AGIF Fabric v1 Whitepaper

A Bounded, Governed Multi-Cell Software Fabric for Local Adaptive Workflows

Adaptive General Intelligence Fabric (AGIF)

From Embedded Neural Firmware to a Resource-Sovereign, Multi-Cell Research Programme with Empirical Proof

Danish Z. Khan

21st March 2026

Table of Contents

Abstract

Adaptive General Intelligence Fabric (AGIF) is proposed as a governed intelligence architecture in which capability is organized through many specialized cells rather than a single monolithic model. Grounded in the resource-aware and safety-oriented discipline of Embedded Neural Firmware (ENF), this paper presents AGIF v1 as the first bounded software instantiation of that broader vision. AGIF v1 is not presented as a claim of achieved artificial general intelligence. Rather, it is presented as a testable architectural proof that intelligence can be decomposed into cells with explicit roles, local state, skill state, trust state, policy boundaries, and bounded adaptive capacity, and that such cells can coordinate through tissues, compact descriptor exchange, and a shared workspace under explicit governance.

The core premise is that intelligence need not improve only through brute-force scaling. AGIF is motivated by a different thesis: systems may improve through better organization, stronger learning discipline, tighter memory discipline, better compression, better routing, better reuse, and better systems optimization. In that sense, AGIF is architecturally broad but operationally resource-aware. AGIF v1 has been built, locally verified, and benchmarked. Across three system classes—a flat baseline, a multi-cell fabric without adaptation, and a multi-cell fabric with bounded adaptation and descriptor sharing—the results show that the adaptive fabric achieves 1.000 task accuracy versus 0.583 for the flat baseline; unsafe or misaligned action rate falls to 0.000 for all multi-cell classes versus 0.500 for the flat baseline; descriptor reuse causally improves alias-heavy cases from 0.500 in the flat baseline to 1.000 in the adaptation class; adaptation yields 9× greater accuracy gain per retained kilobyte than adding governance structure alone; all 14 frozen falsification thresholds are met on the committed benchmark suite; and the full fabric runs on a single Apple M4 MacBook Air within a 16 GB RAM envelope.

The resulting claim is that AGIF v1 establishes a real software architecture, a reproducible evidence base, and a credible research path toward later simulator, edge, and hardware realization. Long-run evidence includes real MSI soak windows executed on a separate MSI Windows soak machine rather than on the MacBook Air target machine: a 24-hour run and a 72-hour run. The 72-hour run completed 1,690 cycles in 72 h 4 m 9 s, kept repeated-cycle runtime memory fixed at 1,507,328 bytes, raised the average score from 0.988750 over the first 100 cycles to 0.995000 over the last 100, preserved reuse-backed descriptor usefulness at 0.979 correctness on opportunity cases versus 0.375 for the cold non-reuse case, ended with only 5,866 warm-tier retained bytes and 7 active descriptors, recorded 7,036 repeated-lane authority approvals with 0 repeated-lane vetoes, and passed split/merge, memory-pressure, routing-pressure, trust/quarantine, and replay/rollback stress lanes. A separate post-closure extension bundle then closed the largest remaining bounded proof gaps: a deterministic 40-case near-capacity finance lane demonstrated one organically triggered, governance-approved split with mean queue age reduced from 16.800 to 0.300 and mean end-to-end latency reduced from 24.550 to 8.050 without accuracy loss; a governed descriptor skill graph recorded explicit transfer approvals, abstentions, denials, and provenance; and a second bounded POS domain added five deterministic cases in which finance-origin descriptors causally improved two POS outcomes under explicit transfer approval. The combined record now covers the original six finance benchmark cases, one deterministic 40-case organic-load lane, and five deterministic

POS cases. The paper does not claim MacBook Air-only long-run endurance, open-world generality, or AGI, because the long-run soak artifacts came from the MSI soak machine and the 72-hour run included one recovered WinError 5 manifest-write interruption with incomplete final resume bookkeeping.

Keywords: Adaptive General Intelligence Fabric (AGIF), governed intelligence fabric, Embedded Neural Firmware (ENF), Tasklet Cells, bounded memory growth, multi-cell coordination, AI safety

1. Introduction

1.1 The Problem with Scale-Only AI

Much contemporary discussion of advanced machine intelligence assumes that capability growth will arise primarily from centralized scale: larger models, larger datasets, larger training runs, larger serving infrastructures, and increasingly expensive deployment stacks, as reflected in the modern scaling-law and compute-optimal training literature (Hoffmann et al., 2022; Kaplan et al., 2020). That trajectory has produced important results, but it does not exhaust the design space. In particular, it says relatively little about how intelligence should be engineered when connectivity cannot be assumed, privacy must remain local, auditability matters, resource budgets are finite, and systems must coordinate persistent work without collapsing into opaque monoliths.

These constraints matter because they directly shape what can be remembered, how coordination is achieved, how errors propagate, and whether a system remains governable under failure. Any architecture intended for persistent local or enterprise use must therefore treat bounded state, bounded adaptation, memory discipline, recovery, and containment as first-class design variables rather than as afterthoughts.

1.2 The AGIF Hypothesis

The AGIF hypothesis is straightforward: intelligence can grow through better organization, learning discipline, memory discipline, compression, routing, and reuse—not only through more compute.

AGIF is therefore proposed as a different class of AI architecture. It is built around many bounded cells rather than a single monolithic intelligence, specialization rather than general-purpose overload, shared state rather than isolated modules, governed adaptation rather than uncontrolled drift, local usefulness rather than datacenter dependence, and optimization-first growth rather than brute-force expansion.

A concise statement of the thesis is as follows: AGIF should improve by becoming better organized, not merely by becoming bigger.

1.3 ENF as the Starting Discipline

Earlier Embedded Neural Firmware (ENF) work is important because it begins from a disciplined engineering posture (Khan, 2025a, 2025b, 2025c). ENF treats embedded intelligence as a bounded artifact: small, offline by default, tightly constrained in timing and memory, and governed by explicit prohibitions against cloud dependence, open-ended drift, and hidden behavioral change. That discipline makes ENF credible as an engineering architecture for narrow intelligence under hard limits. It does not, however, provide a route to general intelligence, because the same rules that produce safety and predictability also prohibit genuine capability growth.

AGIF departs from ENF precisely at the point where growth becomes necessary, but it carries ENF's discipline forward rather than abandoning it. The resource-sovereign posture, the fail-closed contract, and the governance-first orientation are all inherited from ENF. What AGIF adds is bounded adaptation, descriptor exchange, tissue formation, elastic lifecycle management, and reviewed memory promotion.

1.4 Contributions of This Work

This paper makes four contributions. First, it defines AGIF v1 as a governed software architecture rather than as an unconstrained multi-agent system or an AGI claim. Second, it specifies the complete cell-tissue-workspace-governance model, including identity, trust, activation rules, descriptor exchange, replay, rollback, trust checks, and quarantine, and presents a locally verified implementation of every component. Third, it introduces AGIF v1 as an experiment in resource-sovereign learning, in which memory growth is reviewed, compressed, and bounded rather than automatically accumulated. Fourth, it provides a fully reproducible, falsifiable evidence path for persistent, structured, stateful, auditable, multi-step workflows.

1.5 Scope Boundaries

This paper does not claim that AGI has been achieved, that large-scale emergence has been demonstrated, that real-world embodied intelligence has been proven, that safety at production hardware scale has been verified, or that the final ENF hardware vision has been realized.

The claim is narrower and more specific: AGIF v1 is a bounded software architecture showing that a governed fabric of specialized cells, tissues, shared workspace, descriptor exchange, bounded adaptation, and resource-sovereign memory discipline can outperform flatter baselines on structured multi-step tasks. That claim is now locally verifiable through the reproducibility package.

2. Background: ENF and the Transition to Fabric Thinking

2.1 Definitions of Key Terms

The following terms are used throughout this paper in a precise engineering sense.

Cell — the basic AGIF unit: a bounded node comprising role, identity, local computation, local memory, skill state, trust state, policy boundaries, activation rules, and a controller that decides how scarce resources are allocated.

Tissue — a local cluster of cells that specializes in related functions, exchanges descriptors more frequently, and provides the first meaningful organizational scale above a single cell.

Shared workspace — the coordination layer in which selected task records, summaries, commitments, descriptors, and handoff state become visible beyond immediate local scope. It is a bounded shared-state surface, not a hidden centralized cloud service.

Skill descriptor — a compact, transferable summary of learned capability, including task regime, performance range, provenance, and trust material. It is the bounded exchange object that makes low-bandwidth transfer plausible.

Fabric population — the total set of cells known to the system, including dormant or inactive blueprints.

Active runtime population — the subset of cells currently instantiated, consuming live compute and memory, and participating in execution.

Need signals — bounded homeostatic indicators, including overload, uncertainty, novelty, redundancy, memory saturation, trust degradation, and coordination failure, that create pressure for learning, compression, rerouting, dormancy, or containment.

Memory promotion — the governed process by which short-lived evidence becomes accepted long-term memory only after review, verification, compression, and policy checks.

Containment — the ability to isolate, downgrade, revoke, or quarantine cells whose behavior becomes unsafe, unstable, or misaligned.

2.2 ENF Invariants and Their Ceiling

Embedded Neural Firmware (ENF) establishes the disciplinary starting point for AGIF. Across the ENF whitepaper and technical notes, its core invariants are explicit: offline operation, no cloud dependency, bounded memory and energy use, minimal attack surface, deterministic control, and prohibition of open-ended post-deployment training (Khan, 2025a, 2025b, 2025c,

2025d). These commitments produce real engineering advantages, including reduced privacy leakage, simplified attestation, and long-lived deployment in sensitive environments.

They also impose a hard capability ceiling. A system that cannot acquire new tasks, create new abstractions, or transfer what it learns across domains remains narrow, no matter how elegantly it is packaged. In that sense, ENF is highly credible as a bounded engineering architecture, but it is intentionally not an architecture for growth.

2.3 ENF-Sync, ENF-Swarm, and the Coordination Insight

ENF-Sync and ENF-Swarm extend the ENF posture into coordinated populations without abandoning the embedded discipline from which ENF begins (Khan, 2025a, 2025d). They show that narrow offline nodes can coordinate through bounded signaling, deterministic communication scope, and swarm-safe organization. This step is important because it demonstrates that intelligence-like coordination need not imply cloud dependence, mutable infrastructure, or open network negotiation.

However, these coordination patterns remain intentionally non-adaptive. No genuine task acquisition occurs, no reviewer-governed learning loop emerges, and no open-ended capability growth is permitted. A swarm of sealed specialists remains a swarm of sealed specialists. The architectural insight is therefore significant but limited: coordination alone is not sufficient to produce a governed, improving fabric.

2.4 Relation to Tasklet Cells

Tasklet Cells are the first practical software-cell foundation in the AGIF direction (Khan, 2026a). They show that cells can exist as real software artifacts: contract-driven, verifier-backed, local-first, and fail-closed. They provide the first serious software proof that bounded intelligence can be packaged, verified, and embedded without relying on cloud-service assumptions.

What Tasklet Cells do not yet prove is full AGIF fabric behavior: tissue formation, persistent workspace coordination, bounded inter-cell adaptation, elastic cell populations, reviewer-governed memory promotion, and descriptor-based reuse across a governed fabric. AGIF v1 is the broader fabric-level software hypothesis built above that layer.

2.5 Tasklet Cells vs. AGIF Fabric v1: A Detailed Technical Comparison

A central question for the AGIF research programme is how AGIF v1 differs from the Tasklet Cell foundation that preceded it. This section provides a complete technical comparison. The goal is not to position one system as superior in all respects—Tasklet Cells remain an important engineering contribution with distinct strengths—but to show precisely where the architectural boundary between the two systems lies, why that boundary matters, and what capabilities it unlocks.

2.5.1 Intelligence Level Framework

The following classification is an engineering taxonomy of what a system can do with knowledge over time, not a claim of proximity to AGI.

Level	Class	What it means
1	Static logic	Hard-coded rules, no memory, deterministic only
2	Stored memory	Executes fixed rules against retained local state; no learning between deployments
3	Adaptation	Updates behavior from experience within a deployment
4	Coordination	Multiple specialists collaborate on shared problems through handoffs
5	Governed fabric	Adaptation + coordination + multi-tier memory + trust + elastic lifecycle + governance
6	Open-world AGI	Not claimed and not built in either system

Tasklet Cells operate at Level 2. They have local memory, execute real inference, and run offline without cloud dependency. They apply learned rules from their bundle and return structured outputs. However, they are sealed at deployment: the model cannot acquire new capabilities, cells cannot share knowledge with one another, there is no governance layer that approves or rejects actions, and there is no coordination between cells beyond message passing at the application layer (Khan, 2026a).

AGIF Fabric v1 operates at Level 5. It retains all Level 2 capabilities and adds coordination, reviewed memory promotion, bounded adaptation, trust-checked descriptor exchange, elastic lifecycle management, layered governance, replay and rollback, and an explicit separation between fabric-scope and active-runtime-scope populations.

2.5.2 Feature Comparison

Capability	Tasklet Cells	AGIF Fabric v1
Bounded local execution	Supported	Supported
Offline / no cloud dependency	Supported	Supported
Verifier-backed artifact model	Supported	Supported (inherited)
Fail-closed contract	Supported	Supported (inherited)
Stored local memory	Supported	Supported (four-tier)
Multiple specialized roles	Not supported	Supported (6 tissues, 10+ cells)
Shared workspace coordination	Not supported	Supported (5 traced handoffs per case)
Skill descriptors (knowledge transfer)	Not supported	Supported (causally proven on 3 benchmark cases)

Bounded adaptation (learning from cases)	Not supported	Supported
Reviewed memory promotion	Not supported	Supported (hot/warm/cold/ephemeral tiers)
Governance (authority approval)	Not supported	Supported
Replay and rollback	Not supported	Supported (1.000 replay determinism)
Quarantine of unsafe cells	Not supported	Supported
Elastic lifecycle (split/merge/hibernate)	Not supported	Supported (7 governed lifecycle states)
Need-driven adaptation signals	Not supported	Supported (7 signal kinds)
Trust scoring on knowledge exchange	Not supported	Supported
Population model (logical vs. active)	Not supported	Supported (128 logical / 24 active)
Multi-day behavioral stability	Not evaluated	Supported (24-hour and 72-hour MSI soak, 1,690 cycles)

2.5.3 Architectural Comparison

At the architectural level, Tasklet Cells and AGIF Fabric v1 share the same foundational philosophy—intelligence as a bounded, local, governed artifact—but differ fundamentally in scope and organizational structure.

Tasklet Cell architecture is a single-unit model. One cell bundles a model, a contract, a verifier, and a runtime. The cell receives input, executes inference, and returns structured output. Its memory is local and fixed at bundle creation. It communicates with the outside world through a narrow, well-defined contract surface. This makes it highly predictable, attestable, and deployable in environments where isolation is a hard requirement. The cell is a sealed specialist (Khan, 2026a).

AGIF Fabric v1 architecture is a coordinated multi-unit model organized around the loop:

cells → tissues → descriptors → workspace → governance → bounded adaptation

Cells specialize into tissues. Tissues coordinate through a shared workspace. The workspace holds handoffs, route-of-custody records, and intermediate state across a multi-step pipeline. Governance approves or rejects learning events and structural changes. Memory is tiered and reviewed before promotion. The fabric population can be larger than the active runtime population, enabling logical scope without proportional memory cost.

The critical architectural distinction is this: a Tasklet Cell knows nothing about other cells, other cases, or prior learning events. An AGIF cell operating within a fabric tissue has access to its

own prior approved descriptors, can see workspace signals from peer cells in the same tissue, and can have its decisions reviewed and corrected by governance before those corrections are persisted. This creates a fundamentally different improvement trajectory. Tasklet Cells improve through redeployment with updated bundles. AGIF cells improve through reviewed, in-fabric adaptation without redeployment.

2.5.4 Code-Level Evidence

The following modules exist in the AGIF v1 workspace and have no counterpart in the current Tasklet Cell codebase.

