CoMemBench: Benchmarking Collaborative Memory Boundaries across Multi-Agent Workflow Topologies

Anonymous authors
Paper under double-blind review

Abstract

Multi-agent workflows require task-relevant information to be shared across agents, while irrelevant, stale, unverified, or incompatible information must remain isolated. We call this task-conditioned scope of information a *collaborative memory boundary*. Workflow topology determines which intermediate artifacts are applicable to which downstream workers and when they cease to be valid, thereby providing a structural stress dimension for sharing and isolation. Existing memory benchmarks primarily evaluate retention and retrieval, whereas multi-agent benchmarks emphasize coordination and end-to-end completion, leaving topology-conditioned memory boundaries largely unmeasured. We introduce CoMemBench, an execution-grounded benchmark for collaborative memory sharing and isolation across multi-agent workflow topologies. It constructs 800 composite workflows across four domains from source-grounded dependency graphs, with node-local specifications, verifiable artifact handoffs, native evaluators, and matched isolation challenges. CoMemBench measures workflow completion, verified node progress, required-handoff reliability, isolation robustness, and token cost. Experiments reveal a sharing–isolation trade-off: broader context improves information availability but can weaken isolation, while system rankings shift across topologies and artifact violations. CoMemBench is available at https://huggingface.co/datasets/anonymous-submission-333/anonymous-submission.

1 Introduction

Large language model agents increasingly tackle complex tasks through teams of specialized workers (Zhu et al., 2025; Ke et al., 2026). These workers act on different instructions, observations, and execution histories, making selective information reuse central to collaboration. A downstream task must receive the evidence, state, or intermediate result it requires, while contextually inapplicable information—such as a sibling-branch result, an unverified output, or a superseded state—must not alter its execution. We call this task- and state-dependent distinction a *collaborative memory boundary*. Maintaining such a boundary has two requirements: required informa-

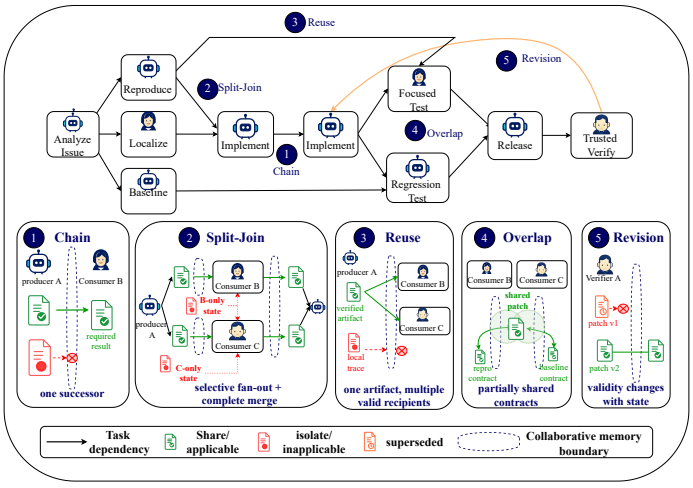


Figure 1: **Topology-conditioned memory boundaries.** Five dependency motifs change where workflow records are required, branch-specific, reused, partially shared, or superseded.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Comparison with related benchmarks.

Benchmark	Multi-Agent Workflow	Workflow Topology Diversity	Artifact Handoffs	Sharing Evaluation	Isolation Evaluation	Containerized Execution
LoCoMo (Maharana et al., 2024)	–	–	–	–	–	–
LongMemEval (Wu et al., 2025)	–	–	–	–	–	–
MemoryAgentBench (Hu et al., 2026a)	–	–	–	–	–	–
GroupMemBench (Yang et al., 2026a)	–	–	–	–	–	–
GateMem (Ren et al., 2026)	–	–	–	–	✓	–
PAC-Bench (Park et al., 2026)	✓	–	–	–	✓	–
PeopleJoin (Jhamtani et al., 2025)	–	–	–	–	–	–
SILO-BENCH (Zhang et al., 2026b)	✓	✓	✓	–	–	–
MultiAgentBench (Zhu et al., 2025)	✓	–	–	–	–	–
TaskBench (Shen et al., 2024)	–	✓	–	–	–	–
MASBench (Ke et al., 2026)	✓	✓	✓	–	–	–
Toolathlon (Li et al., 2026a)	–	–	–	–	–	✓
CoMemBench	✓	✓	✓	✓	✓	✓

tion must remain available to the appropriate consumer, and inapplicable information must not change that consumer's verified behavior.

Workflow topology makes this boundary vary across consumers and execution states. Figure 1 illustrates five recurring cases. A chain transfers a required result to its successor; a split–join keeps branch-specific records separate before combining their outputs; reuse makes one verified result relevant to several consumers; overlap creates both shared and branch-specific dependencies; and bounded revision supersedes an earlier result after verification failure. These motifs do not define different kinds of memory; they change the consumers, workflow states, and dependency regions under which the same record is applicable. The same intermediate result may therefore be required along one edge, inapplicable to a sibling branch, reusable by several consumers, or superseded after revision. Our central question is whether multi-agent systems can maintain both sides of the collaborative memory boundary as these structural conditions change.

Existing benchmarks cover complementary aspects of this problem, including long-context memory, governed or privacy-constrained group memory, multi-agent orchestration, and executable tool use (Maharana et al., 2024; Wu et al., 2025; Hu et al., 2026a; Yang et al., 2026a; Ren et al., 2026; Park et al., 2026; Zhu et al., 2025; Shen et al., 2024; Ke et al., 2026; Li et al., 2026a). As Table 1 shows, they do not jointly evaluate whether systems deliver required information and resist contextually inapplicable information across topology-diverse, executable workflows. A checkmark denotes a capability explicitly instantiated and evaluated. End-to-end success alone cannot distinguish a missing required handoff from an otherwise valid consumer being altered by an inappropriate record.

We introduce CoMemBench, an execution-grounded benchmark of 800 source-grounded workflows across four domains. CoMemBench makes three contributions: **benchmark formulation**, operationalizing collaborative memory boundaries through declared handoffs and matched isolation interventions across diverse workflow topologies; **dataset and construction**, providing executable workflows with node-local specifications, verifiable artifact handoffs, and native evaluators; and **evaluation and findings**, separating terminal completion, verified progress, required sharing, and isolation robustness across systems, backbones, and memory mechanisms. Experiments show that broader information availability can improve progress without preserving isolation, while system behavior varies across workflow topology and artifact-violation type.

2 Related Work

2.1 Agent Memory and Information Boundaries

Memory evaluation has expanded from fixed-corpus evidence use (Bai et al., 2024; Hsieh et al., 2024; Petroni et al., 2021; Thakur et al., 2021) to persistent and interactive settings. MemoryBank, LoCoMo, LongMemEval, RealTalk, and StoryBench evaluate long-term or multi-session recall and reasoning (Zhong et al., 2024; Maharana et al., 2024; Wu et al., 2025; Lee et al., 2025; Wan & Ma, 2025), while MemoryAgentBench adds incremental ingestion, test-time learning, and selective

