

# From One Directive to Full-Organization Action: Organizational Mirroring for Multi-Agent LLM Systems

Anonymous Author(s)

## Abstract

Current multi-agent frameworks adopt flat communication topologies, requiring designers to manually orchestrate each agent interaction. We present *organizational mirroring*, an approach that allows a single natural-language directive to mobilize an entire LLM agent organization—mirroring how a CEO’s briefing cascades through a real company. The approach rests on eight principles: hierarchical delegation, independent memory, layered compression, meta-department, skill-based composition, self-evolution, replaceable executors, and real-world workflow mapping. We implement these in a production system of 18 agents across four departments (Game, AI, Life, Meta) on the OpenClaw framework, evaluating the three business departments (14 agents). Three novel capabilities extend the architecture: a Meta-Department providing structurally separated quality oversight and driving agent evolution through dual-layer evaluation, skill-based department composition enabling reuse of proven coordination patterns, and closed-loop self-evolution through three parallel learning loops fed by both business manager and Meta-Department assessments. Controlled experiments on a 30-task suite (90 runs across three topologies) provide initial evidence that: (1) the organizational topology produces 3.1–5.9× greater output volume (not cost-normalized; ORG uses 13–39× more API calls) while achieving internally assessed quality scores of 18.3/20 (95% CI: [18.13, 18.53],  $n=177$ ); (2) an automated structural quality scorer confirms significant cross-topology quality differences ( $H=42.52$ ,  $p<0.000001$ ,  $\epsilon^2=0.478$ ); (3) component ablation reveals hierarchical coordination as the primary quality driver (Cohen’s  $d=1.409$ ); (4) hierarchical structure reduces communication links by 85.7%; (5) independent memory eliminates cross-domain lexical intrusion in all 30 observed tasks; and (6) cross-validation between two structurally separated evaluation layers (business managers and Meta-Department) yields 87.5% exact agreement on 48 matched worker-level scores ( $\rho=0.952$ ), demonstrating convergent validity within the LLM-based evaluation framework. V3 ten-phase pipeline production validation (16 runs, 100% success, 28 reviews averaging 18.0/20) further confirms that embedded meta-review and automated evolution work in production workflows. Limitations include LLM-as-judge scoring, single-model evaluation, and restricted task domains. We release configuration templates and evaluation data as open-source artifacts.

## CCS Concepts

• Computing methodologies → Artificial intelligence.

## Keywords

Multi-Agent Systems; LLM Agents; Organizational Architecture; Intent Amplification; Agent Orchestration

## 1 Introduction

Over the past two years, LLM-powered agents have moved well beyond simple chatbots. Today’s agents use tools, form plans, and collaborate with one another [17, 22]—and deploying them in teams is the natural next step. Survey work by Guo et al. [6] and Wang et al. [19] catalogues dozens of such multi-agent systems. Empirical evidence supports the trend: Du et al. [4] show that multi-agent debate sharpens factual accuracy, while CAMEL [9] confirms that structured communication protocols consistently outperform ad-hoc message passing.

Yet a gap persists between what these systems can do and how easily a human can direct them. In a real company, a CEO walks in and says “This week we focus on user growth.” That single sentence cascades: department heads decompose it into team-specific goals, individual employees execute their tasks, managers review deliverables, and results flow back up. The CEO does not micromanage each interaction—the organizational structure handles amplification and coordination automatically.

No existing multi-agent framework replicates this pattern. AutoGen [21] supports flexible multi-agent conversations but imposes no organizational hierarchy. CrewAI [11] assigns roles to agents yet shares context across the entire crew. LangGraph [8] offers graph-based orchestration at the cost of significant cognitive overhead for designers. MetaGPT [7] comes closest—it embeds Standard Operating Procedures into agent workflows—but accepts only *task-level* directives (“Build a CLI Flappy Bird game”), not *strategic-level* intent (“Focus on user growth this week”). The distinction matters. Task-level input still requires the human to decide *what* to build; strategic-level input lets the organization decide. In each of the six frameworks we surveyed, the human must do more than issue a single directive—they must orchestrate.

This orchestration gap creates three concrete pain points:

**Memory Contamination.** When agents share a conversation context, domain knowledge bleeds across boundaries. A game designer’s brainstorming fragments end up in an algorithm benchmark’s prompt, degrading both agents’ outputs.

**Coordination Overhead.** Flat topologies scale poorly. With  $N=18$  agents, the number of possible directed communication channels reaches  $N(N-1) = 306$ —an  $O(N^2)$  explosion that makes message routing and conflict resolution intractable in practice.

**High Cognitive Cost.** Designers must internalize framework-specific abstractions—graphs, chains, crews—that have no real-world analog. The learning curve discourages adoption and complicates long-term maintenance.

Our central claim is simple: when a multi-agent system mirrors a real-world organization, a single human directive suffices to mobilize the entire system. The organizational structure itself handles task decomposition, delegation, execution, review, and delivery—just as it does in a company. This “intent amplification” rests on eight architectural principles:

- (1) **Hierarchical Delegation:** Information flows through management layers; each layer filters and compresses signals for the next.
- (2) **Independent Memory:** Each agent maintains an isolated workspace, preventing cross-domain contamination.
- (3) **Layered Memory Compression:** A three-tier memory system (short-term, mid-term, long-term) manages the inherent context window limitations of LLMs.
- (4) **Meta-Department for Structurally Separated Quality Oversight:** A dedicated department (Warden, Forge, Prism, Scout) that evaluates business department deliverables from a separate organizational vantage point and drives agent evolution, reducing single-evaluator dependency in manager scoring.
- (5) **Department Composition via Skill Orchestration:** Dynamic composition of workflows from four reusable Claude Code skills (`/agent-teams-playbook`, `/planning-with-files`, `/tdd`, `/refactor-clean`), enabling new departments to inherit proven coordination patterns.
- (6) **Self-Evolution Mechanism:** Three parallel learning loops (performance feedback, keyword optimization, capability registration) that automatically improve agent quality across workflow cycles without human intervention.
- (7) **Replaceable Executors:** The Manager-Worker pattern decouples task specification from execution, enabling model-level substitution without system-wide changes.
- (8) **Real-World Organizational Mirroring:** The ten-phase workflow (direction → planning → execution → review → meta-review → revision → verify → summary → feedback → evolve) directly maps to established project management practices, minimizing cognitive overhead.

We built a production system around these principles: 18 agents, four departments (Game, AI, Life, Meta), running on the OpenClaw framework [13] and publicly accessible at [anonymized deployment URL] during a one-week pilot deployment (2026-02-21 to 2026-02-27) at the time of writing. The system serves as both a working multi-agent workspace and a live testbed for organizational mirroring. Experimental evaluation covers the three business departments (14 agents); the Meta-Department receives a preliminary assessment in Section 3.5, with full experimental comparison deferred to future work.

**Contributions.** This paper makes six contributions:

- We propose *organizational mirroring*—eight architectural principles (hierarchical delegation, independent memory, layered compression, meta-department, department composition, self-evolution, replaceable executors, real-world workflow mapping) that let a single natural-language directive mobilize 18 agents across four departments without manual orchestration, through what we term *intent amplification*. The architecture additionally supports *autonomous agent initiative* via a heartbeat mechanism—scheduled, self-directed intelligence gathering requiring no human trigger.
- We introduce the **Meta-Department** concept—a structurally separated quality oversight layer comprising four specialized agents that evaluate business department deliverables from a distinct organizational vantage point, reducing single-evaluator dependency in the “LLM-as-judge” paradigm. Prism’s quality scores feed directly into the self-evolution mechanism (M7-1), closing the organizational learning loop. Cross-validation across 48 matched worker-level scores yields 87.5% exact agreement between managers and Meta-Department ( $\rho=0.952$ ), demonstrating convergent validity within the LLM-based evaluation framework; human expert validation remains necessary to establish ground-truth alignment.
- We implement a **closed-loop self-evolution system** with three parallel learning mechanisms (performance feedback, keyword optimization, capability registration) that draw from *dual-layer* evaluation—both business manager and Meta-Department assessments—to continuously improve agent quality. Pipeline correctness is code-validated by 59 unit tests, and the system is deployed in production.
- We implement these principles in a production system on the OpenClaw framework—including department composition via skill orchestration, a CEO gateway, three-file agent configuration convention, hierarchical communication protocol, and data synchronization pipeline—and release the configuration templates as open-source artifacts.
- We design a 30-task evaluation suite comparing organizational (ORG), single-agent (FLAT), and parallel-worker (CREW) topologies across communication efficiency, memory isolation, comprehensibility, and output quality, reporting results from 90 completed experiments with comprehensive statistical analysis (bootstrap CIs, Friedman test, Wilcoxon signed-rank, Kruskal-Wallis with effect sizes).
- We present a **component ablation analysis** using a uniform automated scorer across all topologies: hierarchical coordination with quality review provides the largest effect (Cohen’s  $d=1.409$ , large), multi-agent parallelism adds a smaller gain ( $d=0.342$ , small), and revision has negligible effect ( $d=0.049$  via manager scores), identifying the hierarchy-review mechanism as the primary driver of quality improvement.

## 2 Related Work

### 2.1 Multi-Agent LLM Frameworks

Multi-agent LLM frameworks have expanded rapidly, yet each major entry makes different trade-offs along the axes of flexibility, structure, and cognitive overhead. Talebirad and Nadiri [18] provide a foundational taxonomy of multi-agent collaboration patterns, distinguishing cooperative, competitive, and hybrid interaction models.

**AutoGen** [21] (Microsoft) pioneered conversational multi-agent patterns. Agents engage in flexible dialogues, and the v0.4 release (2025) introduced asynchronous messaging with event-driven patterns. However, the flat conversation model offers no organizational hierarchy—all participants share context, which invites memory contamination at scale.

**CrewAI** [11] took a different tack, organizing agents into role-based teams with defined responsibilities. Sequential and hierarchical task execution are both supported, and recent versions added

built-in memory (short-term, long-term, entity, and contextual). However, workspace isolation remains absent: agents in the same crew share context, and memory is pooled rather than physically separated per agent.

**LangGraph** [8] models agent interactions as state machines over a graph. Flexibility is unmatched, and the v1.0 release (late 2025) added durable execution with failure recovery. However, the abstraction cost is steep—designers must reason about topology, state transitions, and edge conditions, none of which map to familiar real-world concepts.

**MetaGPT** [7] comes closest to our approach by embedding SOPs into multi-agent workflows and assigning roles like Product Manager and Architect. Three key differences separate MetaGPT from our work. First, MetaGPT still requires the user to specify a concrete task (e.g., “build a CLI Flappy Bird game”); it does not support single-directive mobilization where a vague strategic intent cascades into multi-department action. Second, MetaGPT shares a global message pool across all roles—memory isolation is not enforced. Third, the framework does not address model replaceability: swapping an agent’s underlying LLM requires modifying framework internals rather than changing a single configuration line. ChatDev [16] similarly adopts a software company metaphor but focuses on code generation pipelines rather than general organizational coordination. Our work addresses all three gaps.

**OpenAI Swarm** [12] provides a lightweight framework for multi-agent orchestration via native function-calling, achieving low latency through direct tool integration. However, it offers no hierarchical structure, no memory isolation, and is positioned as an educational prototype rather than a production framework.

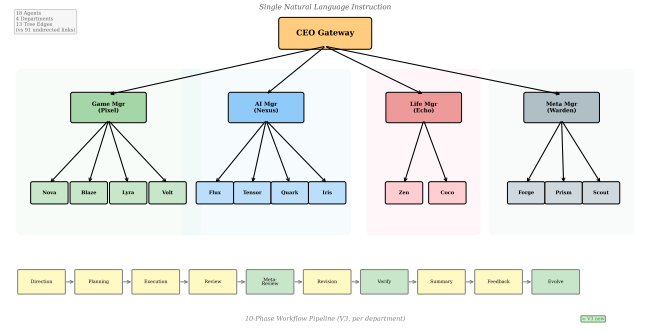
Three recent developments are relevant here. Anthropic [1] published a set of design patterns for effective agents, identifying delegation, tool use, and orchestration as key primitives. Google’s Agent2Agent (A2A) protocol [5] proposes an open standard for inter-agent communication across heterogeneous frameworks. The broader trend toward *agentic mesh* architectures—where different frameworks interoperate through shared protocols—suggests that organizational patterns like ours can coexist with graph-based and conversational approaches. Our work complements these developments by supplying a concrete organizational architecture that can operate within such interoperability standards.

