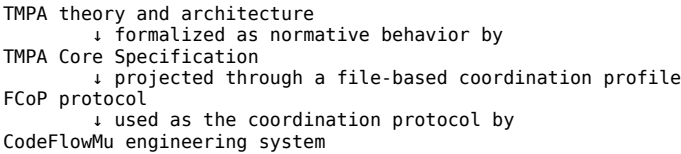


Figure 4. Current guidance and implementation relationship among TMPA, Core, FCoP, and CodeFlowMu.

TMPA theory supplies the architectural direction; Core makes the required behavior normative; FCoP supplies the independently specified coordination protocol; and CodeFlowMu is the running engineering system that uses FCoP to implement TMPA-guided work [37]. Neither FCoP nor CodeFlowMu defines the broader TMPA architecture, and a fixed-bundle product result does not establish universal conformance.

The operational FCoP stack is:

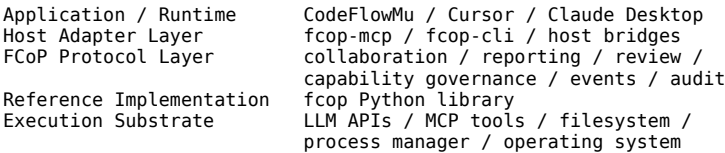


Figure 5. Operational FCoP stack and its separation from TMPA theory.

TMPA is not a runtime component in this stack: it guides the semantics that the stack is intended to realize. The fcop package is the reference implementation of FCoP; CodeFlowMu is the broader engineering and runtime environment. This separation preserves guidance authority, historical accuracy, and conformance boundaries.

5. Research Design and Evaluation Method

This work is a design-science architecture study following the artifact-centered guidance of Hevner et al. and the problem-to-communication sequence of Peffers et al. [34], [35]. It constructs a governance artifact, examines its internal properties, instantiates it through a reference profile, and evaluates bounded engineering evidence. It is not a causal trial, a performance benchmark, or a claim of ecosystem adoption.

5.1 Design-Science Procedure

The study follows six linked stages:

Table 6. Design-science procedure and research-question traceability.

Stage	Research operation	Output and research question
problem diagnosis objectives for a solution	analyze coordination gaps in chats, logs, workflows, and the XiaoDian lineage derive minimum-infrastructure, responsibility, lifecycle, conflict, and reconstruction requirements	DR1–DR8; RQ1 DR1–DR8; RQ1–RQ2
artifact construction demonstration	define objects, streams, authority, lifecycle, and reconstruction rules trace the FCoP protocol and the pinned TMPA-guided CodeFlowMu and WP-13 paths to TMPA concepts	TMPA architecture; RQ2 implementation feasibility; RQ3; XiaoDian lineage excluded from evaluation

Stage	Research operation	Output and research question
evaluation	examine invariants and counterexamples; execute the Reference Reader and product evidence matrix	structural and bounded engineering results; RQ1–RQ3
communication	publish versioned paper, specification, implementation report, fixtures, and evidence boundaries	reproducible review surface

The units of analysis are a governance object, a governed work item, a reconstructed view, and a deployment profile. These units prevent product features, individual files, and architecture-level claims from being treated as interchangeable evidence.

5.2 Evidence and Claim Protocol

Evidence is classified as **specified**, **implemented**, **demonstrated**, or **independently adopted**. Conformance evidence additionally records fixed source revisions, evidence hashes, execution prerequisites, expected and actual outputs, and one of four product verdicts: PASS, PARTIAL, NOT RUN, or FAIL. A fixture-oracle match is reported separately from product execution.

This protocol limits three common inference errors. Historical artifacts are not retroactively presented as purpose-built conformance tests; prerequisite failures are not converted into passes or product failures; and author-produced cases are not treated as independent adoption. The procedure supports structural and implementation-feasibility claims, but it does not support causal claims about productivity, cost, reliability, or organizational outcomes.

5.3 FCoP as Reference Instantiation

FCoP (File-based Coordination Protocol) is the reference profile used to examine whether TMPA semantics can be realized on an ordinary project-visible filesystem. The current guidance and implementation relation is:

TMPA theory → Core requirements → FCoP protocol profile → CodeFlowMu engineering system

Figure 6. Reference-instantiation path used in the engineering evaluation.

