HIVEMIND: Role-Bound Agent Swarms for Enterprise Continuity — Institutional Memory, Knowledge Commons, and Coordinated Intelligence Under Clearance

Oscar Serra

June 19, 2026

Abstract

The most expensive failure in a mid-sized company is not a data breach — it is a resignation. When a senior salesperson leaves after seven years, the relationship history, the negotiation instincts, and the undocumented customer quirks walk out with them, and the replacement spends months rebuilding what was never written down. This paper presents **HIVEMIND** — a multi-agent AI architecture for corporate deployment on OpenClaw/TinkerClaw infrastructure whose central design commitment is **organizational continuity**: the institutional knowledge lives in the agents, and the agents outlive the humans who use them. The key architectural move is **role-bound agents** — one canonical agent per job function (a single *sales rep* agent, a single *field engineer* agent), shared by every human who holds that role, rather than a personalized agent per individual. This decision makes talent retention a non-issue (the agent never resigns), collapses cross-department coordination to one channel per function, and reduces onboarding to a single sentence: the new hire tells the agent which territory they now own. Knowledge enters the swarm through a **knowledge cascade** — a centralized, version-controlled Markdown vault (company canon, industry context, customer-relationship model, product catalog) that agents mount via symlinks, authored and maintained by former trainers reborn as **knowledge stewards** who now train the AI once instead of each hire repeatedly. Security is the enabling constraint, not the headline: we retain a five-tier data classification model mapped to Linux filesystem permissions and formally grounded in the Bell-LaPadula model, now applied at role granularity, with a controlled declassification pipeline, nightly compliance sweeps, and human-in-the-loop quarantine. We ground the architecture in a mid-sized family business (~50–200 employees) and a rewritten Sales Coordination case study in which customer ownership is resolved contextually by a single shared agent rather than through inter-agent email routing. We address GDPR and the EU AI Act, a comprehensive risk model, bounded-autonomy operational governance (end users signal IT rather than self-administer their gateway), failure recovery, and a phased deployment strategy. We frame HIVEMIND as an architectural design paper, distinguishing what is proposed, what is operationally specified, and what remains to be empirically validated.

Keywords: multi-agent systems, enterprise AI, organizational continuity, institutional memory, knowledge management, role-bound agents, knowledge commons, OpenClaw, agent swarms, GDPR, EU AI Act, clearance models, Bell-LaPadula

A Design Paper on Deploying Multi-Agent AI Systems That Outlive the People Who Use Them

1. Introduction — The Enterprise AI Deployment Problem

1.1 The Most Expensive Problem in a Mid-Sized Company

The first generation of practical AI assistants is personal. OpenClaw, TinkerClaw, and comparable platforms are designed around a single operator — one human, one agent, one workspace. The agent accumulates contextual memory, learns preferences, and acts as a trusted extension of that individual's cognition. The single-operator design is by now reasonably mature: persistent memory architectures, curiosity-driven self-improvement, and trust tiers governing tool access are all well-understood for the one-human-one-agent case. The individual agent is a solved-enough problem.

But the individual agent paradigm has a ceiling, and that ceiling is the organization. The interesting enterprise problem is not "how do I give one person a better assistant." It is a set of problems that have nothing to do with individual productivity and everything to do with how knowledge moves — and fails to move — through a company over time:

- **The talent leak.** A senior salesperson with seven years of relationship history resigns. The replacement inherits a CRM full of names and none of the context: which customer never answers the phone before noon, which one always asks for a discount and never takes it, which contract has an unwritten handshake behind it. Months of productivity evaporate, and the same loss recurs with every departure. This is the single most expensive recurring failure in a mid-sized company, and no wiki has ever solved it, because the knowledge was never the kind that gets written down.
- **Knowledge silos.** A rep learns in a client meeting that a competitor launched a new product. It lands in a personal notebook. The sales director never hears it. The intelligence dies where it was born.
- **Onboarding cost.** Every new hire is trained from near-zero by someone whose actual job is something else. The training is repeated per hire, lost on turnover, and inconsistent across trainers.

1.2 The Continuity Thesis

HIVEMIND's central commitment is **organizational continuity**: the institutional knowledge lives in the agents, and the agents outlive the humans who use them.

The architectural move that delivers this is **role-bound agents**. Instead of a personalized agent per individual — María gets LUNA, Pedro gets ATLAS — there is *one* canonical agent per job function: a single **sales rep agent**, shared by every salesperson; a single **field engineer agent**, shared by every engineer. The human is a transient occupant of a role; the agent is the durable holder of the role's accumulated knowledge.

This one decision cascades into three properties that a per-person design cannot offer:

1. **Talent retention becomes a non-issue.** The agent never resigns. When a salesperson leaves, nothing is snapshotted, migrated, or scrubbed — the agent simply keeps serving the next occupant. Offboarding collapses to a single edit: the new hire tells the agent "I now cover the northern territory," and a workload-division file is updated.

2. **Cross-department coordination collapses to one channel per function.** With one sales agent and one engineering agent, a sales-to-engineering handoff is a single, stable, auditable channel — not an N×M mesh of personal agents that has to be re-wired every time someone joins or leaves.

3. **Onboarding becomes inheritance.** A new hire does not start from zero. They start from everything the role has ever learned, and contribute back to it.

1.3 The Core Tension

Continuity and coordination create value only if they do not also create exposure. The central engineering tension is between two legitimate needs:

The need for integration. The more an agent knows about the company, the better it serves. A sales agent that holds the *whole team's* customer history — every touchpoint, complaint, and special arrangement — is vastly more effective than one scoped to a single person's slice. Role-bound agents maximize this by construction: the shared agent already holds everything the role knows.

The need for boundaries. Salary data, M&A strategy, infrastructure credentials, legal proceedings — these must be protected across role and tier boundaries. An AI agent that leaks this is worse than no agent, because the leak is invisible, scalable, and hard to attribute.

HIVEMIND resolves this tension through deep, role-level integration with clearance-based boundaries enforced at the infrastructure level, not merely at the prompt level. Security is the enabling constraint that makes the continuity architecture deployable — it is not the headline, and a deployment that led with it would be solving the wrong problem first.

1.4 Architecture Overview

Before presenting the detailed design, we provide a high-level view of the HIVEMIND architecture (Figure 1).

A role agent is shared by every human who holds that role. Where a department contains a single role (e.g., a sales floor of identically-scoped reps), Level 1 and Level 2 may merge: the role agent *is* the department agent. The three-level structure is retained for departments with multiple distinct roles at different scopes.

1.5 The Per-Agent Cognitive Stack

Single-agent cognition has, over the past few years, been tackled one problem at a time: persistent memory, fast retrieval, stable identity, self-improvement, multi-model deliberation, and layered security each have well-understood solutions for the one-human-one-agent case. HIVEMIND runs all of them at once, across many agents, under concurrency and clearance constraints. Each agent in the swarm is not a thin role wrapper around a model — it is a full cognitive stack of six named subsystems referenced throughout this paper:

- **Append-only memory (the memory subsystem).** Interactions are stored as append-only timestamped events, with older events compressed into lightweight summaries — effectively unlimited memory within bounded storage (Section 2.9).
- **Retrieval index (the retrieval subsystem).** A topic-keyword index over memory segments enabling sub-second lookup across the full interaction history (Section 3.6).
- **Stable persona (the identity subsystem).** A persona definition read at session start gives each agent a distinct, stable identity — a sales agent behaves differently from an IT agent because their persona files encode different values, vocabulary, and domain knowledge.