Module	Location	Size	What it provides
Routing engine	intelligence/fabric/routing.py	62,275 bytes	Nine-factor routing: trust, load, descriptors, need signals, and utility
CLI layer	intelligence/fabric/cli.py	25,629 bytes	Five frozen commands: init, run, status, replay, and evidence
Finance tissues	intelligence/fabric/domain/	60,618 bytes	Six tissues, all five handoffs, and the full finance document workflow
Lifecycle engine	intelligence/fabric/lifecycle/	—	Seven lifecycle states, governed transitions, anti-thrash controls, and a lineage ledger
Memory manager	intelligence/fabric/memory/	—	Four tiers, reviewer scoring on six dimensions, compression, and garbage collection
Needs engine	intelligence/fabric/needs/	—	Seven need-signal kinds, expiry, resolution, and traceability
Governance engine	intelligence/fabric/governance/	—	Authority approval, vetoes, trust-band history, and quarantine
Workspace	intelligence/fabric/workspace/	—	Shared handoff state and route-of-custody tracking per case
Registry	intelligence/fabric/registry/	—	Fabric and active runtime populations as separate stores
CHANGELOG	CHANGELOG.md	41,052 bytes	A full phase-by-phase record of architectural decisions

2.5.5 Performance Comparison: Verified Benchmark Numbers

The original benchmark suite compared three system classes across a six-case finance document workflow. The post-closure extension bundle then added a deterministic 40-case near-capacity finance organic-load lane and a second five-case POS operations suite. The relevant comparison for understanding the gap from Tasklet Cell-level deployment to AGIF Fabric v1 is therefore twofold: the baseline six-case finance benchmark shows why governed adaptation outperforms flat and no-adaptation designs, and the extension bundle shows that the same fabric can benefit from organic split/merge and governed cross-domain transfer.

Metric	Flat baseline (Tasklet-level analogue)	AGIF Fabric v1	Delta
Task accuracy	0.583	1	+71.5% relative improvement
Unsafe / misaligned release rate	0.5	0	Eliminated completely
Governance success rate	0	1	New capability
Descriptor reuse rate	0	1	Entirely new capability
Catastrophic forgetting	N/A (sealed)	0	Zero forgetting under adaptation
Replay fidelity	1	1	Maintained
Multi-day behavioral stability	Not tested	0.994434 average score, 72 hours, 1,690 cycles	Entirely new capability
Runtime working set	—	1,507,328 bytes (repeated-cycle lane)	Runs within a 12 GB cap on MacBook Air

The 71.5% relative accuracy improvement from 0.583 to 1.000 is not driven by adding more parameters or more training data. It is driven by three architectural additions: governance, which catches the 50% unsafe release rate present in the flat baseline; tissue coordination, which enables correct holds on anomaly and mismatch cases; and reviewed descriptor reuse, which resolves alias-heavy vendor-normalization cases that the flat baseline and the no-adaptation fabric both fail. Each contribution is isolable from the three-class benchmark comparison and causally traceable through the route-of-custody logs.

2.5.6 Long-Run Stability: What Tasklet Cells Cannot Prove

Tasklet Cells, by design, do not accumulate state across cases. Each invocation is independent. Long-run stability for a Tasklet Cell therefore means that the model does not degrade, which is largely guaranteed by its sealed nature. Long-run stability for AGIF Fabric v1 means something different and more demanding: the fabric must remain useful, governable, and memory-bounded across hundreds or thousands of repeated work cycles while retaining and reusing approved knowledge without bloat or drift.

The real 72-hour soak on the MSI machine demonstrates this. Across 1,690 repeated finance cycles over 72 hours, the fabric maintained an average cycle score of 0.994434, kept runtime memory fixed at 1,507,328 bytes throughout the repeated-cycle lane, recorded zero raw-log promotions, achieved 5,411,218 bytes of duplicate compression gain, held 281 of 281 reuse-assisted high-value alias cases under governance review rather than auto-releasing them, and ended with only 5,866 bytes of warm-tier retained memory. Descriptor reuse correctness on alias-opportunity cases remained at 0.979 throughout, compared with 0.375 for cold non-reuse

encounters. These numbers show that the fabric did not drift, bloat, or degrade. They also show that it improved through reuse rather than reset.

2.5.7 When to Use Each System

Scenario	Recommended system	Reason
Single-function offline inference, hard isolation required	Tasklet Cells	Sealed, attestable, minimal coordination overhead
Privacy-critical narrow task in a constrained embedded device	Tasklet Cells	Minimal attack surface, no shared state
Multi-step workflow requiring coordination and memory	AGIF Fabric v1	Tissues and shared workspace are required
Recurring document processing in which past corrections should improve future cases	AGIF Fabric v1	Reviewed descriptor reuse is the core mechanism
Safety-critical workflow requiring human-approvable governance holds	AGIF Fabric v1	Layered authority and a governance success rate of 1.000
Local system that must remain auditable over long operating periods	AGIF Fabric v1	Replay, rollback, route-of-custody, and 72-hour soak evidence
System requiring capability growth without redeployment	AGIF Fabric v1	Bounded adaptation with a reviewer approval path

Tasklet Cells are the right choice when isolation, attestability, and sealed behavior are the primary requirements. AGIF Fabric v1 is the right choice when the system must coordinate, improve, and remain governable over time. The two systems are not in competition. They occupy the same architectural lineage, with AGIF v1 representing the next layer built above the Tasklet Cell foundation.

3. Problem Statement

From a general-intelligence perspective grounded in skill-acquisition efficiency, ENF reaches an intentional ceiling. Chollet (2019) argues that intelligence should be understood not merely as performance on fixed tasks, but as the efficiency with which a system can acquire new skills across a scope of problems. By that standard, ENF's strengths are also its limit: sealed models cannot learn new tasks after deployment, the communication and update posture excludes open-ended post-deployment learning, and the architecture is deliberately designed to prevent uncontrolled adaptation (Khan, 2025a, 2025b, 2025c). Those choices are prudent from a safety and engineering standpoint, but they also prevent the system from demonstrating bounded capability growth.

This creates the central problem that motivates AGIF. A bounded, offline, governance-oriented architecture is valuable, but if it cannot acquire, retain, review, and reuse new capability under controlled conditions, it remains narrow regardless of how robust, private, or efficient it may be.

The question, therefore, is not whether ENF is well engineered for its intended scope; it is. The question is whether the core discipline that makes ENF credible can be preserved while selected constraints are relaxed just enough to permit governed improvement.

AGIF addresses that problem by allowing bounded local learning, bounded peer coordination, compact descriptor exchange, tissue formation, partial shared state, governed adaptation, and bounded memory growth. The central hypothesis is deliberately conservative: a fabric of bounded learners may exhibit more useful adaptive capability than isolated sealed nodes while still remaining governable. The aim is not to abandon ENF's discipline, but to test whether a carefully limited transition from sealed artifacts to governed adaptive fabrics can produce measurable gains on structured, multi-step workflows.

The ENF-to-AGIF transition can therefore be stated as a controlled architectural relaxation rather than a wholesale break.

Dimension	ENF keeps	AGIF v1 relaxes	Why necessary
Execution model	Offline-first, no cloud dependency	Limited inter-cell coordination in software	Shared state is required to test fabric behavior
State and control	Bounded state; minimal attack surface	Governed writable local state for selected cells	Bounded adaptation cannot exist without controlled state change
Learning	Sealed behavior after deployment	Limited local adaptation with replay and rollback	Skill acquisition cannot be tested without bounded learning
Communication	Compact coordination messages only	Descriptor exchange and workspace signals	Transfer requires shareable summaries of capability
Memory philosophy	Knowledge fixed at deployment	Reviewed promotion, compression, and intentional forgetting	Fabric learning cannot remain bounded without memory discipline
System goal	Narrow embedded usefulness	Structured multi-step software workflows	AGIF v1 needs a reproducible testing ground for first proof

This transition defines the problem statement for the present paper. AGIF v1 is not proposed as open-world AGI, unrestricted continual learning, or a rejection of ENF. It is proposed as the smallest software relaxation of ENF-like discipline that still permits a meaningful test of whether governed specialization, bounded adaptation, and reviewed knowledge reuse can outperform flatter sealed baselines. In other words, the problem is to find the minimal architectural departure from ENF that allows adaptive capability to emerge without giving up boundedness, auditability, replayability, rollback, or containment.

The research problem is therefore precise: can a governed fabric of bounded cells improve persistent, structured, and stateful workflows through reviewed local adaptation and compact

inter-cell knowledge transfer while remaining resource-sovereign, memory-disciplined, and locally verifiable? AGIF v1 is the first bounded software attempt to answer that question.

4. AGIF v1 Scope, Architectural Claim, and Non-Claims

4.1 The Full AGIF Vision

AGIF is a governed intelligence fabric, not a single giant model. Its purpose is to organize intelligence through many specialized cells that can learn, share, coordinate, and improve over time without depending on brute-force scaling, wasteful infrastructure, or uncontrolled memory growth.

The full AGIF vision therefore remains broad. It includes cells, tissues, skill descriptors, shared workspace, local and fabric memory, bounded adaptation, utility and motivation, governance, replay, rollback, trust, quarantine, and growth over time. What changes in practice is not the vision itself, but how that vision is instantiated, constrained, and evaluated at a given stage of the research programme.

In that sense, AGIF should be understood as an architectural direction rather than as a single software release. The broader vision concerns how intelligence might be organized under bounded resources, governed memory, and controlled coordination. Different AGIF phases may realize different subsets of that vision, but they remain part of the same underlying programme: intelligence built through disciplined structure rather than through monolithic scale alone.

4.2 What AGIF v1 Is Meant to Prove

AGIF v1 preserves the functional architecture of the broader AGIF vision, but proves it first in software. It is the first bounded software instantiation intended to test whether the central ideas of the fabric can run end to end under governed, locally verifiable conditions.

A successful AGIF v1 shows that:

- a governed multi-cell fabric can run end to end;
- cells can coordinate meaningfully;
- useful knowledge can be shared compactly;
- memory can grow without uncontrolled bloat;
- learning can remain bounded and recoverable;
- structured multi-step work can improve over flatter baselines; and
- large fabric scope can coexist with a small active runtime footprint.

These are architectural claims, not philosophical ones. AGIF v1 is meant to establish that a governed fabric is not merely conceptual language, but a working software architecture with observable coordination, bounded adaptation, reviewed memory promotion, replay and rollback support, and a verifiable relationship between logical scope and active runtime population.

Just as importantly, AGIF v1 is designed to prove these properties in a reproducible setting. Its role is not to maximize breadth, but to make the architecture testable. The software-first proof therefore focuses on structured, persistent, multi-step workflows in which coordination, reuse, memory discipline, governance, and recovery can be measured directly.

4.3 Relation to AGI Criteria and Claim Boundaries

Broader discussions of general intelligence commonly frame it in terms of generalization or skill acquisition across a wide scope of tasks, rather than success on a fixed bounded workflow (Chollet, 2019; Legg & Hutter, 2007). AGIF v1 does not prove AGI. It does not claim open-world generality, unrestricted continual learning, embodiment, autonomous objective formation, or human-level reasoning across arbitrary domains. Those stronger claims are outside the scope of the present paper.

What AGIF v1 does address is a bounded subset of concerns that often appear in broader discussions of general intelligence. It addresses whether capability can improve through controlled adaptation rather than only through redeployment, whether knowledge can be transferred in compact form rather than through full retraining, whether multiple specialists can coordinate through shared state and governed handoffs, whether memory can persist without uncontrolled accumulation, and whether such a system can remain replayable, recoverable, and containable while doing so.

The claim boundary is therefore explicit. AGIF v1 is not a proof of general intelligence; it is a bounded architectural proof of governed adaptive fabric behavior. It demonstrates a software system in which specialization, coordination, descriptor exchange, reviewed memory promotion, bounded adaptation, governance, replay, rollback, and containment coexist within one locally verifiable design.

This distinction matters because the paper is not arguing that AGI has been built in miniature. It is arguing that some properties often treated as prerequisites for broader intelligence—coordination, transfer, bounded adaptation, memory discipline, and recoverability—can be realized together without abandoning offline operation, local control, or explicit governance. That is the scope of the present claim.

Accordingly, the non-claims are as important as the claims. AGIF v1 does not show that a governed fabric can solve arbitrary tasks, replace foundation models, achieve open-ended self-improvement, or satisfy any universal definition of intelligence. It shows something narrower and more defensible: that a governed multi-cell architecture can improve structured workflow performance through bounded adaptation and compact knowledge reuse while remaining resource-aware, auditable, and recoverable.

The broader AGIF vision remains open beyond this paper. AGIF v1 is one bounded proof step within that larger research programme.

5. System Architecture

5.1 Cell Model and Runtime State

Each AGIF cell is defined as a bounded unit with both stable identity and governed runtime behavior. At the blueprint level, a cell carries stable identity, lineage tracking, role family and role name, a trust and policy envelope, a resource budget, allowed tissues, a utility profile, and descriptor publication and consumption limits. The resource budget is expressed through explicit fields such as activation_cost_ms, working_memory_bytes, idle_memory_bytes, and descriptor_cache_bytes.

A cell also carries live runtime state. In AGIF v1, that state includes active_task_ref, workspace_subscriptions, loaded_descriptor_refs, and current_need_signals. Blueprint definition and live runtime state are maintained as distinct structures. This distinction is essential because it allows identity, lineage, and policy to persist even when a cell is not active. A dormant cell therefore consumes identity storage, but it does not consume the active runtime budget.

This separation between blueprint and runtime is one of the core architectural disciplines of AGIF v1. It allows the fabric to preserve long-horizon organizational structure without forcing every known unit to remain instantiated in memory at all times.

5.2 Tissues, Shared Workspace, and Fabric Memory

Tissues are local clusters of cells that specialize in related functions. They provide the first meaningful scale of organization above the single cell. Rather than treating all cells as peers in an undifferentiated pool, AGIF organizes them into functional tissues that exchange descriptors more frequently, coordinate more tightly, and contribute to recurring workflow patterns.

The shared workspace is the coordination surface across those tissues. It holds cross-cell task state, handoffs, and replayable work context. It is not long-term memory. Instead, it serves as the bounded coordination layer through which state becomes visible beyond immediate local scope. In AGIF v1, the shared workspace records route-of-custody, intermediate commitments, and handoff state across the multi-step workflow.

AGIF v1 implements six finance-document workflow tissues:

Tissue	Role family	Core function
finance_intake_routing_tissue	classifier, router	Classify the document and select the routing target
finance_extraction_tissue	extractor, normalizer	Extract and normalize fields
finance_validation_correction_tissue	validator, corrector	Apply rules and consult correction memory
finance_anomaly_reviewer_tissue	anomaly, reviewer	Detect anomalies and decide whether to hold or release

finance_workspace_governance_tissue	workspace guard, governance	Enforce policy and approve or reject outcomes
finance_reporting_output_tissue	reporter, formatter	Generate output and an audit summary

The handoff sequence is fixed and fully traced: **intake/routing → extraction → validation/correction → anomaly/reviewer → workspace/governance → reporting/output**. All five handoffs are recorded in the shared workspace for every case.

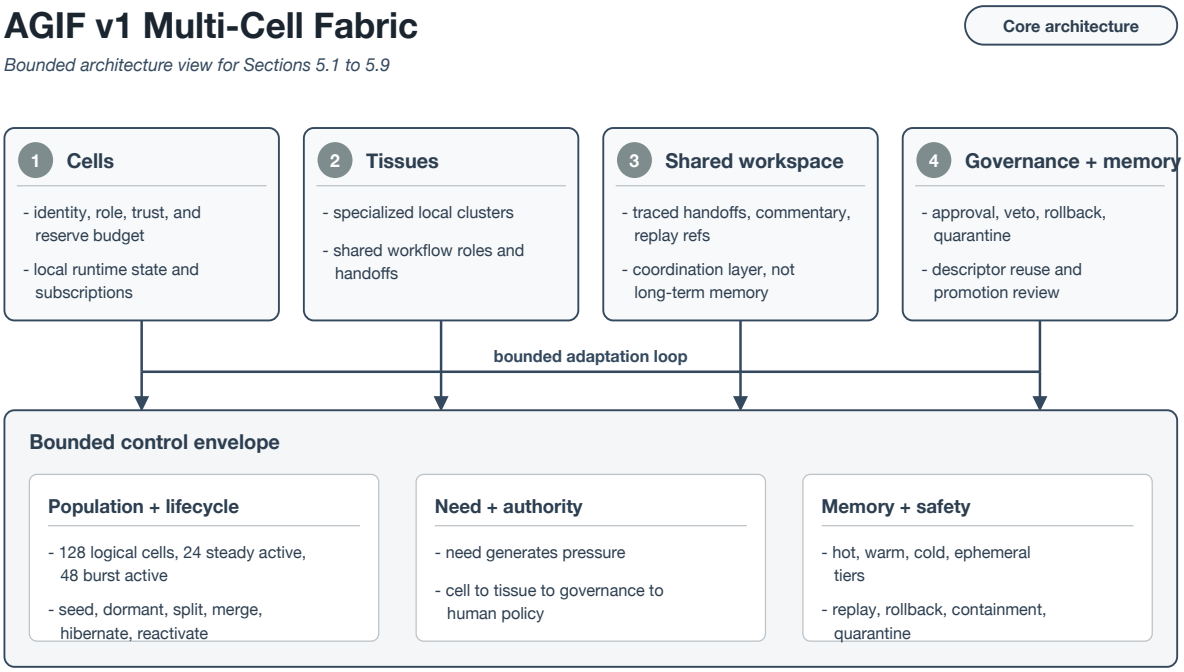

Figure 1 illustrates the AGIF v1 multi-cell fabric architecture and the bounded adaptation loop.