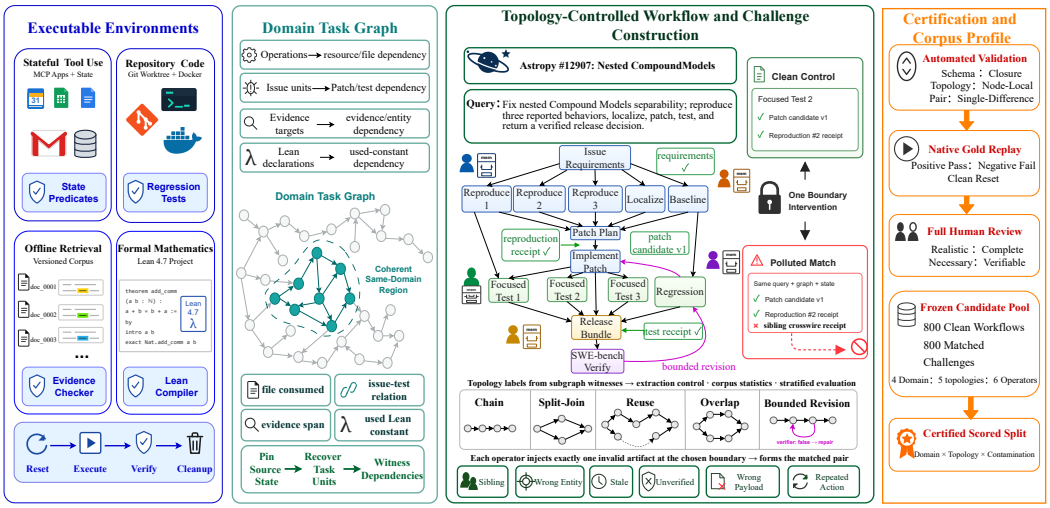


Figure 2: Overview of the CoMemBench construction pipeline.

forgetting (Hu et al., 2026a). These benchmarks move beyond static reading but primarily study memory accumulated and used by one focal agent, rather than its selective availability among agents executing interdependent tasks.

GroupMemBench evaluates speaker-grounded memory (Yang et al., 2026a), GateMem evaluates utility, access control, and active forgetting in a shared pool (Ren et al., 2026), and PAC-Bench studies private memories under two-agent privacy constraints (Park et al., 2026). They show that disclosure matters beyond recall, but scope memory mainly through speakers, identities, roles, or access policies. CoMemBench instead asks which artifact a consumer requires at its current task state and which plausible artifact must remain outside its context, making sharing and isolation a boundary conditioned by task dependencies, verification, and version.

2.2 Multi-Agent and Workflow Benchmarks

AgentBench, MindAgent, and LLM-Coordination evaluate interactive or controlled collaboration (Liu et al., 2024; Gong et al., 2024; Agashe et al., 2025), and MultiAgentBench varies star, chain, tree, and graph *interaction protocols* (Zhu et al., 2025). TaskBench and MASBench instead vary task dependencies, including chains, DAGs, depth, aggregation, and parallelism (Shen et al., 2024; Ke et al., 2026). These works establish the importance of organization and workflow structure, but neither interaction topology nor end-to-end success reveals whether memory was selectively exposed according to task dependencies.

τ-bench and AppWorld provide stateful executable outcomes (Yao et al., 2025; Trivedi et al., 2024); Toolathlon and Toolathlon-GYM provide long-horizon application workflows and reproducible environments (Li et al., 2026a; Zhang et al., 2026a); and PeopleJoin and SILO-BENCH study distributed information (Jhamtani et al., 2025; Zhang et al., 2026b). These benchmarks ground outcomes or collaboration, but lack controlled tests of required sharing and isolation. CoMemBench extracts topology-diverse workflows from source task graphs, validates them natively, and tests typed, versioned handoffs with matched clean–polluted runs, separating completion, progress, sharing, and isolation.

3 The CoMemBench Benchmark

CoMemBench represents each instance as an executable task graph with node-local specifications, typed and version-aware handoffs, resources, and a domain-native evaluator. Its pipeline recovers domain task graphs, extracts topology-controlled workflows, adds matched isolation challenges, and certifies their execution. Figure 2 summarizes the design.

3.1 WORKFLOW AND BOUNDARY OPERATIONALIZATION

An executable workflow comprises a request q, task graph $G = (V, E)$, node-local specifications $\mathcal{X} = \{x_v\}_{v \in V}$, resources and initial state $\mathcal{R}$, and a domain-native evaluator Φ:

$$\mathcal{W} = (q, G, \mathcal{X}, \mathcal{R}, \Phi). \tag{1}$$

Nodes denote executable task units, while edges encode witnessed task dependencies rather than physical agents or an agent communication network. A required edge $e = (u, v)$ carries a handoff contract $h_e = (u, v, \tau, z, \nu, \sigma)$ that records the producer and consumer, semantic type, entity scope, version, and verification requirement. The corresponding artifacts are benchmark-visible workflow records, including evidence, plans, candidate results, execution-state and verification receipts, and domain deliverables.

For a consumer v at workflow state s, an information item m is applicable when it can legitimately support v under the current task dependencies and state; otherwise it is inapplicable. The *collaborative memory boundary* is the task- and state-conditioned distinction between these cases. Its required-sharing side preserves information needed by the consumer, while its isolation side prevents inapplicable information from changing the consumer's verified behavior. This notion covers episodic records, structured state, and procedural or skill-like memory, independent of how a system stores or retrieves them.

COMEMBENCH makes the two sides observable at declared workflow handoffs. A required artifact provides a positive probe for sharing; a matched, contextually inapplicable artifact provides a negative intervention for isolation. Handoff metadata—including semantic type, entity scope, version, and verification state—supplies the task- and state-specific evidence used to construct and validate these probes. Evaluation then uses artifact exposure and native execution receipts, allowing systems to realize memory through transcripts, retrieval stores, structured representations, or procedural mechanisms.

3.2 EXECUTABLE ENVIRONMENTS

COMEMBENCH executes 800 workflows in domain-native environments. Across 200 workflows per domain, stateful tasks draw from 25 pinned MCP servers detailed in Appendix D.1, activating only the 4–10 declared per task; code tasks span 12 repositories and use issue-specific Docker environments evaluated by the official SWE-bench harness against fixed base commits and FAIL-TO-PASS/PASS-TO-PASS tests; retrieval tasks use 200 immutable corpus snapshots containing 16,335 document records, with document-level and exact-span verification; and mathematical tasks compile submitted proofs in isolated copies of three commit-locked Lean 4.7 projects. These native evaluators produce node-level receipts and terminal workflow verdicts.

Each run follows a prepare–execute–verify–cleanup protocol bound to pinned assets, environment configurations, source revisions, and evaluator versions. Stateful tasks additionally record state digests and cleanup receipts, while infrastructure outcomes remain separate from method failures.

3.3 RECOVERING DOMAIN TASK GRAPHS

From sources with realistic requirements and executable evaluators, we recover task units and dependencies in four domains: stateful tool use, repository code change, offline retrieval, and formal mathematics. Source instances provide requirements, assets, and evaluators rather than ready-made multi-agent workflows. We pin assets by task identifier, commit, corpus snapshot, or environment digest; shared entities, resources, state transitions, and evaluator contracts then define graph edges.

Recovery follows each source's native structure. We align Toolathlon operations with resources, initialization, and evaluator predicates (Li et al., 2026a; Zhang et al., 2026a); expand SWE-bench Verified issues into reproduction, localization, patch, and test units (Jimenez et al., 2024); decompose BrowseComp+ items into evidence targets, entity constraints, and synthesis (Chen et al., 2025); and recover declaration graphs and used constants from three miniCTX Lean 4.7 projects (Hu et al., 2024). We exclude declarations transitively depending on `sorryAx` and retain only edges witnessed by consumed files, issue–test links, evidence requirements, or Lean constants.

3.4 TOPOLOGY-CONTROLLED WORKFLOW AND CHALLENGE CONSTRUCTION

Topology-controlled workflow extraction. Within each domain graph, we select a connected, scenario-coherent subgraph with one terminal objective, complete dependency closure, necessary witnessed edges, and native-evaluator coverage. We assign topology labels from explicit graph witnesses: *chain* denotes a serial dependency; *split–join*, independent branches that reconverge; *reuse*, one typed artifact consumed by multiple downstream regions; *overlap*, consumers with partially shared predecessor sets; and *bounded revision*, one verifier-triggered repair and superseding artifact. Labels are multi-label because a workflow may contain several motifs.