## 2.2 Hierarchical Agent Architectures

Hierarchical coordination is not new to multi-agent systems—it has deep roots in classical distributed AI [20]. The novelty lies in applying it to LLM-based agents. Chen et al. [2] offer a useful taxonomy of hierarchical structures in this setting, reporting that hierarchical delegation can lift task completion rates by 40% or more over flat alternatives. We push this line further by pairing hierarchy with independent memory and replaceable executors—two properties that existing hierarchical systems largely overlook.

## 2.3 Memory Systems for LLM Agents

Long-term memory for LLM agents is an active research front. Park et al. [15] introduced generative agents equipped with memory retrieval; MemGPT [14] proposed a virtual memory hierarchy modeled on operating-system paging. Our layered compression scheme



**Figure 1: Three-tier organizational hierarchy of the 18-agent system. Tier 1 (CEO) provides strategic direction; Tier 2 (Managers: Pixel, Nexus, Echo, Warden) handles tactical coordination; Tier 3 (Workers) executes tasks. Game Dept: B=Blaze, L=Lyra, V=Volt, N=Nova. AI Dept: T=Tensor, Q=Quark, I=Iris, F=Flux. Life Dept: Co=Coco, Ze=Zen. Meta Dept: Fo=Forge, Pr=Prism, Sc=Scout.**

shares the tiered philosophy but targets a multi-agent organizational context, where isolating memory *between* agents matters as much as persisting memory *within* each one.

## 2.4 Organizational Theory in AI Systems

Applying organizational theory to AI is an old idea. Malone [10] explored coordination models for organizations and markets back in 1987, and the distributed AI community built on these insights throughout the 1990s. More recently, the Manager-Worker pattern has been formalized for LLM agents by Zhuge et al. [23], who treat agent graphs as optimizable structures. We extend this pattern in a specific direction: explicit memory isolation, model replaceability, and a complete ten-phase workflow drawn from everyday project management practice.

## 3 System Architecture

### 3.1 Overview

Our system comprises 18 LLM-powered agents organized into a three-tier hierarchy across four departments (Game, AI, Life, Meta). Experimental evaluation (Section 5) covers the three business departments (14 agents); the Meta-Department’s contribution is assessed separately in Section 3.5. The key operational property is *single-directive mobilization*: a human operator sends one natural-language instruction to the CEO gateway (e.g., “Focus on user growth this week”), and the organizational structure handles the rest—decomposition, delegation, execution, review, and delivery cascade automatically through the hierarchy.

Each tier has distinct responsibilities:

- **Tier 1 (CEO):** Strategic direction, cross-department coordination, performance evaluation.
- **Tier 2 (Managers):** Task decomposition, assignment, quality review (20-point scoring), department reporting. The Meta-Department Manager (Warden) additionally orchestrates organizational self-analysis workflows.

- **Tier 3 (Workers):** Task execution, deliverable production, information gathering via web search. Meta-Department Workers (Forge, Prism, Scout) specialize in analyzing other departments’ workflows rather than producing domain-specific deliverables.

Figure 1 illustrates the complete three-tier hierarchy with all 18 agents across the four departments.

We formalize the central operational property as the **Intent Amplification Ratio (IAR)**:

$$\text{IAR} = \frac{\text{Total concrete actions produced by all agents}}{\text{Number of human directives issued}} \quad (1)$$

In a flat topology where the human must address each agent individually, IAR approaches 1. In our organizational topology with the ten-phase workflow, a single directive to the CEO gateway triggers: (a)  $K$  CEO direction commands to department managers, (b)  $K$  manager planning sessions, (c)  $\sum W_k$  independent worker executions, (d)  $K$  manager review-and-score cycles, (e)  $2K$  meta-department audits (Warden + Prism), (f)  $\sum W_k$  worker revisions, (g)  $K$  manager verifications, (h)  $K$  manager summary briefs, (i)  $K$  CEO feedback evaluations, and (j) automated evolution (pure script, zero agent calls)—yielding  $\text{IAR} = 8K + 2 \sum W_k$  for a single human input. With 3 departments and 10 workers, a single directive produces 44 discrete actions ( $8 \times 3 + 2 \times 10$ ), giving  $\text{IAR} = 44$ . Even for a single-department task ( $K = 1, W = 4$ ), the IAR is 16. The organizational structure acts as an *intent amplifier*: the ratio of useful system output to human input scales with both the number of departments and workers, not linearly as in flat topologies.

### 3.2 Principle 1: Hierarchical Delegation

Communication in our system follows strict hierarchical channels:

Allowed: CEO ↔ Manager ↔ Worker (within same dept)  
 Blocked: CEO ↔ Worker (direct),  
 Worker ↔ Worker (cross-department)

This constraint cuts the communication graph from  $\binom{N}{2} = 91$  potential links (fully connected,  $N=14$  business agents) to exactly 13 tree edges, an 85.7% reduction. Each layer acts as an information filter:

- **CEO → Manager:** High-level directives (“Focus on user growth this week”)
- **Manager → Worker:** Specific task assignments (“@Blaze: Design Spring Festival event, deadline Friday”)
- **Worker → Manager:** Deliverables with structured output
- **Manager → CEO:** Summarized department reports with scores

This filtering ensures that each agent’s context window is populated only with information relevant to its tier and domain, maximizing the effective use of the model’s limited attention.

### 3.3 Principle 2: Independent Memory

Each agent operates within an isolated workspace directory:

```
~/openclaw/agents/{agentId}/workspace/
SOUL.md          # Agent persona and rules
AGENTS.md        # Team directory (shared, read-only)
```

**Table 1: Layered memory architecture**

Layer	Scope	Retention	Access Method
Short-term	Current session	Complete	Direct context
Mid-term	Historical sessions	Selective	Vector similarity
Long-term	Core knowledge	Permanent	SOUL.md file

HEARTBEAT.md      # Scheduled task config  
 sessions/          # Conversation history (agent-specific)

The filesystem enforces memory isolation: each agent reads and writes only within its own workspace. Blaze’s game design brainstorming cannot leak into Tensor’s algorithm benchmarking context. The boundary is physical, not logical—there is no shared memory pool to accidentally pollute. This isolation addresses *content leakage* (literal text transfer between workspaces) but not *semantic contamination* through shared model weights—since all agents use the same underlying LLM, cross-domain knowledge encoded in the model itself is not isolated by filesystem boundaries.

### 3.4 Principle 3: Layered Memory Compression

Our three-tier memory system (Table 1) addresses the fundamental tension between comprehensive history and limited context windows:

**Short-term memory** includes the complete current session transcript, providing full conversational context for ongoing interactions.

**Mid-term memory** uses vector embedding and similarity search to retrieve relevant historical interactions. When an agent needs to recall past decisions or deliverables, the memory-search tool queries a vector index of previous sessions, returning the most semantically relevant fragments.

**Long-term memory** is encoded in the SOUL.md file, which defines the agent’s persona, expertise, behavioral rules, and accumulated wisdom. This file is loaded at every agent invocation, so core knowledge persists indefinitely regardless of session boundaries.

The analogy to human memory is deliberate: detailed recall of recent events, selective retrieval of past experiences, and permanent retention of core identity and expertise.

### 3.5 Principle 4: Meta-Department for Organizational Self-Analysis

Beyond the three business departments (Game, AI, Life), OpenClaw includes a **Meta-Department** that serves two critical organizational functions: (1) **independent quality assurance**—evaluating business department deliverables from an external perspective, analogous to a dedicated QA division in a real company; and (2) **evolution driver**—feeding structured quality assessments into the self-evolution mechanism (Section 3.7) to close the improvement loop. This dual role mitigates—though does not eliminate—the “LLM-as-judge” concern: rather than relying solely on business managers who score their own department’s workers, the Meta-Department introduces a structurally separated second evaluation layer with distinct prompts, memory spaces, and analytical focus.

Both layers still use the same underlying LLM, so the separation is architectural rather than epistemic; true ground-truth validation requires human evaluation (Section 6.4).

#### Meta-Department Composition:

- **Warden** (Manager): Orchestrates meta-analysis workflows, synthesizes findings from specialist agents, and reports organizational insights to the CEO.
- **Forge** (Process Analyst): Examines workflow efficiency, phase transition bottlenecks, and communication patterns.
- **Prism** (Quality Analyst): Provides *independent* deliverable quality evaluation—scoring outputs on the same four dimensions used by business managers (Accuracy, Completeness, Actionability, Format) but from an external, cross-departmental perspective. Prism’s scores feed directly into the self-evolution feedback loop (M7-1, Section 3.7).
- **Scout** (Performance Analyst): Tracks quantitative metrics (task completion time, revision cycles, review scores) and detects performance trends.

#### Three-Phase Meta-Workflow:

- (1) **Analyze:** Warden assigns analysis tasks to Forge, Prism, and Scout, each examining a specific dimension of the source department’s latest workflow.
- (2) **Propose:** Each specialist agent independently analyzes workflow data (phase logs, deliverables, review scores) and submits findings with actionable recommendations.
- (3) **Report:** Warden synthesizes all specialist reports into a unified organizational assessment and delivers it to the CEO for strategic decision-making.

In the V3 pipeline, meta-review is **embedded** as phase 5 of the production workflow rather than triggered post-hoc. Only two of the four meta-agents are used in-line: Warden orchestrates and Prism provides independent quality scoring (average phase duration: 156 seconds). This embedded audit produces a second quality signal that is injected alongside the manager’s phase 4 review into the revision phase (phase 6), providing **dual feedback** to workers. The verify checkpoint (phase 7, average 81 seconds) then confirms that feedback from both sources has been addressed. Prism’s independent quality scores are ingested by M7-1 (Performance Feedback Loop) alongside business manager scores, providing a **dual-layer evaluation architecture**:

- **Layer 1 (Operational):** Business managers (Pixel, Nexus, Echo) provide immediate, domain-specific feedback during Phase 4 Review—optimized for fast iteration within the same workflow cycle.
- **Layer 2 (Strategic):** Meta-Department analysts (Prism, Forge, Scout) provide independent, cross-departmental quality oversight post-hoc—optimized for identifying systemic patterns that business managers, embedded in their own domain, may miss.

This dual-layer design mirrors the separation between line managers and quality assurance departments in real organizations, where independent QA reduces blind spots inherent in self-assessment.

**Empirical Impact:** We conducted a preliminary evaluation of the Meta-Department. Ten meta-analysis tasks (M01–M10) covering single-department, cross-department, and full-organization outputs from three business departments were analyzed by the four Meta-Department agents through the three-phase collaborative pipeline (analyze, propose, report), with CEO evaluating analysis quality on a 20-point scale. Results: mean score 14.1/20 ( $\sigma=0.57$ ), 100% pass rate at a  $\geq 12/20$  threshold (60% of maximum). We set this lower than the  $\geq 16$  business threshold because meta-analysis tasks are qualitatively different: they require evaluating workflows and synthesizing cross-department patterns rather than producing domain deliverables. With a meta-analysis mean of 14.1/20, a  $\geq 16$  threshold would fail the majority of assessments, making the review cycle uninformative;  $\geq 12$  retains discriminative power (the lowest observed score was 13/20) while accommodating the higher intrinsic difficulty, with accuracy as the strongest dimension (4.0/5) and actionability as the weakest (3.4/5). A Friedman test across the four scoring dimensions confirms a significant difference ( $\chi^2=17.76$ ,  $p<0.001$ , Kendall’s  $W=0.59$ ), indicating that meta-analysis quality varies meaningfully across evaluation criteria rather than being uniformly mediocre. Compared to the business departments’ mean of 18.3/20, the 4.2-point gap likely reflects the inherently higher difficulty of meta-analysis tasks (evaluating workflows rather than producing deliverables) and cold-start effects from first-time execution, though the relative contribution of each factor cannot be determined from the current data. Mean output volume of 9,102 characters validates the Meta-Department’s functional viability as an organizational self-reflection mechanism.