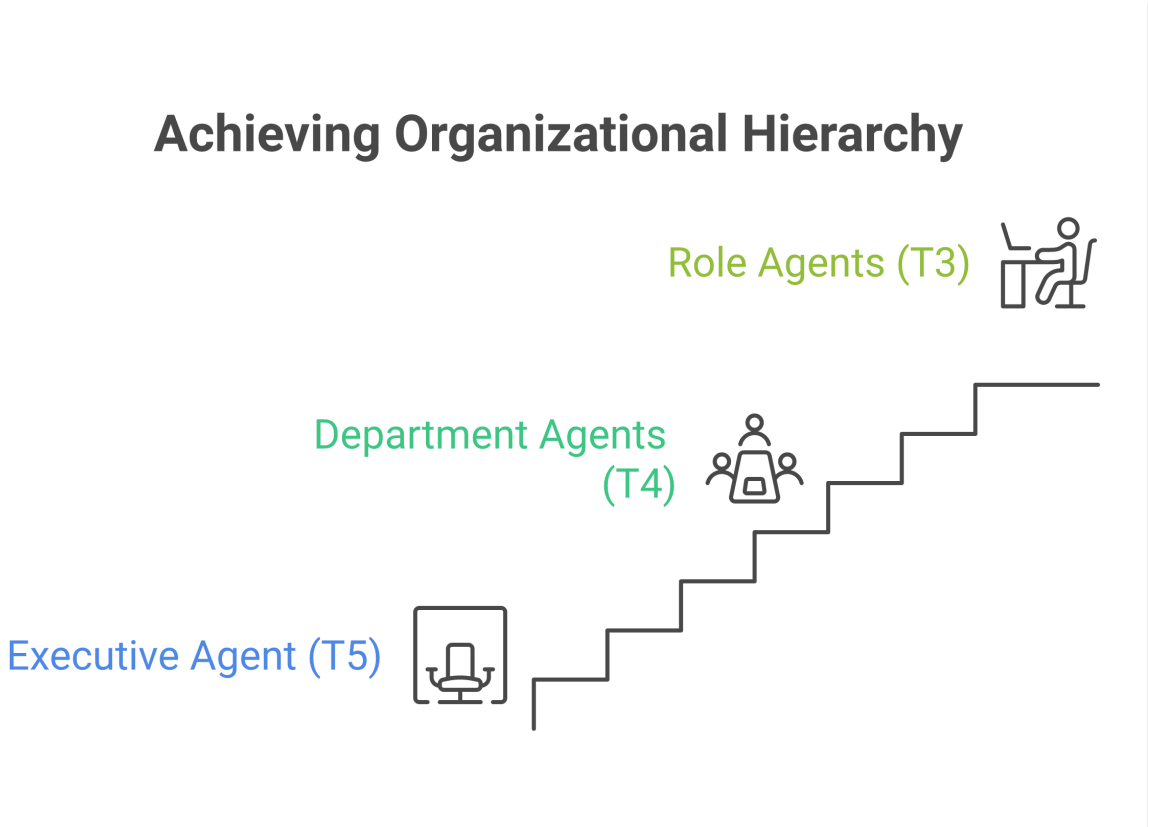

Figure 1. Figure 1. HIVEMIND's three-level agent hierarchy. A single **executive** agent (L3-EXEC, T5) sits over one **department** agent per function (L2-SALES / L2-IT / L2-HR, T4), each over a **role** agent shared by everyone who holds that job function (L1-SALES-REP / L1-FIELD-ENGR / L1-SUPPORT, T3). Queries flow downward only; structured intelligence pushes flow upward; lateral traffic between role agents is mediated by the department agent — and every inter-level edge is gated by the clearance lattice, enforced by the Linux kernel, not the model.

- **Self-improvement (the reflection subsystem).** A nightly cycle reviews the day's errors and adjusts behavioral rules, so the swarm improves without centralized retraining.
- **Multi-model deliberation (the deliberation subsystem).** Complex cross-department decisions trigger several models debating and synthesizing a consensus, so no single model's blind spot drives a critical decision.
- **Layered security (the security subsystem).** A framework enforcing clearance and defending against prompt-injection that could compromise one agent and cascade through the swarm.

For brevity the paper refers to these by short names — ENGRAM (memory), HIPPOCAMPUS (retrieval), CORTEX (identity), CEREBELLUM (reflection), SYNAPSE (deliberation), and AEGIS (security) — purely as labels for the six functions above. Each is a well-studied single-agent subsystem in its own right; here they matter only as the per-agent capabilities the swarm composes.

Running the full stack on every agent would be wasteful, so HIVEMIND tiers it (Table 1): cognitive cost should track decision consequence.

Table 1. Cognitive stack by agent tier (under role-binding).

Capability	Role agent (L1)	Department (L2)	Executive (L3)
Memory (ENGRAM)	shared role memory	full + department aggregate	full + organization-wide read
Retrieval (HIPPOCAMPUS)	role index	role + mediated cross-role	+ all-department read
Persona (CORTEX)	per-role template + thin per-person layer	per-department	bespoke (single agent)
Self-improvement (CEREBELLUM)	shared role rule-base	department-level reflection	full reflection
Deliberation (SYNAPSE)	off (escalate instead)	on for cross-role decisions	on for cross-department/strategic
Security (AEGIS)	clearance ceiling T2–T3	T4 + clearance validator	T5 + dead switch + two-person rule

Two choices are deliberate. First, role agents do not run SYNAPSE deliberation: a decision consequential enough to need multiple models arguing is consequential enough to escalate to the department agent (Section 3.3). Second, role-tier self-improvement writes to a *shared role rule-base* — which under role-binding is not a special case but the default: a lesson the sales agent learns ("this customer always disputes the first invoice") is immediately available to every salesperson, declassified through the same pipeline as any downward flow (Section 4.4). Role-binding and a shared rule-base are the same idea applied to behavior rather than knowledge.

1.6 Related Work and Positioning

The organizational framing of multi-agent systems is not unique to this paper; over the past year, academic, open-source, and industrial efforts have independently converged on organizational structure as the coordination model for multiple agents. Three coordination topologies are now visible: **fan-out-and-synthesize** (a coordinator dispatches one problem to several models and merges answers — no persistent roles); **coordinator-with-workers** (a single prompt spawns tool-restricted workers communicating through shared files — Claude Code's coordinator mode is the industrial exemplar); and **org-chart** (persistent hierarchical roles, work distributed as

tickets, per-agent budget tracking). These are points on a spectrum of persistence and structure, not competitors. HIVEMIND sits at the high-structure end and adds the two things the leaner topologies omit: a clearance lattice enforced below the model, and the full per-agent cognitive stack of Section 1.5.

- **Paperclip** (open-source "zero-human company" orchestration) mirrors HIVEMIND's structural thesis closely — each role is a separate agent, work flows through tickets, per-agent budgets are tracked centrally — and is direct evidence the org-chart paradigm works. What it lacks is exactly HIVEMIND's subject: no clearance or information-flow model (any agent reads anything), no governed shared memory, no persona persistence. HIVEMIND can be read as Paperclip with a Bell-LaPadula spine and a full cognitive stack — and role-binding goes one step further than Paperclip's per-role agents by making the role agent *shared across occupants* for continuity.
- **Coordinator/teammate execution models** contribute useful vocabulary: a *teammate* in a separate context communicating through a file-based mailbox maps naturally onto HIVEMIND's clearance-filtered channels (Section 3.3) — every message is a file with owner, group, and permission bits, which is precisely the enforcement substrate. **Single-agent role-switching** (e.g., GStack: one process, multiple role prompts) marks the low-cost end of the design space and is a legitimate small-organization alternative (Section 10).
- **headroom** (Chopra, 2026; ~24.7k stars, Apache-2.0) is a production context-compression system built on **Compress-Cache-Retrieve (CCR)**: it caches originals and retrieves them on demand, reporting 60–95% token savings at near-zero accuracy delta with a runnable eval. It bears directly on HIVEMIND's memory subsystem and creates one tension and one template — reversible retrieval collides with the lattice (Section 2.10), while its shipped accuracy-delta eval is the discipline HIVEMIND borrows for its redaction-fidelity eval (Section 4.4.2).
- **addyosmani/agent-skills** (~56.8k stars) contributes a **doubt-driven-development** loop whose reusable kernel — a *fresh-context adversarial reviewer denied the source* — maps onto HIVEMIND's hardest problem, implication-leakage at the declassification boundary. Section 4.4.1 re-purposes it as a leakage oracle: run across a clearance boundary, a quality check becomes an information-flow test.

The convergence argument. That organizational coordination patterns emerged independently across academic, open-source, and industrial work suggests corporate structure is not merely a convenient metaphor for multi-agent systems — it may be the natural coordination topology. Companies already encode "who may know what" and "who reports to whom" in org charts and filesystem permissions; mapping agents onto that structure is less an invention than a recognition. What distinguishes HIVEMIND is the claim that this same structure, taken seriously, *is* the security model — the org chart is the access-control graph, not a layer on top of it.