Query and node-local realization. For each extracted graph G_i, we construct a natural request q_i and a local task x_v for every node. The request specifies the objective, entities, resources, deliverables, constraints, and acceptance conditions without prescribing an execution plan, following Toolathlon's distinction between realistic underspecification and information required for deterministic execution (Li et al., 2026a). Each node satisfies *node-local closure*: its task, authorized environment view, and declared incoming artifacts contain the information required for execution. Every required handoff records its semantic type, entity scope, version, and verification requirement. Together, node-local closure and declared handoff contracts instantiate the required-sharing side of the collaborative memory boundary: they specify which upstream records each consumer needs and make their delivery and use observable.

Matched isolation challenges. For every clean workflow, we select a target consumer supported by a local topology witness and construct a matched variant that inserts exactly one contextually inapplicable artifact into the target consumer's benchmark-visible incoming-artifact view. The challenge label, injection plan, and evaluator predicate remain hidden from the evaluated system. Six operators instantiate concrete violations: *sibling crosswire* routes an artifact from the wrong branch; *incompatible payload* violates the handoff's semantic contract; *unverified result* lacks a required success receipt; *completed-as-pending* presents an already completed side effect as still requiring execution; *wrong entity* binds a type-compatible artifact to a different resource; and *stale version* exposes a superseded artifact. The target topology characterizes the structural setting of the tested handoff, while the operator specifies how the injected artifact becomes inapplicable. Split–joins support crosswires, bounded revisions support stale versions, and chain, reuse, and overlap expose handoff, entity, payload, or verification mismatches; Figure 3 reports the resulting combinations. Each pair otherwise shares its query, graph, environment binding, tools, and non-target inputs. The polluted target is scored only when its clean counterpart verifies successfully. The clean and polluted variants therefore instantiate the two sides of the same tested consumer relation: the clean run establishes that required handoffs support a solvable target, while the matched intervention tests whether one inapplicable artifact changes its verified behavior.

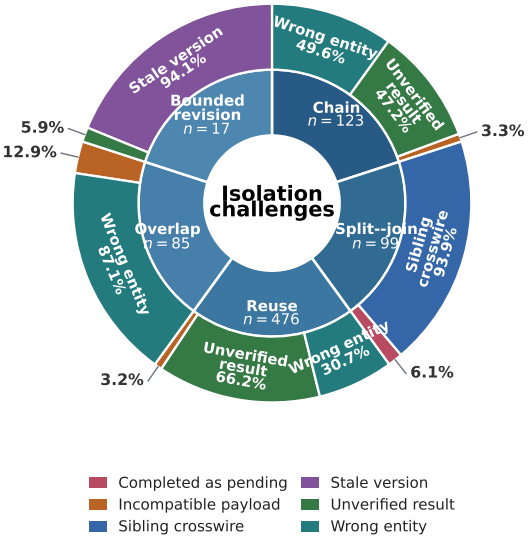


Figure 3: Topology–operator challenge distribution.

3.5 CERTIFICATION AND CORPUS PROFILE

Certification. A workflow enters the scored split only after three gates. Automated checks enforce source locks, graph closure, acyclicity outside declared revision edges, topology witnesses, node-local closure, control–polluted equality, evaluator coverage, and removal of private material. A private *Gold execution plan* then runs the workflow from a clean reset through the released harness and evaluator; targeted negative mutations must fail, while repeated executions must produce equivalent receipts

Table 2: Structural demand and multi-label topology coverage.

	Structural demand (mean)							Execution-topology coverage (count)					
Domain	N	Tasks	Depth	Width	Handoffs	Fan-outs	Joins	Chain	Split–join	Reuse	Overlap	Revision	Types/WF
Stateful tool use	200	7.78	6.02	2.30	17.17	4.62	4.75	200	200	199	197	70	4.33
Repository code	200	11.58	6.00	4.78	19.97	3.81	4.81	200	200	200	200	74	4.37
Offline retrieval	200	9.35	8.35	2.00	19.04	6.35	3.94	200	200	200	200	126	4.63
Formal mathematics	200	11.70	4.59	5.58	14.47	2.88	3.56	200	200	200	190	144	4.67
All / Avg.	800	10.10	6.24	3.66	17.66	4.41	4.26	800	800	799	787	414	4.50

and semantic state digests. Finally, human reviewers assess query plausibility and completeness, node-local executability, edge necessity, and evaluator–goal alignment. Rejected workflows are repaired and recertified or excluded, and infrastructure defects quarantine the affected family rather than count as method failures. Audit records retain each decision and its receipts, source-grouped splits limit near-duplicate leakage, and Gold plans never enter prompts, public traces, or scores.

Corpus profile. The certified release contains 800 clean workflows, evenly distributed across four domains, and 800 matched isolation challenges. Table 2 summarizes per-workflow structural demand and multi-label topology coverage: a workflow contributes every motif present in its graph, whereas receipt-routed conditional revisions are counted as revision boundaries rather than ordinary source-graph edges. Across the release, a workflow averages 10.10 executable task units, 17.66 required handoffs, and 4.50 topology types. The four domains pair these structures with distinct executable artifacts—mutable application state, patches and test evidence, corpus evidence spans, and proof terms with compiler receipts—so the profile captures both graph-level coordination demand and domain-native verification rather than graph size alone.

4 Evaluation Protocol

We evaluate workflow execution at two levels. **SR** measures clean-workflow terminal success, and **Verified Node Completion Rate** (VNCR), $|\mathcal{C}|^{-1}\sum_{q\in\mathcal{C}}|\mathcal{V}_q|/|\mathcal{U}_q|$, measures the macro-average verified fraction of required nodes. **Verified Handoff Success** (VHS), $|\mathcal{H}^+|/|\mathcal{H}|$, is the fraction of eligible consumers that complete successfully after their mandatory direct predecessors verify and their declared artifacts are delivered. **Isolation Challenge Success** (ICS), $|\mathcal{P}^+|/|\mathcal{P}|$, is polluted-target success among matched pairs whose corresponding clean target succeeds. For the single ICS value in the system comparison, we set $\mathcal{P}=\bigcup_{d\in\mathcal{D}}\mathcal{P}_d$ and micro-pool eligible matched target pairs across the four domains, so each pair, rather than each domain, receives equal weight. Accordingly, VHS evaluates the required-sharing side of a collaborative memory boundary, whereas ICS evaluates its isolation side; SR and VNCR provide the surrounding task-completion context.

Results are reported by domain and further stratified by target topology and pollution operator. Benchmark and infrastructure failures are separated from method failures, token cost is analyzed in Section 5.5, and formal eligibility and accounting rules appear in Appendix F.

5 Experiments

We compare systems across domains and backbones, isolate memory mechanisms under a fixed executor, and stratify results by topology and pollution operator to test whether verified progress, required sharing, and isolation improve together.

5.1 Experimental Setup

Evaluated systems. We group ten baselines by collaboration abstraction: *general-purpose runtimes*, Codex and Hermes (OpenAI, 2026; Nous Research, 2026); *multi-agent frameworks*, Deep Agents, CrewAI, and AutoGen (LangChain AI, 2026; CrewAI, Inc., 2026; Wu et al., 2024); *fixed collaboration methods*, Peer Review, SMoA, and OMAC (Xu et al., 2023; Li et al., 2025; 2026b); and *automated MAS design*, ADAS and EvoMAS (Hu et al., 2025; 2026b). Each receives the same node-local

Table 3: System and backbone results.