FCoP is not identical to TMPA. It realizes a file-based subset through named textual artifacts, lifecycle directories, append-oriented transition evidence, schemas, runtime tools, and adapters [7]–[12]. CodeFlowMu is the engineering system developed under TMPA guidance that uses FCoP as its coordination protocol in a persistent multi-role development and work environment [28], [37].

Table 7. Mapping from the FCoP protocol profile to TMPA concepts.

FCoP element	TMPA interpretation
artifact identifier or filename stem	stable transport-visible object identity
sender and recipient	declared writer and intended responsibility transfer
lifecycle path and transitions	profile-defined state and transition evidence
references, ref_task, subject_ref	typed causal or governance links
REVIEW plus approval evidence	independent verdict and decision
supersedes	correction without destructive rewrite
archive/history	retained terminal evidence

Under this profile, one task has one stable transport-visible carrier; reports, reviews, issues, approvals, and corrections remain separate artifacts. Writers publish independently, while readers inspect the available source set. The published runtime specification, schemas, packages, adapters, governance middleware, ADRs, and documentation establish implementation and distribution paths, not broad adoption [8]–[16].

5.4 Case and Corpus Procedure

The author reports the historical lineage as XiaoDian AI → early TMPA → FCoP → CodeFlowMu, followed by feedback into current TMPA formalization [25]. The XiaoDian source has no fixed public snapshot and is therefore used only to disclose design provenance, not as an evaluated observation. Public, pinned

FCoP, CodeFlowMu, and WP-13 artifacts provide the evidence used in RQ3. Their operational results supply bounded revision input without becoming the authority that defines TMPA.

The live CodeFlowMu browser [13] is retained as an illustrative interface reference only: its build and dataset are not immutably identified, so it is excluded from the reproducible evaluation. CodeFlowMu claims in this paper rely on the locked I1.0 evidence package [28]. The NL2SQL view remains an illustration of the intended object chain, not a representative production benchmark or a scored observation.

The consolidated corpus preserves historical mapped evidence and adds a direct S1.0 product run [28]. The evaluation has two tracks. The **analytical track** examines invariants, counterexamples, equal-input determinism, illegal transitions, conflicts, and three-valued judgments. The **engineering track** fixes the normative inputs by raw-byte hash; locks product and dependency revisions; inventories and hashes evidence; executes each criterion through the CodeFlowMu product Reader; validates the result envelope; and reports Reference Reader fixtures separately from product behavior. The WP-13 path is treated as evidence-admission behavior, not as S1.0 conformance evidence or proof that an agent no longer hallucinates [36].

The resulting evidence supports bounded structural and implementation feasibility: durable project-visible coordination, role-separated review, lifecycle gates, conflict preservation, deterministic reconstruction, archive preservation, and restart-related recovery. It establishes a 14/14 product result across 71 mandatory assertions only for the exact S1.0 bundle and declared CodeFlowMu V1.8.0 evidence revision. It does not establish lower comparative cost, representative SME performance, independent adoption, portability across deployment profiles, or conformance for arbitrary inputs and deployments.

5.5 Evidence Dependence and Analysis Boundary

The cases are **not independent replications**. TMPA, FCoP, CodeFlowMu, XiaoDian, the Reference Reader, and the C01–C14 mapping share an author and an engineering lineage. FCoP and CodeFlowMu evidence therefore tests projection and implementation paths inside one related ecosystem; aggregating their test counts does not increase the number of independent observations.

The evidence set was selected for traceability to the current architecture and for availability of inspectable artifacts, not through random sampling. The analysis uses descriptive counts, invariant checks, counterexamples, and artifact tracing; it performs no null-hypothesis test, effect-size estimate, or causal comparison. Earlier FAIL, PARTIAL, and NOT RUN results remain in their immutable I0.6–I0.8 histories and in the frozen S1.0 candidate baseline; A1.0 reports the later I1.0 external product run separately rather than rewriting those earlier conditions.