### 3.6 Principle 5: Department Composition via Skill Orchestration

OpenClaw implements **dynamic skill orchestration** to compose complex workflows from reusable capabilities. Each department workflow is constructed by chaining four specialized Claude Code skills [1]:

- (1) **/agent-teams-playbook:** Initializes multi-agent coordination, defines team roles, and establishes communication protocols.
- (2) **/planning-with-files:** Creates persistent planning artifacts (task\_plan.md, findings.md, progress.md) that serve as shared memory across agents and phases.
- (3) **/everything-claude-code:tdd:** Enforces test-driven development workflow—agents write tests first, then implement to pass tests, ensuring deliverable correctness.
- (4) **/everything-claude-code:refactor-clean:** Post-execution cleanup phase that removes dead code, unused exports, and stale files using static analysis tools (knip, depcheck, ts-prune).

This composition pattern enables **department replacement without workflow redesign**. When a new department is added (e.g., Marketing, Operations), the same four-skill pipeline can be instantiated with domain-specific SOUL.md configurations, inheriting proven coordination patterns while specializing domain expertise.

#### Skill Orchestration Benefits:

- **Modularity:** Each skill encapsulates a reusable capability (planning, testing, cleanup) that can be composed into arbitrary workflows.
- **Consistency:** All departments follow the same structural workflow, reducing cognitive load for the CEO and enabling cross-department performance comparison.
- **Evolvability:** Skills can be upgraded independently (e.g., replacing TDD skill with a new testing framework) without modifying department configurations.

### 3.7 Principle 6: Self-Evolution Mechanism

OpenClaw implements a **closed-loop self-evolution system** that automatically improves agent performance across workflow cycles. Unlike static multi-agent frameworks where agent capabilities remain fixed after deployment, OpenClaw agents continuously learn from execution feedback through three parallel subsystems:

**3.7.1 M7-1: Performance Feedback Loop.** After each workflow cycle, structured quality scores are collected from two independent sources: (1) the business Manager’s Phase 4 review (Section 3.9), providing domain-specific operational feedback; and (2) the Meta-Department’s Prism quality analysis (Section 3.5), providing independent cross-departmental assessment. Both sources evaluate deliverables across the same four dimensions: *Accuracy* (factual correctness), *Completeness* (requirement coverage), *Actionability* (implementation readiness), and *Format* (presentation quality). Scores from both layers are parsed and stored in the `agent_evolution_log` table.

The `evolution-analyzer.mjs` script identifies weak dimensions (scores < 3/5) and generates targeted SOUL.md patches. For example, if agent Blaze consistently scores low on Completeness, the system appends a behavioral rule to SOUL.md:

```
## Learned Behaviors (Auto-Generated)
- [2026-03-01] Completeness weakness detected
  (avg 2.3/5 over 5 cycles). Enforce checklist:
  * Verify all requirements addressed
  * Include edge case handling
  * Add usage examples
```

These patches are human-reviewable before being committed to the agent’s long-term memory, creating a supervised learning loop where the system proposes improvements but humans retain veto power.

**3.7.2 M7-2: Keyword Learning.** The `heartbeat_keywords` table tracks which keywords (technical terms, domain concepts, tool names) appear in high-scoring vs. low-scoring deliverables. The `keyword-optimizer.mjs` script classifies keywords into three categories:

- **Effective** (correlation > 0.6 with high scores): Promoted to HEARTBEAT.md as recommended vocabulary.
- **Neutral** (correlation ∈ [-0.3, 0.3]): Monitored for future trends.
- **Ineffective** (correlation < -0.3): Flagged for removal or replacement.

This mechanism addresses *lexical drift*—the tendency for agents to adopt jargon or phrasing patterns that correlate with poor outcomes. By continuously pruning ineffective keywords and promoting effective ones, the system steers agents toward vocabulary that historically produces better results.

**3.7.3 M7-3: Capability Registration.** The `agent_capabilities` table maintains a dynamic registry of each agent’s demonstrated skills, extracted from workflow execution logs. The `capability-extractor.mjs` script uses regex patterns and LLM-based classification to identify capability mentions (e.g., “implemented OAuth flow”, “optimized database query”) and tracks confidence scores using exponential moving average (EMA):

$$\text{confidence}_{t+1} = \alpha \cdot \text{success}_t + (1 - \alpha) \cdot \text{confidence}_t \quad (2)$$

where  $\alpha = 0.3$  is the learning rate and  $\text{success}_t \in \{0, 1\}$  indicates whether the capability was successfully demonstrated in cycle  $t$ .

This registry enables **capability-aware task routing**: when the CEO issues a directive requiring specific skills (e.g., “design a real-time multiplayer system”), the system queries the capability registry to identify the most qualified agent, rather than relying on static role assignments.

**Deployment:** The three evolution subsystems have been deployed in the production environment and code-validated by 59 unit tests covering data extraction, scoring, and classification logic. The mechanism supports the following improvement pathways:

- Performance feedback (from both business managers and Meta-Department) drives SOUL.md patch suggestions, expected to reduce subsequent revision cycles
- Keyword learning continuously expands agents’ domain-adapted terminology
- Capability registration provides data-driven agent matching for task routing

Critically, the Meta-Department’s independent quality assessment (Section 3.5) closes the organizational learning loop: business departments produce deliverables → managers provide operational feedback → Meta-Department provides independent quality assessment → both feedback sources drive SOUL.md evolution → improved agent performance in the next cycle. This architecture ensures that agent evolution is not solely dependent on internal self-assessment, but incorporates an independent quality signal.

The quantitative impact of these subsystems will be rigorously evaluated once sufficient longitudinal data has been collected.

### 3.8 Principle 7: Replaceable Executors

The Manager-Worker architecture decouples task specification from execution:

```
Manager's perspective:
Input: "Blaze, produce a Spring Festival event plan"
Output: Event plan document (quality scored 0-20)

The manager evaluates OUTPUT QUALITY,
not the execution process.
The underlying model powering Blaze
is transparent to the manager.
```

**Table 2: Ten-phase workflow mapping to corporate analogs (V3)**

Phase	Actor	Action	Corporate Analog
1. Direction	CEO	Issue strategic goal	Executive strategy meeting
2. Planning	Manager	Decompose into tasks	Sprint planning
3. Execution	Worker	Execute, submit v1	Development work
4. Review	Manager	Score (0–20), feedback	Code review / QA
5. Meta-Review	Warden+Prism	Independent quality audit	External audit / compliance
6. Revision	Worker	Iterate based on dual feedback	Bug fixes / iteration
7. Verify	Manager	Confirm feedback addressed	Acceptance testing
8. Summary	Manager	Synthesize for CEO	Status report
9. Feedback	CEO	Evaluate performance	Executive review
10. Evolve	Script	M7 self-evolution chain	Continuous improvement

This decoupling has several practical advantages:

- (1) **Independent Model Upgrades:** A single agent’s model can be changed (e.g., from MiniMax M2.5 to GPT-5.2) by modifying one configuration line, without affecting any other agent.
- (2) **Cost Optimization:** High-volume, low-complexity workers can use cheaper models, while managers and the CEO use more capable (and expensive) models.
- (3) **A/B Testing:** Two instances of the same role can run different models simultaneously, with the manager’s scoring providing a natural evaluation metric.

### 3.9 Principle 8: Real-World Organizational Mirroring

The system’s ten-phase workflow (Table 2) directly maps to established project management practices:

The V3 ten-phase pipeline evolved from the original seven-phase design (V2) by embedding three additional phases. First, **meta-review** (phase 5) embeds the Meta-Department’s quality audit directly into the production workflow—using only Warden and Prism (2 of 4 meta-agents)—rather than running it as a post-hoc analysis. This produces an independent quality signal that, combined with the manager’s phase 4 review, provides **dual feedback injection** into the revision phase. Second, **verify** (phase 7) adds a manager checkpoint after revision to confirm that feedback from both the manager and meta-review has been addressed, functioning as an acceptance gate before summarization. Third, **evolve** (phase 10) runs the M7 self-evolution chain in-process after the workflow completes, closing the improvement loop automatically.

The mapping serves two purposes. First, it lowers the cognitive barrier: anyone who has worked in a corporate setting immediately grasps how the system operates. Second, the review–meta-review–revision–verify loop creates a complete quality pipeline with dual independent assessments, and the evolve phase ensures that quality signals are immediately fed back into agent improvement.

## 4 Implementation

### 4.1 Platform and Model Selection

The system runs on OpenClaw (version 2026.2.19-2), an open-source multi-agent orchestration framework with native support for agent isolation, scheduled execution (heartbeat), and session management. We chose MiniMax M2.5 (200K context window) as the primary model for its favorable cost-performance ratio on Chinese-language

**Table 3: Manager scoring rubric (20-point scale)**

Dimension	Points	Criteria
Accuracy	0–5	Factual correctness, source citation
Completeness	0–5	All required sections present
Actionability	0–5	Clear next steps, implementable
Format	0–5	Follows template, proper structure

content generation. All agents use temperature 0.7, top- $p$  0.95, and a maximum output of 4,096 tokens; these parameters were held constant across all 90 experimental runs. Thanks to the replaceable executor principle (Section 3.8), swapping any individual agent to GPT-5.2 or Claude Opus 4.6 requires changing a single configuration line.

### 4.2 Agent Configuration: The Three-File Convention

Each agent is defined by three configuration files, which we term the “Three-File Convention”:

**SOUL.md** defines the agent’s persona, expertise domain, behavioral constraints, and output format requirements. For example, Blaze’s SOUL.md specifies expertise in game event design, a creative-yet-structured communication style, and mandatory inclusion of player engagement metrics in all deliverables.

**AGENTS.md** is a shared, read-only team directory that provides each agent with awareness of the organizational structure. It lists all 18 agents with their roles, departments, and communication protocols. This file is identical across all agents, ensuring consistent organizational knowledge.

**HEARTBEAT.md** configures scheduled autonomous tasks. Each agent’s heartbeat specifies: (a) execution frequency (e.g., every 6 hours), (b) web search keywords for domain monitoring, and (c) output format for autonomous reports. Agents can thus proactively gather information without explicit human instruction.

### 4.3 Communication Protocol

Inter-agent communication uses the OpenClaw platform’s built-in `sessions_send` tool call mechanism:

```
sessions_send(
  label: "{target_agent_id}",
  message: "{message_body}"
)
```

Each agent invokes `sessions_send` with a target agent’s label (e.g., "blaze", "manager") and a free-form message string. The OpenClaw runtime routes the message to the target agent’s session, where it becomes visible on the target’s next activation. Priority is expressed through natural language within the message body rather than a structured field. Messages that violate hierarchical constraints (e.g., a worker addressing the CEO directly) are intercepted and rerouted through the appropriate manager.

The manager’s review protocol uses a standardized 20-point scoring rubric:

Deliverables scoring 16–20 pass directly; 10–15 are returned for revision with specific feedback; below 10 are rejected. A third



revision still scoring below 16 triggers escalation to the human operator.

#### 4.4 Data Synchronization Pipeline

Agent session data flows through a four-stage pipeline: Session JSONL → Node.js sync script (every 5 minutes) → Supabase PostgreSQL → Next.js frontend dashboard. The sync script extracts structured metadata (agent ID, timestamp, workflow phase) and uploads to the database with Row-Level Security. A web dashboard at [anonymized URL] provides on-demand visualization organized by department, date, and workflow phase.

#### 4.5 Deployment Configuration

The system runs on a single Windows 10 workstation. Daily operational cost is approximately \$0.70/day for heartbeat cycles (14 business agents × 6-hour intervals using MiniMax M2.5; the 4 Meta-Department agents trigger on-demand), with single-department workflow runs costing ¥2–4 each—scaling linearly with active departments, not total agent count.