1.7 Paper Organization

Section 2 defines the information architecture, the formal security model, the knowledge-commons cascade, and the clearance-aware memory model. Section 3 presents the role-agent hierarchy, the naming convention, the multiple orthogonal axes along which the network operates (clearance, supervision, bottleneck reduction, workload sharing, self-modification, knowledge propagation), and cross-role retrieval. Section 4 details the Sales Coordination case study under contextual ownership, including the doubt reviewer and the redaction-fidelity eval. Section 5 addresses privacy and legal considerations — GDPR and the EU AI Act — which motivate the compliance architecture in Section 6. Section 7 covers technical implementation, including an inter-agent protocol specification and operational governance. Section 8 analyzes risks and mitigations.

Section 9 presents a phased deployment strategy. Section 10 sketches future directions. Section 11 collects limitations. Section 12 concludes.

2. Information Architecture — The Corporate Filesystem Model

2.1 The Shared Server as Mental Model

Every mid-sized company running on-premises or hybrid infrastructure has a shared file server. Top-level folders for each department, subfolders for specific functions, and a web of permissions that determines who can read or write to what. This folder hierarchy is a materialized representation of the company's information governance model, encoding decades of institutional decisions in chmod bits and ACLs.

HIVEMIND maps directly onto this existing model. The advantages:

1. **Familiarity.** IT administrators understand filesystem permissions without learning a new access control paradigm.
2. **Auditability.** Filesystem access is loggable, diff-able, and understood by existing compliance tools.
3. **Trust.** Employees already accept IT-controlled data access. Extending this to AI agents feels less invasive than a new control plane.
4. **Enforcement.** Linux filesystem permissions are enforced by the kernel, not by a language model's good intentions. This is a critical distinction we return to in Section 8.

2.2 Data Classification Tiers

We define five classification tiers:

Tier	Label	Examples	Default Access
T1	**PUBLIC**	Company website, product brochure, public pricing, press releases	All agents, external systems
T2	**INTERNAL**	Procedures, internal manuals, product specifications, org chart	All role agents
T3	**CONFIDENTIAL**	Financial data, contracts, email correspondence, CRM records	Role-relevant agents only
T4	**RESTRICTED**	Salaries, infrastructure passwords, M&A plans, margin data, HR records	Senior management + relevant department heads
T5	**SECRET**	Board-level strategy, legal proceedings, acquisition targets, whistleblower reports	Executive agents only

These tiers map to Linux filesystem paths:

```
/corporate/
+-- public/          # directory: 0755, files: 0644
+-- internal/        # directory: 0750, group: all-employees; files: 0640
+-- confidential/    # directory: 0750, per-department group; files: 0640
+-- restricted/      # directory: 0750, group: management; files: 0640 + per-dept ACLs
‘-- secret/          # directory: 0700, owner: agent_strategos; files: 0600
```

Each agent process runs as a distinct Linux user. User group membership determines tier access. The kernel enforces this — no prompt instruction can override a `permission denied`.

Note: directory permissions (e.g., 0755) control traversal; file permissions within (e.g., 0644) control read/write access to content. Both layers must be correctly configured to ensure the intended access boundaries.

2.3 The "Need to Know" Principle

Military and intelligence communities operate on "need to know": having clearance for a classification level does not automatically grant access to all information at that level. We apply the same principle. Clearance tier establishes a ceiling, not a floor. Within a tier, access is further scoped by role and department via POSIX ACLs. A Finance department agent with T4 clearance can access salary data (T4, Finance) but not infrastructure passwords (T4, IT).

This limits blast radius: a compromised agent's accessible data surface is bounded by role scope, not just clearance tier.

2.4 Formal Security Model

HIVEMIND's information flow model is grounded in the Bell-LaPadula (BLP) lattice model, adapted for the multi-agent enterprise context.

Definitions. Let $S = \{s_1, s_2, \ldots, s_n\}$ be the set of agents (subjects), $O = \{o_1, o_2, \ldots, o_m\}$ the set of data objects, and $L = \{T1 < T2 < T3 < T4 < T5\}$ the totally ordered security lattice. Each subject s has a clearance $c(s) \in L$ and a role scope $R(s) \subseteq \mathcal{D}$ (where $\mathcal{D}$ is the set of departments). Each object o has a classification $\ell(o) \in L$ and a department assignment $d(o) \in \mathcal{D}$.

Simple Security Property (no read-up). Agent s may read object o only if:

$$c(s) \geq \ell(o) \quad \text{and} \quad d(o) \in R(s)$$

An agent cannot read data classified above its clearance tier, and even at its tier, access is restricted to its role-relevant departments.

Star Property (*-property, no write-down). Agent s may write to object o only if:

$$\ell(o) \geq c(s)$$

An agent cannot write data to a lower classification level, preventing information laundering from restricted tiers to public ones.

Declassification Exception. The *-property creates a practical problem: department agents (T4) need to send filtered summaries to role agents (T3). We handle this through a controlled declassification mechanism. A Level 2+ agent may generate a* new* data object o' with $\ell(o') < \ell(o)$ under these constraints:

1. The declassification is explicitly invoked (not implicit in normal operation).
2. The output o' passes through a template-based redaction pipeline (see Section 4.4) or receives human approval.
3. A declassification event is logged with: source object hash, output object hash, classification change, agent ID, timestamp, and redaction method used.
4. The original object o is not modified.

This exception is the formal basis for redacted summaries. It is narrow by design — only structured declassification through approved channels is permitted.

Enforcement layers. The BLP properties are enforced at two independent layers, and — importantly — *both* can deny a violation, not merely log one:

- **Primary (kernel-level):** Linux filesystem permissions and POSIX ACLs enforce the Simple Security Property. The *-property is enforced by restricting write permissions: agent users have write access only to their own workspace and approved output directories at or above their classification level. The kernel denies the read or write outright.
- **Secondary (runtime tool-boundary):** A pre-execution policy hook — AEGIS — evaluates each tool call *before it runs* and can return a hard block, denying the action even when the runtime is otherwise operating in a permissive mode. This is a genuine second *enforcing* layer rather than advisory defense-in-depth: where the kernel denies the underlying read, AEGIS denies the tool call that would launder it (the aggregation or exfiltration step), so a clearance breach must defeat *two* independent deny gates, not one. Gateway routing rules, agent system prompts, and clearance validators add a third, softer layer of defense-in-depth that the model can in principle err around; the kernel and the tool-boundary hook cannot be talked past by a prompt.

The architectural consequence is worth stating plainly, because earlier multi-agent security analyses (including our own) treated application-layer enforcement on an LLM runtime as an observe-only "physics limit" — something that could flag a violation but never stop one. That is no longer true. A pre-execution hook that returns a block decision *prevents* the tool call. HIVEMIND therefore has two layers that can each *prevent* (not merely record) a violation, which materially strengthens the aggregation and social-engineering mitigations of Section 8: the synthesis-and-exfiltrate tool call can be stopped at the tool boundary, not only surfaced in the nightly sweep.

2.5 Information Flow as a Directed Graph

We model inter-agent information sharing as a directed graph with clearance-enforced edges. Define:

- A_i = agent i with clearance tier c_i and role scope R_i
- $\text{share}(A_i \to A_j, d)$ = agent i shares data item d with classification c_d to agent j

The fundamental rule:

$$\text{share}(A_i \to A_j, d) \text{ is permitted iff } c_j \geq c_d \text{ and } d(d) \in R_j$$

No agent may receive data classified above its clearance tier or outside its role scope. Upward information requests are structurally prohibited (Section 3.3). Downward flows require controlled declassification (Section 2.4).

2.6 Why Bell-LaPadula Is Necessary but Not Sufficient

Bell-LaPadula is the right primary model because HIVEMIND's dominant enterprise risk is **confidentiality failure**: the wrong role agent learning about salaries, credentials, legal matters, or board strategy. But Bell-LaPadula is not the only relevant security model, and the design is stronger when that is stated explicitly.

- **Role-Based Access Control (RBAC).** RBAC explains *who should access what* in organizational terms and maps well to departments, supervisors, and executives. HIVEMIND uses RBAC operationally through Linux users, groups, ACLs, and tool allowlists. However, RBAC alone does not formalize information flow across multi-step agent interactions.
- **Chinese Wall.** Brewer-Nash is relevant where conflict-of-interest boundaries matter — for example, if a consulting firm runs multiple client-specific agent swarms. HIVEMIND does not fully implement Chinese Wall semantics, but future multi-tenant versions should.