The submission-level unit of inference is consequently limited to: (a) architectural coherence under the stated assumptions; (b) execution of identified mechanisms in pinned author-controlled artifacts; and (c) bounded demonstration of evidence admission, role separation, recovery, and audit behavior. Organizational effectiveness, lower cost, reliability improvement, and general adoption remain hypotheses for independent study.

6. Canonical Reconstruction Contract

A TMPA implementation may store evidence in different substrates, but it must expose a canonical textual projection sufficient for deterministic reconstruction. The reader receives a set of **source candidates**, not a trusted ordered log. It parses and validates candidates, preserves duplicate observations, distinguishes conflicting content under the same identifier, applies profile rules, and emits:

1. a canonical candidate set;
2. a partial-order process and responsibility graph;
3. an authoritative issue set;
4. statuses such as authoritative, partial, disputed, quarantined, or rejected.

A conforming reconstruction procedure performs the following logical steps:

```
RECONSTRUCT(source_candidates, profile):
```

```

parse and canonicalize candidates
  validate schemas, identifiers, types, and integrity evidence
  preserve duplicate observations and conflicting candidates
  group valid objects by writer stream and check local sequence
  construct edges from sequence, references, lifecycle, and profile relations
  validate role authority, separation of duties, and lifecycle legality
  detect missing references, prohibited cycles, and unresolved conflict
  derive lifecycle, responsibility, review, and recovery state
  emit canonical graph and issue set

```

Figure 7. Substrate-neutral canonical reconstruction procedure.

The procedure does not silently “repair” invalid evidence. A malformed object, illegal transition, missing reference, duplicate identifier with different content, or contradictory review remains observable through the issue set. A later authorized resolution may change the authoritative interpretation without erasing the earlier evidence.

6.1 Determinism

Let (S) be a fixed final set of source candidates and (P) a fixed profile. The reader is deterministic when every enumeration or delivery permutation of (S) produces the same canonical candidate set, graph, statuses, and issue set.

This is a set-based requirement, not a claim that every intermediate view is complete. Before all evidence arrives, the reader may report partial or disputed state. Once the available source set is identical, output must be invariant to discovery order.

The proof obligation follows from four constraints:

- canonicalization and validation depend on candidate content and profile rules, not enumeration order;
- within-stream order is derived from explicit sequence evidence;
- cross-stream order is derived from explicit references and lifecycle relations;
- conflicts are preserved as sets and issues rather than resolved by “last arrival wins.”

The current paper provides a proof sketch and executable fixture oracle, not a mechanized proof. The S1.0 C11 fixture enumerates declared permutations and the CodeFlowMu V1.8.0 product path records PASS for that fixed fixture. This is a product-level verdict for the declared bundle, not a proof over arbitrary graphs, encodings, filesystems, or hostile platforms.

6.2 Integrity, Identity, and Truth

Digest and signature evidence must be interpreted narrowly. A digest can reveal covered-byte modification. A verified signature can bind bytes to a key under an external trust model. Neither proves that the signed statement is factually correct.

A malicious, compromised, or mistaken participant can publish a schema-valid, lifecycle-valid, digest-consistent, and even correctly signed report whose claims are false. TMPA Core can identify the object, preserve it, test its declared authority, and relate it to independent review; factual verification requires tool receipts, reproducible outputs, independent data sources, or domain-specific validation outside Core.

This distinction produces three separate questions:

Table 8. Separation of integrity, authenticated identity, and factual truth.

Question	Required mechanism
Were these bytes changed?	digest or tamper-evident storage
Which authenticated principal signed them?	signature, key, and identity profile
Are the claims true?	independent verification or domain evidence

TMPA must not claim the guarantee of a surrounding identity, policy, or cryptographic system unless the deployment actually verifies that evidence under the corresponding profile.

7. Evaluation Results

The separately maintained TMPA Core Specification S1.0 is the sole normative source for the current C01–C14 criteria and all SHALL/MUST clauses. The Implementation Case Report I1.0 reports the current bounded S1.0 engineering-evidence baseline. I0.6–I0.8 remain immutable historical baselines for their earlier inputs and product revisions. This section evaluates the research questions without reproducing the specification.