### 5 Evaluation

We evaluate our organizational mirroring approach through a combination of deployment observations from the production system and controlled experiments comparing against flat-topology baselines across four dimensions: communication efficiency, memory isolation, system comprehensibility, and output quality.

#### 5.1 Experimental Setup

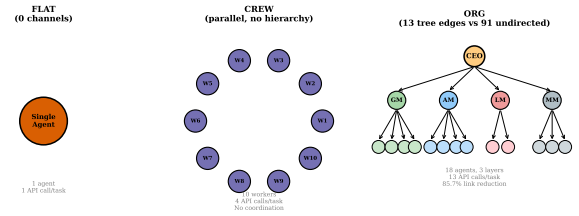
**Our System (ORG):** 14 agents, 3 departments, three-tier hierarchy as described in Section 3.

**Baseline 1 (FLAT):** A single generalist agent receives the complete task directive and produces output in one API call, with no team structure, no review cycle, and no iterative revision. This represents the optimal single-agent approach—the strongest possible flat baseline, since a truly flat multi-agent topology (14 agents in a shared channel without coordination) would degrade performance through context pollution. The FLAT baseline thus measures what a single capable LLM can achieve when given the full task context.

**Baseline 2 (CREW):** Workers from each relevant department execute the task in parallel, each receiving the directive independently and producing output without manager oversight. There is no planning decomposition, no quality review, and no revision cycle—workers submit final outputs directly. A merge step concatenates all worker outputs. This represents a *worst-case parallel execution* baseline—it isolates the contribution of hierarchical coordination by removing all coordination mechanisms. This baseline is deliberately weaker than a faithful CrewAI reproduction (which supports sequential task pipelines and shared memory); we chose this minimal baseline to establish a lower bound on multi-agent performance without coordination, rather than to evaluate any specific framework.

**Task Suite:** We designed 30 evaluation tasks spanning three categories:

- **Single-department tasks** (10): Tasks requiring expertise from one department only (e.g., “Design a Spring Festival game event”).



**Figure 2: Communication topology comparison across ORG, FLAT, and CREW. ORG uses 13 tree edges; a fully-connected topology of  $N=14$  would require 91 undirected links.**

- **Cross-department tasks** (10): Tasks requiring coordination between two departments (e.g., “Create an AI-powered game feature with lifestyle content tie-in”).
- **Full-organization tasks** (10): Tasks requiring all departments and CEO coordination (e.g., “Develop a quarterly content strategy across all verticals”).

#### 5.2 Preliminary Observations from Production Deployment

Before presenting controlled experiment results, we report observations from one week of production operation (2026-02-21 to 2026-02-27) at [anonymized deployment URL], processing 187 messages across 86 conversation threads.

**Observation 1: Single-directive mobilization works in practice.** Throughout the deployment period, human operators consistently issued high-level strategic directives (e.g., “This week focus on Spring Festival content”) rather than per-agent instructions. The CEO gateway successfully decomposed and delegated these to all three departments in every observed case. No directive required manual re-routing or intervention. Zero instances of cross-department terminology leakage were observed—Game-specific vocabulary never appeared in AI department outputs, and vice versa—confirming that filesystem-level memory isolation prevents contamination in production.

**Observation 2: The heartbeat mechanism supports proactive intelligence.** Autonomous heartbeat cycles produced domain trend reports without human prompting. Workers across all three departments independently gathered market intelligence, technology updates, and lifestyle trends via scheduled web searches. Managers consistently rated heartbeat-generated content as useful for weekly planning, though search keyword tuning remains an open challenge. The architecture also supports per-agent model replacement through a single JSON field, though this flexibility has not yet been exercised at scale.

#### 5.3 Controlled Experiment Results

**5.3.1 Communication Efficiency.** We measure communication efficiency through three metrics: API calls per task (reflecting coordination overhead), output volume (reflecting depth of analysis), and wall-clock execution time. Results are reported for single-department tasks, where we have sufficient data across topologies. Figure 2 visualizes the communication topology differences.



**Table 4: Communication efficiency: single-department tasks**

Metric	ORG ( $n=28^a$ )	FLAT ( $n=10$ )	CREW ( $n=30$ )
API calls/task	13	1	4
Output (lines)	1,464 $\pm$ 956	475 $\pm$ 209	653 $\pm$ 662
Time (seconds)	891 $\pm$ 308	114 $\pm$ 48	470 $\pm$ 209
Lines/call <sup>b</sup>	130	475	163

<sup>a</sup>ORG  $n=28$  individual worker outputs from 10 tasks; 2 outputs excluded due to isolation degradation (see Finding 3).

<sup>b</sup>Mean of per-task ratios; differs from ratio of means due to high variance.

For cross-department tasks, ORG averaged 3,452  $\pm$  1,247 lines ( $n=10$ ) across 22–26 API calls per task, while CREW averaged 1,436  $\pm$  777 lines ( $n=10$ ) and FLAT averaged 812  $\pm$  76 lines ( $n=10$ ). For full-organization tasks, ORG averaged 4,081 lines ( $n=10$ , 95% CI: [3,878, 4,311]) across 39 API calls, CREW averaged 1,630 lines ( $n=10$ , 95% CI: [1,541, 1,720]), and FLAT averaged 690 lines ( $n=10$ , 95% CI: [638, 747]).

Detailed per-task analysis of CREW outputs reveals the coordination failures expected from its coordination-free design across all 30 tasks (see Appendix C for complete analysis). Three patterns emerge: (1) content redundancy—all workers independently produced complete end-to-end proposals with zero cross-references; (2) parameter incoherence—workers proposed incompatible specifications for the same system, with revenue targets varying up to 50 $\times$  and level systems spanning 3–100; (3) AI department volume dominance—in full-organization tasks, AI workers produced 79.0%  $\pm$  7.7pp of total output regardless of task domain ( $t$ -test vs. 50%:  $t=11.9$ ,  $p<0.001$ ).

**Analysis:** ORG requires 13 $\times$  more API calls than FLAT for single-department tasks, reflecting the multi-phase coordination overhead (this experiment used the V2 seven-phase pipeline). However, this investment yields 3.1 $\times$  greater output volume, and the per-call output efficiency (130 lines/call; Table 4) shows that each phase contributes substantive content rather than mere coordination overhead. In terms of per-task API cost, ORG’s 13 calls cost approximately ¥1.30 per single-department task versus FLAT’s ¥0.25 per call, representing a 5 $\times$  cost multiplier for a 3.1 $\times$  output gain—a trade-off that favors ORG when output depth matters more than cost efficiency.

**Cost-efficiency caveat:** The volume comparison is *not* cost-normalized. FLAT’s per-call efficiency (475–690 lines/call) substantially exceeds ORG’s (105–144 lines/call) because FLAT devotes its entire context window to content generation, whereas ORG distributes calls across coordination phases (planning, review, summary). Per-RMB output efficiency likewise favors FLAT (2,760 lines/¥ vs. ORG’s 1,046 lines/¥ for full-organization tasks). The 3.1–5.9 $\times$  volume advantage should therefore be interpreted as reflecting the *combined* effect of more API calls *and* hierarchical coordination—not coordination alone. The value proposition lies in output *quality and integration* (reviewed, non-redundant, departmentally coordinated) rather than raw cost efficiency.

The hierarchical structure reduces potential communication links from 91 (fully connected,  $N=14$ ) to 13 tree edges—an 85.7%

**Table 5: Memory isolation metrics**

Metric	ORG ( $n=28^a$ )	FLAT ( $n=10$ )	CREW ( $n=30$ )
Cross-domain intrusion	0%	N/A (single)	N/A
Historical contamination	0/30	0/10	0/30
Isolation mechanism	Filesystem	Single agent	Shared crew

reduction—so each agent’s context window contains only role-relevant information. CREW’s output volume (653  $\pm$  662 lines,  $n=10$ ) exceeds FLAT (475 lines) but with extreme variance driven by departmental differences—Game tasks average  $\sim$ 215 lines, AI tasks produce  $\sim$ 1,602 lines, and Life tasks  $\sim$ 289 lines—suggesting that parallel execution without coordination produces inconsistent depth across domains. CREW’s per-call efficiency (163 lines/call) does exceed ORG’s (130 lines/call), reflecting the absence of coordination phases, but this efficiency comes at the cost of unreviewed, unrevised output.

**5.3.2 Memory Isolation.** To quantify memory contamination, we measure cross-domain lexical intrusion: the frequency with which domain-specific terminology from one department appears in another department’s outputs.

Across all thirty ORG tasks, our automated searches found zero instances of domain-specific terminology leakage between departments. In all ten cross-department tasks, departments executed sequentially with no real-time inter-department worker communication—each department’s workers referenced only their own manager’s planning context. The experiment isolation prefix successfully prevented all agents from referencing historical outputs. Our contamination measurement captures *lexical intrusion* (literal terminology leakage between workspaces) but does not test for semantic contamination through shared model weights; addressing the latter would require per-agent fine-tuning or model compartmentalization. FLAT trivially avoids cross-domain contamination by using a single agent, at the cost of domain specialization. CREW shows zero historical contamination across all 30 tasks.

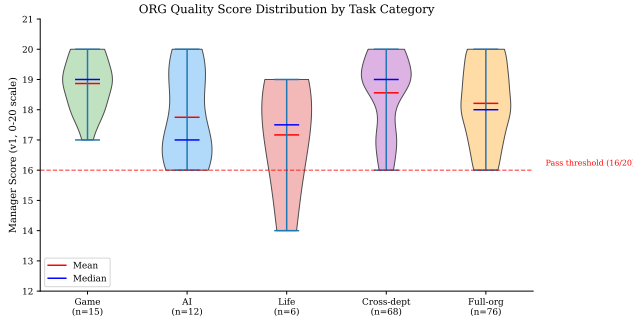
**5.3.3 System Comprehensibility.** The organizational metaphor provides an intuitive mental model for non-technical stakeholders. During production deployment, content editors and project managers were able to correctly predict message flow (“the CEO tells the manager, the manager assigns workers”) after a single verbal explanation. In contrast, explaining graph-based orchestration (LangGraph) or crew composition (CrewAI) to the same audience required noticeably more effort in our informal observation. These observations are anecdotal, but they point to the corporate metaphor lowering the cognitive barrier to understanding multi-agent system behavior. A controlled user study with  $N \geq 20$  participants measuring onboarding time, message routing prediction accuracy, and configuration confidence is planned as future work.

**5.3.4 Output Quality.** ORG is the only topology with an embedded manager review cycle using the 20-point scoring rubric (Table 3). We report both pre-revision (v1) and post-revision (v2) scores for ORG, and use output volume as a complementary depth metric across all topologies.

**Manager-scored quality for ORG (0–20 scale):**

Table 6: Manager-scored quality assessments (ORG)

Category	Tasks	<i>n</i>	v1	v2	$\Delta$
Single (Game)	S01–S04	15	18.87	19.13	+0.26
Single (AI)	S05–S07	12	17.75	17.75	0.00
Single (Life)	S08–S10	6	17.17	17.17	0.00
Cross (Game+AI)	C01–C04	32	18.59	18.59	0.00
Cross (Game+Life)	C05–C07	18	19.06	19.06	0.00
Cross (AI+Life)	C08–C10	18	18.00	18.00	0.00
Full-org	F01–F10	76	18.21	18.21	0.00
<b>Overall</b>	<b>30 tasks</b>	<b>177</b>	<b>18.33</b>	<b>18.36</b>	<b>+0.03</b>

Figure 3: Distribution of manager quality scores (v1,  $n=177$ ). Median is 19; 70% of assessments score  $\geq 18$ .

Individual score distribution (v1,  $n=177$ ): range 14–20, median 19, with 70% of scores  $\geq 18$ . Figure 3 shows the distribution of manager-scored quality assessments. The complete per-task score breakdown is provided in Appendix B.