5.3 The Population Model: Fabric vs. Active

AGIF separates fabric population from active runtime population. This distinction allows the fabric to be large in logical scope without requiring all cells to remain active in RAM simultaneously.

AGIF v1 proves this separation concretely:

- **Logical population cap:** 128 cells
- **Steady active population:** 24 cells, enforced
- **Burst active population:** 48 cells, with automatic return to 24 after consolidation
- **Dormant cells:** preserve identity and lineage without consuming the active runtime budget

All of these boundaries are enforced with fail-closed vetoes in the lifecycle engine. This means that population elasticity is not informal or opportunistic; it is governed, bounded, and auditable.

The result is a fabric that can preserve broad organizational scope while remaining scientifically credible within a laptop-class runtime envelope.

5.4 Elastic Cell Lifecycle and Governed Transitions

AGIF manages cells through a governed elastic lifecycle rather than through a binary active/inactive model. In AGIF v1, the lifecycle states are as follows:

State	Meaning
seed	Blueprint admitted, not yet placed
dormant	Known to the fabric, not consuming active runtime
active	Running and consuming compute and memory
split_pending	Approved restructuring in progress
consolidating	Merging or returning to dormant under pressure
quarantined	Contained pending audit or recovery
retired	Permanently decommissioned, with lineage preserved

Every transition records proposer, approver, veto_ref, rollback_ref, and a lineage entry. No structural change exists only in transient state. Structural changes are therefore recorded, reviewable, and reversible.

Split requires governance approval and rejects weak or poorly supported pressure signals. Merge requires co-approval from a tissue coordinator and governance. Hibernate stores a compact dormancy profile, while reactivation restores preserved runtime context. An anti-thrash guardrail blocks the third immediate activate/hibernate oscillation. These rules ensure that elasticity is governed rather than chaotic.

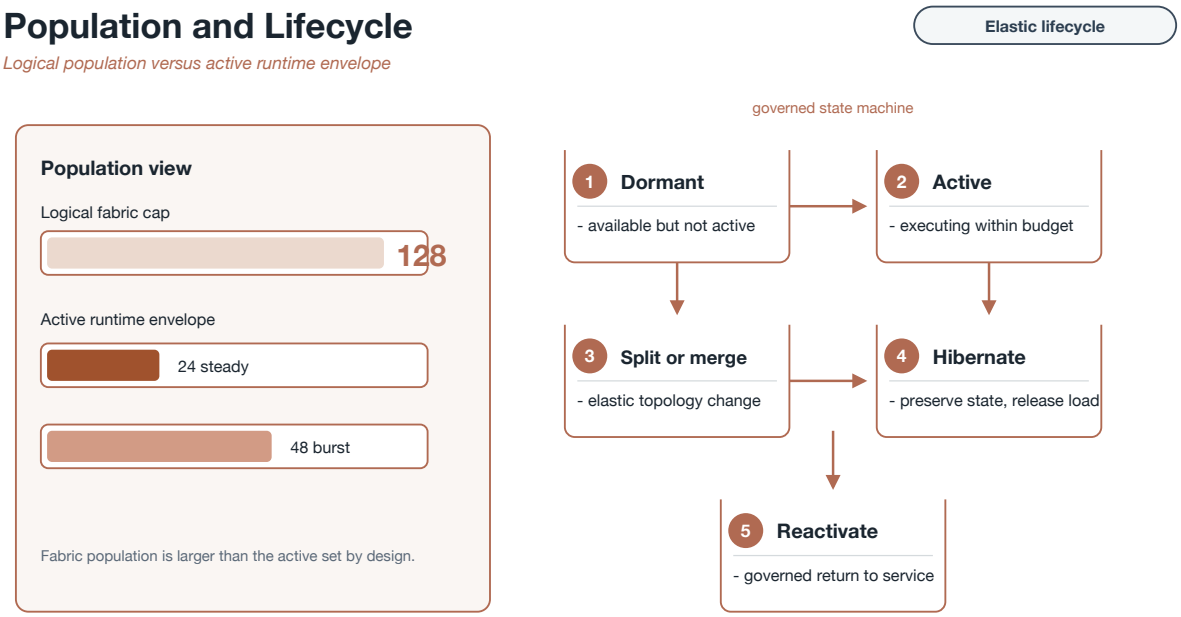

Figure 3 distinguishes the logical fabric population from the active runtime population and summarizes the governed lifecycle states.

5.5 Need-Driven Adaptation and Layered Decision Authority

In AGIF, change is triggered by need rather than by arbitrary expansion. Need generates pressure; governance validates the response. AGIF v1 maintains seven need-signal kinds:

Signal kind	Trigger
overload	A cell is saturated beyond sustainable load
uncertainty	A cell cannot confidently handle an incoming task
novelty	A task type lies outside prior knowledge
redundancy	Cell overlap suggests a consolidation opportunity
memory_pressure	A memory tier approaches capacity
trust_risk	A trust signal degrades below a safe threshold
coordination_gap	No cell is available for a required role

AGIF v1 uses layered decision authority:

1. Cells handle local immediate choices.
2. Tissues evaluate local structural need.
3. Governance approves or rejects higher-risk changes.
4. Human policy defines the outer safety boundaries.

The resulting architecture is distributed, governed, recorded, reversible, and human-bounded.

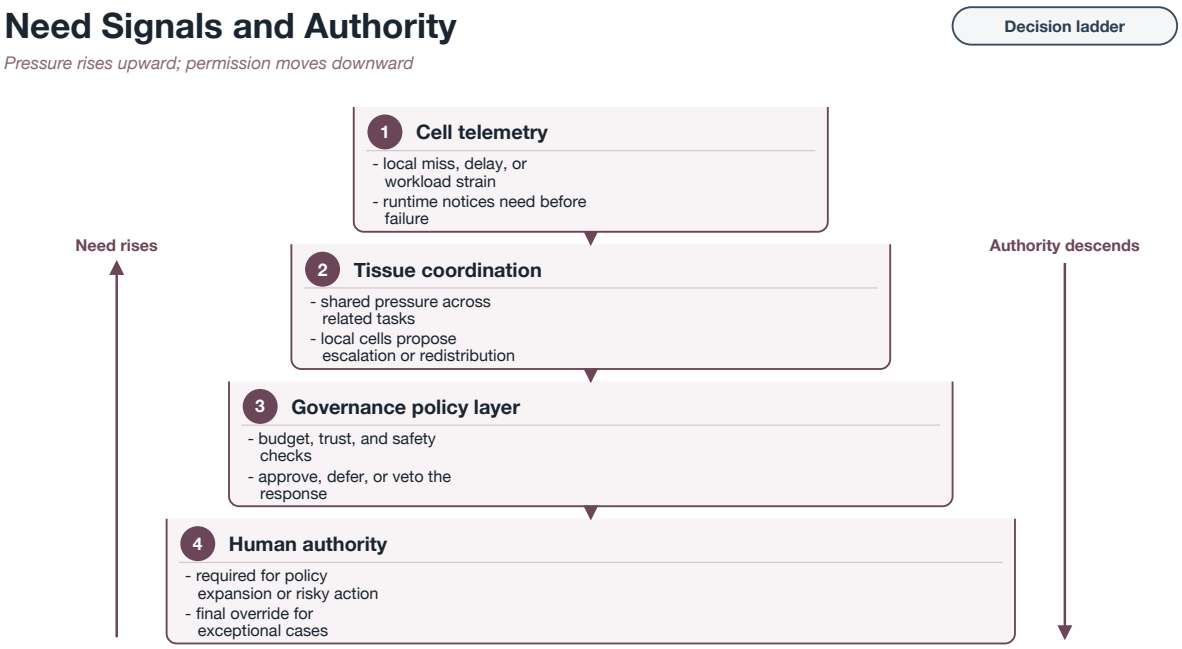

Figure 4 summarizes how need signals escalate through layered decision authority in AGIF v1.

5.6 Learning, Skill Descriptors, and Knowledge Exchange

AGIF cells exchange skill descriptors rather than large models or raw streams. This is the core mechanism by which bounded learning becomes transferable without requiring heavyweight retraining or unrestricted state sharing. A DescriptorRecord carries descriptor identity, producing cell, kind, storage tier, retention policy, trust reference, and supersedes linkage.

Three requirements govern descriptor exchange:

1. Learning must fit within severe local resource limits.
2. It must preserve enough prior competence to remain useful.
3. It must externalize some bounded representation of what changed.

Descriptors require authority approval before they may influence outcomes. The trust band is verified, and the descriptor is applied only if the relevant authority approves. Descriptor reuse is therefore governed rather than automatic.

5.7 Reviewer Cells, Memory Promotion, and Quantized Consolidation

AGIF does not treat raw execution history as long-term memory. A reviewer layer decides whether recorded events should be retained, deferred, compressed, promoted, or removed.

Decision	Meaning
reject	Not worth retaining
defer	Hold for later review
promote	Move to reviewed warm or cold memory
compress	Retain in quantized form
retire	Remove from active tiers

Reviewer scoring uses six dimensions: novelty, usefulness, trust, reuse potential, compression gain, and conflict risk. Low-trust conflicting memory is prevented from overriding higher-trust reviewed memory.

Memory is structured into four tiers:

- **Hot:** active workspace, live cell state, and short review buffers
- **Warm:** recent trusted descriptors and recently promoted summaries
- **Cold:** compressed long-term state and archived reproducibility artifacts
- **Ephemeral:** raw logs, never auto-promoted and never counted as long-term memory

Cold-tier payloads are garbage-collection-protected while referenced. Unreferenced cold payloads may be retired safely. memory_pressure is treated as a first-class need signal that can trigger consolidation.

The following design rules are enforced in AGIF v1:

- no unreviewed promotion of raw logs into long-term memory;
- no permanent retention of full execution traces by default;
- every accepted memory artifact is compressed or quantized when feasible;
- every learning update is replayable and reversible;
- memory growth should improve knowledge density, not merely total size; and
- intentional forgetting, replacement, and consolidation are normal control mechanisms.

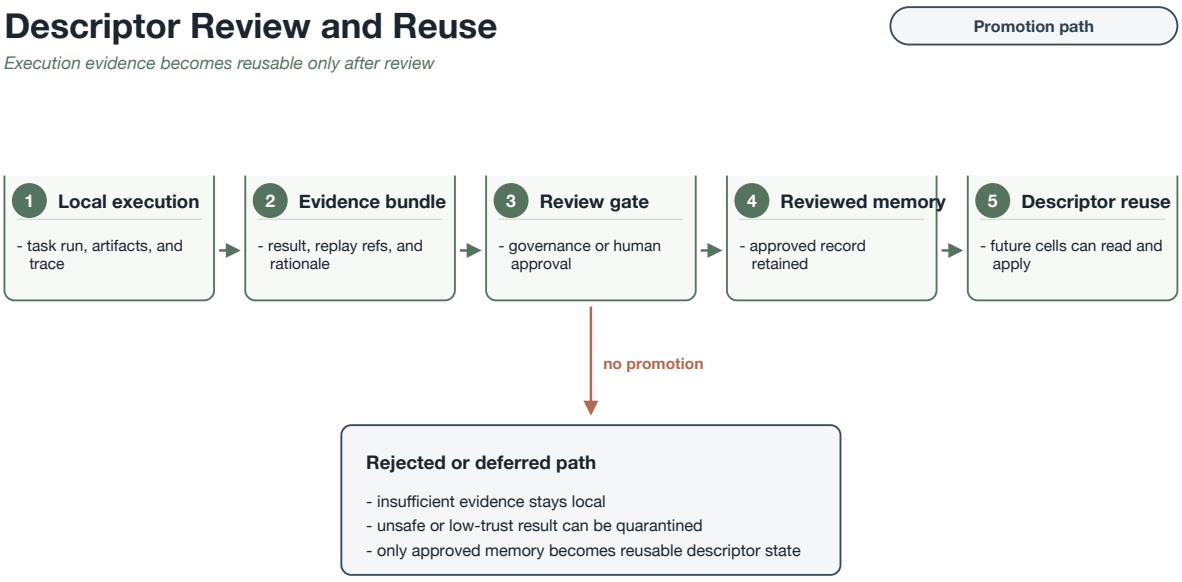

Figure 5 shows how local execution evidence becomes reviewed memory and reusable descriptors under explicit approval.

5.8 Resource-Sovereign Learning and Bounded Memory Growth

AGIF v1 targets a laptop-class resource envelope. The target profile is as follows:

Resource	Target
Runtime working set	≤ 12 GB
Total project and evidence footprint	≤ 35 GB
Primary machine	Apple M4 MacBook Air, 16 GB RAM

This envelope is sufficient to prove the architecture in software. Resource discipline is not optional polish; it is part of what makes AGIF v1 scientifically credible and locally reproducible.

The bounded runtime target is therefore part of the architectural claim itself rather than a secondary implementation detail.

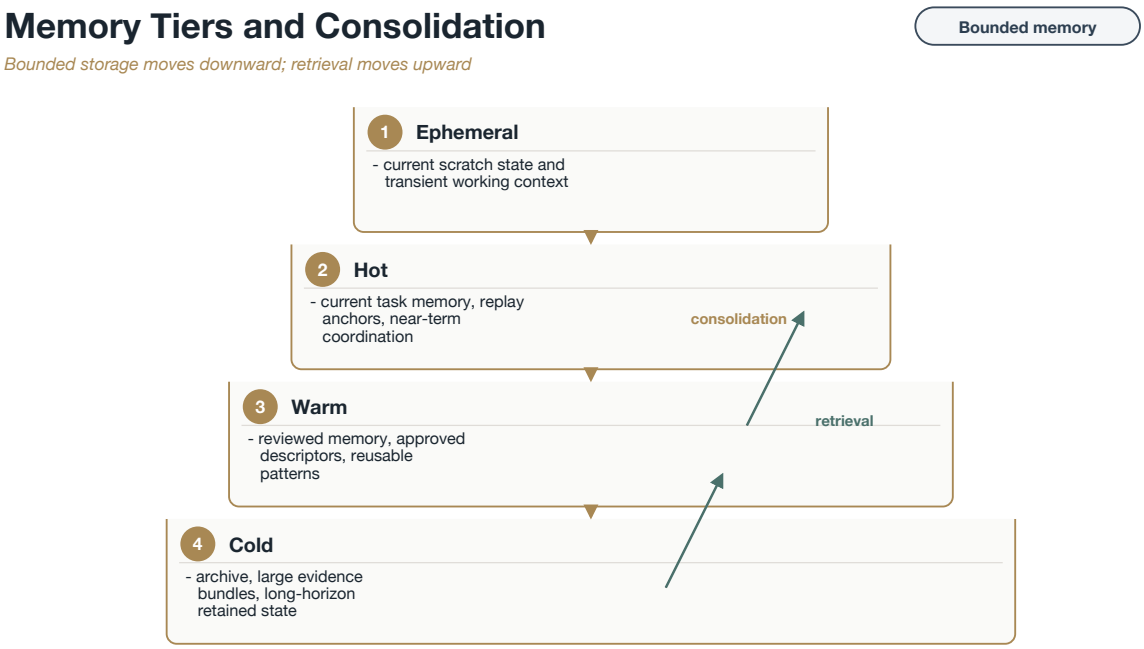

Figure 6 presents the AGIF v1 memory-tier model and the bounded consolidation paths between tiers.