7.1 Findings by Research Question

Table 9. Findings, evidence, and inference boundaries by research question.

Research question	Finding and evidence	Boundary
RQ1: governance-state sufficiency	ordinary conversation and execution surfaces do not, by themselves, preserve enough explicit authority, lifecycle, conflict, and recovery state for deterministic governance reconstruction; supported by problem diagnosis, DR1–DR8, and object/reconstruction analysis	no comparative field experiment has measured failure rates against alternative architectures
RQ2: minimum architecture	stable carriers, single-writer streams, explicit authority and lifecycle, typed references, three-valued judgment, and source-preserving deterministic reconstruction form a coherent minimum contract; supported by invariants, counterexamples, the determinism proof sketch, and Core S1.0	the proof is not mechanized, and minimality is an architectural argument rather than a universal lower-bound proof
RQ3: engineering feasibility and boundary	FCoP supplies the protocol profile; CodeFlowMu V1.8.0 executes its own product Reader against the exact S1.0 bundle with 14/14 PASS across 71 mandatory assertions; the archive locks source, inputs, commands, results, and integrity records; WP-13 supplies separate evidence-admission observations	all execution remains author-produced; independent adoption, comparative SME burden, cross-profile portability, and behavior outside the fixed bundle are not established

The result is therefore strongest for architectural coherence and fixed-bundle implementation feasibility. It is weaker for organizational effectiveness and ecosystem generalization, which require independent, representative, and comparative evidence.

7.2 Conformance-Domain Summary

The current product result is summarized by architectural domain; exact criterion definitions remain in Core S1.0 § 10.2.

Table 10. CodeFlowMu V1.8.0 product results for the exact S1.0 bundle.

Domain	Criteria	Product-level result
object, immutability, provenance, and integrity	C01, C02, C03, C08	4 PASS
authority and lifecycle	C05, C06, C07	3 PASS
ordering, reference, and conflict	C04, C09, C10, C12	4 PASS
determinism, recovery, and history	C11, C13, C14	3 PASS

This 14/14 result is a criterion-level product verdict for one exact input bundle and declared product revision. It is not a declaration of authenticated governance, correctness for arbitrary source sets, full protocol validity, or independent certification.

7.3 S1.0 Author-Run Product Baseline

I1.0 publishes `tmpa-i1.0-codeflowmu-v1.8.0-s1.0-evidence-20260811.zip`, whose outer SHA-256 is `9b92b06c3e46c8019362f55075be4ed066cb7a9c0d9859945b6c3b7de8840d04` [28]. The run fixes TMPA Core S1.0 at frozen candidate commit `942cbb097eb3d662662f96a2269818ec9d7ca2ed` and CodeFlowMu V1.8.0 at evidence commit `c1e1f724293e8048fc3a956b6f6df8cf83f54830`. The normative inputs are byte-identical to the frozen GitHub objects; their aggregate bundle digest is `sha256:f98764987760cdc8ac356b1265fc98485f33345e7d6ffc8575ccb059ddd34daa`, and

the aggregate result digest is sha256:0f0f642449db1853371861751a7a8ea36dce00013f53e32012a5e4dae45f4c39.

The result envelope records `product_reader_called: true` and `reference_reader_called: false`. CodeFlowMu's internal TMPA Runtime suite passes 24/24 tests; Runtime records 1,522 passed / 0 failed / 1 skipped; Shell records 791/791; Protocol validation and type checking exit successfully; and the FCoP reference implementation locked at commit `da79dfe9f597c9e422ce9edec22157f915a21` records 1,210 passed / 2 skipped. These supporting suite counts are context, not additional C01–C14 observations.

Table 11. Product-level S1.0 verdicts for the I1.0 baseline.

Verdict	Criteria
PASS	C01, C02, C03, C04, C05, C06, C07, C08, C09, C10, C11, C12, C13, C14
PARTIAL	none
NOT RUN	none
FAIL	none

The archive contains 889 manifest-covered files, including exact inputs, product source, dependency locks, command records, raw results, pre-fix failures, remediation history, and a reduced clean-machine reproducer. Publication review recomputed all fourteen manifest digests, fourteen result digests, 71 mandatory assertions, the input digest, and aggregate result digest without discrepancy. This establishes package consistency and traceability; it does not constitute an independent product rerun or certification.