Backbone	System	Stateful Tool			Repository Code			Offline Retrieval			Formal Mathematics			ICS
		SR	VNCR	VHS	SR	VNCR	VHS	SR	VNCR	VHS	SR	VNCR	VHS	
Qwen3.8-27B	Codex	0.0	5.1	32.7	0.0	33.2	43.0	4.1	23.3	10.7	0.0	9.1	5.3	0.0
	Hermes	0.0	3.2	18.2	0.0	25.2	30.2	4.1	35.7	14.4	0.0	6.2	2.0	0.0
	Deep Agents	0.0	5.5	1.4	0.0	57.7	66.8	2.0	37.8	9.5	0.0	11.4	3.4	38.1
	CrewAI	0.0	38.3	67.1	0.0	44.1	54.8	12.0	53.1	38.1	0.0	10.8	9.3	12.9
	AutoGen	0.0	33.9	56.8	0.0	48.8	55.1	18.0	50.7	37.3	0.0	10.3	8.7	20.0
	Peer Review	0.0	34.9	61.8	4.0	49.7	55.1	0.0	24.3	5.0	0.0	14.0	11.5	19.1
	SMoA	0.0	43.8	72.4	0.0	44.5	51.0	16.0	53.3	45.3	0.0	13.3	4.7	10.1
	OMAC	0.0	31.8	55.3	0.0	52.9	62.1	8.0	39.9	38.9	0.0	7.4	5.8	17.3
	ADAS	0.0	18.3	36.6	0.0	17.0	37.3	0.0	12.5	0.0	0.0	6.5	8.2	7.4
	EvoMAS	0.0	21.0	39.0	4.0	67.6	73.2	0.0	13.6	0.0	0.0	19.4	10.8	26.8
DeepSeek-V4-Flash	Codex	0.0	2.7	0.0	0.0	17.1	33.2	0.0	5.1	0.0	0.0	9.5	3.9	0.0
	Hermes	0.0	5.6	9.4	0.0	26.5	26.1	2.0	45.7	17.1	0.0	4.2	0.0	0.0
	Deep Agents	0.0	4.3	1.8	0.0	18.8	26.1	0.0	5.3	0.0	0.0	1.4	0.0	0.0
	CrewAI	0.0	14.4	29.6	0.0	40.1	45.0	2.0	33.7	9.7	0.0	9.2	4.1	6.1
	AutoGen	0.0	15.0	28.8	0.0	63.8	71.9	4.0	40.5	24.8	0.0	10.2	6.9	15.6
	Peer Review	0.0	17.6	24.0	0.0	52.0	65.2	2.0	23.3	17.2	0.0	9.9	6.0	17.1
	SMoA	0.0	21.5	51.9	2.0	76.8	85.8	2.0	41.0	32.9	0.0	8.2	3.2	14.9
	OMAC	0.0	66.2	82.5	4.0	84.2	89.4	10.0	65.5	57.6	0.0	12.3	5.3	4.4
	ADAS	0.0	17.8	52.0	8.0	67.6	80.9	4.0	29.9	16.1	0.0	11.5	2.0	14.3
	EvoMAS	0.0	12.9	22.2	6.0	58.8	68.9	0.0	13.6	4.1	0.0	17.6	9.6	9.8
GPT-6-Luna	Codex	0.0	6.0	2.0	0.0	13.3	17.2	0.0	3.5	0.0	0.0	4.0	0.0	0.0
	Hermes	0.0	6.0	43.8	0.0	25.9	25.4	0.0	31.4	11.5	0.0	14.7	0.0	0.0
	Deep Agents	0.0	3.0	0.0	0.0	12.5	32.1	0.0	8.9	0.0	0.0	0.0	0.0	0.0
	CrewAI	0.0	35.7	63.1	0.0	52.5	64.4	0.0	41.6	8.2	0.0	8.4	6.0	13.3
	AutoGen	0.0	19.0	69.2	0.0	37.6	73.3	4.0	41.6	31.1	0.0	12.0	2.4	11.8
	Peer Review	0.0	13.7	35.8	0.0	3.1	20.0	0.0	10.2	0.0	0.0	4.0	0.0	0.0
	SMoA	0.0	29.6	68.5	0.0	49.2	64.1	2.0	43.5	17.0	0.0	11.2	2.5	15.1
	OMAC	0.0	39.7	66.9	0.0	35.3	72.4	0.0	28.5	12.5	0.0	7.8	1.4	4.4
	ADAS	0.0	26.8	69.8	0.0	26.7	31.3	2.0	36.0	9.4	0.0	6.5	7.4	7.1
	EvoMAS	0.0	43.6	63.7	0.0	55.5	67.9	0.0	17.0	5.2	0.0	18.0	16.4	7.0

task, authorized tools, and declared inputs while retaining its own routing, scheduling, and memory decisions; Appendix E gives exact configurations.

Controlled memory comparison. To isolate memory, a fixed topology-preserving supervisor–worker executor compares LOCAL-ONLY, FULL CONTEXT, SHARED FLAT RAG, Mem0 (Chhikara et al., 2025), A-Mem (Xu et al., 2025), and PlugMem (Yang et al., 2026b). Prompts, workers, tools, graph, evaluator, budgets, and retries remain fixed; Appendix E specifies each implementation.

Backbones and protocol. System comparisons use Qwen3.8-27B, DeepSeek-V4-Flash, and GPT-6-Luna on a fixed domain-stratified split; the memory comparison fixes Qwen3.8-27B. Clean and polluted runs share isolated task-scoped environments and the declared consumer boundary. Manifests record model and framework settings, unsupported combinations are "–", and scoring and accounting follow Section 4 and Appendix F.

5.2 SYSTEM AND BACKBONE COMPARISON

Table 3 tests whether greater verified progress preserves both sides of the boundary. It reports domain-specific SR, VNCR, and VHS, while one system-level ICS micro-pools eligible matched target pairs across domains. All entries are percentages; Section 4 and Appendix F define scoring.

Task success and verified progress. No supported configuration solves a Stateful Tool or Formal Mathematics workflow, and the best SR is only 8.0 in Repository Code and 18.0 in Offline Retrieval. Yet some zero-SR configurations reach 84.2 VNCR and 89.4 VHS, verifying substantial subgraphs and downstream consumers without closing the terminal objective. The VNCR–VHS gap further shows that upstream completion does not guarantee a successful required handoff; workflow completion, verified progress, and sharing reliability expose distinct failure stages.

Table 4: Memory-mechanism results under controlled multi-agent execution.

Memory mechanism	Stateful Tool				Repository Code				Offline Retrieval				Formal Mathematics			
	SR	VNCR	VHS	ICS	SR	VNCR	VHS	ICS	SR	VNCR	VHS	ICS	SR	VNCR	VHS	ICS
Controls																
LOCAL-ONLY	0.0	74.6	84.0	97.1	0.0	77.1	89.5	89.4	2.0	63.6	83.9	60.9	0.0	13.6	26.9	0.0
FULL CONTEXT	2.0	86.9	86.7	72.4	16.0	61.7	89.1	71.9	6.0	77.1	88.6	80.6	0.0	11.8	26.8	0.0
SHARED FLAT RAG	2.0	86.9	86.4	70.6	18.0	72.4	90.8	73.9	8.0	74.1	88.4	75.0	0.0	14.4	29.7	20.0
External memory systems																
MEM0	0.0	83.8	85.5	86.3	2.0	77.9	89.6	89.1	4.0	64.9	84.7	73.9	0.0	12.2	22.7	52.4
A-MEM	4.0	85.8	86.6	87.9	18.4	58.2	88.4	63.9	6.0	71.8	85.6	71.9	0.0	12.3	26.2	0.0
PLUGMEM	0.0	83.1	85.1	90.0	0.0	68.2	88.9	91.3	4.0	56.9	83.9	52.9	0.0	14.2	29.0	33.3

Sharing versus isolation. Required sharing and isolation do not improve together. General-purpose runtimes are generally weak in verified progress, while explicit multi-agent organizations improve particular domains without dominating both boundary requirements. With Qwen3.8-27B, SMoA reaches 43.8/72.4 VNCR/VHS on Stateful Tool but only 10.1 ICS; Deep Agents instead attains 38.1 ICS despite weak progress. With DeepSeek-V4-Flash, OMAC leads progress in Stateful Tool, Code, and Retrieval yet reaches only 4.4 ICS, below Peer Review's 17.1; rankings shift again under GPT-6-Luna. Because VHS tests required handoffs and ICS conditions on a successful clean target, stronger execution and coordination do not by themselves preserve both sides of the boundary.