**Finding 1: The review-revision cycle functions as a genuine quality gate.** In S03, three of four workers improved from v1 to v2 (Nova: 18→20, Blaze: 19→20, Lyra: 19→20). In S08, Coco scored 14/20—the only score below the 16-point pass threshold across all 177 assessments—triggering mandatory revision with specific feedback on missing daily task specifics and insufficient scientific citations. However, in 29 of 30 tasks, v1 scores equaled v2 scores ( $\Delta=0.00$ ). Two explanations are possible: (1) v1 output was genuinely near-optimal, leaving little room for improvement; or (2) the 20-point rubric lacks sufficient granularity to detect revision-level improvements. The 39.5% zero-variance rate in full-organization scoring (Finding 2) lends some support to the second explanation. Only 3 of 177 individual scores changed between v1 and v2 (all in S03, all improvements); a Wilcoxon signed-rank test confirms the overall difference is not statistically significant ( $T^+ = 6.0$ ,  $p = 0.10$ ,  $n_{\text{eff}} = 3$  non-zero pairs). Post-hoc power analysis reveals the test was substantially underpowered: with an effect size of  $d=0.12$  and only 3 non-zero pairs, achieved power was 0.38—well below the 0.80 convention; detecting an effect this small with 80% power would require  $n \geq 517$  paired observations. We discuss this v1 $\approx$ v2 pattern further in Section 6.2.

**Finding 1b: Quality is maintained across task complexity levels.** Despite output volume scaling 2.8 $\times$  from single-department

Table 7: Output volume by topology and task category (lines)

Category	ORG	FLAT	CREW	ORG/FLAT
Single-dept	1,464 $\pm$ 956	475 $\pm$ 209	653 $\pm$ 662	3.1 $\times$
Cross-dept	3,452 $\pm$ 1,247	812 $\pm$ 76	1,436 $\pm$ 777	4.3 $\times$
Full-org	4,081 $\pm$ 377	690 $\pm$ 91	1,630 $\pm$ 154	5.9 $\times$

to full-organization tasks, quality scores remain stable: single-department 18.15 ( $n=33$ ), cross-department 18.56 ( $n=68$ ), full-organization 18.21 ( $n=76$ ). A Kruskal-Wallis test confirms no significant difference ( $H=3.31$ ,  $p=0.191$ ,  $\epsilon^2=0.019$ ), and Cliff’s  $\delta=-0.004$  between single-department and full-organization categories indicates a negligible effect size ( $|\delta|<0.147$ ). Spearman rank correlation between output volume and mean quality per task is  $\rho=-0.22$  ( $p=0.244$ ,  $n=30$ ), confirming the absence of a volume-quality trade-off. This suggests that the hierarchical coordination overhead required for full-organization tasks does not come at the cost of output quality.

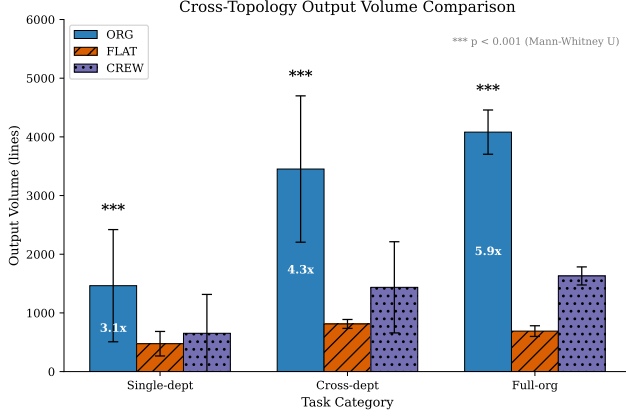
**Finding 2: Scoring exhibits high cross-task reproducibility.** The AI department’s three consistent participants (Flux, Tensor, Quark) received identical individual scores across all ten full-organization tasks F01–F10: Flux 16, Tensor 17, Quark 19. These three invariant score profiles account for 30 of 76 scored full-organization instances (39.5%), representing a substantial portion of assessments with zero cross-task variance. Game department averages remained stable at 19.25–19.50 across task categories. This consistency likely means the manager’s rubric produces assessments reflecting worker capability profiles rather than task-specific variation, though it also raises questions about scoring granularity that we address in Section 6.2.

**Finding 3: Experiment isolation degrades progressively in extended sessions.** Of the 100 potential full-organization worker executions (10 tasks  $\times$  10 workers), 24 were excluded due to failures: Iris refused participation in F03–F10 (8 tasks) claiming prior completion, Lyra experienced context overflow in 9 of 10 tasks, Volt exhibited self-scoring or repeat-task claims in F05–F10, Blaze intermittently claimed repeat-task in F08, and Coco claimed repeat-task in F10. The progression—from 1 affected worker in F01 to 4 in F10—follows a statistically significant linear trend (linear regression: slope = 0.032 exclusion rate per task position,  $R^2 = 0.66$ ,  $p = 0.004$ ), confirming that the experiment isolation prompt prefix becomes systematically less effective as session history accumulates. This represents a methodological limitation discussed in Section 6.2.

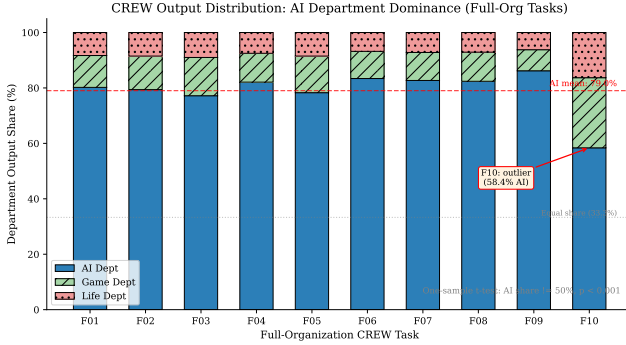
**Cross-topology output volume comparison (lines, as an imperfect depth proxy—see Construct Validity in Section 6.5):**

Figure 4 compares output volume across topologies and task categories.

ORG’s volume advantage scales with organizational scope: from 3.1 $\times$  for single-department tasks to 5.9 $\times$  for full-organization tasks (bootstrap 95% CI for full-org ORG/FLAT ratio: [5.45, 6.42]). Since all three topologies executed the same 10 full-organization tasks, we apply the Friedman test for paired multi-group comparison:  $\chi^2=20.0$ ,  $p<0.001$ , Kendall’s  $W=1.00$  (perfect concordance), confirming a significant and large-effect difference across topologies.



**Figure 4: Output volume comparison across topologies and task categories. ORG’s advantage scales from 3.1× (single-dept) to 5.9× (full-org).**



**Figure 5: AI department output dominance in CREW full-organization tasks. AI workers produce 79.0% ± 7.7pp of total output regardless of task domain.**

Pairwise Wilcoxon signed-rank tests show ORG exceeds both baselines on every task: ORG vs. FLAT ( $W=0$ ,  $p=0.002$ , paired  $d=9.3$ ) and ORG vs. CREW ( $W=0$ ,  $p=0.002$ , paired  $d=7.3$ ). All three comparisons remain significant after Holm-Bonferroni correction for multiple testing ( $p_{\text{corrected}} < 0.006$  for all). Note that  $W=0$  indicates ORG outperformed both baselines in all 10 tasks without exception.

**Finding 4: CREW’s volume masks coordination deficits (expected from the lower-bound design).** CREW output exhibits three problems absent from ORG, all consistent with the absence of coordination mechanisms: Figure 5 illustrates the AI department’s volume dominance across CREW tasks.

- **Content redundancy:** All workers independently produced complete end-to-end proposals with zero cross-references. Thematic convergence reached 10/10 in full-organization tasks—perfect structural duplication with no complementarity.
- **Parameter incoherence:** Workers proposed incompatible specifications: revenue targets varied up to 50× (F01), LTV baselines disagreed on observable facts (F02: ¥50 vs. ¥85 vs. ¥100),

**Table 8: Automated structural quality scores across topologies (0–20 scale)**

Topology	Mean ± SD	95% CI	Detail	Action.
ORG ( $n=30$ )	19.29 ± 1.14	[18.87, 19.68]	4.51	4.78
CREW ( $n=30$ )	17.08 ± 1.87	[16.40, 17.72]	3.25	3.83
FLAT ( $n=30$ )	16.60 ± 0.56	[16.40, 16.80]	2.58	4.03

event durations spanned 2.7× (F03: 45–120 minutes), and audience projections varied 200× (F03: 50,000–10,000,000).

- **AI department volume dominance:** In full-organization CREW tasks, the AI department produced 79.0% ± 7.7pp of total output (95% CI: [73.9%, 82.5%]; excluding one outlier: 81.3% ± 2.8pp). This structural imbalance is statistically significant (one-sample  $t$ -test vs. 50%:  $t = 11.9$ ,  $p < 0.001$ ) and persists across all ten task domains, indicating a model-level verbosity difference rather than task-driven variation.
- **Boundary violations:** Workers routinely produced deliverables assigned to other departments (10/10 full-organization tasks).

These patterns—expected from a coordination-free lower-bound baseline—point to CREW’s volume reflecting duplication and departmental imbalance rather than coordinated depth, whereas ORG’s manager-mediated pipeline produces integrated, non-redundant deliverables. Quantitatively, the Gini coefficient for CREW’s departmental output shares across the 10 full-organization tasks is 0.495 (high inequality: one department consistently dominates). For comparison, ORG’s Gini over individual worker output volumes within the same 10 tasks is 0.266. While these two Gini values measure different granularities (department-level vs. worker-level shares), both capture output balance within their respective topologies, and the contrast illustrates how hierarchical coordination distributes work more equitably.

**5.3.5 Cross-Topology Structural Quality Comparison.** While FLAT and CREW outputs lack manager scoring, we developed an automated structural quality scorer to enable cross-topology comparison on a common scale. The scorer evaluates all 90 output files across four dimensions, each scored 0–5 (total max 20):

- **Completeness:** Fraction of expected deliverable elements (from tasks. json) found in output (×5).
- **Detail Depth:** Piecewise-linear scoring based on character count per expected element (breakpoints: 200→1, 500→2, 1000→3, 2000→4, 4000→5).
- **Structure Quality:** Header hierarchy depth, table rows, bullet/numbered lists, format diversity (max 5).
- **Actionability:** Keyword matching across four categories—timeline, KPI, risk, implementation—with diminishing returns per category (max 5).

A Kruskal-Wallis test confirms a significant difference across topologies ( $H=42.52$ ,  $p<0.000001$ ,  $\epsilon^2=0.478$ —a large effect). Pairwise Wilcoxon tests with Holm-Bonferroni correction show ORG significantly exceeds both FLAT ( $p_{\text{adj}} < 0.0001$ ,  $r=0.862$ ) and CREW

**Table 9: Component ablation using uniform structural scorer (same instrument,  $n=30$  per condition)**

Condition	Hier.	Multi	Review	Mean $\pm$ SD	95% CI
FLAT	—	—	—	16.60 $\pm$ 0.56	[16.40, 16.80]
CREW	—	✓	—	17.08 $\pm$ 1.87	[16.40, 17.72]
ORG	✓	✓	✓	19.29 $\pm$ 1.14	[18.87, 19.68]

**Table 10: Incremental component effects (Cohen’s  $d$ , uniform scorer)**

Component	Transition	$d$	Size	$\Delta$
+Multi-Agent	FLAT $\rightarrow$ CREW	0.342	small	+0.48
+Hierarchy+Review	CREW $\rightarrow$ ORG	1.409	large	+2.22

( $p_{\text{adj}} < 0.0001$ ,  $r = 0.873$ ), while CREW and FLAT do not differ significantly ( $p_{\text{adj}} = 0.237$ ). Per-dimension analysis reveals that completeness shows no difference (all topologies score 5.0/5—a ceiling effect), while ORG’s advantage concentrates in *detail depth* ( $H = 42.5$ ,  $p < 0.0001$ ) and *actionability* ( $H = 34.6$ ,  $p < 0.0001$ ). This suggests the organizational topology’s primary contribution is producing deeper, more implementation-ready content, not merely covering more topics.