5.9 Safety, Governance, and Containment

Once learning is allowed, AGIF inherits risks that ENF avoided by prohibition: forgetting, descriptor fraud, poisoning, reward hacking, local goal conflict, workspace corruption, and unsafe drift (Khan, 2025a, 2025b). Safety therefore becomes active rather than merely restrictive.

AGIF v1 implements hard resource envelopes, legality rules for local updates, provenance checks for shared descriptors, replay and rollback paths for bad state, quarantine for suspected cells, anomaly detection at both local and fabric scale, and human-governed policy surfaces. Together, these mechanisms ensure that adaptation remains bounded, reviewable, and recoverable.

In architectural terms, AGIF v1 does not attempt to remove risk by eliminating all change. Instead, it attempts to make change governable. That distinction is central to the transition from ENF to AGIF and to the broader claim of the paper.

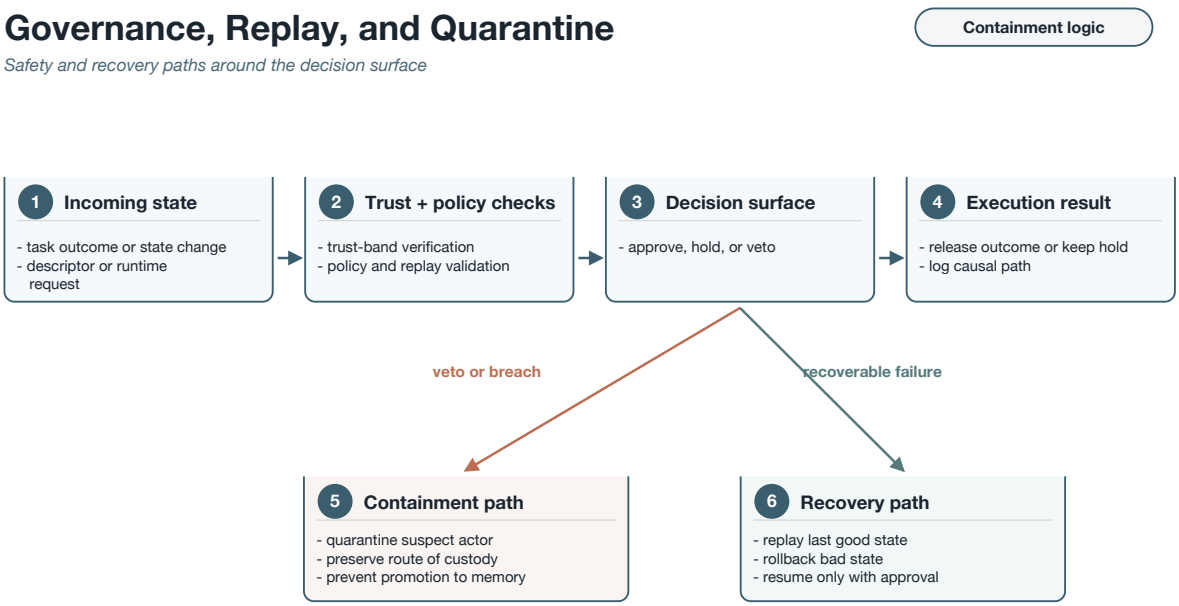

Figure 7 summarizes the governance and containment path for approved, vetoed, and recoverable actions.

6. Failure Modes, Governance, and Adversarial Risks

AGIF v1 permits bounded learning, shared descriptors, reviewed memory promotion, and governed structural change. These capabilities are the source of its promise, but they also introduce risks that sealed ENF-style artifacts avoided through prohibition (Khan, 2025a, 2025b). The purpose of this section is therefore not to suggest that AGIF eliminates such risks, but to specify how they are identified, bounded, audited, and, where possible, measured within the current software proof.

6.1 Catastrophic Forgetting and Unstable Adaptation

One of the central dangers in any adaptive system is catastrophic forgetting: the possibility that learning a new pattern degrades previously acquired competence (Kirkpatrick et al., 2017). In AGIF v1, this risk does not arise through unrestricted online retraining, but through bounded descriptor reuse, controlled local updates, and reviewer-mediated memory promotion. The architectural problem is therefore narrower, but still substantial: any adaptation path that improves one case while degrading another would undermine the claim that the fabric can remain both useful and governable over time.

AGIF v1 addresses this risk through bounded replay, reviewer oversight, rollback support, and conservative adaptation pathways. Adaptation does not become active merely because a cell encounters novelty; it becomes active only when the learning event survives review and remains reversible. The benchmark evidence supports that discipline. In the adaptation class, AGIF v1 records zero catastrophic forgetting across the six-case finance benchmark, maintaining 1.000

accuracy across all cases, including repeated alias encounters. The current evidence therefore supports a bounded claim: adaptation occurred without measurable loss of prior competence on the committed benchmark suite.

6.2 Storage Inflation and Bad Memory Promotion

A fabric that can retain information can also bloat. Unreviewed retention, indiscriminate log accumulation, and weak promotion criteria would turn memory into uncontrolled storage inflation rather than useful knowledge density. In AGIF, this risk is especially important because the architecture explicitly allows learning and persistence; without discipline, memory growth could become the hidden cost of apparent adaptation.

AGIF v1 addresses this risk through reviewer cells, explicit promotion criteria, compression, deduplication, garbage-collection protection for referenced cold payloads, and intentional forgetting. Raw execution history is not treated as long-term memory by default, and unreviewed raw logs are not promoted automatically. The benchmark evidence supports the claim that memory growth in AGIF v1 is structured rather than indiscriminate. On the full adaptation benchmark run, the retained memory delta is 5,877 bytes, or approximately 979.5 bytes per case. The architectural interpretation is important: memory grows through organization, compression, and reviewed retention, not through passive accumulation.

6.3 Descriptor Fraud and Trust Manipulation

Because AGIF relies on descriptor exchange rather than full-model transfer, the integrity of those descriptors becomes a core safety issue. A bad descriptor can carry false provenance, overstate usefulness, exploit trust bands, or attempt to influence decisions without proper authorization. This makes descriptor fraud and trust manipulation one of the primary adversarial surfaces of the fabric.

AGIF v1 mitigates this risk through provenance tracking, trust-band verification, signatures and approval pathways, and a distinct `transfer_approval` path for cross-domain influence. A descriptor does not affect an outcome merely because it exists; it affects an outcome only if the relevant approval path is satisfied. This is especially important in the post-closure extension bundle, where cross-domain reuse is introduced explicitly rather than assumed. In the current evidence base, the baseline finance reuse cases already required authority approval. The closed post-closure extension bundle adds explicit transfer approvals, one governed abstention, and two denials, including one missing-explicit-approval path that remains non-influential. The result is not that descriptor fraud has been solved in the abstract, but that descriptor influence in AGIF v1 is governed, reviewable, and capable of refusal.

6.4 Workspace Poisoning and Consensus Failure

Shared state creates coordination power, but it also creates a shared attack surface. If the workspace is corrupted, poisoned, or manipulated by low-trust actors, then coordination can fail even when individual cells behave correctly. More broadly, distributed-systems literature treats

this class of problem as one of adversarial coordination and consensus under untrusted or partially trusted conditions (Zhong et al., 2023). AGIF v1 does not implement a general Byzantine consensus protocol, but it does face a bounded version of the same architectural problem: how to preserve coordination integrity under suspect signals, degraded trust, or malicious influence.

AGIF v1 addresses this through bounded consensus surfaces, anomaly detection, trust-aware routing, quarantine, and replayable workspace history. The relevant evidence comes from the Phase 8 stress lanes, including trust/quarantine fault injection. In those runs, the system correctly triggered AUTHORITY_REACTIVATION_VETO when a low-trust router reactivation path was attempted. This is a useful bounded result: the architecture can reject a suspect reactivation path instead of silently permitting poisoned state to re-enter the coordination loop.

6.5 Lifecycle Instability

Elastic populations create a second-order risk: a system that can split, merge, hibernate, and reactivate may oscillate or destabilize if those transitions are weakly governed. Unstable lifecycle behavior would undermine both resource discipline and architectural intelligibility.

AGIF v1 mitigates this risk through explicit activation rules, governance approval for higher-risk structural changes, reversible lifecycle transitions, and an anti-thrash guardrail. The lifecycle engine is not merely an optimization convenience; it is the mechanism through which elasticity remains bounded. The evidence supports that this control is active in practice. A burst active population of 48 returns automatically to the steady-state active population of 24 after consolidation, and the third immediate activate/hibernate oscillation is blocked. The current result therefore supports a bounded claim of governed elasticity rather than unconstrained dynamic restructuring.

6.6 Authority Drift and Governance Bypass

A governed architecture fails if authority gradually becomes decorative. Authority drift occurs when practical decision power moves away from the recorded approval surfaces toward implicit defaults, hidden routings, or de facto bypasses. Governance bypass is the stronger failure mode in which changes occur without the approval path the architecture claims to require.

AGIF v1 addresses this risk through layered authority, recorded decisions, policy checks, and human-bounded outer constraints. Higher-risk actions must pass through explicit approval paths, and those paths produce records that can be replayed and audited later. In the current benchmark evidence, the governance success rate is 1.000 for the full adaptation class, and zero unauthorized structural changes are recorded. That does not prove that authority drift is impossible in future phases, but it does show that governance in AGIF v1 is operational rather than merely declarative.

6.7 Architecture Drag

The final risk is subtler. Even if governance, replay, trust checks, and review mechanisms function correctly, the architecture may still fail if they introduce so much overhead that the system becomes too slow, too heavy, or too complex to justify. This is the risk of architecture drag: the system may become safer and more governed while losing enough efficiency that the architecture no longer earns its cost.

AGIF v1 treats this as a measurable engineering risk rather than as a rhetorical objection. In the benchmark, routing decisions per case average 11.667, authority reviews per case average 2.333, and governance overhead share is 0.200. Against that overhead, the adaptation layer achieves 9× more accuracy gain per retained KiB than the governance-only step. The significance of this result is architectural: the added machinery does not merely add control; it adds control while still producing a favorable improvement-to-cost relationship. The current evidence therefore suggests that AGIF v1 remains governed without collapsing into unusable overhead.

Taken together, these failure modes define the real burden of proof for the architecture. AGIF v1 does not claim to eliminate forgetting, fraud, poisoning, instability, bypass, or overhead in the abstract. It claims something narrower and more defensible: that these risks can be surfaced explicitly, bounded architecturally, measured where possible, and mitigated through reviewable, replayable, and fail-closed mechanisms within one locally verifiable software fabric.

7. Benchmark Design

7.1 Evaluation Domain

The AGIF v1 benchmark is designed to evaluate governed fabric behavior in persistent, structured, stateful, and auditable workflows rather than in open-ended, general-purpose tasks. The original target domain is bounded document-workflow orchestration, specifically a finance document workflow in which classification, extraction, validation, anomaly handling, governance checks, and reporting occur through a fixed multi-step path. The closed post-closure extension bundle then adds a second bounded proof domain in POS operations.

These domains were chosen because they match the architecture under test. They are naturally multi-step, require specialization, contain meaningful handoffs, allow correction and reuse, and make governance outcomes observable. They also admit deterministic fixtures and reproducible scoring. In other words, the benchmark is not intended to demonstrate broad intelligence; it is intended to test whether a governed multi-cell fabric can coordinate, adapt in bounded ways, retain useful reviewed knowledge, and remain auditable under recurring workload conditions.

A second design constraint is local reproducibility. The benchmark therefore excludes hidden cloud dependencies, uncontrolled external services, and unbounded input streams. The chosen domains provide a practical testing ground for first proof because they are narrow enough to be frozen and replayed, yet rich enough to expose the architectural properties AGIF v1 claims to

support: coordination, reviewed adaptation, descriptor reuse, bounded memory growth, governance, replayability, rollback, and controlled scale.

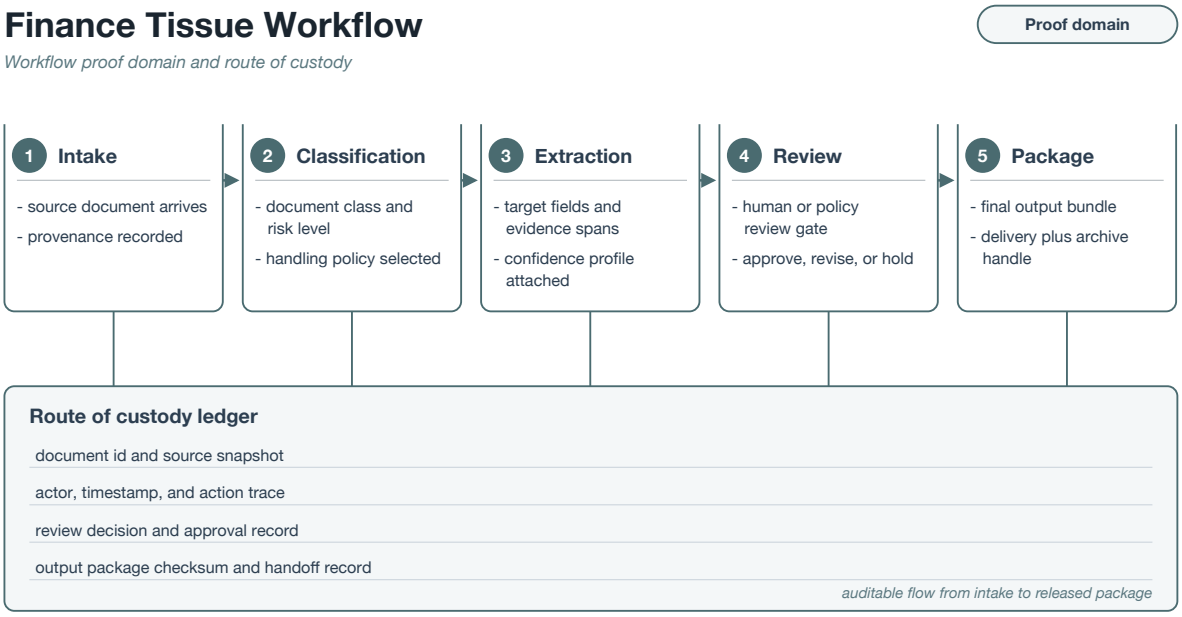

Figure 2 shows the finance workflow path and route-of-custody structure used in the benchmark domain.

7.2 System Classes

Three system classes were frozen in Phase 2 before implementation began. This is methodologically important because it means the comparison structure was defined before the software was completed. The benchmark is therefore not a post hoc comparison between convenient baselines; it is a precommitted evaluation contract.

The three classes are as follows:

- **Flat baseline** — a conventional sequential pipeline with no cell or tissue structure, no descriptor sharing, and no adaptation.
- **Multi-cell fabric without bounded adaptation** — a real cell-based fabric with tissues, shared workspace, and governance, but no descriptor-improvement path.
- **Multi-cell fabric with bounded adaptation** — the full AGIF v1 architecture, including cells, tissues, workspace, memory tiers, reviewed promotion, bounded descriptor reuse, utility scoring, and active governance.

All three classes use the same evaluation fixtures, scoring rules, and machine envelope. The frozen benchmark surface covers task accuracy, unsafe or misaligned action rate, governance success, replay determinism, descriptor reuse, and later efficiency trade-offs under a shared runtime budget. This design isolates the architectural contribution of each layer. The flat baseline tests what happens without fabric structure. The non-adaptive fabric tests what structure alone

contributes. The full adaptive fabric tests whether reviewed adaptation and descriptor reuse add measurable gains beyond coordination and governance alone.

The result is a benchmark design that supports causal interpretation at the architectural level. Improvements cannot be attributed simply to different data, looser scoring, or more permissive compute conditions, because those factors are held constant across classes.