- **Biba integrity model.** Bell-LaPadula protects confidentiality; Biba protects integrity. In practice, enterprise agent systems need both. HIVEMIND's audit logs, two-person review for T5 summaries, configuration versioning, and controlled write paths are partial integrity controls, but a full formal integrity model remains future work.
- **Zero Trust architecture.** Zero Trust contributes the operational assumption that no agent, process, or network path should be trusted implicitly. HIVEMIND adopts this stance in gateway authentication, least-privilege tool access, and process isolation.

Accordingly, HIVEMIND should be read as a **hybrid control architecture**: Bell-LaPadula provides the confidentiality lattice; RBAC defines organizational scope; Zero Trust informs runtime assumptions; and partial Biba-like controls support integrity. This clarification matters academically because it prevents overclaiming: the paper does not present Bell-LaPadula as a complete model of enterprise agent security, only as the correct primary backbone for secrecy-preserving information flow.

2.7 The Knowledge Commons and the Cascade Model

The clearance filesystem of Sections 2.2–2.6 answers *what an agent is permitted to read.* It does not answer the prior question: *what should every agent simply know?* A sales agent that has clearance to read the product catalog but has never been told what industry the company operates in, what kind of customers it serves, or how the company expects its people to talk to those customers is technically secure and practically useless. HIVEMIND therefore introduces a second information structure alongside the clearance lattice: a **knowledge commons** — a centralized, version-controlled corpus of Markdown documents that encodes the company's shared, teachable knowledge, mounted into each agent's workspace by symlink.

The commons is deliberately mostly low-tier (T1/T2). It is not where secrets live; it is where *context* lives — the body of knowledge a human trainer would once have transferred to a new hire over their first weeks.

The cascade. An agent's effective knowledge is assembled from four layers, in order of decreasing breadth and increasing specificity — analogous to a CSS cascade or to inherited configuration files:

```
company canon      →  role slice          →  workload map        →  per-person layer
(T1/T2, all agents)   (T2, this role)        (T2/T3, this role)     (thin, current human)
"who we are"          "how this job works"   "who covers what"      "who am I serving now"
```

1. **Company canon (T1/T2, mounted by every agent).** Who the company is and how it operates. Proposed taxonomy:

   - *Company canon* — history, mission, values, org chart.
   - *Industry & market* — the sector, its regulations, the competitive landscape, the domain terminology a newcomer would not know.
   - *Customer-relationship model* — whether the company sells B2B, B2C, B2B2C, or a mix; the buyer personas; the shape and length of the sales cycle; the norms of the relationship (a B2B supplier nurturing a ten-year account behaves nothing like a B2C brand optimizing a checkout).
   - *Product/service catalog* — what the company makes or delivers, specifications, positioning.
   - *Process & SOPs* — how recurring work is done.
   - *Glossary / ontology* — company-specific acronyms and terms.
   - *Voice & communication standards* — how the company talks to customers and to itself.

2. **Role slice (T2, mounted by the role agent).** The job-specific layer: a sales rep's qualification methodology, objection-handling playbook, and pricing-approval ladder; a field engineer's safety procedures and equipment manuals. This is the layer that makes one role agent canonical across all its occupants — every salesperson's agent reads the *same* sales playbook.

3. **Workload map (T2/T3).** A `workload-division.md` recording who currently owns what — which rep covers which territory or account set, which engineer is on which project. This is operational data, not a secrecy boundary (all reps share the same clearance and could in principle see all accounts; the map records *responsibility*, not *permission*). It is the file a new hire edits on their first day and the mechanism by which offboarding becomes a one-line change (Section 5.7).

4. **Per-person interaction layer (thin).** A role agent is shared, so it holds no per-person *memory store*. But it does keep a small, non-sensitive profile of each human it serves — name, role tenure, communication preferences ("prefers bullet points," "wants the number first") — enough to address each person appropriately without fragmenting the shared knowledge into per-person silos. This is the deliberate compromise between usefulness and simplicity: shared knowledge, personalized manners.

Why symlinks, and why they are safe. Each agent workspace mounts the relevant commons paths by symbolic link rather than by copy. This gives a single source of truth: a steward edits one canonical file and the change is visible to every agent that links it, with zero copy-drift. Crucially, symlinks introduce no clearance hole. On Linux, opening a symlink resolves to the target and the kernel checks permissions on the *target* (and traversal on its resolved path components), not on the link. A symlink placed in a T3 workspace pointing at a higher-tier file therefore grants nothing — the agent's UID still fails the target's permissions. Since the commons is T1/T2, every agent already has read access to it; the symlink merely makes shared knowledge appear in-workspace without weakening any boundary.

```
/corporate/knowledge/              # the commons: dir 0755, files 0644 (T1/T2)
+-- canon/                         # company identity, values, org chart
+-- industry/                      # sector, regulation, competitors, terms
+-- customers/                     # B2B/B2C model, personas, relationship norms
+-- catalog/                       # products / services, specs, positioning
+-- process/                       # SOPs
+-- glossary/                      # acronyms, ontology
+-- voice/                         # communication standards
‘-- roles/
    +-- sales-rep/                 # role slice: playbooks, pricing ladder
    |   ‘-- workload-division.md   # who covers what (T2/T3)
    +-- field-engineer/
    ‘-- support/

# A sales-rep agent’s workspace links the slices it needs:
/agents/sales-rep/workspace/knowledge/canon     -> /corporate/knowledge/canon
/agents/sales-rep/workspace/knowledge/customers -> /corporate/knowledge/customers
/agents/sales-rep/workspace/knowledge/role      -> /corporate/knowledge/roles/sales-rep
```

2.8 Knowledge Freshness and Provenance

A commons maintained by humans rots. Stale industry knowledge is more dangerous than absent knowledge, because the agent states it with the same confidence as the truth. Each commons document therefore carries review metadata in its frontmatter:

```yaml
owner: knowledge-steward-sales        # who is accountable for this file
last_reviewed: 2026-05-01
review_cadence: quarterly             # how often it must be re-verified
classification: T2
```

A scheduled job (Section 7.6) flags any document past its review cadence and routes it to its owning steward. This makes knowledge staleness a tracked, owned, auditable property rather than a silent decay — the same discipline a living-documentation system applies to code. Write access to the commons is scoped: stewards hold write permission on their owned paths, gated through the Git review workflow of Section 7.8. Agents read the commons; they do not write to it directly.

2.9 Clearance and the Append-Only Event Store

One interaction between the clearance lattice and the per-agent memory model (ENGRAM, Section 1.5) deserves explicit treatment, because it is a genuine edge case rather than a clean consequence of Bell-LaPadula. Each role agent's memory is an append-only event store in which older events are periodically compacted into lightweight summary objects. Compaction is lossy by design — a summary stands in for many original events — and it is performed *within* an agent's own clearance domain. That domain confinement keeps the interaction safe, but the invariants are worth stating so an implementer does not break them:

- **Compaction never crosses tiers.** A summary inherits the maximum classification of the events it compresses. The compactor may read T4 events in a T4 store and produce a T4 summary; it may not produce a lower-tier summary as a side effect. Downgrading is reserved for the explicit declassification pipeline (Section 2.4), never the compactor.
- **A compacted pointer must not become a confused-deputy.** If a summary is referenced by a higher-tier agent (e.g., `L3-EXEC` reads `L2-SALES`'s compacted history), the reference is resolved under the *reader's* clearance against the *summary's* classification — exactly the Simple Security Property (Section 2.4). The dangerous failure would be a low-clearance agent holding a pointer that, when dereferenced by a privileged service on its behalf, returns content above its tier. HIVEMIND forbids server-side dereferencing of memory pointers on behalf of a requester: an agent can only follow pointers into stores it could already read directly. Pointers are not capabilities.
- **Compaction is itself an audited write.** Each compaction event is logged with input event hashes, output summary hash, and classification, so the nightly sweep (Section 6.3) can verify no compaction silently changed an object's tier.

The general lesson: an append-only store plus lossy compaction does not by itself violate the lattice, but it introduces *new objects* (summaries) and *new references* (pointers) that must each be assigned a classification and resolved under the reader's clearance. Treating compaction output as a first-class classified object closes the gap.