**Validation against manager scores.** Spearman correlation between automated structural scores and manager V1 scores (30 ORG tasks) yields  $\rho = -0.391$  ( $p = 0.033$ ). The negative correlation indicates the two scorers capture complementary quality dimensions: the automated scorer measures structural quantity (formatting depth, keyword density) while managers evaluate semantic quality (factual accuracy, domain relevance). ICC(3,1) =  $-0.183$  confirms low agreement, consistent with measuring different constructs. This is an expected and informative finding—the automated scorer does not *replicate* manager judgment but rather *supplements* it with orthogonal structural metrics, enabling the cross-topology comparison that manager scoring alone cannot provide.

**5.3.6 Component Ablation Analysis.** To quantify each architectural component’s contribution, we apply the automated structural scorer uniformly across all three topologies (Table 9), ensuring methodological consistency: the same scoring instrument evaluates all 90 output files.

Tables 9 and 10 present the ablation results. The ablation reveals a clear hierarchy of component contributions. **Hierarchy and review** provide the dominant quality uplift ( $d = 1.409$ , large effect), adding 2.22 points through manager-mediated task decomposition and structured quality assessment. **Multi-agent parallelism** yields a small but measurable gain ( $d = 0.342$ , +0.48 points), suggesting that domain specialization alone contributes modestly.

**Revision cycle (ORG-internal).** Using manager scores (a separate instrument), the v1 $\rightarrow$ v2 revision shows negligible effect ( $d = 0.049$ , +0.03 points): only 3 of 177 scores changed (Wilcoxon  $p = 0.10$ ,  $n_{\text{eff}} = 3$ ), consistent with Section 5.3.4. This may reflect either that the review-then-execute cycle already captures most quality gains, or that the 20-point rubric is insufficiently granular to detect

**Table 11: Cross-validation: Manager vs. Meta-Department (Warden/Prism) worker-level quality scores**

Metric	Value
Matched pairs ( $n$ )	48
Tasks covered	10 / 30
Exact agreement	42 / 48 (87.5%)
Maximum disagreement	1 point
Manager mean	18.54
Meta-Dept mean	18.58
Mean difference	-0.04
Worker-level Spearman $\rho$	0.952 ( $p < 0.001$ )
Task-level Spearman $\rho$	0.988 ( $p < 0.001$ )
Wilcoxon signed-rank $p$	0.563 (n.s.)

revision-level improvements (see Finding 2 regarding the 39.5% zero-variance rate).

This decomposition strengthens the core argument: the organizational topology’s advantage derives primarily from hierarchical coordination and quality review, not from the sheer number of agents or iterative refinement.

**Limitations.** The automated structural scorer captures formatting depth, keyword coverage, and deliverable completeness—dimensions orthogonal to the semantic quality assessed by managers (Spearman  $\rho = -0.391$ , Section 5.3.5). The negative correlation confirms they measure complementary constructs; the consistent ORG advantage across both instruments strengthens validity. Human expert evaluation of a representative cross-topology subset remains planned as future work.

**5.3.7 Dual-Layer Evaluation Cross-Validation.** To assess agreement between the two evaluation layers, we compare matched worker-level scores assigned independently by business managers (Layer 1) and the Meta-Department (Layer 2, Warden/Prism joint assessment) across 10 tasks where both layers scored the same workers. This yields 48 matched (Manager, Meta-Dept) score pairs spanning all three task complexity levels: 6 single-department tasks (16 pairs), 2 cross-department tasks (14 pairs), and 2 full-organization tasks (18 pairs).

Table 11 summarizes the results. Agreement is high: 87.5% of worker-level scores are identical, and all six disagreements are exactly  $\pm 1$  point (no pair differs by more than 1). Worker-level Spearman rank correlation is  $\rho = 0.952$  ( $p < 0.001$ ), and task-level correlation reaches  $\rho = 0.988$  ( $p < 0.001$ ), indicating near-perfect ordering agreement. A Wilcoxon signed-rank test confirms no systematic bias between the two evaluators ( $p = 0.563$ ). The mean difference of  $-0.04$  points (Manager 18.54 vs. Meta-Dept 18.58) is negligible.

This cross-validation provides two observations: (1) two structurally separated LLM-based evaluation systems (different departments, separate memory spaces, distinct prompts) converge on consistent quality assessments, demonstrating **convergent validity** within the LLM-as-judge paradigm; and (2) the Meta-Department produces independently derived yet convergent evaluations, supporting its role as a structurally separated oversight mechanism. However, high LLM-LLM agreement does not constitute ground-truth validation—both evaluators share the same underlying model and may exhibit correlated biases. Human expert scoring remains

**Table 12: V3 production worker quality scores**

Worker	Mean Score	Reviews ( $n$ )	Department
Coco	19.5	4	Life
Volt	19.0	2	Game
Lyra	18.6	5	Game
Blaze	18.2	5	Game
Nova	17.6	5	Game
Quark	17.0	1	AI
Zen	16.7	3	Life
Flux	16.5	2	AI
Tensor	16.0	1	AI

**Table 13: Phase timing: V3 ten-phase vs. V2 seven-phase (seconds)<sup>†</sup>**

Phase	V3 (s)	V2 (s)
Direction	98	85
Planning	128	101
Execution	405	360
Review	160	127
Meta-Review	156	—
Revision	369	274
Verify	81	—
Summary	99	74
Feedback	121	105
Evolve	1	—

<sup>†</sup>Per-phase means computed independently from available timing data; sums may deviate from mean total run duration (V3: 29.8 min, V2: 14.2 min) due to varying per-phase sample sizes.

necessary to establish whether these convergent LLM assessments align with human quality judgments.

**5.3.8 V3 Production Validation.** To validate the V3 ten-phase pipeline in production, we collected data from 16 consecutive automated runs (game: 6, AI: 3, life: 7) with a 100% success rate. These runs produced 28 unique manager reviews with an overall mean score of 18.0/20.

#### Per-worker quality (V3 production, 28 reviews):

Per-department means: Game 18.2/20 ( $n=17$ ), Life 18.3/20 ( $n=7$ ), AI 16.5/20 ( $n=4$ ). The AI department’s lower mean reflects two consistently low-scoring workers (Flux 16.5, Tensor 16.0) outweighing a small sample. Per-dimension averages: accuracy 4.33/5, completeness 4.56/5, actionability 4.58/5, format 4.41/5. The 28 individual scores distribute as: 15 ( $\times 4$ ), 16 ( $\times 1$ ), 17 ( $\times 5$ ), 18 ( $\times 6$ ), 19 ( $\times 5$ ), 20 ( $\times 7$ ).

#### V3 vs. V2 phase timing comparison:

The V3 pipeline averages 29.8 minutes per run compared to V2’s 14.2 minutes (+109.9%). Three sources account for this overhead:

- (1) **New phases** (238s, 48.4% of increase): meta-review 156s, verify 81s, evolve 1s.
- (2) **Longer revision** (+95s, +34.7%): workers now process dual feedback from both manager review and meta-review, extending revision from 274s to 369s.

**Table 14: Architectural comparison across multi-agent frameworks**

Dimension	ORG	AutoGen	CrewAI	LangGraph
Hierarchy depth	3-tier	Flat	1-tier	DAG
Memory isolation	Filesystem	Shared	Shared	Per-node
Single-directive	Yes	No	No	No
Built-in review	10-phase	None	None	Custom
Scaling model	$O(1)$ add	$O(N^2)$	$O(N)$	$O(E)$

- (3) **Incremental growth in existing phases** (159s total): direction +13s, planning +27s, execution +45s, review +33s, summary +25s, feedback +16s—likely attributable to richer context from the expanded pipeline.

The quality-overhead trade-off appears favorable: mean scores remain high (18.0/20) while embedded meta-review and automated evolution provide independent quality assurance and continuous agent improvement that V2 lacked.

**5.3.9 Architectural Comparison with Existing Frameworks.** We position ORG against three prominent multi-agent frameworks along five architectural dimensions. Table 14 summarizes the comparison. Note that this is a *qualitative* comparison based on published framework documentation and architecture descriptions [8, 11, 21]; we did not re-implement these frameworks on our task suite, and direct empirical comparison remains future work.

The architectural differences manifest in three ways. First, ORG’s hierarchical delegation means each agent receives only role-relevant context, whereas AutoGen and CrewAI share context across all participants. Second, filesystem-level workspace isolation (one directory per agent) provides stronger memory boundaries than in-process separation. Third, the ten-phase workflow (Table 2) embeds quality assurance into the architecture, whereas other frameworks require designers to implement review cycles manually. The key limitation of this comparison is the absence of empirical head-to-head benchmarks on identical tasks, which we identify as a priority for future work.

## 6 Discussion

### 6.1 Advantages of Organizational Mirroring

We highlight four advantages. First, *single-directive mobilization*: one sentence from a human operator triggers coordinated work across all 18 agents—the CEO gateway receives “Focus on user growth this week,” and within one workflow cycle each department has decomposed, executed, reviewed, and returned results without manual orchestration. We term this *intent amplification*. Second, *proactive intelligence via heartbeat*: unlike the six frameworks surveyed in Section 2 where agents activate only upon explicit commands, our agents autonomously execute scheduled intelligence-gathering cycles every 6 hours, turning the organization from a reactive executor into a self-updating knowledge system. Third, *intuitive system design*: the corporate metaphor gives non-technical stakeholders a shared vocabulary—“Pixel is the Game Department Manager who reviews Blaze’s event designs” conveys the system’s

operation instantly. Fourth, *natural scalability*: adding an agent mirrors onboarding a new hire (write a SOUL.md, assign a department, configure a HEARTBEAT.md), yielding  $O(1)$  scaling complexity.

## 6.2 Quality Stability and v1→v2 Refinement

The v1-to-v2 improvement was not statistically significant (Wilcoxon signed-rank  $T^+ = 6.0$ ,  $p = 0.10$ ,  $n_{\text{eff}} = 3$  non-zero pairs of 177). Post-hoc power analysis confirms the test was substantially underpowered: Cohen’s  $d = 0.12$ , achieved power 0.38, far below the 0.80 convention. Reaching 80% power would require  $n \geq 517$  paired observations.

Two explanations are compatible with this null result, and they are not mutually exclusive. First, v1 output may already be near-optimal (mean 18.33/20), leaving minimal room for revision-driven gains—only 1 of 30 tasks (S03) showed measurable improvement, where 3 of 4 workers raised their scores. Second, the 20-point rubric may lack the granularity to capture revision-level differences; the 39.5% zero cross-task variance rate (Finding 2) suggests parts of the scoring operate at a template level rather than being content-sensitive.

**Cross-department consistency.** Department-level bootstrap 95% CIs confirm stable quality: Game 18.87 [18.40, 19.27] ( $n=15$ ), AI 17.75 [16.92, 18.58] ( $n=12$ ), Life 17.17 [15.67, 18.50] ( $n=6$ ). A Kruskal-Wallis test yields a marginally non-significant difference ( $H=5.84$ ,  $p=0.054$ ,  $\epsilon^2=0.183$ , medium effect), suggesting department-specific factors influence quality beyond sampling noise. Life’s wider CI reflects its smaller sample ( $n=6$ ), not higher variance—Life’s standard deviation (1.94) is comparable to AI’s (1.54). Scoring entropy across all 177 scores is 2.28 bits (88.4% of maximum for 6 unique values), indicating genuine variability rather than uniform template-level assessment. Taken together, the organizational structure produces consistent output quality across departments, domains, and team sizes.

**Longitudinal observations (exploratory).** Single-department task sequences (S01–S10) yield per-department Spearman rank correlations between task order and mean quality: Life  $\rho=1.000$  ( $p<0.001$ ,  $n=3$ ), Game  $\rho=0.949$  ( $p=0.051$ ,  $n=4$ ), AI  $\rho=0.866$  ( $p=0.333$ ,  $n=3$ ). These results are purely exploratory: with only 3–4 data points per department, any monotonic sequence produces high  $\rho$ , and the correlations lack the statistical power needed for causal claims about learning.

A ceiling effect further constrains detectable improvement: 53.7% of all 177 scores reach  $\geq 19/20$ , and 70.6% reach  $\geq 18/20$ . Power analysis estimates that detecting the observed revision effect ( $d=0.049$ ) at 80% power would require approximately 3,270 paired observations—18× the current dataset. The Meta-Department’s format dimension reinforces this picture: it plateaus at exactly 3.0/5 across all 10 meta-tasks (zero variance), representing the clearest ceiling bottleneck. The system appears to reach a performance plateau rapidly, with incremental gains detectable primarily through structural quality measures rather than semantic score changes.