5.3 MEMORY MECHANISM COMPARISON

Table 4 fixes the executor, workers, handoffs, tools, and evaluator, varying only additional cross-worker exposure. It retains domain-specific ICS because memory policies may interact with domain-native artifacts. Relative to LOCAL-ONLY, which preserves declared handoffs but disables extra cross-worker memory, FULL CONTEXT raises Stateful Tool VNCR from 74.6 to 86.9 and Code SR from 0.0 to 16.0, but lowers ICS from 97.1 to 72.4 and from 89.4 to 71.9. SHARED FLAT RAG reaches 90.8 Code VHS yet only 73.9 ICS, showing that broader or semantically relevant exposure can improve availability while weakening task- and state-dependent isolation.

Structured stores likewise do not dominate all outcomes. A-Mem reaches 18.4 Code SR but only 63.9 ICS, whereas PlugMem preserves 91.3 Code ICS with 0.0 SR and falls to 52.9 ICS on Retrieval. Formal Mathematics remains unsolved at workflow level. Thus retention, linking, and procedural organization may help particular domains without consistently preserving both required access and isolation.

5.4 TOPOLOGY AND ISOLATION ANALYSIS

We stratify Qwen3.8-27B traces by target topology in Table 5 and pollution operator in Appendix Table 9; both summarize Table 3. Topology identifies where a consumer boundary is tested, while the operator identifies why an artifact is inapplicable. Within each cross-domain stratum, VNCR pools canonical nodes, VHS eligible consumers, and ICS matched pairs whose clean target succeeds; workflows are not assigned wholesale to every motif. All entries are percentages.

Boundary maintenance is not a single system-level capability. Chain consumers have the highest VHS for every system, while branch separation, reuse, overlap, and revision change which records must be routed, separated, or superseded. Rankings do not follow one difficulty order: EvoMAS reaches 72.7 ICS on split–join but 14.3 on reuse, whereas Deep Agents reaches 80.0 on overlap but 28.6 on reuse. Protecting one local boundary therefore does not ensure robustness when information is reused or partially shared.

Appendix Table 9 provides the complementary semantic diagnosis: robustness to one violation does not reliably transfer to another. Topology determines where an artifact can affect execution, whereas semantics, scope, verification, and version determine whether it should affect that consumer. These strata explain complementary boundary pressures but are descriptive rather than causal topology–operator comparisons.

Table 5: Topology-stratified task progress and isolation success.

System	Chain			Split–join			Reuse			Overlap			Bounded revision		
	VNCR	VHS	ICS	VNCR	VHS	ICS	VNCR	VHS	ICS	VNCR	VHS	ICS	VNCR	VHS	ICS
Codex	29.7	62.3	0.0	20.2	29.1	0.0	22.8	25.9	0.0	21.9	29.6	0.0	20.7	31.2	0.0
Hermes	28.4	56.5	0.0	20.1	20.5	–	22.7	14.8	0.0	21.4	19.8	–	25.0	0.0	–
Deep Agents	42.4	81.3	33.3	33.4	40.1	42.9	38.3	39.2	28.6	36.1	40.4	80.0	32.5	30.8	–
CrewAI	45.7	81.1	20.0	40.0	49.7	0.0	47.3	49.6	16.2	42.8	50.0	0.0	43.3	47.5	0.0
AutoGen	**47.9**	80.9	33.3	39.9	46.8	25.0	46.9	46.6	19.4	42.6	47.0	0.0	43.3	52.2	0.0
Peer Review	46.6	**85.1**	44.4	32.5	43.1	0.0	36.5	43.0	14.3	34.9	43.9	0.0	28.1	53.4	20.0
SMoA	47.0	79.5	25.0	**41.8**	51.7	0.0	**49.4**	53.8	11.9	**44.1**	52.2	0.0	**44.6**	**55.0**	0.0
OMAC	45.3	84.4	28.6	36.5	51.0	16.7	43.3	50.3	13.3	39.1	51.1	25.0	33.9	38.1	20.0
ADAS	16.5	56.7	0.0	14.2	26.9	33.3	16.4	25.3	0.0	15.4	27.7	0.0	14.5	42.6	0.0
EvoMAS	46.6	79.3	30.0	33.8	**51.8**	72.7	37.3	**54.8**	14.3	36.1	**52.8**	0.0	20.3	26.7	0.0

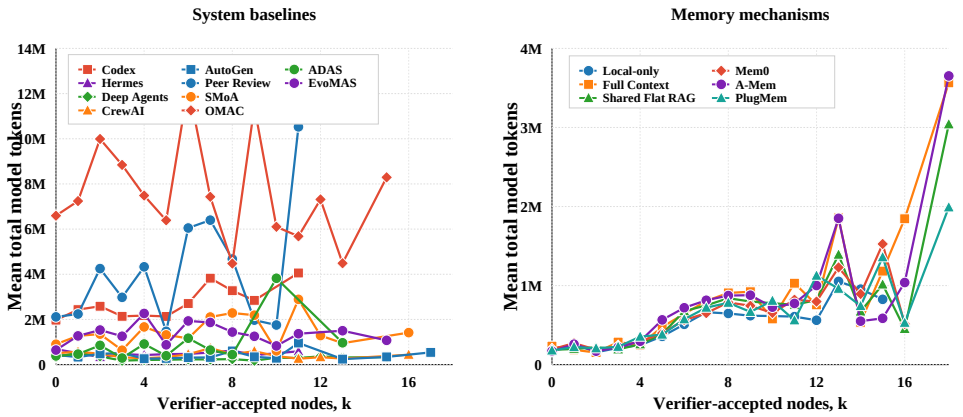


Figure 4: Progress-conditioned token cost; panels use separate linear scales.

5.5 Cost Analysis

Figure 4 compares tokens at equal verifier-accepted progress. Deep Agents' supervisor–subagent loop averages 0.31M tokens per task, versus 8.01M for OMAC's multi-round process and 3.61M for Peer Review's create–review–revise loop. None dominates SR, VHS, or ICS, so coordination cost guarantees neither reliable handoffs nor isolation (Appendix F).

Compared with the 489k-token Local-only control, Full Context and Shared Flat RAG use 672k and 621k tokens without consistently improving both sharing and isolation; PlugMem costs 521k but varies by domain. Efficient collaborative memory therefore requires selective exposure, not simply more context.

6 Conclusion

Multi-agent workflows must reuse upstream information without allowing contextually inapplicable records to alter downstream execution; topology makes this boundary vary across consumers and states. CoMemBench operationalizes this problem with 800 executable workflows spanning four domains and diverse topologies. Declared handoffs measure required sharing, while matched challenges and native evaluators test isolation through downstream behavior. Across systems, backbones, and memory mechanisms, completion, verified progress, handoff reliability, and isolation remain distinct: broader access can improve progress without preserving isolation, and rankings vary by topology and violation. Effective collaborative memory therefore requires selective, task- and state-conditioned exposure rather than simply more context.