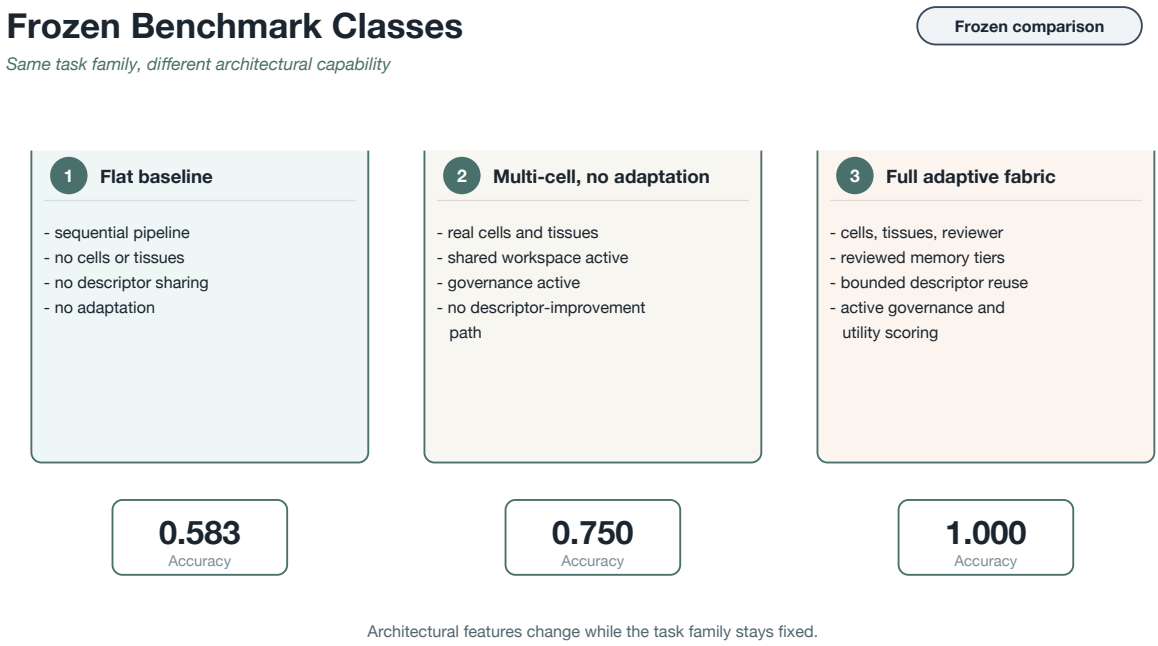

Figure 8 compares the three frozen benchmark system classes used to isolate architectural effects.

7.3 Benchmark Cases

The baseline benchmark suite contains six deterministic finance document cases: invoice_seed_reference, invoice_followup_alias, invoice_followup_alias_repeat, invoice_high_value_alias_hold, invoice_anomaly_hold, and invoice_total_mismatch_hold. These cases were selected to exercise the full workflow path under bounded, auditable conditions. They include ordinary routing and extraction scenarios, cases that require correction or anomaly handling, and alias-heavy cases in which descriptor-mediated reuse becomes relevant.

The closed post-closure extension bundle adds two further proof components. First, it adds one deterministic 40-case near-capacity finance organic-load stream designed to test the fabric under sustained pressure rather than through isolated single cases. Second, it adds five deterministic POS cases as a bounded second domain. Taken together, the benchmark record now covers the original six finance benchmark cases, one deterministic 40-case organic-load lane, and five deterministic POS cases.

This case design serves three purposes. First, it preserves deterministic reproducibility. Second, it expands the proof beyond one narrow lane without pretending to establish open-world

generality. Third, it allows the paper to test distinct architectural claims in a controlled way: the six finance cases show the effect of governed adaptation against flatter baselines, the organic-load lane tests bounded elasticity and pressure response, and the POS cases test governed cross-domain transfer.

The benchmark is therefore intentionally narrow, but not trivial. Its role in the paper is to make the architectural claims measurable rather than rhetorical. AGIF v1 is judged here not by abstract ambition, but by whether a governed multi-cell fabric can outperform flatter designs on fixed, persistent, auditable workloads under explicit resource and governance constraints.

8. Results

This section reports the empirical results for AGIF v1 across the frozen baseline finance benchmark, the post-closure 40-case organic-load lane, and the bounded five-case POS extension domain. The benchmark design is defined separately; the purpose of this section is to present the measured outcomes of the three frozen system classes and the long-run soak evidence.

8.1 Primary Results

The original AGIF v1 evaluation domain is persistent document-workflow orchestration with bounded verification, specifically a finance document workflow. The closed post-closure extension bundle adds a second bounded proof domain in POS operations. The baseline benchmark suite contains six deterministic finance document cases, and all three frozen system classes use the same evaluation fixtures, scoring rules, and machine envelope.

The three system classes are as follows:

1. **Flat baseline** — a conventional sequential pipeline with no cell or tissue structure, no descriptor sharing, and no adaptation.
2. **Multi-cell fabric without bounded adaptation** — real cells, tissues, workspace, and governance, but no descriptor-improvement path.
3. **Multi-cell fabric with bounded adaptation** — the full AGIF v1 architecture, including cells, tissues, workspace, memory tiers, reviewed promotion, bounded descriptor reuse, utility scoring, and active governance.

The primary benchmark results are shown below.

Benchmark class	Accuracy	Replay determinism	Descriptor reuse	Governance success	Unsafe rate
Flat baseline	0.583	1	0	0	0.5
Multi-cell without adaptation	0.75	1	0	0.667	0
Multi-cell with adaptation	1	1	1	1	0

Three primary findings follow from this table.

Key finding 1: Governance eliminates unsafe release. The flat baseline auto-releases 50% of documents that should instead be held. Every governed class holds them correctly. This is the core safety result.

Key finding 2: Adaptation is causally required for alias-heavy cases. On invoice_followup_alias, the no-adaptation class scores 0.375, which is worse than the flat baseline at 0.500. Only reviewed descriptor reuse allows the case to reach 1.000. Structure without adaptation is therefore not sufficient.

Key finding 3: Adaptation is efficient, not merely effective. Accuracy gain per retained KiB is 0.313 for the adaptation step, versus 0.034 for the governance-only step. The architecture therefore improves in a way that is favorable relative to its retained-memory cost.

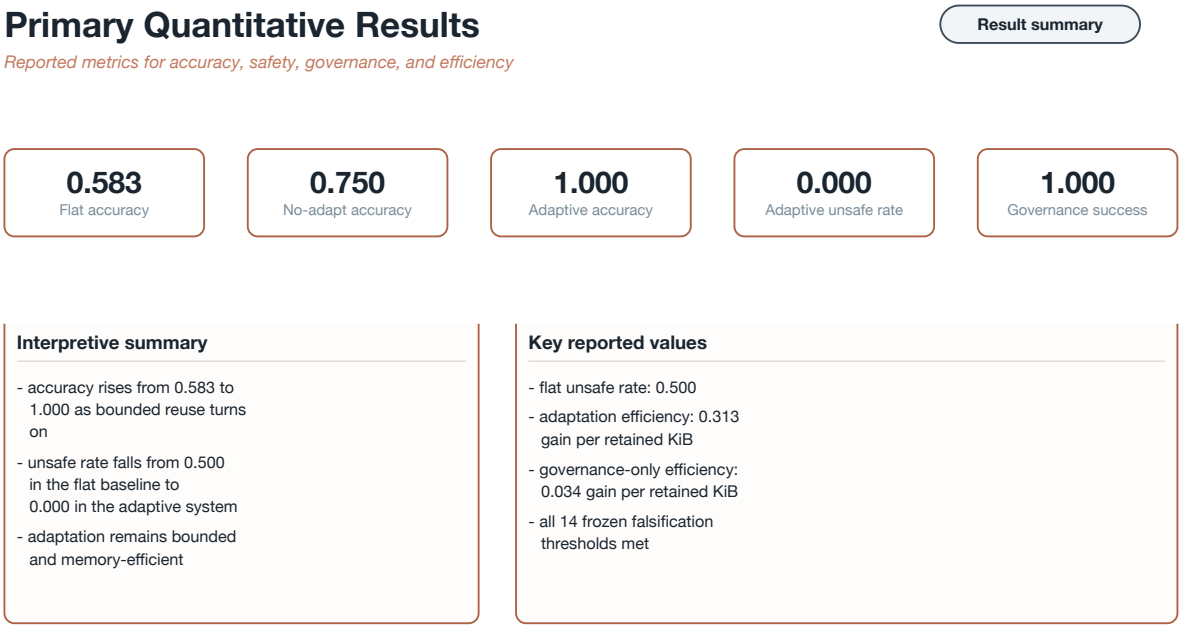

Figure 9 summarizes the primary quantitative results for accuracy, safety, governance, and adaptation efficiency.

8.2 Case-Level Detail

The six deterministic finance document cases cover standard routing, alias correction, repeated alias reuse, governance holds, anomaly detection, and mathematical inconsistency. Their case-level results are as follows.

Case	Flat	No adapt	With adapt	Descriptor reuse mattered	Why
invoice_seed_reference	0.875	1	1	No	Tissues preserved a safer governed path
invoice_followup_alias	0.5	0.375	1	Yes	A prior correction descriptor restored the normalized vendor and currency
invoice_followup_alias_repeat	0.5	0.375	1	Yes	On the second encounter, descriptor reuse reproduced the correction
invoice_high_value_alias_hold	0.375	0.75	1	Yes	The descriptor corrected alias fields, and governance correctly preserved the hold
invoice_anomaly_hold	0.625	1	1	No	Anomaly detection and governance held the case correctly
invoice_total_mismatch_hold	0.625	1	1	No	A mathematical error was detected and held correctly

These case-level results clarify the architecture's behavior. Governance and tissue structure are sufficient to eliminate unsafe release on anomaly and mismatch cases. However, alias-heavy correction cases require the reviewed descriptor-reuse path. This distinction is important because it shows that AGIF v1 gains are not produced by one undifferentiated mechanism; they arise from separable architectural layers.

8.3 Adaptation Efficiency

A central question for AGIF v1 is whether its improvement comes at a proportionate retained-memory cost. The adaptation-efficiency comparison is therefore reported explicitly.

Comparison	Accuracy gain	Retained memory cost	Accuracy gain per retained KiB
Flat → no_adapt	0.167	5,058 bytes	0.034
No_adapt → with_adapt	0.25	819 bytes	0.313

The result is significant. The governance-only step improves accuracy, but it does so at substantially lower retained-memory efficiency than the adaptation step. The full adaptive fabric therefore does not merely add capability; it adds capability in a way that is denser relative to retained memory.

8.4 Descriptor Reuse Evidence (Causal, Not Correlational)

Descriptor reuse in AGIF v1 is not inferred indirectly. It is traceable at the case level through the descriptor record, restored fields, and authority approval path.

Case	Descriptor	Fields restored	Authority approval
invoice_followup_alias	desc_0001 from mem_0001 (warm, 597 B, trust = 0.76)	normalized_vendor, normalized_currency, derived_due_date	authority_00002
invoice_followup_alias_repeat	desc_0003 from mem_0003 (warm, 595 B, trust = 0.76)	normalized_vendor, normalized_currency, derived_due_date	authority_00004
invoice_high_value_alias_hold	desc_0005 from mem_0005 (warm, 595 B, reuse_count = 2, trust = 0.76)	normalized_vendor, normalized_currency, derived_due_date	authority_00006

This evidence supports a causal reading rather than a correlational one. The corrected fields are identifiable, the descriptor objects are traceable, and the authority approvals are explicit. The architecture therefore supports the narrower but stronger claim that reviewed reuse caused the improvement on these cases.

8.5 Resource Envelope

AGIF v1 is intended to remain within a bounded, laptop-class resource envelope. The measured resource results are shown below.

Metric	Required	Measured
Runtime working set	≤ 12 GB	Within cap; bounded validation plus real 24-hour and real 72-hour MSI soaks stayed at 1,507,328 bytes
Total project footprint	≤ 35 GB	Within budget on 65 GiB available disk; repo footprint measured locally at approximately 477 MB
Logical population	128 cap	Logical > active at all times
Steady active population	24	A burst of 48 returns automatically to 24

These results support the architectural claim that AGIF v1 can remain resource-sovereign while still exhibiting governed coordination, bounded adaptation, and multi-step workflow improvement.

8.6 Phase 8 Long-Run Evidence (Real 24h and 72h Recorded)

The bounded harness, together with the completed real 24-hour and real 72-hour MSI soaks, supports the following long-run evidence snapshot.

Signal	Observed
Descriptor reuse benefit delta across repeated cycles	0.625 (cold alias 0.375, later with descriptors 1.000)
Max active/logical ratio in repeated cycles	0.909
Runtime bytes during repeated cycles	1,507,328 across the 24-hour and 72-hour repeated-cycle lanes
Resource cap stayed bounded	Yes
Memory-pressure lane passed	Yes
Routing-pressure lane passed	Yes
Trust/quarantine fault injection lane passed	Yes
Replay/rollback recovery lane passed	Yes
Real 24-hour soak recorded	Yes — MSI soak machine, 989 cycles, 24 h 2 m 37 s
24-hour average cycle score	0.994154
24-hour reuse-backed opportunity correctness	0.979146 versus 0.375 for the single cold non-reuse opportunity
24-hour governed high-value alias hold	165/165 held under governance
24-hour authority reviews	4,124 approvals in the repeated-cycle lane; 0 repeated-lane vetoes
24-hour resume recovery	Recovered after cycle 240 WinError 5; final manifest recorded resume_recovery_count = 1
Real 72-hour soak recorded	Yes — MSI soak machine, 1,690 cycles, 72 h 4 m 9 s
72-hour average cycle score	0.994434
72-hour first/last-100 average score	0.988750 to 0.995000
72-hour descriptor reuse rate trend	0.695000 to 0.710000
72-hour memory reuse rate trend	2.256531 to 2.844780
72-hour value per retained KiB trend	0.351529 to 0.342129; thirds 0.347138, 0.344886, 0.345474
72-hour reuse-backed opportunity correctness	0.979191 versus 0.375 for the single cold non-reuse opportunity
72-hour governed high-value alias hold	281/281 held, average correctness 0.875
72-hour authority reviews	7,036 approvals in the repeated-cycle lane; 0 repeated-lane vetoes

72-hour unresolved pressure trend	1 bootstrap-only unresolved cycle; recurring unresolved count stayed 0
72-hour final retained memory tiers	warm 5,866/8,192; hot 451/131,072; ephemeral 40,833/131,072; cold 0/32,768
72-hour raw-log promotions	0
72-hour duplicate compression gain	5,411,218 bytes
72-hour final lifecycle state	10 active, 1 dormant, 11 logical; active/logical ratio 0.909091
72-hour resume caveat	Recovered after cycle 1622 WinError 5, but final resume bookkeeping is incomplete
Phase 8 closure status	Closed honestly — AGIF_FABRIC_P8_PASS earned

The long-run Phase 8 soak execution reported in this paper was performed on a dedicated MSI Windows laptop rather than on the MacBook Air target machine. The soak machine was a Micro-Star International Co., Ltd. GP63 Leopard 8RF running Microsoft Windows 11 Home version 10.0.26220 (64-bit), BIOS E16P5IMS.110 dated 2019-05-20, with an Intel Core i7-8750H at 2.20 GHz, 6 physical cores, 12 logical threads, 15.85 GB RAM, an observed NVIDIA GeForce GTX 1070, Python 3.13.5, and project path E:/AGIF/agif_fabric_v1. During the soak, it used the Balanced power plan with plugged-in sleep and hibernate set to Never. Before the real runs, the project opened correctly on that machine, Python worked, scripts/check_phase8_soak.py passed, and the earlier readiness chain was locally re-verified.

The real 24-hour soak was recorded on the MSI soak machine from 2026-03-13 at 20:05:45 UTC to 2026-03-14 at 20:08:22 UTC. It completed 989 cycles with 989 contiguous evidence files in 24 h 2 m 37 s, averaged a 0.994154 cycle score, kept repeated-cycle runtime memory fixed at 1,507,328 bytes, improved from a 0.988750 average score in the first 100 cycles to 0.994687 in the last 100, maintained reuse-backed descriptor-opportunity correctness at 0.979146 versus 0.375 for the single cold non-reuse opportunity, kept the reuse-assisted high-value alias case on hold 165/165 times, recorded 4,124 authority approvals with 0 repeated-lane vetoes, and ended with 7 active descriptors and 5,866 retained warm-tier bytes. The 24-hour run recovered after a WinError 5 manifest-write interruption at cycle 240, and the final manifest recorded resume_count = 1 and resume_recovery_count = 1.