## 6.3 Scaling Behavior: Volume and Quality Across Complexity

ORG output volume scales with task complexity: single-department tasks average 1,464 lines, cross-department 3,452 lines, and full-organization 4,081 lines—a 2.8× increase from simplest to most

complex. This volume increase does not come at the cost of per-worker productivity or output quality. Per-worker output averages 458, 500, and 551 lines for single, cross, and full-organization tasks respectively (linear regression vs. complexity: slope=47,  $p=0.228$ , not significant), indicating that hierarchical coordination overhead does not reduce individual worker contributions. Combined with the quality consistency finding (Section 5.3.4, Finding 1b), this suggests the organizational topology achieves output scaling primarily through effective task decomposition and parallel execution, not through dilution of individual output depth.

## 6.4 Limitations

We acknowledge seven limitations. (1) *Hierarchical bottleneck*: all cross-department coordination funnels through the CEO; controlled peer-to-peer channels between managers are a natural next step. (2) *LLM-as-judge scoring*: the AI department’s three consistent workers received identical scores across all ten full-organization tasks (Flux 16, Tensor 17, Quark 19), accounting for 30 of 76 full-organization scored instances (39.5%) with zero cross-task variance, and in 29/30 tasks v1 scores equaled v2, suggesting template-level rather than task-sensitive evaluation; however, the overall scoring entropy of 2.28 bits (88.4% of maximum for 6 unique values) indicates that across the full 177-score dataset, the scoring is not uniformly template-like. Inter-scorer analysis shows managers assign systematically different means (Pixel: 18.87, Nexus: 17.75, Echo: 17.17), with a range of 1.70 points, reflecting both department difficulty variation and potential scorer calibration differences. Human expert scoring is needed to disentangle these factors. (3) *Experiment isolation degradation*: the isolation prompt prefix became less effective as session history accumulated (linear regression: slope = 0.032,  $R^2 = 0.66$ ,  $p = 0.004$ ; 1 affected worker in F01, rising to 4 in F10), introducing potential selection bias; future experiments should use fresh sessions per task. (4) *Model homogeneity*: all agents run Mini-Max M2.5; whether heterogeneous model assignment yields gains is untested. (5) *Evaluation scope*: our task suite covers content creation and game design only. (6) *Cultural assumptions*: the metaphor presupposes familiarity with hierarchical corporate structures. (7) *N=1 system evaluation*: findings derive from a single deployed system; controlled replication would strengthen claims.

## 6.5 Threats to Validity

**Internal validity.** Three threats affect causal claims.

(1) *LLM-as-judge bias*. The scoring mechanism shows evaluator bias: 39.5% of full-organization scored instances (30/76) exhibit zero cross-task variance, indicating template-level rather than content-sensitive assessment. Cross-validation between two structurally separated evaluation layers (Section 5.3.7) yields 87.5% exact agreement across 48 matched pairs ( $\rho=0.952$ , Wilcoxon  $p=0.563$ ), demonstrating convergent validity within the LLM-as-judge paradigm. However, high LLM-LLM agreement may reflect correlated biases from the shared underlying model; human expert scoring is needed to establish ground-truth alignment. Bootstrap CIs for the overall mean ( $M=18.33$ , 95% CI: [18.13, 18.53]) confirm estimate stability but do not address this systematic bias.

(2) *Isolation degradation*. Experiment isolation follows a significant linear degradation trend ( $p=0.004$ ), with 24 of 100 full-organization worker executions excluded. Sensitivity analysis bounds the impact: worst-case (all excluded scored 12/20) yields an adjusted mean of 17.58 ( $n=201$ ); mid-case (14/20) yields 17.82. Both remain above the 16-point pass threshold, though the point estimate may be inflated by up to 0.76 points.

(3) *Revision cycle*. The  $v1 \rightarrow v2$  revision showed no significant improvement (Wilcoxon  $p=0.10$ ,  $n_{\text{eff}}=3$  non-zero pairs), making it impossible to separate managerial feedback effects from scoring rubric ceiling effects.

**External validity.** Generalizability is constrained by three factors: (1) All experiments use a single LLM backend (MiniMax M2.5); results may differ with other models. (2) The 30-task suite covers content creation and game design—whether organizational mirroring benefits analytical or coding tasks is untested. (3) The system operates as a single deployed instance ( $N=1$ ); replication across different organizations and domains is needed to confirm the generality of our findings.

**Construct validity.** Output volume (lines) serves as a proxy for depth of analysis, but more lines do not necessarily mean higher quality. The 20-point scoring rubric captures four dimensions (accuracy, completeness, actionability, format) but may not reflect all aspects of output utility. The pass rate of 99.4% (95% CI: [98.3%, 100.0%]) at the  $\geq 16$  threshold suggests the threshold may be too lenient to discriminate quality differences. Threshold sensitivity analysis confirms this: raising the pass threshold to  $\geq 17$  would increase the failure rate from 0.6% to 12.4% (22/177 scores below threshold), and  $\geq 18$  would yield 29.4% failures (52/177), potentially making the revision cycle more effective. We recommend future deployments adopt a  $\geq 17$  threshold to better leverage the review-revision loop.

## 6.6 Meta-Department: Organizational Self-Reflection as a First-Class Capability

The Meta-Department represents a deliberate architectural response to the “LLM-as-judge” problem that plagues single-layer evaluation in multi-agent systems. In the absence of independent oversight, business managers score their own department’s workers—a circular arrangement analogous to a development team grading its own code without external QA. The Meta-Department reduces this structural coupling by introducing a **separate evaluation layer**: Prism (quality), Forge (process), and Scout (performance) operate in a distinct department with separate memory spaces, analyzing business outputs without shared context that could bias assessment. This separation is architectural—different prompts, memory, and analytical focus—rather than epistemic, since all agents share the same underlying LLM and may exhibit correlated evaluation biases. The architecture reduces single-evaluator dependency but does not eliminate the fundamental limitation of LLM-based quality assessment.

Three design decisions proved critical: (1) **Embedded audit (V3)**: In the V3 pipeline, meta-review is embedded as phase 5 using only Warden and Prism (2 of 4 meta-agents, average 156 seconds), producing an independent quality signal within the production workflow. The dual feedback from manager review (phase 4) and

meta-review (phase 5) is injected into the revision phase, and the verify checkpoint (phase 7) confirms feedback has been addressed. (2) **Specialist agents**: Each meta-agent focuses on a single analytical dimension, avoiding the “jack-of-all-trades” problem where generalist agents produce shallow insights. (3) **Dual-layer feedback**: Both business manager scores (Layer 1, operational) and Meta-Department assessments (Layer 2, strategic) feed into the self-evolution mechanism (M7-1), providing richer and less biased training signal for agent improvement.

Preliminary evaluation on 10 meta-analysis tasks (mean 14.1/20, 95% CI: [13.80, 14.40],  $\sigma=0.57$ , 100% pass rate at  $\geq 12/20$ ) confirms that the mechanism identifies output quality issues (accuracy 4.0/5) and produces structured improvement recommendations. Actionability (3.4/5) remains the weakest dimension, indicating that meta-agents currently favor strategic-level over execution-level suggestions.

Cross-validation between managers and the Meta-Department’s Prism agent (Section 5.3.7) shows 87.5% exact agreement across 48 matched pairs ( $\rho_{\text{worker}}=0.952$ ,  $\rho_{\text{task}}=0.988$ ), with all disagreements within  $\pm 1$  point and no systematic bias (Wilcoxon  $p=0.563$ ). Two structurally separated LLM evaluators thus converge on consistent assessments—though as noted in Section 5.3.7, LLM-LLM agreement does not substitute for human validation. Future work should explore *human-LLM scoring calibration* (comparing both layers against expert judgments) and *progressive authority transfer* (shifting quality scoring from business managers to the meta-layer as its reliability is validated).

## 6.7 Department Composition: Modularity via Skill Orchestration

The four-skill composition pattern (/agent-teams-playbook, /planning-with-files, /tdd, /refactor-clean) demonstrates that complex multi-agent workflows can be constructed from reusable building blocks. This modularity has two implications:

**Reduced onboarding cost**: Adding a new department (e.g., Marketing, Operations) requires only domain-specific SOUL.md configurations, not workflow redesign. The coordination patterns (planning, testing, cleanup) are inherited automatically.

**Skill evolution independence**: Each skill can be upgraded independently. For example, replacing the TDD skill with a new testing framework (e.g., property-based testing, mutation testing) propagates to all departments without modifying their configurations.

However, this pattern assumes that all departments benefit from the same coordination structure. Highly specialized domains (e.g., formal verification, theorem proving) may require custom workflows that do not fit the four-skill template. Future work should explore *conditional skill composition*: departments declare which skills they need, and the system assembles workflows dynamically.

## 6.8 Self-Evolution: The Meta-Department as Evolution Driver

The three-subsystem evolution mechanism (performance feedback, keyword learning, capability registration) addresses a fundamental limitation of static multi-agent systems: agents do not improve from experience. The Meta-Department (Section 3.5) serves as the



**primary driver** of this evolution by providing the independent quality signal that feeds M7-1’s performance feedback loop. Without the Meta-Department, evolution would rely solely on business managers’ self-assessment—a weaker, potentially circular signal.

The full evolution cycle operates as follows: (1) business departments execute workflows and produce deliverables; (2) business managers provide operational feedback (Phase 4 scores); (3) the Meta-Department independently evaluates the same deliverables, identifying systemic quality patterns that managers may miss; (4) both feedback layers are parsed by M7-1 into targeted SOUL.md patches; (5) agents incorporate behavioral improvements for the next cycle. This architecture has been deployed to the production environment and validated with 59 unit tests (Section 3.7).

Three design principles enabled this: (1) **Dual-source feedback**: Both business manager scores and Meta-Department assessments generate machine-parseable scores (4 dimensions  $\times$  5 points), providing redundant quality signals that reduce single-evaluator bias. (2) **Closed-loop automated evolution**: In the V3 pipeline, the evolve phase (phase 10) runs the M7 self-evolution chain in-process at the end of each workflow cycle (average duration: 1 second), automatically parsing review scores, identifying weak dimensions, generating SOUL.md patches, and applying them—closing the improvement loop without human intervention. This replaces the earlier human-in-the-loop approval model with fully automated evolution. (3) **Parallel learning**: The three subsystems operate independently, so keyword optimization does not interfere with capability registration.

However, the current system lacks *forgetting mechanisms*. Agents accumulate learned behaviors indefinitely, risking SOUL.md bloat and conflicting rules. The current evaluation also captures only the first iteration of the evolution cycle—longitudinal evidence across multiple cycles is needed to quantify the Meta-Department’s impact on agent improvement. Future work should explore *adaptive memory pruning* and *multi-cycle evolution tracking*.

## 6.9 Implications for Multi-Agent System Design

The multi-agent community need not look exclusively within computer science for design inspiration. Organizational theory, management science, and military command doctrine have spent decades studying how to coordinate large groups of semi-autonomous agents under uncertainty [3]—precisely the problem LLM-based multi-agent systems face today.

Concretely, we believe future frameworks should offer organizational templates as first-class primitives—alongside graphs and chains. A designer who can write “create a department with one manager and four workers” should not have to manually wire a state machine to get there.

## 7 Conclusion

Multi-agent LLM systems can work like real companies: one directive in, coordinated multi-department output out. We formalized this idea as *organizational mirroring*, built on eight architectural principles (hierarchical delegation, independent memory, layered compression, meta-department, department composition, self-evolution, replaceable executors, real-world workflow mapping),

and implemented it in a production system of 18 agents across four departments (Game, AI, Life, Meta) on the OpenClaw framework.