AI USE STATEMENT

Generative AI tools were used for language and presentation editing and routine implementation assistance. The authors designed the study, verified the data and results, and take full responsibility for the paper and released artifacts.

REPRODUCIBILITY STATEMENT

We will release the benchmark data, clean–polluted pair specifications, schemas, evaluation code, prompts, and environment configurations. Each run records system and model versions, decoding and budget settings, source and evaluator revisions, and asset checksums. Appendix A documents the released records, Appendix B describes construction and certification, and Appendices D–F specify the execution substrates, baseline implementations, and scoring protocol.

REFERENCES

Saaket Agashe, Yue Fan, Anthony Reyna, and Xin Eric Wang. LLM-Coordination: Evaluating and analyzing multi-agent coordination abilities in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 8053–8072, 2025. doi: 10.18653/v1/2025.findings-naacl.448. URL https://aclanthology.org/2025.findings-naacl.448/.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, et al. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

CrewAI, Inc. Crewai: Framework for orchestrating role-playing autonomous agents. https://github.com/crewAIInc/crewAI, 2026. Software documentation and source code.

Ran Gong, Qiuyuan Huang, Xiaojian Ma, Yusuke Noda, Zane Durante, Zilong Zheng, Demetri Terzopoulos, Li Fei-Fei, Jianfeng Gao, and Hoi Vo. MindAgent: Emergent gaming interaction. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3154–3183, 2024. doi: 10.18653/v1/2024.findings-naacl.200. URL https://aclanthology.org/2024.findings-naacl.200/.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Jiewen Hu, Thomas Zhu, and Sean Welleck. miniCTX: Neural theorem proving with (long-) contexts. *arXiv preprint arXiv:2408.03350*, 2024. URL https://arxiv.org/abs/2408.03350.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, 2025.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=ZgQ0t3zYTQ.

Yuntong Hu, Yuting Zhang, Matthew Trager, Yi Zhang, Shuo Yang, Wei Xia, and Stefano Soatto. EvoMAS: Evolutionary generation of multi-agent systems. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b.

Harsh Jhamtani, Jacob Andreas, and Benjamin Van Durme. LLM agents for coordinating multi-user information gathering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 17800–17826, 2025. doi: 10.18653/v1/2025.findings-acl.916. URL https://aclanthology.org/2025.findings-acl.916/.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Zixuan Ke, Yifei Ming, Austin Xu, Ryan Chin, Xuan-Phi Nguyen, Prathyusha Jwalapuram, Jiayu Wang, Semih Yavuz, Caiming Xiong, and Shafiq Joty. MAS-Orchestra: Understanding and improving multi-agent reasoning through holistic orchestration and controlled benchmarks. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=3fGXBm4c3S.

LangChain AI. Deep agents: The batteries-included agent harness. https://github.com/langchain-ai/deepagents, 2026. Software documentation and source code.

Dong-Ho Lee, Adyasha Maharana, Jay Pujara, Xiang Ren, and Francesco Barbieri. RealTalk: A 21-day real-world dataset for long-term conversation. *arXiv preprint arXiv:2502.13270*, 2025. URL https://arxiv.org/abs/2502.13270.

Dapeng Li, Zhen Tan, Peng Qian, Yuheng Li, Kunal Chaudhary, Li Hu, and Jianwei Shen. SMoA: Improving multi-agent large language models with sparse mixture-of-agents. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 54–65. Springer, 2025.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, Junteng Liu, Zhaochen Su, Yiyang Guo, Fan Zhou, Lueyang Zhang, Juan Michelini, Xingyao Wang, Xiang Yue, Shuyan Zhou, Graham Neubig, and Junxian He. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=z53s5p0qhf.

Shijun Li, Hilaf Hasson, and Joydeep Ghosh. OMAC: A holistic optimization framework for LLM-based multi-agent collaboration. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. AgentBench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Nous Research. Hermes agent. https://github.com/NousResearch/hermes-agent, 2026. Software documentation and source code.

OpenAI. Codex: Lightweight coding agent that runs in your terminal. https://github.com/openai/codex, 2026. Software documentation and source code.

Minjun Park, Donghyun Kim, Hyeonjong Ju, Seungwon Lim, Dongwook Choi, Taeyoon Kwon, Minju Kim, and Jinyoung Yeo. PAC-BENCH: Evaluating multi-agent collaboration under privacy

constraints. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 31030–31056, 2026. doi: 10.18653/v1/2026.findings-acl.1552. URL https://aclanthology.org/2026.findings-acl.1552/.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. KILT: A benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2523–2544, 2021. doi: 10.18653/v1/2021.naacl-main.200. URL https://aclanthology.org/2021.naacl-main.200/.

Zhe Ren, Yibo Yang, Yimeng Chen, Zijun Zhao, Benshuo Fu, Zhihao Shu, Bingjie Zhang, Yangyang Xu, Dandan Guo, and Shuicheng Yan. GateMem: Benchmarking memory governance in multi-principal shared-memory agents. *arXiv preprint arXiv:2606.18829*, 2026. URL https://arxiv.org/abs/2606.18829.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. TaskBench: Benchmarking large language models for task automation. In *Advances in Neural Information Processing Systems*, volume 37, pp. 4540–4574, 2024. doi: 10.52202/079017-0148. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/085185ea97db31ae6dcac7497616fd3e-Abstract-Datasets_and_Benchmarks_Track.html.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Luanbo Wan and Weizhi Ma. StoryBench: A dynamic benchmark for evaluating long-term memory with multi turns. *arXiv preprint arXiv:2506.13356*, 2025. URL https://arxiv.org/abs/2506.13356.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, 2025.

Zhenran Xu, Senbao Shi, Baotian Hu, Jindi Yu, Dongfang Li, Min Zhang, and Yuxiang Wu. Towards reasoning in large language models via multi-agent peer review collaboration. *arXiv preprint arXiv:2311.08152*, 2023.

Jingbo Yang, Kwei-Herng Lai, Xiaowen Wang, Shiyu Chang, Yaar Harari, and Evgeniy Gabrilovich. GroupMemBench: Benchmarking LLM agent memory in multi-party conversations. *arXiv preprint arXiv:2605.14498*, 2026a. URL https://arxiv.org/abs/2605.14498.

Ke Yang, Zixi Chen, Xuan He, Jize Jiang, Michel Galley, Chenglong Wang, Jianfeng Gao, Jiawei Han, and ChengXiang Zhai. PlugMem: A task-agnostic plugin memory module for LLM agents. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026b.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R. Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Puzhen Zhang, Weijie Bai, Wendong Fan, and Guohao Li. Toolathlon-GYM: A large-scale environment for real-world tool-use agents. GitHub repository, 2026a. URL https://github.com/eigent-ai/toolathlon_gym.

Yuzhe Zhang, Feiran Liu, Yi Shan, Xinyi Huang, Xin Yang, Yueqi Zhu, Xuxin Cheng, Cao Liu, Ke Zeng, Terry Jingchen Zhang, and Wenyuan Jiang. SILO-BENCH: A scalable environment for evaluating distributed coordination in multi-agent LLM systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29379–29398, 2026b. doi: 10.18653/v1/2026.acl-long.1354. URL https://aclanthology.org/2026.acl-long.1354/.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946. URL https://ojs.aaai.org/index.php/AAAI/article/view/29946.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Zhe Wang, Zhenhailong Wang, Cheng Qian, Xiangru Tang, Heng Ji, and Jiaxuan You. MultiAgentBench: Evaluating the collaboration and competition of LLM agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8580–8622, 2025. doi: 10.18653/v1/2025.acl-long.421. URL https://aclanthology.org/2025.acl-long.421/.