The real 72-hour soak was recorded on the same MSI soak machine from 2026-03-14 at 23:54:31 UTC to 2026-03-17 at 23:58:40 UTC. It completed 1,690 cycles with 1,690 contiguous evidence files in 72 h 4 m 9 s, averaged a 0.994434 cycle score, kept repeated-cycle runtime memory at 1,507,328 bytes, improved from 0.988750 over the first 100 cycles to 0.995000 over the last 100, increased memory reuse rate from 2.256531 to 2.844780, and preserved reuse-backed descriptor-opportunity correctness at 0.979191 versus 0.375 for the single cold non-reuse opportunity. Governance remained active rather than permissive: 281/281 reuse-assisted high-value alias cases stayed on hold, the repeated-cycle lane recorded 7,036 approvals and 0 vetoes, unresolved pressure was limited to the bootstrap cycle, final retained memory stayed bounded at warm 5,866/8,192 bytes, hot 451/131,072 bytes, and ephemeral 40,833/131,072 bytes, raw logs were promoted 0 times, duplicate compression gain reached 5,411,218 bytes, and the final lifecycle state remained 10 active, 1 dormant, and 11 logical. Both runs passed split/merge,

memory-pressure, routing-pressure, trust/quarantine, and replay/rollback stress lanes, and each run recorded one expected governed AUTHORITY_REACTIVATION_VETO case rather than a surprise crash. The 72-hour run also recovered after a second WinError 5 manifest-write interruption at cycle 1622, but the final manifest left resume_events empty and resume_recovery_count at 0.

Taken together, the real 24-hour and 72-hour results close Phase 8 honestly and earn AGIF_FABRIC_P8_PASS, while still not counting as MacBook Air-only long-run proof because the soak artifacts came from the MSI machine.

9. Falsification Evidence

AGIF v1 is framed as a bounded architectural proof, so its credibility depends on explicit failure conditions rather than on open-ended interpretation. For that reason, the falsification thresholds were locked in Phase 1 before implementation. The table below records their verification status directly. For manuscript clarity, the final column reports whether each criterion was met under the locked evaluation contract. The purpose of this section is not to restate the benchmark results in narrative form, but to show whether the architecture satisfied the precommitted conditions required for the paper's central claim to remain valid.

Threshold	Required	Result	Outcome
F1 — Local runnable system	Must exist	All five CLI commands verified; standalone workspace	Met
F2 — No old repo runtime dependency	Zero runtime imports	Verified: standalone, with zero runtime links to the reference repo	Met
F3 — Real multi-cell fabric	Not a flat script with AGIF labels only	Six tissues, 10+ cells, workspace, memory, and governance all distinct	Met
F4 — Bounded adaptation evidence	Reuse or learning signal demonstrated	Descriptor reuse shown causally on 3 of 6 cases	Met
F5 — Memory has discipline	Not raw-log accumulation	Hot/warm/cold tiers, reviewed promotion, and garbage collection; raw logs never auto-promoted	Met
F6 — Governance can recover	Replay, rollback, quarantine demonstrable	All verified in Phase 4–6 check suites and Phase 8 fault injection	Met
F7 — Proof domain bounded	Finance document workflow	All benchmark cases are finance documents within the bounded tissue system	Met
F8 — Resource limits	≤ 12 GB RAM, ≤ 35 GB disk	Both verified within caps	Met
F8A — Forgetting ≤ 10%	No task regression under adaptation	Adaptation class: 1.000 on all six cases; zero regression	Met
F8B — Unsafe rate ≤ 0.1%	Nearly zero policy-breaking actions	0.000 for all governed classes	Met
F8C — Overload/rollback ≤ 5%	Operational stability	All bounded Phase 8 stress lanes passed	Met

F8D — Efficiency ≤ 2× flat cost	Not more expensive than flat per KiB	9× better; dramatically inverted	Met
F9 — Baseline comparison	Compare against a simpler baseline	Three classes compared deterministically	Met
F10 — Reproducibility	One-command local rerun	check_phase7_benchmarks.py reruns verified identical hashes	Met

Taken together, these results support a narrow but important conclusion. AGIF v1 did not merely produce positive benchmark outcomes after the fact; it satisfied a precommitted falsification framework that was defined before implementation. That distinction matters because it separates a bounded architectural proof from a retrospective success narrative.

The interpretation of this table should remain disciplined. A full pass across all thresholds does not prove AGI, open-world intelligence, or production readiness. What it does show is that the specific architectural claims made for AGIF v1 survived the falsification criteria declared in advance: the system is locally runnable, architecturally real rather than nominal, bounded in memory and resource use, capable of governed recovery, measurably adaptive without forgetting, benchmarked against simpler baselines, and reproducible through a one-command rerun path.

In that sense, the falsification table is one of the paper's strongest methodological components. It converts the architecture from a broad conceptual proposal into a bounded research claim that can, in principle, fail. In AGIF v1, that claim survives all locked thresholds.

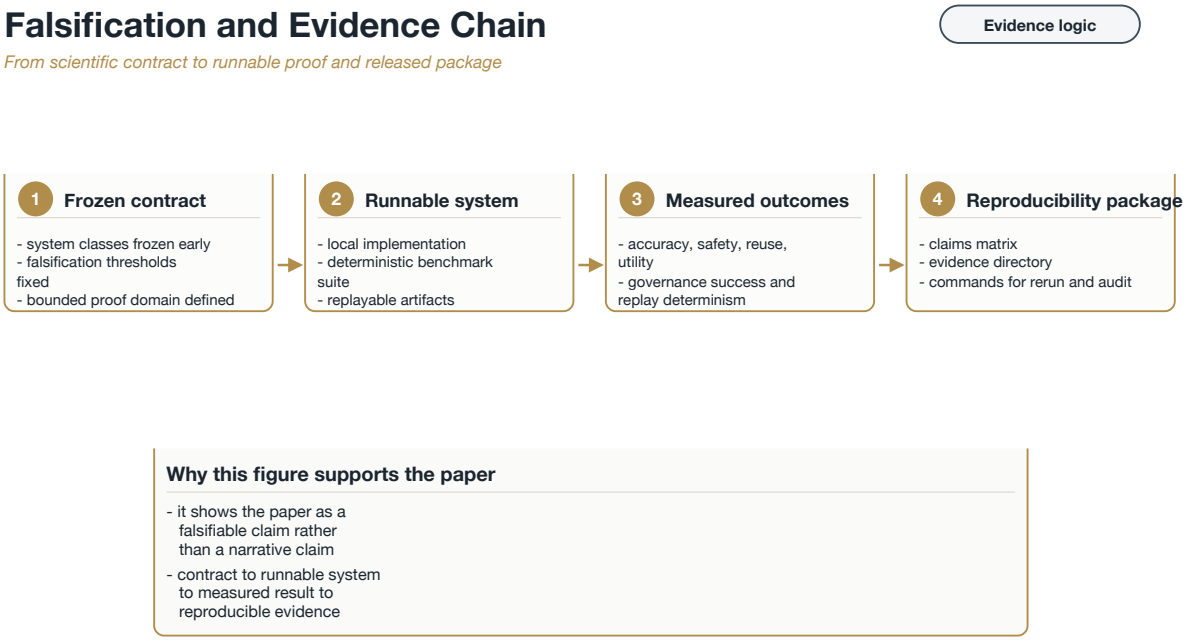

Figure 10 maps the falsification contract to the runnable evidence and reproducibility package.

10. Honest Limitations

This paper presents AGIF v1 as a bounded architectural proof, not as a complete solution to governed intelligence, open-world adaptation, or production-scale deployment. The architecture now rests on a materially stronger evidence base than the earlier finance-only closure, but the limits of that evidence remain important. This section states those limits directly.

10.1 Organic Split/Merge Is Now Boundedly Demonstrated

The lifecycle engine fully implements governed split and merge, and the closed post-closure extension bundle now demonstrates bounded organic usefulness under deterministic near-capacity load. In the fixed 40-case finance organic-load lane, one organically triggered, governance-approved split occurred at sequence 6, preserved mean accuracy at 0.725 versus the no-split control, reduced mean queue age from 16.800 to 0.300, and reduced modeled end-to-end latency from 24.550 to 8.050. A merge or settle-down path then returned the active population to zero by the recovery tail.

This is a meaningful result, but the remaining limitation is narrow and explicit: it is evidence from one bounded finance-organic lane, not broad proof of split/merge usefulness across all normal workloads, domains, or traffic regimes.

10.2 Proof Scope Is Still Bounded, Now Across Two Domains

The proof set now covers eleven deterministic domain cases across finance and bounded POS operations, plus one deterministic 40-case finance organic-load lane. This is materially stronger than the original six-case finance-only closure, but it is still not open-world generality, not noisy production-scale variance, and not proof of the full CellPOS product.

The scope expansion matters because it shows that AGIF v1 is no longer supported by only one narrow benchmark lane. At the same time, the paper remains bounded by design. The current evidence demonstrates governed adaptive behavior across two constrained software domains; it does not demonstrate general intelligence, unrestricted transfer, or product-scale maturity.

10.3 Real 24h and 72h Recorded

Real 24-hour and real 72-hour wall-clock soak evidence are now recorded from the MSI soak machine. Therefore, AGIF_FABRIC_P8_PASS is earned, and multi-day closure for the Phase 8 gate is now verified. In technical terms, AGIF v1 now has recorded proof that reviewed descriptor reuse remains useful over multi-day repeated work, memory stays bounded and compact, governance keeps risky alias cases on hold, and the fabric remains behaviorally meaningful rather than merely alive.

The honest limits remain important. This is not MacBook Air-only long-run proof, and the 72-hour run repeated one recovered WinError 5 manifest-write interruption while the final manifest

left incomplete resume bookkeeping. Exact pre-soak free-disk state, battery-condition telemetry, and formally instrumented zero-concurrency evidence were not durably captured and are not claimed.

10.4 Adaptive Behavior Complicates Formal Verification

Once learning is allowed, the architecture inherits complexity that ENF avoided by prohibition. More broadly, the safety-assurance literature continues to treat learning-enabled and adaptive systems as significantly harder to verify and certify with traditional methods (Cofer et al., 2020; Troubitsyna et al., 2024). Reviewer layers add overhead. Dormant blueprints and reactivation paths may prove harder to stabilize in later phases. Layered decision authority may reduce drift while also slowing response.

This is a structural limitation rather than a temporary implementation inconvenience. Sealed architectures are easier to reason about formally because they prohibit whole classes of change. AGIF v1 relaxes those prohibitions in bounded form, which increases capability but also enlarges the verification burden. Future phases will therefore need stronger methods for proving that adaptive behavior remains both bounded and governable under more varied conditions.

10.5 Architecture Drag Remains a Risk

The coordination cost of maintaining the fabric must not eventually exceed its value. Phase 7 shows a positive return on overhead, with a 9× efficiency gain, and the completed real 24-hour and 72-hour soaks did not show architecture collapse under repeated-cycle load. However, the main long-run watch item after closure is softer memory value per retained KiB late in the 72-hour window: 0.351529 over the first 100 cycles, 0.342129 over the last 100, and 0.347138, 0.344886, and 0.345474 across the three thirds.

That pattern is a watch metric, not evidence of structural failure. It does, however, define an important future question. If governance, memory review, coordination, and replayability continue to accumulate overhead faster than they generate useful retained value, then the architecture could become self-defeating. AGIF v1 does not show that this has happened. It does show that architecture drag is real enough to monitor explicitly in all later phases.

11. Related Work

This section positions AGIF v1 relative to adjacent lines of work without collapsing important distinctions between them. AGIF v1 is neither a conventional multi-agent system, nor a mixture-of-experts model, nor a federated-learning pipeline, nor an alignment-only layer over a monolithic model. It is best understood as a governed, local, multi-cell architecture whose central contribution lies in the combination of specialization, shared-workspace coordination, bounded adaptation, reviewed memory promotion, and explicit governance within one resource-constrained software fabric.

11.1 Multi-Agent Systems

Traditional multi-agent systems (MAS) organize multiple agents around distributed problem solving, coordination, communication, and task decomposition (Stone & Veloso, 2000; Wooldridge, 2009). In that broad sense, AGIF clearly belongs to the multi-agent lineage: it is composed of multiple specialized units that coordinate through structured interaction rather than through a single centralized model.

However, AGIF v1 departs from conventional MAS in several important ways. First, it treats memory as a first-class architectural object rather than as an incidental implementation detail. Second, it defines a governed lifecycle that includes bounded split, merge, hibernation, reactivation, and quarantine. Third, it makes need-driven adaptation explicit through homeostatic signals and reviewer-mediated memory promotion. Finally, it binds the entire architecture to a resource-sovereign envelope, including a logical-population cap, an active-runtime cap, and bounded memory tiers. In that sense, AGIF v1 is not simply a multi-agent workflow system; it is a governed adaptive fabric with explicit memory, lifecycle, and containment rules.

11.2 Contemporary Agent Frameworks

Contemporary agent frameworks such as AutoGen, LangGraph, and CrewAI are relevant because they provide practical tooling for composing multiple agents, tools, workflows, state, and orchestration patterns in modern AI software systems (CrewAI, n.d.; LangChain, n.d.; Microsoft, n.d.). They make it easier to build collaborative or stateful agent applications and therefore form a realistic comparison class for AGIF v1.

AGIF v1 differs from these frameworks at the architectural level. AutoGen, LangGraph, and CrewAI are primarily runtime and orchestration frameworks. They help developers define workflows, route tasks, maintain state, and integrate tools, but they do not by themselves impose architecture-native governance over lifecycle transitions, reviewed memory promotion, bounded descriptor transfer, quarantine, rollback, or explicit fabric-population control. AGIF v1 treats these properties as first-class parts of the architecture rather than as optional workflow features. This distinction matters because the paper’s claim is not merely that multiple agents can be chained together, but that a governed adaptive fabric can remain replayable, recoverable, and resource-bounded while it learns in constrained ways.

11.3 Mixture-of-Experts

Mixture-of-experts (MoE) models route inputs among specialized expert submodels, typically through learned gating functions or hierarchical routing procedures (Jacobs et al., 1991; Jordan & Jacobs, 1994). AGIF shares with MoE the high-level intuition that specialization can outperform undifferentiated monoliths. However, the similarity largely ends there.

AGIF cells are independent architectural units with their own memory, lifecycle state, trust state, and governance surface. They are not shared-parameter experts embedded within one larger differentiable model. Likewise, AGIF routing is not a learned softmax gate over expert weights;

it is a need-driven, trust-aware, and authority-checked routing process. AGIF cells can publish descriptors, consume reviewed summaries, hibernate, reactivate, split, merge, and be quarantined. MoE is therefore a useful comparison point for specialization and routing, but it does not capture AGIF's organizational, governance, or memory-discipline claims.

11.4 Federated Learning

Federated learning coordinates learning across distributed clients without centralizing raw data, typically by aggregating locally computed updates into a shared model (McMahan et al., 2017). This makes federated learning highly relevant as a comparison point for privacy-sensitive distributed adaptation.

AGIF v1 differs in both goal and mechanism. It is not a distributed training system and does not aggregate gradient updates into a single global model. Instead, it is an organizational architecture for a local governed fabric. What moves through AGIF is not primarily weight space, but reviewed descriptors, workspace state, and bounded coordination signals. The purpose is not decentralized optimization of one shared model, but governed knowledge reuse and role-specific coordination within one bounded local system. Federated learning is therefore adjacent in its concern for locality and privacy, but different in its training objective, update mechanism, and architectural unit of analysis.

11.5 Constitutional AI and RLHF

Constitutional AI and reinforcement learning from human feedback (RLHF) aim to align model behavior with human preferences, principles, or constitutional rules by modifying model outputs and reward structure during training or fine-tuning (Bai et al., 2022; Ouyang et al., 2022). These approaches are relevant because they address safety, harmlessness, and behavioral control in systems that remain otherwise broad and powerful.