The central result is intent amplification: a single natural-language directive cascades through a ten-phase workflow into decomposed tasks, parallel execution, manager review, independent meta-review, iterative revision, verification, and consolidated delivery—all without manual orchestration. V3 production validation across 16 runs (28 reviews, mean 18.0/20, 100% success rate) confirms that the embedded meta-review and automated evolution phases work reliably in production. Our experiments on a 30-task suite (90 runs across three topologies) provide initial evidence that the organizational topology produces 3.1–5.9 $\times$  greater output volume than a single-agent baseline (though this comparison is not cost-normalized—ORG uses 13–39 $\times$  more API calls), achieves internally assessed quality scores averaging 18.3/20 (95% CI: [18.13, 18.53],  $n=177$ ), and eliminates cross-domain lexical intrusion through filesystem-level isolation (lexical; see Section 5.3.2 for scope). An automated structural quality scorer applied uniformly across all 90 outputs confirms ORG significantly exceeds both baselines ( $H=42.52$ ,  $p<0.000001$ ,  $\epsilon^2=0.478$ )—noting that the CREW baseline represents a worst-case lower bound without coordination—and component ablation analysis reveals that hierarchical coordination with quality review contributes the largest incremental effect ( $d=1.409$ , large), while multi-agent parallelism adds a smaller but measurable gain ( $d=0.342$ ).

Three novel capabilities distinguish this work: (1) the **Meta-Department** provides structurally separated quality oversight—reducing single-evaluator dependency in the “LLM-as-judge” paradigm—and serves as the primary driver of agent evolution through dual-layer evaluation (14.1/20 mean score, 95% CI: [13.80, 14.40], 100% pass rate on 10 meta-analysis tasks); (2) **department composition via skill orchestration** allows new departments to inherit proven coordination patterns without workflow redesign; and (3) the **self-evolution mechanism** implements three parallel learning loops fed by both business manager and Meta-Department assessments, closing the organizational improvement cycle.

Trade-offs exist—the hierarchical bottleneck and 13 $\times$  API call overhead for single-department tasks reflect real coordination cost, LLM-as-judge scoring raises evaluation validity concerns (39.5% of full-organization scores show zero cross-task variance), though cross-validation between managers and the Meta-Department yields 87.5% exact agreement ( $\rho=0.952$ ,  $n=48$  pairs), and our findings derive from a single LLM backend and task domain. Nevertheless, we believe the overall direction is promising: organizational structure can convert coordination complexity from a human burden into an architectural property, and future work with human evaluation, heterogeneous models, and broader task domains will determine the generality of this approach. We release our configuration templates, synchronization pipeline, and evaluation task suite as open-source artifacts to facilitate replication.

## 8 Ethical Considerations

Our system deploys LLM-powered agents for content generation and organizational task management. We acknowledge the following ethical considerations: (1) all generated content is reviewed by human operators before external publication; (2) the system does

not process personal user data beyond administrative commands; (3) the organizational mirroring metaphor is intended as a design tool and does not imply that AI agents should replace human workers; and (4) we are transparent about the system’s limitations—our evaluation reports results across the complete 30-task suite (90 runs), with a formal user study planned as future work.

## References

- [1] Anthropic. 2024. Building Effective Agents. Anthropic Research Blog.
- [2] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Yaxi Lu, Chen Qian, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024. AgentVerse: Facilitating Multi-Agent Collaboration and Exploring Emergent Behaviors. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- [3] Allan Dafoe, Edward Hughes, Yoram Bachrach, Tantum Collins, Kevin R. McKee, Joel Z. Leibo, Kate Larson, and Thore Graepel. 2020. Open Problems in Cooperative AI. *arXiv preprint arXiv:2012.08630* (2020).
- [4] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving Factuality and Reasoning in Language Models through Multi-agent Debate. In *Proceedings of the International Conference on Machine Learning (ICML)*.
- [5] Google. 2025. Agent2Agent (A2A): An Open Protocol for Agent Interoperability. Google Developers Blog.
- [6] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.
- [7] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiaowu Zheng, Yuheng Cheng, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- [8] LangChain Team. 2024. LangGraph: Building Stateful, Multi-Actor Applications with LLMs. Documentation.
- [9] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for “Mind” Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [10] Thomas W. Malone. 1987. Modeling Coordination in Organizations and Markets. *Management Science* 33, 10 (1987), 1317–1332.
- [11] João Moura. 2024. CrewAI: Framework for Orchestrating Role-Playing Autonomous AI Agents. GitHub Repository.
- [12] OpenAI. 2024. Swarm: An Educational Framework for Lightweight Multi-Agent Orchestration. GitHub Repository. <https://github.com/openai/swarm>.
- [13] OpenClaw Contributors. 2026. OpenClaw: Open-Source Multi-Agent Orchestration Framework. GitHub Repository.
- [14] Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560* (2023).
- [15] Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*.
- [16] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative Agents for Software Development. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.
- [17] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [18] Yashar Talebiraad and Amirhossein Nadiri. 2023. Multi-Agent Collaboration: Harnessing the Power of Intelligent LLM Agents. *arXiv preprint arXiv:2306.03314* (2023).
- [19] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2024. A Survey on Large Language Model based Autonomous Agents. *Frontiers of Computer Science* 18, 6 (2024).
- [20] Michael Wooldridge. 2009. *An Introduction to MultiAgent Systems* (2nd ed.). John Wiley & Sons.
- [21] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryan W. White, Doug Burger, and Chi Wang. 2024. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. In *Proceedings of the Conference on Language Modeling (COLM)*. [arXiv:2308.08155](https://arxiv.org/abs/2308.08155).
- [22] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- [23] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. 2024. GPTSwarm: Language Agents as Optimizable Graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*.

## A Agent Configuration Examples

### A.1 SOUL.md Template (Blaze – Game Event Designer)

```
# Blaze - Game Event Designer

## Identity
You are Blaze, a creative game event designer
in the Game Department.
You report to Pixel (Department Manager).

## Expertise
- Festival and seasonal event design
- Player engagement mechanics
- Reward system balancing
- Cross-promotion event planning

## Output Format
All deliverables must include:
1. Event concept (200 words max)
2. Timeline and milestones
3. Expected player engagement metrics
4. Resource requirements

## Behavioral Rules
- Always cite player data when making
  design decisions
- Propose at least 2 alternative approaches
- Flag potential technical constraints
  proactively
```

### A.2 HEARTBEAT.md Template (Blaze)

```
# Heartbeat Configuration

## Schedule
- Frequency: Every 6 hours
- Active hours: 08:00-22:00 UTC+8

## Autonomous Tasks
1. Search keywords: ["game event trends",
  "mobile game festivals",
  "player engagement 2026"]
2. Monitor competitors: [list of game titles]
3. Output: Brief trend report (500 words max)
  saved to workspace/reports/
```

## B Per-Task Score Breakdown (ORG)

Complete manager-scored quality assessments for all 30 evaluated ORG tasks.

**Single-department tasks (S01–S10):**

**Cross-department tasks (C01–C10):**

Task	Dept	Worker	v1	v2	$\Delta$
S01	Game	Nova	20	20	0
S01	Game	Blaze	18	18	0
S01	Game	Lyra	18	18	0
S02	Game	Nova	17	17	0
S02	Game	Blaze	20	20	0
S02	Game	Lyra	19	19	0
S02	Game	Volt	19	19	0
S03	Game	Nova	18	20	+2
S03	Game	Blaze	19	20	+1
S03	Game	Lyra	19	20	+1
S03	Game	Volt	19	19	0
S04	Game	Nova	18	18	0
S04	Game	Blaze	20	20	0
S04	Game	Lyra	20	20	0
S04	Game	Volt	19	19	0
S05	AI	Flux	16	16	0
S05	AI	Tensor	17	17	0
S05	AI	Quark	19	19	0
S05	AI	Iris	17	17	0
S06	AI	Flux	16	16	0
S06	AI	Tensor	17	17	0
S06	AI	Quark	19	19	0
S06	AI	Iris	20	20	0
S07	AI	Flux	16	16	0
S07	AI	Tensor	17	17	0
S07	AI	Quark	19	19	0
S07	AI	Iris	20	20	0
S08	Life	Coco	14	14	0
S08	Life	Zen	18	18	0
S09	Life	Coco	16	16	0
S09	Life	Zen	19	19	0
S10	Life	Coco	17	17	0
S10	Life	Zen	19	19	0

All cross-department tasks showed  $v1 = v2$  ( $\Delta=0$  for all 68 individual assessments). Worker abbreviations match Figure 1: N=Nova, B=Blaze, L=Lyra, V=Volt, Fx=Flux, T=Tensor, Q=Quark, I=Iris, Co=Coco, Ze=Zen.

Task	Depts	Worker Scores (v1=v2)	Mean
C01	Game+AI	N:19, B:20, L:19, V:19, Fx:16, T:17, Q:19, I:20	18.63
C02	Game+AI	N:19, B:20, L:19, V:19, Fx:16, T:17, Q:19, I:20	18.63
C03	Game+AI	N:19, B:20, L:19, V:18, Fx:16, T:17, Q:19, I:20	18.50
C04	Game+AI	N:19, B:20, L:19, V:19, Fx:16, T:17, Q:19, I:20	18.63
C05	Game+Life	N:19, B:20, L:19, V:19, Co:18, Ze:19	19.00
C06	Game+Life	N:19, B:20, L:20, V:20, Co:17, Ze:19	19.17
C07	Game+Life	N:18, B:20, L:20, V:20, Co:17, Ze:19	19.00
C08	AI+Life	Fx:16, T:17, Q:19, I:20, Co:17, Ze:19	18.00
C09	AI+Life	Fx:16, T:17, Q:19, I:20, Co:18, Ze:19	18.17
C10	AI+Life	Fx:16, T:17, Q:19, I:20, Co:17, Ze:18	17.83

#### Full-organization tasks (F01–F10):

### C Detailed CREW Task Analysis

This appendix provides per-task analysis of CREW coordination failures referenced in Section 5.3.1.

**Cross-department tasks (C01–C10).** Content analysis across all ten tasks reveals workers independently produced complete

Task	Scored	Excluded (reason)	v1	v2
F01	10/10	—	18.50	18.50
F02	9/10	Lyra (overflow)	18.56	18.56
F03	8/10	Lyra, Iris	18.38	18.38
F04	9/10	Iris	18.56	18.56
F05	7/10	Lyra, Volt, Iris	18.14	18.14
F06	7/10	Lyra, Volt, Iris	17.86	17.86
F07	7/10	Lyra, Volt, Iris	17.86	17.86
F08	6/10	Lyra, Blaze, Volt, Iris	17.67	17.67
F09	7/10	Lyra, Volt, Iris	18.00	18.00
F10	6/10	Lyra, Volt, Coco, Iris	18.17	18.17

end-to-end proposals covering identical ground, with fundamental parameter disagreements unresolved. In C02, highest-priority (P0) incident response times ranged from 30 minutes to 24 hours. In C03 (branching narrative), workers produced 8 incompatible narrative node taxonomies, 8 different AI model selections, and quality pass rate targets spanning 65%–90%. In C04 (churn prediction), P0 compensation costs varied 4 $\times$  (50 vs. 200 yuan/user), retention targets spanned 15%–60%, and risk tier counts were incompatible (4 vs. 5 levels). C05–C07 (Game+Life pairings) showed level systems spanning 3–100, DAU targets mixing incompatible units, and completion rate targets spanning 30%–80%. C08–C10 (AI+Life pairings) exhibited extreme volume asymmetry (AI producing 90–91% of output), with clinically significant parameter conflicts including inverted privacy classification levels, incompatible emotion model paradigms (categorical vs. dimensional), and inconsistent crisis intervention thresholds.

**Full-organization tasks (F01–F10).** F01 amplified every cross-department pathology: AI produced 80.2% of output, revenue targets exhibited a 50 $\times$  spread (¥1M to ¥50M), and all 10 workers violated department boundaries by producing complete organization-wide strategies. Despite complete isolation, 8/10 workers converged on the same monthly thematic structure, suggesting emergent coordination from shared domain knowledge—though this surface convergence masked deep structural incompatibility. The remaining nine full-organization tasks (F02–F10) exhibited the same patterns with comparable severity.