Table 6: Pinned sources and domain-graph recovery.

Domain	Source snapshot	Task units	Dependency evidence
Stateful tool use	Toolathlon-GYM `45bb4935`	Source-authored operations over named resources and side effects	Resource flow, state transitions, evaluator predicates, and typed source-workflow dependencies
Repository code	SWE-bench Verified `c104f84`	Issue analysis, reproduction, localization, patching, and regression checks for one issue	Base commit, issue–test links, modified symbols, and FAIL-TO-PASS/PASS-TO-PASS contracts
Offline retrieval	BrowseComp+ `144cff8e`	Query constraints, fixed-corpus evidence partitions, entity resolution, answer evidence, and synthesis	Hidden qrels, stable document identifiers, exact spans, constraint coverage, and provenance requirements
Formal mathematics	HepLean `7448822a`; HTPI `8eeebaec`; PNT `23650db8`	Lean declarations in an elaborated target cone	Proof-term used constants, source modules and lines, and Lean 4.7 elaboration

A Benchmark Schema and Runtime Protocol

The release separates reusable workflow definitions from executions. A `workflow_family` record stores the source references, global request, node-local tasks, typed dependency edges, topology witnesses, execution contract, and split. A `benchmark_instance` binds one family to either its clean control or matched polluted variant; the two variants share the family, scenario, and environment initialization. The public package contains 800 families, 1,600 instances, and 8,975 node-local contexts. Private evaluator oracles and pollution plans are distributed only with the operator package.

At execution time, a worker receives a `node_view`: its role and task, authorized local context, declared incoming artifacts, handoff contract, output type, and an opaque capability token. An artifact records its semantic type, producer, scenario and entity scope, version, verification state, payload, and provenance. The native evaluator returns a verification receipt, and only a verified receipt permits a required artifact to satisfy a downstream dependency. Systems that control routing may emit a `memory_proposal` containing an artifact, producer, and proposed recipients; the harness records the proposal and the artifacts actually exposed to each consumer.

These records describe the benchmark-visible boundary, not an architecture's internal memory. A system may use a transcript, vector store, structured memory, or no persistent store, but private reasoning and store contents are neither inspected nor treated as ground truth. Replay instead follows the manifest-fixed case initialization, node views, native actions, receipts, issued artifacts, handoffs, consumed-artifact identifiers, and pollution events. Schema validation, content hashes, and receipt signatures make an exposure trace replayable without assuming a common internal memory representation.

B Data Sources and Construction Details

Table 6 records the pinned sources and the evidence used to recover task dependencies. Source repositories are frozen at the listed revisions; their license and attribution files are preserved. Toolathlon-GYM is Apache-2.0, SWE-bench and BrowseComp+ are MIT licensed, and the three Lean projects retain their project-specific upstream licenses. MCP-server components remain subject to their own licenses rather than being relicensed by CoMemBench.

For each source record, we first build a domain-level graph and then extract a connected subgraph with one terminal objective, complete ancestor closure, and a native evaluator. No workflow joins unrelated source cases or crosses domains. Selection retains only edges with the evidence in Table 6 and favors subgraphs that instantiate the released topology motifs without adding synthetic dependencies. The request is then realized from the selected source goal, entities, resources, deliverables, constraints, and acceptance conditions; node tasks expose only the local information and declared predecessors required to perform their work. This produces a new multi-agent workflow while preserving the source task and its executable semantics.

Splits are assigned at family level, so the clean and polluted variants and all node views of a workflow remain together. Repository and Lean-project group checks additionally prevent the same repository

or proof project from crossing splits. The released assignment contains 177 development and 623 test families, balanced at 200 families per domain overall.

C Quality Assurance and Audit Protocol

Certification combines exhaustive machine gates with semantic review. JSON Schema validation covers families, instances, contexts, node views, artifacts, proposals, and private oracles. Graph gates check identifier closure, reachable terminal objectives, acyclicity outside declared revision edges, topology witnesses, and a dependency-evidence record on every edge. Node-local closure is checked against the task, authorized resources and capabilities, and required incoming handoffs; an undeclared path, tool, predecessor, or output semantic fails the gate. All 800 families pass the released graph and execution-contract audits.

Evaluator closure is tested by replay rather than final-text judging. The deterministic graph-and-handoff oracle accepts the known-good trace and rejects its targeted negative trace for all 800 families. Domain-native evaluators then determine node and terminal success during benchmark execution; all 200 retrieval families additionally pass document, qrel, and exact-span closure. For isolation challenges, every plan must identify a real donor, a target handoff, an invalidity reason, a local topology witness, and a deterministic observable failure predicate. The candidate gate passes for all 800 matched plans, and its single-difference manifest certifies that query, graph, evaluator, environment, and non-target inputs are unchanged.

Human review is repair-oriented rather than a majority-vote annotation task. Reviewers compare each candidate with its source record and native evaluator using four questions: whether the request is natural and complete, whether every node is executable from its local view, whether each retained edge is necessary for its consumer, and whether evaluator acceptance entails the requested outcome. A disputed item is adjudicated by rechecking the source evidence and evaluator contract; it is then repaired and rerun through all gates or excluded. We therefore do not treat an automatic language-model score as evidence of realism or report an artificial inter-annotator agreement statistic.

Finally, a public–private leakage audit rejects any public occurrence of Gold answers or patches, qrel mappings, pollution operators, target labels, failure predicates, or evaluator-only fields. Source-group split checks prevent repository and proof-project leakage, and digest comparisons detect reset drift. The final release audit reports no schema, graph, oracle-replay, source-invariant, or public leakage findings.

D Native Evaluators and Environment Setup

D.1 MCP application environment

The application environment comprises 25 MCP servers. MCP defines the agent-facing action protocol, while the corresponding resources reside in task workspaces, PostgreSQL-backed applications, and locally managed services. Before a workflow begins, CoMemBench materializes its initial files and application state in a family-isolated workspace, database, and service namespace. At each node, the harness starts only the servers admitted by the node capability contract. It checks process startup and tool-schema discovery before exposing the tools, and records the arguments, native observations, and state changes of executed calls. Environment initialization, state export, evaluator invocation, and cleanup remain benchmark control operations and are not available to evaluated agents.

Table 7: MCP server inventory in the pinned application-task source environment. Runtime identifiers are normalized to readable server names.

Capability group	MCP servers	Count
Files and office documents	Filesystem, Excel, Word, PowerPoint, PDF Tools	5
Collaboration and productivity	Email, Google Calendar, Google Forms, Google Sheets, Notion, Canvas, Memory	7
Research and web content	Local arXiv, arXiv LaTeX, Scholarly Search, Fetch, Playwright, YouTube, YouTube Transcript, HowToCook	8
Business and structured data	Snowflake, WooCommerce, Yahoo Finance, 12306 Rail	4
Local computation	Terminal	1
Total		25

Server source, dependencies, build artifacts, and launch configurations are pinned before evaluation and applied uniformly to all systems. Environment certification records server entrypoints, database routing, initial and final state digests, and cleanup receipts; a task with an unavailable declared server is excluded from method-level scoring.

D.2 Non-MCP native substrates

The other execution substrates retain their native interfaces instead of being artificially wrapped as MCP services. Code outputs are submitted to the official SWE-bench evaluation entrypoint, `swebench.harness.run_evaluation`, which applies the patch inside the instance-specific Docker environment and executes its FAIL_TO_PASS and PASS_TO_PASS tests. Mathematical outputs are inserted into an isolated copy of the corresponding Lean source file and checked with the pinned Lake-managed Lean 4.7 toolchain; the original source is restored after compilation. Retrieval nodes operate over immutable corpus snapshots and submit an answer with document identifiers and exact source spans. The retrieval verifier checks document existence, span provenance, declared constraint coverage, and answer correctness; it evaluates evidence selection and synthesis rather than open-web search behavior. All three substrates emit the same canonical action, artifact, receipt, and failure records used by the MCP path.