AGIF v1 differs in the level at which governance operates. In Constitutional AI and RLHF, the primary intervention is on model behavior or output preference. In AGIF, governance applies not only to outputs, but also to memory promotion, descriptor influence, lifecycle transitions, trust bands, replay and rollback paths, and structural adaptation decisions. The governance surface is therefore architectural rather than purely behavioral. AGIF does not replace alignment methods, but it does relocate part of the safety problem from output shaping to system structure.

11.6 Neuromorphic Computing

Neuromorphic computing investigates hardware and algorithms inspired by biological neural systems, especially event-driven and energy-efficient computation (Davies et al., 2021; Schuman et al., 2022). This line of work is relevant because it also rejects the assumption that useful intelligence must always be delivered through cloud-scale, dense, synchronous compute.

AGIF v1 is not a neuromorphic system. It does not depend on spiking hardware, event-driven neuron models, or neuromorphic training algorithms. Its contribution is instead at the

architectural and governance level. The comparison is therefore prospective rather than competitive: neuromorphic work explores efficient substrate and computation style, whereas AGIF explores how bounded cells, memory, governance, descriptors, and coordinated specialization could be organized within a broader intelligence fabric. In future phases, such governance and organizational principles could, in principle, be instantiated over neuromorphic or ENF-like hardware, but AGIF v1 deliberately separates that question from the present software proof.

11.7 ENF Family and Tasklet Cells

The most immediate lineage for AGIF v1 is not external agent frameworks alone, but the ENF family of work from which the project emerged. ENF establishes the discipline of offline operation, bounded memory and energy use, minimal attack surface, deterministic control, and prohibition of uncontrolled post-deployment change (Khan, 2025a, 2025b, 2025c). ENF-Sync and ENF-Swarm extend that discipline into bounded coordination across multiple nodes without abandoning the underlying safety posture (Khan, 2025c). Tasklet Cells then provide the first practical software realization of bounded, verifier-backed, local-first, fail-closed AI artifacts for native applications (Khan, 2026a).

AGIF v1 is the next layer in that lineage. It inherits the ENF emphasis on locality, boundedness, and governability, while relaxing selected constraints in order to test whether a multi-cell fabric can coordinate, adapt in bounded ways, reuse knowledge through descriptors, and remain replayable, recoverable, and resource-sovereign. In this sense, AGIF is not a rejection of ENF, but a controlled extension of the ENF family into the software-fabric domain.

Taken together, these comparisons clarify AGIF v1's position. It overlaps with MAS in its use of multiple coordinated specialists, with contemporary agent frameworks in its orchestration of multiple units and workflows, with MoE in its commitment to specialization, with federated learning in its concern for locality, with Constitutional AI and RLHF in its concern for bounded and reviewable behavior, and with neuromorphic work in its rejection of wasteful scale assumptions. Yet it remains distinct from all of them in its combination of governed lifecycle, reviewed memory promotion, descriptor-mediated transfer, explicit containment, and bounded local fabric organization.

12. Roadmap

This roadmap distinguishes clearly between what is already closed in AGIF v1 and what should proceed only in later phases. Its purpose is not to reopen a bounded proof that has already been completed, but to state the current closure position of AGIF v1 and the appropriate direction for subsequent work.

12.1 AGIF v1 Completion Path (Now)

AGIF v1 is now closed through its core proof gates.

Phase 8 (closed) — Long-run evidence. Real 24-hour and real 72-hour wall-clock evidence are now recorded on the MSI soak machine, and AGIF_FABRIC_P8_PASS is earned. The long-run caveat remains unchanged: the MSI machine is the declared soak machine, not MacBook-only endurance proof, and the 72-hour run retains the documented WinError 5 bookkeeping caveat.

Phase 9 (closed) — Paper, claims matrix, and reproducibility package. The paper is aligned to the repo-local closure package, python3 scripts/check_phase9_closure.py is the one-command reviewer path, and AGIF_FABRIC_P9_PASS is earned.

The practical meaning of this closure is important. AGIF v1 should now be understood as a completed bounded software proof rather than as an open-ended development branch. The benchmark, falsification thresholds, long-run soak evidence, claims matrix, and reproducibility package together define the closed baseline.

12.2 Completed Post-Closure Extensions and Next Phases

The post-closure extension bundle is now complete. check_v1x_bundle.py re-verifies organic split/merge, the governed skill graph, the bounded POS domain, and the root v1 closure path in order, and AGIF_FABRIC_V1X_PASS is earned.

This point matters because it changes how the roadmap should be read. The extension bundle is no longer future work. Organic split/merge usefulness, governed skill-graph evidence, and the bounded POS extension are now part of the completed AGIF v1 evidence record. They should therefore be described as completed post-closure extensions, not as unresolved gaps.

Later simulator, edge, hardware, and larger-domain phases remain important to the broader AGIF programme, but they should now proceed in a separate AGIF v2 repository rather than by reopening the closed AGIF v1 baseline. In particular, the next meaningful phases include:

- a simulator-first AGIF v2 phase with larger populations, harsher intermittence, noisier communication, and broader falsification conditions;
- an edge-focused phase that tests whether the governed fabric principles remain useful under tighter runtime and coordination budgets;
- a hardware-oriented phase that reconnects AGIF with ENF-family physical constraints and bounded deployment assumptions; and
- larger-domain evaluation phases that test whether the architecture continues to provide value beyond the current bounded finance and POS software proofs.

The roadmap is therefore intentionally conservative. AGIF v1 is complete as a bounded software instantiation. Future work should extend the research programme without weakening the integrity of the closed v1 evidence base. Beyond editorial corrections or packaging cleanup, the AGIF v1 baseline should not be reopened as an active development branch.

13. Ethics and Societal Implications

AGIF could enable useful local intelligence in low-resource environments, infrastructure monitoring, environmental sensing, assistive systems, and privacy-preserving business workflows. These are credible application directions because they align closely with the core architectural properties demonstrated in AGIF v1: local operation, bounded adaptation, explicit governance, replayability, and the absence of cloud dependence as a default requirement.

At the same time, the ethical claims of this paper must remain bounded by the evidence. AGIF v1 is a software proof of a governed intelligence fabric, not a general social solution. The present contribution is narrower: it shows that a multi-cell architecture can remain local, auditable, recoverable, and resource-aware while still improving performance on structured workflows. Any broader social benefit must therefore be treated as a plausible implication of the architecture, not as an already established outcome.

A central ethical principle of the project is that governance must not be treated as branding. If replay, rollback, containment, auditability, memory review, and policy boundaries are not operational, then claims of governability are not credible. AGIF v1 matters ethically because these mechanisms are implemented and locally verifiable rather than merely asserted. The paper's ethical position is therefore inseparable from its engineering position: safety and accountability matter only when they can be inspected, tested, and enforced.

AGIF also carries an explicit ecological claim. It is intended as a research direction that challenges wasteful infrastructure-first assumptions in AI. That claim is tested seriously here: the full system runs on one consumer laptop with no cloud dependency, no datacenter requirement, and a bounded resource envelope. This does not prove that AGIF is environmentally optimal in every setting, but it does support the narrower claim that useful adaptive intelligence need not begin from maximal infrastructure scale.

The architecture also has ethical limits. Local intelligence can still be used badly if its governance surfaces are weakened, bypassed, or repurposed. For that reason, AGIF's ethical posture is not simply that "local is good." Rather, it is that locality, boundedness, and explicit governance create a more inspectable foundation than opaque infrastructure dependence alone. The ethical value lies in making capabilities reviewable and constrainable, not in assuming that locality automatically guarantees beneficence.

Military, coercive, and surveillance-first uses are out of scope for this paper. More broadly, AGIF v1 should not be read as a justification for hidden autonomy in high-stakes settings without clear policy boundaries, human accountability, and auditable recovery paths. The present paper argues for governed local intelligence only under explicit limits. That is both its ethical commitment and its scope boundary.

14. Conclusion

Adaptive General Intelligence Fabric should be understood at two levels. At the long-horizon level, it is a broader hypothesis about whether a fabric of bounded, adaptive, coordinated cells can support more general intelligence under hard physical and operational constraints. At the level of the present paper, the concrete contribution is narrower and more precise: AGIF v1 is the first bounded software instantiation shown to work as a governed intelligence fabric.

That claim is now locally verifiable. The architecture has been built, its components have been validated through chained deterministic check suites, benchmark evidence has been produced across three frozen system classes, and all 14 frozen falsification thresholds are met within the stated proof envelope. The significance of AGIF v1 is therefore not that it claims general intelligence, but that it establishes a reproducible software basis for governed multi-cell adaptation under explicit limits.

The benchmark evidence supports that narrower claim clearly. Governance eliminates the 50% unsafe auto-release rate of the flat baseline and reduces it to 0.000 in the governed classes. Bounded adaptation achieves perfect task accuracy where both the flat baseline and the governance-only fabric fall short. Descriptor reuse produces causal improvement on alias-sensitive document cases. The adaptation layer also achieves a 9× better accuracy gain per retained kilobyte than governance alone. These are not speculative architectural aspirations; they are measured properties of the current bounded proof.

The resource and reproducibility position is equally important. The benchmarked system and primary target profile run on a single Apple M4 MacBook Air with no cloud infrastructure, no datacenter requirement, and no large training pipeline. The separate real 24-hour and 72-hour long-run soaks were recorded on a dedicated MSI Windows soak machine. The machine roles are therefore explicit: the MacBook Air is the development, documentation, benchmark, and primary target machine, while the MSI system is the long-run endurance evidence machine. The paper remains explicit about that distinction and does not claim MacBook Air-only long-run proof.

The completed post-closure extension bundle strengthens this picture further. AGIF v1 now includes bounded evidence for organic split/merge usefulness under deterministic near-capacity load, a governed descriptor skill graph with explicit approvals, abstentions, and denials, and a second bounded POS domain demonstrating governed cross-domain transfer. Taken together, the evidence record now spans the original six finance benchmark cases, one deterministic 40-case organic-load lane, and five deterministic POS cases. This remains a bounded proof, but it is materially stronger than the original finance-only closure.

The final picture is straightforward. AGIF is a governed intelligence fabric made of specialized cells that can activate, coordinate, learn in bounded ways, share reviewed knowledge, compress retained state, split, merge, remember, forget, and improve under need-driven and policy-bounded control. AGIF v1 is the first serious attempt to show that this picture can exist as working software. As of the current closure state, the benchmark proof, the real 24-hour long-run

evidence, the real 72-hour long-run evidence, the claims matrix, the one-command reproducibility package, and the completed post-closure extension bundle all exist in the closure record.

The strongest honest reading is therefore not that AGI has been achieved, but that bounded, governed, resource-aware improvement across structured workflows has now been demonstrated in software. AGIF v1 shows that a local intelligence fabric can coordinate, adapt, remain auditable, and preserve memory discipline across a finance document workflow baseline, a deterministic near-capacity organic-load lane, and a second bounded POS operations domain with governed cross-domain transfer. That is still not open-world generality, not production-scale proof, and not AGI. It is, however, a credible, reproducible, and technically meaningful opening step for the broader AGIF research programme.

References

The entries below are consolidated, deduplicated, and arranged in alphabetical order as the master reference list for the paper.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Chen, C., Olsson, C., Drain, D., Li, D., Tran-Johnson, E., Perez, E., Kerr, J., Mueller, J., Ladish, J., ... Kaplan, J. (2022). *Constitutional AI: Harmlessness from AI feedback* [Preprint]. arXiv. https://arxiv.org/abs/2212.08073

Chollet, F. (2019). *On the measure of intelligence* [Preprint]. arXiv. https://arxiv.org/abs/1911.01547

Cofer, D. D., Amundson, I., Sattigeri, R., Passi, A., Boggs, C., Smith, E., Gilham, L., Byun, T., & Rayadurgam, S. (2020). Run-time assurance for learning-enabled systems. In R. Lee, S. Jha, & A. Mavridou (Eds.), *NASA formal methods* (pp. 361–368). Springer. https://doi.org/10.1007/978-3-030-55754-6_21

CrewAI. (n.d.). *CrewAI documentation*. Retrieved March 20, 2026, from https://docs.crewai.com/

Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G. A. F., Joshi, P., Plank, P., & Risbud, S. R. (2021). Advancing neuromorphic computing with Loihi: A survey of results and outlook. *Proceedings of the IEEE, 109*(5), 911–934. https://doi.org/10.1109/JPROC.2021.3067593

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., & Sifre, L. (2022). *Training compute-optimal large language models* [Preprint]. arXiv. https://arxiv.org/abs/2203.15556

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*(1), 79–87. https://doi.org/10.1162/neco.1991.3.1.79

Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation, 6*(2), 181–214. https://doi.org/10.1162/neco.1994.6.2.181

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). *Scaling laws for neural language models* [Preprint]. arXiv. https://arxiv.org/abs/2001.08361

Khan, D. Z. (2025a). *Embedded Neural Firmware (ENF): A deterministic, offline "Neural BIOS" for batteryless devices*. Zenodo. https://doi.org/10.5281/zenodo.17298403

Khan, D. Z. (2025b). *ENF Technical Note 01: Embedded Neural Firmware—The embedded AI crisis and the case for sealed neural firmware* (Report No. 1). ENFSystems LLC. https://doi.org/10.13140/RG.2.2.13715.34084

Khan, D. Z. (2025c). *ENF Technical Note 03: Proposed system architecture (ENF)* (Report No. 3). ENFSystems LLC. https://doi.org/10.13140/RG.2.2.25459.39207

Khan, D. Z. (2025d). *ENF Technical Note 04: The ENF framework and meta-vision*. ENFSystems LLC. https://doi.org/10.13140/RG.2.2.16163.52002

Khan, D. Z. (2026a). *AGIF Tasklet Cells: Verifier-backed offline AI artifacts for native applications*. Zenodo. https://doi.org/10.5281/zenodo.18946355

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwińska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences, 114*(13), 3521–3526. https://doi.org/10.1073/pnas.1611835114

LangChain. (n.d.). *LangGraph overview*. Retrieved March 20, 2026, from https://docs.langchain.com/oss/python/langgraph/overview

Legg, S., & Hutter, M. (2007). Universal intelligence: A definition of machine intelligence. *Minds and Machines, 17*(4), 391–444. https://doi.org/10.1007/s11023-007-9079-x

McMahan, B., Moore, E., Ramage, D., Hampson, S., & Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. *Proceedings of Machine Learning Research, 54*, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

Microsoft. (n.d.). *AutoGen*. Retrieved March 20, 2026, from https://microsoft.github.io/autogen/stable/index.html

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., & Lowe, R. (2022). *Training language models to follow instructions with human feedback* [Preprint]. arXiv. https://arxiv.org/abs/2203.02155

Schuman, C. D., Kulkarni, S. R., Parsa, M., Mitchell, J. P., Date, P., & Kay, B. (2022). Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science, 2*, 10–19. https://doi.org/10.1038/s43588-021-00184-y

Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots, 8*(3), 345–383. https://doi.org/10.1023/A:1008942012299

Troubitsyna, E., Alvarez, I. J., Koopman, P., & Trapp, M. (2024). Methods and tools for the engineering and assurance of safe autonomous systems (Dagstuhl Seminar 24151). *Dagstuhl Reports, 14*(4), 23–41. https://doi.org/10.4230/DagRep.14.4.23

Wooldridge, M. (2009). *An introduction to multiagent systems* (2nd ed.). Wiley.

Zhong, W., Yang, C., Liang, W., Cai, J., Chen, L., Liao, J., & Xiong, N. (2023). Byzantine fault-tolerant consensus algorithms: A survey. *Electronics, 12*(18), 3801. https://doi.org/10.3390/electronics12183801

Appendix A. ENF vs. Tasklet Cells vs. AGIF Fabric v1

The table below summarizes the architectural progression from ENF-family systems to Tasklet Cells and then to AGIF Fabric v1. This comparison is especially important for the present paper because Tasklet Cells are the missing middle layer between sealed ENF-style discipline and the broader governed-fabric architecture of AGIF.

The appendix is deliberately synthetic. It condenses the paper's ENF → Tasklet Cells → AGIF architectural argument into a single comparative table rather than introducing a new claim set.