7.4 Interpretation: Closed Implementation Gaps, Open Validation Gap

The I0.6 baseline exposed missing product projection plus concrete C02 and C07 failures and unexecuted C08, C11, and C12 paths. I0.7 and I0.8 progressively closed those observed gaps. The I1.0 baseline rebinds the product path to the stable S1.0 machine identities, preserves the frozen candidate's historical product NOT RUN baseline, and registers the later CodeFlowMu V1.8.0 exact-version run separately. No historical verdict or criterion was weakened.

```
exact S1.0 normative bytes
  ↓ hash verification
CodeFlowMu V1.8.0 product Reader
  ↓ C01–C14 execution
schema-valid 14/14 result
  ↓ publication
locked evidence archive + integrity manifest
```

Figure 8. Product-evidence path established by I1.0.

The dominant remaining gap has therefore moved from product projection and stable machine identity to validation independence and external scope. The result remains author-run, related-lineage, fixed-bundle demonstrated evidence. The CodeFlowMu evidence commit was local-only at capture, although the archive includes its source snapshot and patch. Independent reruns, independent adoption, alternative-profile implementations, representative SME deployment, comparative cost, and measurements of organizational outcomes remain necessary before broader claims are supportable. WP-13 remains a bounded evidence-gating case and does not establish S1.0 conformance or hallucination elimination.

8. Discussion, Limitations, and Threats to Validity

TMPA's contribution is a process-governance contract, not a complete enterprise agent platform. It makes work identity, responsibility, review, lifecycle, conflict, and recovery explicit at publication time and reconstructs them from durable evidence. Author-run FCoP evidence indicates that a useful subset can operate in an ordinary project environment without a mandatory broker or coordination database.

The architecture does not replace:

- enterprise IAM, credential issuance, or key management;
- runtime gateways, policy engines, or admission control;

- model evaluation and factual verification;
- OTel, SIEM, CMDB, GRC, or enterprise agent inventory;
- database transactions, distributed consensus, or Byzantine fault tolerance;
- legal compliance programs.

These systems may supply identifiers, policy decisions, execution receipts, and protected storage to TMPA. Their guarantees remain external unless verified and referenced.

8.1 Evidence Maturity

The paper separates four claim levels:

1. **specified** — a rule, schema, or criterion is published;
2. **implemented** — code executes the rule;
3. **demonstrated** — an operational case exhibits the behavior;
4. **independently adopted** — an external system relies on and validates it.

The architecture and criteria are specified. FCoP supplies implementation evidence. The locked Code-FlowMu and WP-13 artifacts supply bounded demonstration evidence. XiaoDian is retained as author-reported provenance only; independent adoption is not established.

The author is also the originator and principal developer of the evaluated systems. This gives access to detailed artifacts but creates self-evaluation and selection risk. The corpus therefore labels author-produced evidence, non-gating field evidence, fixture-only oracle results, and product-level verdicts separately.

8.2 SME-First Claim

“SME-first” is an operational scope, not a claim that every SME has the same needs or that TMPA is unsuitable for larger organizations. The lightweight profile assumes limited platform and operations capacity. Larger deployments may preserve the same semantics through databases, object stores, identity systems, replication, and enterprise control planes.

The decisive empirical question remains whether the benefits justify the discipline and resource cost. Required measurements include:

Table 12. Empirical program required to test the SME-first feasibility claim.

Experiment	Required evidence
deployment burden	dependencies, installation steps, setup time, first-team time, backup, migration
reconstruction	delayed and permuted streams, intermediate partial state, byte-equivalent final output
fault and recovery	duplicates, illegal transitions, missing references, tampering, restart, recovery time
human inspectability	ability to identify ownership, missing evidence, review status, and next responsibility
adoption discipline	onboarding, bypass, correction, fallback to chat, perceived burden, continued use
representative workload	latency, CPU, memory, storage growth, conflict rate, and recovery time

The current corpus provides a normalized baseline but not these complete measurements.