2.10 Compress-Cache-Retrieve Under Clearance

Section 2.9 reached safety by *throwing originals away*: compaction is lossy and a pointer is deliberately not a capability. That is conservative, and it pays a price — every compaction permanently degrades what the agent can later recall. A contemporaneous production system (headroom's Compress-Cache-Retrieve, Section 1.6) suggests the price may be avoidable: it keeps originals in a cache and retrieves them on demand, recovering most lost fidelity at 60–95% token savings. Can HIVEMIND adopt reversibility without reintroducing the confused-deputy hazard?

It can, but only by one substitution: **the retrieve step must be the clearance gate.** We call the variant **Compress-Cache-Retrieve-Under-Clearance (CCR-UC)**, governed by two invariants that mirror the lattice rather than headroom's single-trust-domain assumption:

- **The cache inherits the maximum classification of what it compressed.** A cache built from T4 events is a T4 object. There is no "low-tier cache of high-tier originals" — that object would be precisely the reversible pointer that launders a tier on dereference.
- **Retrieve resolves under the reader's clearance against the cache's classification — the Simple Security Property, unchanged.** A holder of a CCR-UC pointer can expand it only into stores it could already read directly; expansion on behalf of a lower-clearance requester is forbidden.

Under these invariants CCR-UC is not a new exception to Bell-LaPadula — it is the cross-role mediated-retrieval machinery (Section 3.6) pointed inward at an agent's own compacted memory. When a privileged agent expands a cache spanning tiers for delivery to a lower-clearance reader, the expansion passes through the same template-based declassification pipeline (Section 4.4) as any downward flow. The honest boundary against headroom: headroom ships this transform with a runnable accuracy-delta eval and has no notion of classification; HIVEMIND adds the classification semantics and the clearance-checked retrieve, and Section 4.4.2 specifies the eval that would close the remaining gap.

3. Agent Hierarchy Architecture

3.1 Three-Level Structure

Naming convention. To keep the cast legible, every agent is named `L<level>-<SCOPE>`: the leading `L1`/`L2`/`L3` states its level at a glance, and the scope names its function. So `L1-SALES-REP` is unambiguously a Level 1 role agent for sales; `L2-SALES` is the sales department agent; `L3-EXEC` is the executive agent. The reader never has to remember whether "ATLAS" or "STRATEGOS" sat at which tier — the name carries the answer. (System user accounts use the lowercase form `agent_salesrep`, `agent_sales`, `agent_exec`; the `L<level>-<SCOPE>` label is the human-readable alias used throughout this paper.)

Level 1 — Role Agents. One per job function, *shared by every human who holds that role.* A single sales rep agent (`L1-SALES-REP`) serves all salespeople; a single field engineer agent (`L1-FIELD-ENGR`) serves all engineers. Calibrated personality and deep role knowledge; clearance set at the role level (all occupants of a role share one clearance). The agent is the durable holder of the role's accumulated knowledge — it persists across the people who come and go.

Level 2 — Department Agents. One per department (L2-SALES, L2-IT, L2-HR, etc.). Elevated clearance (typically T4, T5 for the executive department). Coordinate the role agents beneath them, aggregate intelligence, enforce department policies, and gate inter-level communication. Where a department contains a single role, the department agent and the role agent are the same entity.

Level 3 — Executive Agent (L3-EXEC). Single top-level agent with T5 clearance and read access to all department agent workspaces. Serves the CEO and senior management. The only agent that sees the complete organizational picture.

3.2 Role Agent Design

Each role agent is a full OpenClaw/TinkerClaw deployment with:

- **Standardized build, canonical across occupants.** There is one sales rep agent build — one persona, one skill set, one mounted role slice — stamped once and shared, not reinvented per hire. Every salesperson talks to an agent that has read the same playbook and holds the same team-wide customer history. Personalization is a *thin* per-person interaction layer (Section 2.7): the agent addresses María and Pedro differently in tone and format, but draws on the same shared knowledge.

- **Shared, role-scoped workspace.** Contains the files the role can access plus the symlinked commons slices it needs. Because the agent is shared, there is no per-person memory store to fragment or migrate; institutional memory accrues to the role.
- **Knowledge via cascade, not injection.** Rather than baking assigned clients and procedures into one person's system prompt, the agent reads them from the commons (company canon, role slice) and the `workload-division.md` (who currently owns what). Updating the company's knowledge updates every agent at once.
- **Tool access scoped by role.** A sales rep agent can access CRM, email, calendar, and product catalog. It cannot access payroll, infrastructure tools, or HR records. Explicit DENIED entries for out-of-scope systems.

This design is what makes talent retention a non-issue. There is nothing to offboard when a person leaves and nothing to build from scratch when one arrives — only a `workload-division.md` to edit. The new occupant inherits, on day one, everything the role has ever learned.

3.3 Information Flow Rules

Rule 1: Downward queries are permitted. Level 3 can query Level 2. Level 2 can query its Level 1 agents. Queries retrieve information, request actions, audit memories, or issue directives.

Rule 2: Upward queries are structurally prohibited. Level 1 agents cannot initiate queries to Level 2 or Level 3. This is enforced by the gateway's session routing topology: role agent sessions are not registered as valid targets for upward message addressing.

Rule 2a: Upward intelligence push is permitted through structured channels. This is a controlled exception to Rule 2. Role agents may push data *upward* to their department agent, but only through pre-defined structured channels with these constraints:

- Each channel has a fixed schema (message type, required fields, classification ceiling).
- The role agent can only push data classified at or below its own clearance tier — it cannot push data it shouldn't have.
- Push channels are defined in the department agent's configuration and cannot be created or modified by role agents.
- The department agent validates all incoming pushes against the channel schema before processing.
- All pushes are logged with sender, channel, content hash, and timestamp.

The distinction from a query is structural: a push deposits information and returns no response. The role agent learns nothing new from the act of pushing. This prevents push channels from being used as covert query mechanisms. See Section 8.2 for abuse analysis.

Rule 3: Lateral communication is mediated, and role-binding shrinks it. Because all occupants of a role share one agent, intra-role peer communication — the old LUNA-to-ATLAS problem — no longer exists as an inter-agent flow; it is internal to a single shared agent. The remaining lateral flow is *cross-role* (sales rep agent to field engineer agent), and role-binding collapses it to one stable channel per function pair rather than an N×M mesh of personal agents. Cross-role communication still routes through the department/topology layer, which applies clearance filtering before forwarding.

Rule 4: Structured data sharing creates approved channels. Pre-approved data sharing patterns (e.g., the workload-division map or a cross-role handoff schema) bypass mediation for specific, predefined data types defined in the department agent's configuration.

3.4 The Department Agent as Coordinator

The department agent is simultaneously:

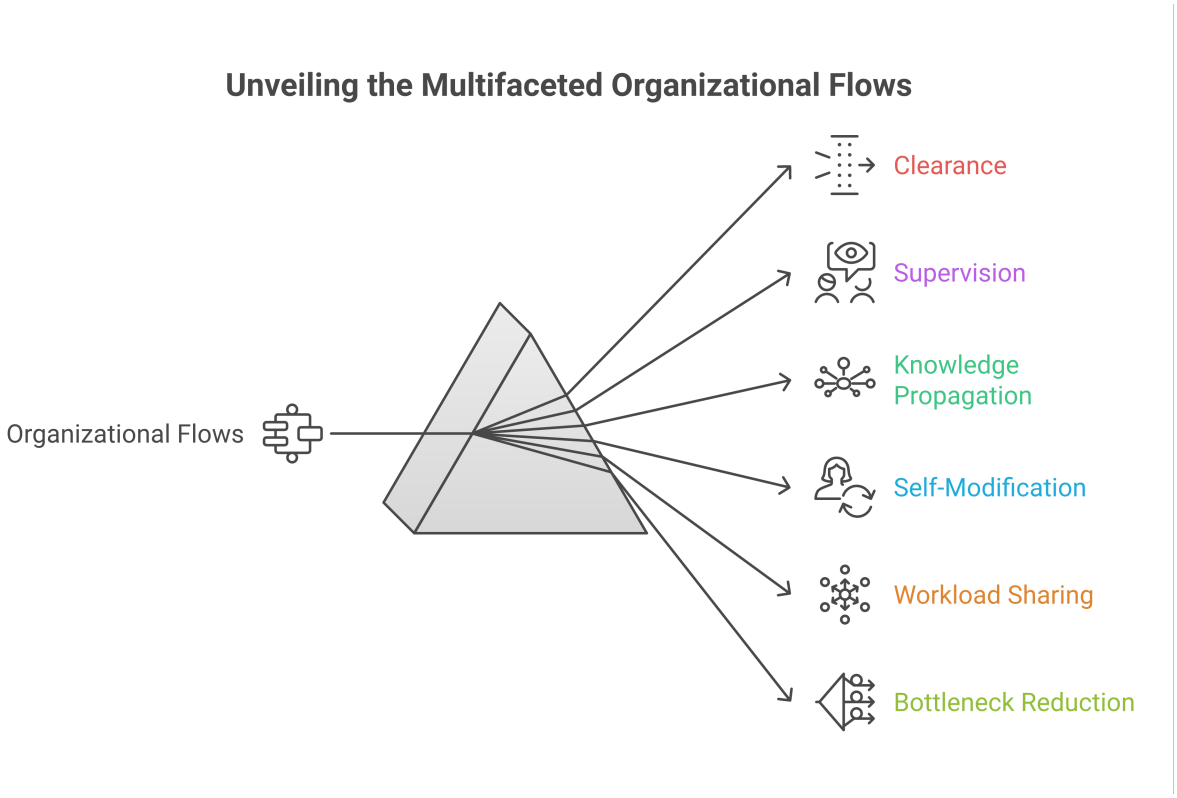

Figure 2. Figure 3. The whole-paper synthesis: one agent network, six axes. The same org chart of role-bound agents (L1 role agents shared across occupants, under L2 department agents, under L3-EXEC) carries six orthogonal flows, each drawn as a differently-colored path — clearance (red, downward), supervision (purple), knowledge propagation (green, from the commons to every occupant at once), self-modification (blue, users → L2-IT → rollout), workload sharing (orange), and bottleneck reduction (teal, one shared agent serving many humans). A mechanism that serves one axis often taxes another; naming the axes makes the trade-offs explicit.

- **A supervisor.** Reads, writes, and audits role agent memory files. Can modify context, correct errors, inject directives.
- **An aggregator.** Synthesizes cross-employee information for department-level queries. What is the total pipeline value for Q2? Which customers have open complaints?
- **A gatekeeper.** All cross-agent and cross-level communication routes through it. Applies clearance filtering before forwarding.
- **A memory keeper.** Maintains institutional knowledge transcending any individual employee.

The department agent runs on a persistent schedule with hooks into communications infrastructure. It monitors data streams and proactively updates role agents' context when relevant events occur.

3.5 The Agent Network Along Multiple Axes

It is tempting to read HIVEMIND as a single clearance hierarchy, because the level structure (L1/L2/L3) and the security model dominate the early sections. But clearance is only *one projection* of the network. The same set of agents and channels is better understood as one graph viewed along several orthogonal axes, each answering a different organizational question and each carrying its own mechanisms. Most design mistakes come from optimizing one axis while silently degrading another; naming the axes makes the trade-offs explicit.

Axis 1 — Clearance (confidentiality): *who may know what.* The axis the paper develops most fully (Section 2). Information flows downward only under controlled declassification; the Bell-LaPadula lattice, filesystem permissions, and redaction pipeline are its mechanisms. Direction: restrictive, downward.

Axis 2 — Supervision (accountability): *who may inspect whom.* A manager — through the department agent — can read a subordinate role agent's memory, outputs, and audit trail. This runs in the *opposite* direction to clearance: oversight reaches *down* the org chart into what subordinates' agents are doing. The distinction matters and is easy to blur: supervision grants read access to an agent's *memory and behavior*, not an elevation of the subordinate's *data* clearance. A manager can see that the sales agent answered a pricing question; that does not give the sales agent the right to see the manager's margin data. Mechanisms: read-only group access to lower-tier workspaces (the `-ro` groups of Section 7.4), the audit store, and the nightly sweep. Tension: supervision collides with the trust deficit (Section 5.2) — too much visible monitoring and people self-censor. The design choice is to monitor *boundaries*, not *performance*.

Axis 3 — Bottleneck reduction (throughput): *where work piles up on one node.* In a human-only company, the senior expert is a chokepoint — every "how do we handle X?" routes to the one person who remembers. The shared role agent dissolves this: the institutional answer is held by the agent and available to every occupant simultaneously, so routine knowledge queries no longer queue behind a single human. The department agent further reduces bottlenecks by answering aggregate questions ("total Q2 pipeline?") without convening twelve people. Mechanism: institutional memory as a shared, always-available resource rather than a scarce human.

Axis 4 — Workload sharing (load balancing): *how work is diverted dynamically.* Because all occupants of a role share one agent, the agent has a network-level view of who is busy, who is idle, and who is best suited to a task. Work can be diverted on an *objective* evaluation — current load, relevant expertise, account ownership, response-time SLAs — rather than landing wherever it first arrived. A lead that arrives for an overloaded rep can be surfaced to a colleague with capacity; a support ticket can be routed to the agent-occupant whose history shows the fastest resolution on that issue type. This requires an objective scoring function (and guardrails against gaming it); the role agent is the natural place to compute it, because it already sees the whole role's state. Tension: workload sharing pushes toward *wider* visibility of each other's queues, which must be reconciled with Axis 1's scoping (Section 11's intra-team-visibility limitation).

Axis 5 — Self-modification (evolution): *how the workforce gets faster over time.* The most distinctive axis. A HIVEMIND deployment is not static — its agents can grow new tools, skills, plugins, and recipes that compound the workforce's speed. But capability growth is governed by the state-of-the-art principle of *division of labor*: **only IT (`L2-IT`) may upgrade, rebuild, or restart the gateway.** End users do not self-modify their infrastructure (Section 7.11). What makes this fluent rather than a bottleneck is a structured improvement channel — detailed in Section 7.11 — designed to be *ingested by IT's own agent*: end users file richly-specified feature requests, `L2-IT` triages the queue, implements (increasingly by vibe-coding against its agent) and functionality-tests, then rolls the improvement out to the shared role agent so every occupant gets it at once. The evolution axis is therefore a controlled loop: distributed sensing of what to improve, centralized authority to implement.

Axis 6 — Knowledge propagation (learning diffusion): *how fast a new fact reaches everyone who needs it.* Distinct from clearance (which is about *restriction*); this axis is about *speed of spread*. When a competitor's move, a regulatory change, or a new product detail enters the system, how long until every relevant agent acts on it? Role-binding makes this near-instant within a role — one update to the shared agent or its commons slice reaches all occupants immediately — and the steward/cascade model (Sections 2.7, 5.9) governs propagation across roles. A company's competitiveness increasingly depends on this axis: the firm whose

knowledge diffuses in hours out-executes the firm whose knowledge diffuses in quarters.

These axes are orthogonal and frequently in tension: clearance restricts, supervision inspects, bottleneck-reduction and workload-sharing widen access, evolution centralizes authority, propagation maximizes spread. A mechanism that serves one axis often taxes another. The contribution of naming them is that every design decision in the rest of this paper can be checked against the question: *which axis does this serve, and which does it cost?*

3.6 Cross-Role Retrieval — Memory Across the Swarm

Role-binding dissolves the hardest version of inter-agent retrieval. Because all occupants of a role share one agent, retrieval *within* a role is internal: when the sales agent assembles a González briefing, every salesperson's touchpoints are already in its own shared memory (HIPPOCAMPUS index over the role's store) — no cross-agent hop is needed. This is a direct consequence of the continuity thesis and a large simplification over a per-person design, where the same briefing required querying twelve separate indexes.

What remains is *cross-role* retrieval — when the sales agent needs context that lives in the field engineer agent's memory (an installation history, a support escalation). HIVEMIND's answer is that **cross-role retrieval is never direct; it is mediated and clearance-filtered by the department or executive layer.** Three properties make this tractable:

1. **Indexes are federated, not merged.** There is no global index any agent can search. Each role agent keeps its own. The department/executive layer holds a *catalog* — for each role index, the topics it covers and each topic's classification ceiling — without holding the underlying segments. The catalog is enough to route a query ("who has material on this account?") without itself being a leak.
2. **Retrieval is a query, not a grant.** When the sales agent needs cross-role context, it issues a request upward; the mediating agent fans it out to the role indexes its catalog flags as relevant, applies the same clearance filtering and template-based redaction as any downward flow (Section 4.4), and returns a synthesized package. The requesting agent never receives raw access to another role's index. The result respects the requester's clearance, not the source's.
3. **Provenance survives the hop.** Each retrieved segment carries its source role, classification, and content hash into the audit log, so a compliance sweep can reconstruct exactly which memory crossed which boundary. Cross-role retrieval is therefore among the most heavily logged operations in the system — appropriately, since it is where one role's knowledge becomes another's.