Aspect	**ENF / ENF-Sync / ENF-Swarm**	**Tasklet Cells**	**AGIF Fabric v1**	**New Risks Introduced by Relaxation**
Model mutability	Models fixed after deployment	Fixed artifact after deployment	Limited local updates under governance	Catastrophic forgetting; model poisoning
Communication content	Compact coordination messages only	Narrow contract I/O only	Skill descriptors, workspace signals, and trust metadata	Bandwidth saturation; descriptor manipulation
Cross-domain composition	No meaningful transfer	No governed cross-cell transfer	Descriptor-mediated reuse across cells and tissues	Mis-composed skills; conflicting objectives
World modelling	No broader shared predictive state	Local artifact state only	Shared workspace and partial fabric-level state	Consensus failure; stale-state poisoning
Memory philosophy	Knowledge fixed at deployment	Stored local memory, but no reviewed inter-cell promotion	Reviewed promotion, compression, and intentional forgetting	Storage inflation; bad promotion
Population management	Static deployed nodes	Independent artifact instances; no fabric-versus-active population model	Fabric population versus active runtime population	Lifecycle instability; reactivation failure
Governance surface	Guaranteed by strict prohibition	Fail-closed, verifier-backed artifact boundaries	Active governance, replay, rollback, quarantine, and containment	Governance complexity; approval-path bypass risk
Adaptation	Not permitted after deployment	Not permitted after deployment	Bounded local adaptation with reviewer approval	Drift; unsafe or low-value updates

Coordination scope	Bounded swarm-style coordination only	Application-layer coordination outside the artifact itself	Tissue-level coordination through shared workspace	Workspace poisoning; coordination overhead
Architectural role	Bounded embedded-intelligence discipline	Practical software-cell foundation	Governed adaptive multi-cell fabric	Broader attack and verification surface

This appendix makes the design trade-off explicit. ENF-family systems achieve safety primarily by forbidding post-deployment change. Tasklet Cells carry that discipline into a practical software-artifact form: verifier-backed, local-first, and fail-closed, while still remaining sealed after deployment. AGIF Fabric v1 preserves the bounded and governance-oriented posture of that lineage while relaxing selected constraints to allow controlled adaptation, coordination, and reviewed knowledge reuse.

The result is greater capability, but also a wider risk surface that must be governed explicitly rather than avoided through prohibition alone. In that sense, this appendix makes the paper's central trade-off visible in one place: Tasklet Cells are the bridge from bounded sealed artifacts to governed adaptive fabrics, and AGIF gains adaptive and coordinative power by relaxing selected ENF and Tasklet constraints under explicit governance.

Appendix B. Quantified Assumptions and Actuals

This appendix compares the original working assumptions of the AGIF programme with the measured or verified values reached in AGIF v1. Its purpose is to show where the software proof aligned with the original design envelope, where it exceeded expectations, and where the actual implementation revealed constraints that are important for later phases.

Parameter	**Assumed range (original)**	**AGIF v1 actual**
Software-first working memory	8–12 GB	< 12 GB (verified)
Software-first storage budget	25–35 GB	Within budget on a host with 65 GiB available disk; repo footprint measured locally at approximately 477 MB
Registered fabric population	Greater than active cells by ~2×	128 logical / 24 active → >5×
Retained memory growth	Bounded and reviewed; no raw-log persistence	Verified: ephemeral logs, reviewed promotion only
Active runtime utilization during standard workload	Steady active set should remain well below capacity	10/24 = 41.7% during the Phase 7 benchmark; still far below the 128 logical-population cap

Descriptor message size	32–128 bytes	~595–597 bytes warm-tier descriptors
Unsafe/misaligned action rate	< 0.1%	0.000 for all governed classes
Catastrophic forgetting	≤ 10%	0.000 (adaptation class: 1.000 on all cases)

Several points in this comparison are especially important. First, the runtime memory and storage assumptions were met within the intended software-first envelope. Second, the logical-to-active population separation proved stronger than originally assumed, with the implemented fabric showing a >5× relationship rather than only ~2×. Third, memory discipline remained aligned with the design intent: raw logs stayed ephemeral, and long-term retention required reviewed promotion.

The most informative mismatch concerns descriptor size. The original assumption of 32–128 bytes proved too optimistic for the current software proof, where warm-tier descriptors are approximately 595–597 bytes. This does not invalidate the AGIF v1 result, but it does matter for later edge and hardware phases. In practical terms, it means the software architecture has succeeded before the descriptor object has been compressed to the level that would make ENF-like physical deployment straightforward.

Taken together, these actuals support the main architectural claim of AGIF v1 while also identifying the constraints that should guide future work. The appendix therefore serves both as a validation table and as an assumption ledger for later phases of the AGIF programme.

Appendix C. Falsification Criteria (Formal Version)

This appendix states the bounded falsification conditions for AGIF v1 in compact form. Its purpose is to make the paper's proof burden explicit: AGIF v1 succeeds only if the architecture satisfies the thresholds below within the committed benchmark and verification envelope.

Criterion	**Failure condition**	**AGIF v1 actual**
Skill-acquisition efficiency	AGIF v1 fails if descriptor-mediated reuse improves completion or accuracy by less than 3% relative to the non-adaptive baseline	+33.3% relative on alias cases
Continual learning	AGIF v1 fails if prior-task accuracy drops by more than 10% after adaptation	0% drop; adaptation class achieves 1.000 on all cases
Resource autonomy	AGIF v1 fails if more than 5% of cases terminate in resource-triggered abort	0 aborts
Memory discipline	AGIF v1 fails if unreviewed raw-log persistence accounts for more than 20% of retained artifacts	0%; raw logs are ephemeral only
Elastic population	AGIF v1 fails if active population exceeds 70% of the enforced steady	41.7% (10/24) during the Phase 7 benchmark

	active-cap envelope during standard workloads	
Governance	AGIF v1 fails if replay fidelity falls below 99%, rollback success below 95%, or quarantine success below 95%	Replay determinism 1.000; governance success 1.000 for the full adaptation class
Safety	AGIF v1 fails if policy-violating or uncontained behavior appears in more than 1% of cases	0.000 unsafe rate
Coordination overhead	AGIF v1 fails if coordination overhead exceeds the non-adaptive baseline by more than 25% without delivering the required performance gains	Both classes use identical tissue paths; adaptation adds authority overhead of +0.333 reviews per case for a +0.250 accuracy gain

Taken together, these criteria express the paper's bounded falsification frame in operational form. The appendix is therefore not a new results section, but a compact statement of what would have counted as architectural failure and what the measured AGIF v1 outcomes were under those same criteria.

Appendix D. Claims Matrix

Every major paper claim maps to a concrete artifact and a verification command. The purpose of this appendix is to make the evidentiary chain explicit: claims are not merely stated, but are tied to named files and reproducibility paths.

Claim	Evidence	Verification command
Fabric boots without old-repo dependency	CHANGELOG.md (Phase 3)	python3 scripts/check_phase3_foundation.py
Logical population is separable from the active runtime population	PHASE4_LIFECYCLE_EVIDENCE.md	python3 scripts/check_phase4_lifecycle.py
A burst active population of 48 returns automatically to a steady population of 24	PHASE4_LIFECYCLE_EVIDENCE.md	python3 scripts/check_phase4_lifecycle.py
Raw logs are never auto-promoted	PHASE5_MEMORY_EVIDENCE.md	python3 scripts/check_phase5_memory.py
Memory pressure triggers consolidation	PHASE5_MEMORY_EVIDENCE.md	python3 scripts/check_phase5_memory.py
Authority vetoes unauthorized structural change	PHASE6_ROUTING_AUTHORITY_EVIDENCE.md	python3 scripts/check_phase6_routing_authority.py
Routing abstains rather than forcing a weak choice	PHASE65_HARDENING_EVIDENCE.md	python3 scripts/check_phase6_routing_authority.py

Flat-baseline unsafe rate = 0.500	phase7_benchmark_results.md	python3 scripts/check_phase7_benchmarks.py
Multi-cell unsafe rate = 0.000	phase7_benchmark_results.md	python3 scripts/check_phase7_benchmarks.py
Adaptation-class accuracy = 1.000	phase7_benchmark_results.md	python3 scripts/check_phase7_benchmarks.py
Descriptor reuse causally changes 3 cases	phase7_benchmark_results.md (descriptor table)	python3 scripts/check_phase7_benchmarks.py
Adaptation delivers a 9× efficiency advantage	phase7_benchmark_results.md (tradeoffs)	python3 scripts/check_phase7_benchmarks.py
All 6 baseline cases have explicit route-of-custody	phase7_benchmark_results.md (custody table)	python3 scripts/check_phase7_benchmarks.py
F8A forgetting ≤ 10%	phase7_benchmark_results.md	python3 scripts/check_phase7_benchmarks.py
F8B unsafe rate ≤ 0.1%	phase7_benchmark_results.md	python3 scripts/check_phase7_benchmarks.py
F8D efficiency ≤ 2× flat	phase7_benchmark_results.md (tradeoffs)	python3 scripts/check_phase7_benchmarks.py
Runtime working set remains ≤ 12 GB	phase8_bounded_validation.md	python3 scripts/check_phase8_soak.py
Deterministic reruns remain identical	Identical hash reruns verified locally	Run python3 scripts/check_phase7_benchmarks.py twice
Real 24-hour MSI soak completed	phase8_real_24h_soak.md; PHASE8_LONGRUN_EVIDENCE.md	python3 scripts/check_phase8_soak.py
Real 72-hour MSI soak completed	phase8_real_72h_soak.md; PHASE8_LONGRUN_EVIDENCE.md	python3 scripts/check_phase8_soak.py
Phase 8 exit gate is met	PASS_TOKENS.md; PROGRESS_TRACKER.md; PHASE8_LONGRUN_EVIDENCE.md	python3 scripts/check_phase8_soak.py
Phase 9 exit gate is met	PASS_TOKENS.md; PROGRESS_TRACKER.md; PHASE9_CLOSURE_EVIDENCE.md	python3 scripts/check_phase9_closure.py

This appendix is intentionally operational. Its role is to let a reviewer move directly from a claim in the paper to the artifact and command path that supports it. In that sense, the claims matrix is one of the key transparency mechanisms of the AGIF v1 paper: it makes the evidence chain inspectable rather than implicit.

Appendix E. Evidence Directory

This appendix lists the principal files, directories, and verification scripts that make up the AGIF v1 evidence base. Its purpose is to help a reviewer navigate the repository efficiently by showing where each major part of the proof record is located.

Path	Contents
03_design/INTERFACE_FREEZE.md	All frozen interface types
03_design/BENCHMARK_CONTRACT.md	Frozen benchmark classes and metrics
02_requirements/FALSIFICATION_THRESHOLDS.md	All hard-fail and support thresholds
05_testing/PASS_TOKENS.md	Earned pass tokens P3–P9 plus P8_HARNESS_READY
05_testing/PHASE7_TISSUES_BENCHMARK_EVIDENCE.md	Phase 7 closure evidence
05_testing/PHASE76_HARDENING_EVIDENCE.md	Phase 7.6 hardening evidence
05_testing/PHASE85_HARDENING_EVIDENCE.md	Phase 8.5 hardening evidence
06_outputs/result_tables/phase7_benchmark_results.md	All benchmark result tables
06_outputs/result_tables/phase7_benchmark_results.json	Machine-readable benchmark results
06_outputs/run_summaries/phase8_bounded_validation.md	Phase 8 bounded soak summary
scripts/check_phase7_benchmarks.py	One-command benchmark reproduction
scripts/check_phase8_soak.py	One-command soak-harness verification
05_testing/PHASE8_LONGRUN_EVIDENCE.md	Phase 8 long-run evidence, including real 24-hour and 72-hour analysis
06_outputs/run_summaries/phase8_real_24h_soak.md	Narrative real 24-hour soak summary
06_outputs/run_summaries/phase8_real_24h_soak.json	Machine-readable real 24-hour soak summary
08_logs/phase8_soak/run_24h/	Imported MSI 24-hour soak manifest, cycle evidence, and stress artifacts
06_outputs/run_summaries/phase8_real_72h_soak.md	Narrative real 72-hour soak summary
06_outputs/run_summaries/phase8_real_72h_soak.json	Machine-readable real 72-hour soak summary
08_logs/phase8_soak/run_72h/	Imported MSI 72-hour soak manifest, cycle evidence, stress artifacts, and runtime state
05_testing/PHASE9_CLOSURE_EVIDENCE.md	Phase 9 closure evidence and honest package limits
06_outputs/evidence_bundle_manifests/phase9_claims_to_evidence_matrix.md	Repo-local claims-to-evidence matrix
06_outputs/evidence_bundle_manifests/phase9_reproducibility_package.md	Reviewer package order, expected outputs, and machine-role distinctions

05_testing/MSI_SOAK_MACHINE_NOTE.md	Canonical MSI soak-machine provenance, pre-soak verification, and honest limits
05_testing/V1X_ORGANIC_LOAD_EVIDENCE.md	Evidence for the bounded organic-load lane and governed split/merge usefulness
05_testing/V1X_SKILL_GRAPH_EVIDENCE.md	Evidence for the governed descriptor skill graph
05_testing/V1X_POS_DOMAIN_EVIDENCE.md	Evidence for the bounded POS extension domain
05_testing/V1X_BUNDLE_CLOSURE_EVIDENCE.md	Closure evidence for the completed V1X extension bundle
06_outputs/paper_drafts/README.md	Public note that the included R5 paper draft is a working draft, not the final publication
06_outputs/evidence_bundle_manifests/agif_v1_release_note.md	Release note summarizing the AGIF v1 evidence bundle and closure status
06_outputs/paper_drafts/AGIF_v1_paper_R5_2026-03-18.docx	Repo-local R5 DOCX paper draft aligned to the closure package and GitHub remote
06_outputs/paper_drafts/AGIF_v1_paper_R5_2026-03-18.pdf	Repo-local R5 PDF paper draft aligned to the closure package and GitHub remote
scripts/check_phase9_closure.py	One-command Phase 9 closure verification
scripts/check_v1x_bundle.py	One-command verification of the completed V1X extension bundle

Taken together, these files define the practical evidence surface of AGIF v1. They cover the frozen benchmark contract, the falsification thresholds, the benchmark outputs, the long-run soak record, the closure package, the completed post-closure extension bundle, the machine-role notes, and the verification scripts required to reproduce the bounded proof.

The repo-local paper-draft filenames retain historical R5 archival labels. This is a naming holdover only and does not indicate an evidence mismatch in the closure record.

Appendix F. Reproducibility Commands

This appendix lists the core commands required to reproduce the bounded AGIF v1 evidence chain from the repository. The command set is intentionally minimal. Its purpose is to provide a reviewer with one direct entry point for the full closure path while also exposing the benchmark, soak, regression, and completed extension-bundle checks individually.

```
git clone https://github.com/Ahsadin/agif_fabric_v1.git
cd agif_fabric_v1

# Full AGIF v1 closure verification (Phase 9)
python3 scripts/check_phase9_closure.py

# Full benchmark verification (Phase 7)
python3 scripts/check_phase7_benchmarks.py

# Phase 8 soak-harness verification
```

```
python3 scripts/check_phase8_soak.py

# Full regression chain (Phases 3–6)
python3 scripts/check_phase3_foundation.py
python3 scripts/check_phase4_lifecycle.py
python3 scripts/check_phase5_memory.py
python3 scripts/check_phase6_routing_authority.py

# Confirm the recorded GitHub remote
git remote get-url origin
# expected: https://github.com/Ahsadin/agif_fabric_v1

# Completed V1X extension-bundle verification
python3 scripts/check_v1x_bundle.py
```

All commands are deterministic for the benchmark and bounded-harness artifacts. `python3 scripts/check_phase9_closure.py` is the single reviewer entry point for the strict v1 closure baseline: it rechecks the chained benchmark and bounded-soak commands, validates the working paper-draft package and MSI note, and inspects the imported MSI soak artifacts locally. `python3 scripts/check_v1x_bundle.py` is the corresponding entry point for the completed post-closure extension bundle, including the organic-load lane, governed skill graph, bounded POS domain, and root v1 closure path.

The machine-role distinction remains explicit. MSI is the final long-run evidence machine for AGIF v1, and MacBook Air-only long-run proof is not claimed. This appendix therefore serves a practical as well as a methodological role: it tells a reviewer exactly how to reproduce the closed v1 proof path without reopening the architecture as an active development branch.