8.3 Digital Employee Profile and Privacy

The term **digital employee** is used only as an engineering label for a persistent AI work role. It does not assert employment status, legal personality, consciousness, human intention, autonomous organizational authority, or replacement of an accountable human or organization.

A future profile may define job scope, acceptance of delegated work, handoff, suspension, reassignment, and retirement. These are application semantics, not changes to Core.

Textual governance improves inspectability but may expose sensitive information. Deployments should minimize content, separate secrets from governance metadata, apply access control and encryption, and define retention and erasure procedures. Immutability of governance history does not require public readability of sensitive payloads.

8.4 Threats to Validity

Construct validity. C01–C14 operationalize governance structure and reconstruction behavior. They do not directly measure factual truth, human usefulness, productivity, or organizational accountability. A criterion pass must not be interpreted as success on those external constructs.

Internal validity. The author selected the architecture, systems, cases, and evidence mappings and also ran the baseline. Version differences and retrospective mapping from historical artifacts to later criteria can affect the result. Exact normative bytes, fixed revisions, hashes, explicit prerequisites, and separate product and fixture verdicts reduce—but do not remove—this risk.

External validity. The principal implementation is a file-based profile, the case set is small, and much of the observed execution is project-local. The findings may not transfer unchanged to database-backed, highly distributed, regulated, adversarial, or high-throughput deployments.

Conclusion validity. The verdict counts are descriptive results for a selected fixed bundle. They are not statistical estimates, evidence of general or independently certified conformance, or causal comparisons with chat, event-log, workflow-engine, or database alternatives.

Reproducibility. The current corpus is author-produced. It now has a stable public repository path, executable reproduction command, environment declaration, and SHA-256 manifest. An independent rerun remains necessary.

8.5 Limitations and Falsification Conditions

TMPA's central claims should remain open to disconfirmation:

Table 13. Falsification conditions for the central TMPA claims.

Claim	Evidence that would weaken or refute it
equal source sets permit deterministic reconstruction	conforming readers produce different canonical graph or issue outputs for the same profile and source set
durable textual evidence improves responsibility recovery	controlled recovery tasks perform no better than relevant alternatives, or cannot identify responsibility and missing evidence reliably
the architecture can operate with minimal infrastructure	required correctness depends on an undeclared coordination database, broker, global clock, or centralized mutable log
the SME-first profile is operationally feasible	deployment, maintenance, storage, or human-discipline costs outweigh measured governance benefits in representative SME use
the semantics are portable across profiles	independent implementations cannot preserve object, authority, lifecycle, conflict, and reconstruction semantics across different storage substrates

8.6 Publication and Reproducibility Boundary

A1.0 is the stable theoretical architecture paper. Core S1.0 is the current normative source, while Implementation Case I1.0 reports the current author-run S1.0 engineering-evidence baseline. I0.6–I0.8 preserve their earlier historical states. The paper may summarize those companion artifacts but must not silently redefine their meaning, merge Reference Reader and product verdicts, or expand a fixed-bundle result into general conformance.

For the current system boundary, TMPA theory and architecture guide CodeFlowMu engineering; Core S1.0 fixes the behavior being evaluated; and FCoP supplies the coordination protocol used by CodeFlowMu. Historical implementation feedback may motivate later theory or specification revisions, but it does not make CodeFlowMu or FCoP the authority that defines TMPA.

The public evidence archive contains the exact normative inputs, CodeFlowMu evidence source, commands, results, regression history, dependency locks, reduced reproducer, and integrity manifest. Before an independently validated claim, at least one unaffiliated rerun remains necessary. Low-resource deployment measurements remain a separate empirical requirement for the SME feasibility claim.

9. Conclusion

TMPA is an **SME-first, minimal-infrastructure textual-message multi-agent asynchronous process architecture**. Text carries durable work and state; each work item has one stable primary carrier; each published object has one writer and belongs to a local serial stream; independent streams progress asynchronously; and aggregation plus deterministic reading reconstructs the partial-order process, responsibility, lifecycle, conflict, recovery, and audit view.