This makes the swarm's collective memory queryable without making it flat: an agent can draw on the whole organization's history, but only through a gate that enforces who is entitled to what.

3.7 Concurrent Action Under Arbitration

The retrieval machinery of Section 3.6 mediates concurrent *reads*. It says nothing about concurrent *writes*, yet the swarm's central substrate — the shared corporate filesystem of Section 2 — is mutable, and agents at the same tier will at times act on the *same* object at the same time. Two department agents may both append to the competitive-intelligence channel of Section 4.5; a steward and a sweep may touch the same commons file. The information plane has a worked-out safety story; the action plane needs the matching one. Mediating who may *know* what is only half the design if nothing arbitrates who may *change* what, concurrently.

The problem, stated cleanly. When N agents act concurrently on one shared store, three failure modes appear:

1. **Collision** — two agents want the same object at the same instant and nothing arbitrates, so one silently overwrites the other.
2. **Commit hazard** — an agent tries to record only its own change, but a coarse-grained "stage everything" gate sweeps in a peer's half-finished work, attributing it to the wrong actor and corrupting the audit trail.
3. **Manual conflict analysis** — a human is forced to reason, after the fact, about which tasks were independent (disjoint objects) and which had to serialize (a shared object), and to hand-stitch the results.

These are the action-plane analogue of the information-plane hazards the rest of the paper addresses (leakage, coordination failure, silo): the same need for *coordination under a protocol rather than a human's manual intervention*, applied to mutation instead of disclosure.

The mechanism: prepare outside the lock, then short-lived per-object leases. A coordination protocol — instantiated as a working orchestrator on the OpenClaw/TinkerClaw filesystem, here named the arbitration layer — splits every worker into two phases.

- **Phase A (no lease, fully parallel).** Each worker reads, diagnoses, and drafts an *exact* change to the object(s) it will touch. This is the bulk of the wall-clock — roughly 95% — and it runs with zero contention, because reading never conflicts (the same reason read-only fan-out parallelizes cleanly in Section 3.6).
- **Phase B (brief lease, serial per object).** A worker acquires a lease on *only* the object(s) its prepared change touches, applies the pre-computed edit, verifies it, and releases — a matter of seconds. Disjoint objects therefore proceed concurrently; only genuine shared-object contention serializes, and even then the object "passes hands" the instant an edit lands. If the base shifted while a worker waited, a staleness guard makes it re-derive its change against the new state rather than apply a stale one.

The elegant consequence the design names: because every write to a given object is serialized through that object's lease against one shared state, *nothing ever diverges — so "clean merge" becomes a non-event; there is nothing to merge.* This is the write-side counterpart to the §3.6 "provenance survives the hop" property: each applied change is recorded as its own atomic, attributable commit — staging *only* that unit's objects, never a coarse "stage everything" — so the audit substrate of Section 6 gets a clean, per-actor write history for free.

The stronger variant: isolation-per-agent with merge-back. To defend against a *peer outside the protocol* — a second, uncoordinated session dirtying the shared tree — each disjoint unit can prepare and apply on its own isolated checkout branched off a clean checkpoint, with overlapping units serialized onto one shared checkout; a final integration phase fast-forwards or cherry-picks the branches back, and any foreign uncommitted work on a contended object is stashed and restored around the landing so it is neither swept in nor reverted. The guarantee this buys — *a non-participating agent's work-in-progress is never touched, and every worker starts from a clean base* — is the same architectural move as the per-principal isolation that separate Linux users provide for clearance (Section 7.5), applied to *write* isolation instead of *read* isolation. Per-principal isolation enforced below the agent's reasoning, on two different planes.

Why this belongs in the swarm design. It is the operational realization of an assumption the architecture otherwise leaves implicit: that department- and executive-tier agents will sometimes act on the same shared corporate state concurrently. The clearance lattice tells them what they may touch; the arbitration layer tells them how to touch it at the same time without a human reconciling the result. It instantiates the design's first principle — that *concurrency is a feature made safe by protocol, not a hazard to be avoided* — with a mechanism rather than an assertion, and it generalizes well beyond code: the shared filesystem of Section 2 is exactly the shared mutable state the protocol governs.

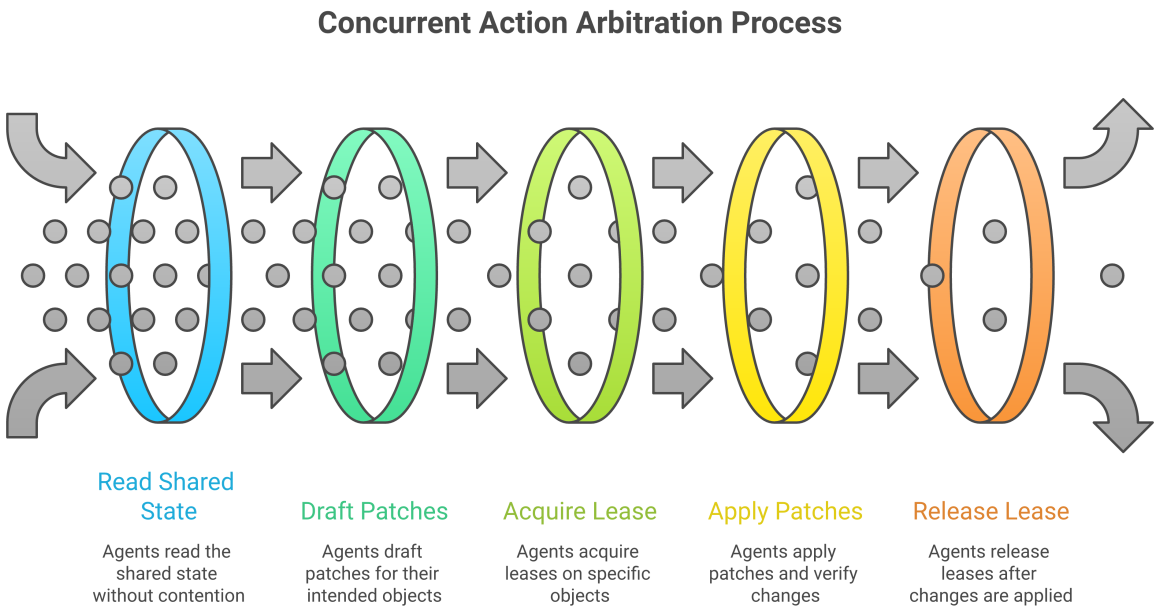

Figure 3. Figure 4. Concurrent action under arbitration. Phase A (left): four workers each read the shared state and draft an exact patch fully in parallel, no lease held — ~95% of wall-clock. Phase B (right): workers acquire a short-lived lease on only the object(s) their patch touches and apply; disjoint objects (1, 2, 4) land concurrently while the two workers contending for object 3 serialize through its lease, the object passing hands the instant an edit lands. Each applied unit is committed as its own attributable change. Because every write to an object is serialized through its lease against one shared state, the streams never diverge — "clean merge" is a non-event.

4. The Sales Coordination Case Study

4.1 Scenario Setup

The company is a mid-sized industrial supply distributor, family-owned, 85 employees. The sales department has 12 salespeople in three regional teams. Data architecture: 2,400 active CRM accounts, 8 years of email archive, call logs, pending offers, negotiation histories, and customer-specific pricing arrangements.

There is **one sales rep agent**, shared by all twelve salespeople. María García (senior rep, 7 years, currently covering the northern territory including González Industrial) and Pedro Martínez (mid-level rep, 3 years, central territory) both interact with the *same* agent. The agent holds the entire team's relationship history natively — it does not have to assemble it from twelve separate memories, because it *is* the single memory. A `workload-division.md` records who currently covers which territory and accounts. Above the role agent sit the **L2-SALES** agent (T4) and **L3-EXEC** (T5).