E Baseline Instantiations and Run Manifests

Table 8 records the exact implementation evaluated for each system. The adapter supplies node-local tasks, authorized actions, and declared incoming artifacts, and records traces without choosing answers, routing, schedules, or memory contents.

Table 8: System baseline implementations.

Baseline	Version / revision	Evaluated instantiation
Codex	CLI 0.152.1	Official `codex exec -json` loop with isolated configuration and native event capture.
Hermes	0.21.0 (`d3e2ace`)	Native `AIAgent.chat` and `delegate_task` execution with sandbox tools.
Deep Agents	0.7.12 (`3db758c`)	Official `create_deep_agent` supervisor–subagent loop with native model and sandbox-tool bindings.
CrewAI	1.15.18 (`b608a35`)	Native `Agent`/`Task`/`Crew` objects with sequential process orchestration.
AutoGen	0.7.5 (`027ecf0`)	`RoundRobinGroupChat` with native messages and tool callbacks.
Peer Review	EvoMAS source `93fd9d6`	Three creators produce candidates, followed by nine reviews, three revisions, and aggregation.
SMoA	Source `1f3444d`	Official sparse-mixture process with its role layers and aggregation policy preserved.
OMAC	Source `6bccfec`	Frozen five-dimensional, three-round LLMLP package; optimized functions and collaboration structure are retained.
ADAS	Source `2702bee`	The selected agent program is frozen before benchmark evaluation and executed through the official agent interface.
EvoMAS	Source `93fd9d6`; smolagents 1.26.0	Frozen evolved role, prompt, and organization configuration executed by `MasRuntime`.

Controlled memory mechanisms. All six mechanisms run in the same supervisor–worker executor. LOCAL-ONLY exposes only worker-local state; FULL CONTEXT exposes every prior visible artifact without retrieval; and SHARED FLAT RAG retrieves the top four artifacts from one workflow-local store using `all-MiniLM-L6-v2`. Mem0 0.1.104 uses its local `add`/`search` API with a workflow-specific Qdrant store. A-Mem (source `0c8039f`) writes verified artifacts as linked notes and retrieves related notes for each node. PlugMem (source `3b2ce75`) uses the official graph write, recall, and reasoning path. Each external store is empty at workflow start; only the memory policy changes across rows.

Every run has an immutable manifest recording source and adapter revisions, model and decoding settings, budgets, configuration hashes, memory policy, data split, environment and evaluator hashes, seed, and trace schema. This separates a baseline's organization from the common execution and scoring substrate.

F METRIC DEFINITIONS AND SCORING DETAILS

For clean controls $\mathcal{C}$, SR is $|\mathcal{C}|^{-1}\sum_{q\in\mathcal{C}}\mathbb{I}[\text{terminal_success}(q)]$. Let $\mathcal{U}_q$ be the canonical required nodes of q and $\mathcal{V}_q \subseteq \mathcal{U}_q$ those with a verified completion; VNCR is $|\mathcal{C}|^{-1}\sum_{q\in\mathcal{C}}|\mathcal{V}_q|/|\mathcal{U}_q|$. Repeated executions and revision retries count once, while method-private nodes never enter either set.

For VHS, let $\mathcal{H}$ be non-root consumers whose mandatory direct predecessors have verified successfully, and let $\mathcal{H}^+$ be the successful subset. A consumer enters $\mathcal{H}^+$ only if every required predecessor artifact is exposed at the declared boundary, current, receipt-valid, explicitly consumed, and followed by native-verifier success. Thus VHS is $|\mathcal{H}^+|/|\mathcal{H}|$; unsupported slices are marked "–".

Each isolation pair z declares one target consumer t_z. Let $\mathcal{P}$ contain pairs for which t_z succeeds in the clean variant, and $\mathcal{P}^+$ those for which it also succeeds in the matched polluted variant. Thus ICS is $|\mathcal{P}^+|/|\mathcal{P}|$; eligibility is not terminal success of the full clean workflow. For the pooled system-level score, $\mathcal{P} = \bigcup_{d\in\mathcal{D}}\mathcal{P}_d$ is the union of eligible pair sets across domains; domain-specific scores restrict $\mathcal{P}$ to the eligible pair set of one domain. Clean and polluted variants share the query, graph, initial state, tools, and non-target inputs; only one evaluator-hidden inapplicable artifact at t_z differs.

Token accounting includes execution, coordination, memory-retrieval context, recovery or repair, and meta-level search or evolution. We report input/output tokens, mean, median, 90th percentile, successful-run mean, and total valid-run tokens divided by successful workflows. Benchmark-visible artifact exposure is reported separately. The eligibility conditions for VHS and ICS follow the definitions in Section 4. Benchmark and infrastructure failures are excluded from method-failure denominators.

G ADDITIONAL EVALUATION DETAILS

Pollution-operator breakdown. Table 9 stratifies the Qwen3.8-27B system runs by the violation that makes the injected artifact inapplicable. Eligible matched pairs are pooled across domains within each operator. EvoMAS is comparatively robust to sibling crosswires but remains vulnerable to unverified results and other observed violations, while Deep Agents performs strongly on wrong-entity challenges but less reliably on unverified results. The variation across columns shows that isolation robustness does not transfer uniformly between artifact violations.

Table 9: Isolation by pollution operator.

System	Sibling crosswire	Incompatible payload	Unverified result	Completed as pending	Wrong entity	Stale version
Codex	0.0	0.0	0.0	–	0.0	0.0
Hermes	–	–	0.0	–	–	–
Deep Agents	42.9	0.0	31.0	–	80.0	–
CrewAI	0.0	0.0	22.2	0.0	0.0	0.0
AutoGen	33.3	0.0	26.3	0.0	0.0	0.0
Peer Review	0.0	0.0	29.6	0.0	0.0	20.0
SMoA	0.0	0.0	17.5	0.0	0.0	0.0
OMAC	33.3	0.0	20.7	0.0	12.5	20.0
ADAS	33.3	0.0	0.0	–	0.0	0.0
EvoMAS	80.0	16.7	20.7	0.0	0.0	0.0

Table 10 reports the aggregate token statistics behind Figure 4. Values include model calls used for execution, coordination, recovery, and memory operations and are computed over runs with complete usage telemetry.

Table 10: Token accounting in thousands of tokens.

System	Mean	Median	Per verified node	Memory mechanism	Mean	Median	Per verified node
Codex	2350	1875	1098	Local-only	489	367	80
Hermes	518	382	194	Full Context	672	380	104
Deep Agents	310	227	89	Shared Flat RAG	621	389	94
CrewAI	461	263	112	Mem0	576	422	91
AutoGen	388	231	97	A-Mem	617	357	102
Peer Review	3608	2265	1022	PlugMem	521	392	90
SMoA	1419	858	333				
OMAC	8012	5536	1981				
ADAS	565	436	290				
EvoMAS	1306	977	343				

Table 11 states the failure attribution applied before metric aggregation. Only method failures enter method-performance denominators; benchmark and infrastructure failures remain separately auditable.

Table 11: Run-outcome attribution.

Outcome	Criterion	Scoring treatment
Success	Native evaluator accepts the required terminal state or target node	Positive outcome
Method failure	A valid run produces a rejected action/state, exhausts its protocol, or terminates without an admissible result	Included as failure
Benchmark failure	Released asset, schema, reset, oracle, or evaluator contract is inconsistent	Quarantined
Infrastructure failure	Required service, container, endpoint, or evaluator process is unavailable independently of system behavior	Excluded and rerun