The architecture arose through **practice → method → theory**: XiaoDian AI exposed the multi-role coordination problem, early TMPA identified the textual asynchronous method, FCoP extracted and matured its reusable file-coordination and review subset, and CodeFlowMu engineered that direction into a persistent development and work system. FCoP and CodeFlowMu results then fed evidence back into the current formalization of objects, invariants, assurance boundaries, and conformance criteria. Current authority runs in the other direction—TMPA theory and Core requirements guide continued CodeFlowMu engineering through the FCoP protocol. The early pipeline establishes origin, not retroactive Core conformance.

A1.0 answers RQ1 and RQ2 at the architectural level and provides a pinned, author-run baseline for RQ3. Under the I1.0/S1.0 evaluation, CodeFlowMu V1.8.0 records **14 product PASS, 0 PARTIAL, 0 NOT RUN, and 0 FAIL** across 71 mandatory assertions for the exact bundle. Its internal TMPA Runtime suite passes 24/24; Runtime records 1,522 passed / 0 failed / 1 skipped; Shell records 791/791; and the locked FCoP reference implementation records 1,210 passed / 2 skipped. The separate S1.0 Reference Reader passes all fourteen synthetic fixtures, but those results remain distinct from product execution. The locked evidence package improves traceability without creating independent validation.

The decisive remaining question is RQ3: whether an organization can sustain the responsibility, review, recovery, and evidence benefits of TMPA in an ordinary project environment at acceptable resource and discipline cost. Product projection and a locked S1.0 evidence package now exist; the broader claim still requires low-resource deployment and recovery measurements, baseline comparisons, representative use, alternative-profile evidence, and independent reproduction. TMPA also does not by itself establish authenticated identity, strong isolation, protected storage, Byzantine resilience, factual truth of participant claims, or ecosystem adoption.

Artifact Availability

The current author-run S1.0 evidence package is `tmpa-i1.0-codeflowmu-v1.8.0-s1.0-evidence-20260811.zip`, published with Implementation Case I1.0 and available from the public evidence path. Its SHA-256 is `9b92b06c3e46c8019362f55075be4ed066cb7a9c0d9859945b6c3b7de8840d04`. Earlier S0.4–S0.6 corpora and I0.6–I0.8 reports remain immutable historical baselines. All product evidence is author-produced unless and until independently rerun.

Data Availability

The paper does not publish production business data. The worked NL2SQL material is an illustrative governance reconstruction rather than a verbatim production export. Conformance fixtures and selected implementation evidence are included in the author-produced corpus; any future public release must preserve redaction, version, provenance, and checksum information.

Competing Interests and Author-Produced Evidence

The paper author is also the originator and principal developer of TMPA, FCoP, and CodeFlowMu. This relationship creates self-evaluation and selection risks. The paper separates specified, implemented, demon-

strated, and independently adopted claims; the current baseline is author-run and does not constitute independent validation or ecosystem adoption.

Author Contributions

Zhu Wei: conceptualization, architecture design, methodology, software and protocol development, evidence curation, investigation, writing, and maintenance of the public artifacts. This single-author contribution statement does not imply independent validation of the evaluated systems.

Ethics and Privacy Statement

This architecture study does not report a human-subject experiment and does not publish production business data. The worked examples and conformance fixtures are technical artifacts. Any future study involving employees, organizational behavior, user performance, interviews, or sensitive operational records must separately address applicable review, informed consent, access control, minimization, retention, and redaction requirements.

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](https://github.com/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” live public demonstration. <https://demo.chedian.cc/>. Observed 2026-07-29. Because the build and dataset are not immutably identified, this source is illustrative only and is excluded from the reproducible evaluation; locked CodeFlowMu claims use [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 / XiaoDian AI Project. “多 AI 角色协同架构规划” [Multi-AI Role Collaboration Architecture Plan]. Author-reported private project archive, March 2026. No fixed public snapshot is available. It is cited only to disclose the author's account of design lineage and is excluded from the evaluated corpus, RQ results, and conformance claims.
- [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 × Application Unified Architecture,” docs/TMPA-GOVERNANCE.md. GitHub, 2026. <https://github.com/joinwell52-AI/codeflowmu/blob/main/docs/TMPA-GOVERNANCE.md>.