This is the structural simplification role-binding buys: most of what the original design solved with inter-agent routing simply does not arise.

4.2 Customer Ownership, Resolved Contextually

There is no Customer Ownership *Registry* synchronized across agents and no email interceptor, because there is only one agent. Ownership is resolved contextually: the agent knows which human it is currently serving and reads `workload-division.md` to know who covers what.

Pedro forwards the agent an email from a González contact:

1. The agent recognizes it is serving Pedro.
2. It reads from the workload map that González is in María's territory.
3. It responds to Pedro: "González Industrial is currently María's account. I'll flag this for her. Here's the current status so you can reply on the logistics question, but please defer to her on any pricing commitment."
4. It surfaces the same item to María the next time she engages (or via a notification, if enabled).

Same outcome as the original cross-ownership alert — but with no inter-agent message, no synchronized registry, and no mailbox interceptor. The "two agents, same customer, divergent responses" failure mode is structurally impossible: there is one agent, with one consistent view.

4.3 Unified Customer History — Now Trivial

When María prepares for a González call, she needs the complete relationship history across the whole team. In the original per-person design this required a cross-agent query-and-assemble dance. With a shared role agent it is trivial:

1. María asks the agent: "Prepare a briefing for the González account for my call tomorrow."
2. The agent already holds every salesperson's González touchpoints in its shared memory — no cross-agent request is needed.
3. The only filtering that still applies is *upward* tier filtering for data the sales role does not clear:
    - Sales interactions (T3): [included] included in full.
    - Customer-specific pricing tiers negotiated by management (T4): [filtered] replaced with a **Redacted Summary** (Section 4.4), delivered via controlled declassification from L2-SALES.

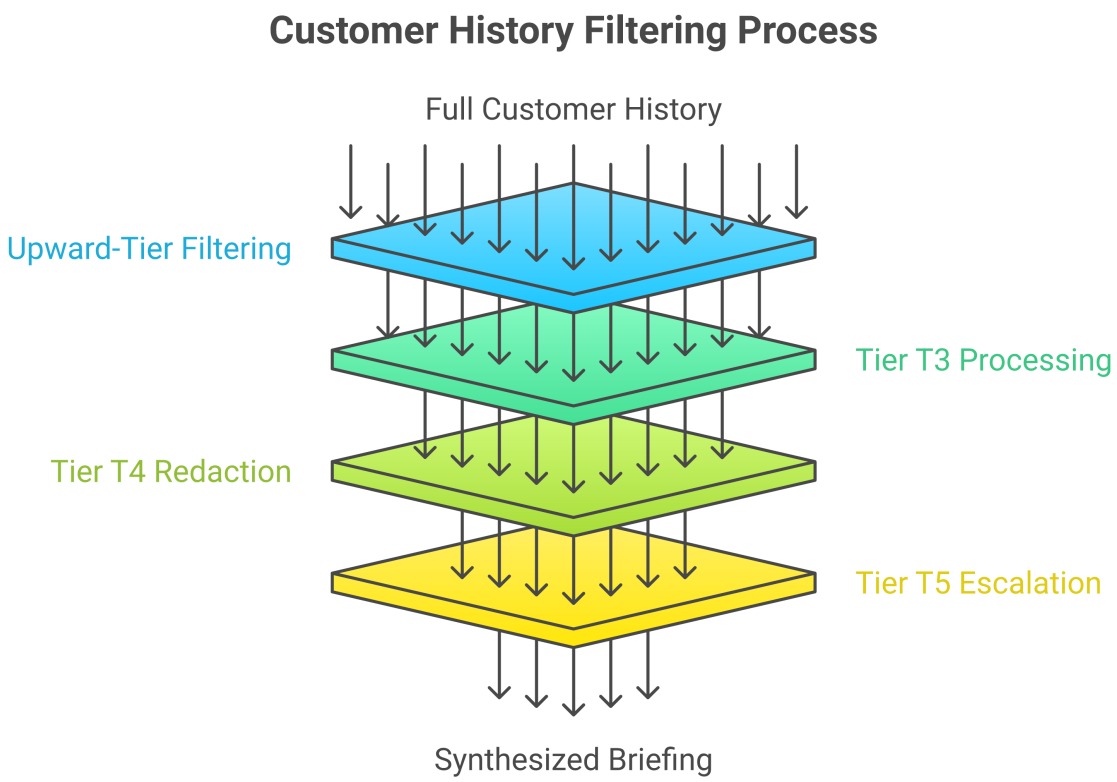

Figure 4. Figure 2. Contextual history with upward-tier filtering, under role-binding. María asks the one shared L1-SALES-REP agent, which already holds the whole team's history natively — no cross-agent fan-out. Only data above the sales role's clearance is filtered: T3 interactions pass verbatim, T4 pricing authority is replaced by a redaction template, and T5 legal matter becomes a caution notice. The result respects the requester's clearance, not the source's, and every declassified segment is logged with classification and hash.

- Legal proceedings involving this customer (T5): [filtered] replaced with a caution notice: "Active legal matter. Escalate any contract discussions to management before committing."

4. The agent synthesizes the briefing.

María walks into the call with the whole team's institutional knowledge, without accessing data above the sales role's clearance. Note the boundary that *moved*: ownership (María vs. Pedro) is now an operational overlay, not a clearance wall — both reps share T3-sales clearance and the agent could surface either's customer history. The hard secrecy boundary remains where it belongs, between the sales role (T3) and management data (T4/T5). The trade-off this implies for intra-team confidentiality is discussed in Section 11.

4.4 Redacted Summaries and Leakage Control

The redacted summary is a critical mechanism — and the single most vulnerable point in the architecture. When a higher-clearance agent synthesizes information for a lower-clearance recipient, there is an inherent risk that the LLM leaks restricted content through phrasing, implication, or context.

The risk: An LLM told to "summarize this pricing negotiation without revealing the specific margins" might produce: "The CFO has authorized significant flexibility on pricing for this account, particularly if they commit to volume." This technically redacts the number but reveals the existence and direction of the authorization — information the salesperson should not have.

Template-based redaction pipeline. To mitigate this, HIVEMIND uses a structured redaction approach rather than relying on free-form LLM summarization:

1. **Classification and tagging.** L2-SALES identifies restricted elements in the source data and tags them by category (pricing authority, legal status, margin data, personnel information).

2. **Template selection.** Each category maps to a pre-approved redaction template:

   - Pricing authority → "Pricing for this account is managed by [ROLE]. Refer pricing inquiries to [ESCALATION_CONTACT]."
   - Legal status → "There is an active matter affecting this account. Escalate contract discussions to management before committing."
   - Margin data → "Account profitability is under management review. Standard pricing applies unless otherwise directed."
   - Personnel information → [Omitted entirely; no template generated.]

3. **Template population.** The templates are filled with non-sensitive parameters only (role titles, contact names, standard procedures). No LLM generation is involved in the redacted output itself.

4. **Fallback to human review.** If the restricted content does not fit any pre-approved template, the redaction request is queued for human review rather than generating a free-form summary. The requesting agent receives a holding response: "Additional context for this account requires management approval. Request submitted."

5. **Logging.** Every redaction event is logged: source classification, template used, output hash, recipient clearance. Logged redactions are reviewed in weekly compliance audits.

This template approach sacrifices some contextual richness — a human manager could write a more nuanced briefing — but it eliminates the information-theoretic leakage risk of LLM-generated summaries. The templates are authored by management and compliance, not by the LLM.

When is free-form summarization acceptable? Only for T2→T1 declassification (internal to public), where the sensitivity gap is minimal and the risk of meaningful leakage is low. All T3+ declassification uses the template pipeline.

4.4.1 The Doubt Reviewer — A Fresh-Context Leakage Oracle

The template pipeline defends against leakage by *construction* — no LLM generates the redacted text, so none can be coaxed into over-disclosing. But it has a residual failure mode §4.4 itself names and the §8.2 clearance validator cannot catch: **implication-leakage**, where the output contains no restricted token yet still reveals the existence and direction of an authorization. The validator inspects for *known restricted formats* (salary ranges, credential patterns, document IDs); a sentence like "pricing on this account is unusually flexible right now" carries none of those and sails through, while leaking exactly the fact the recipient should not have. A format-aware checker is structurally blind to inference-leakage, because the leak is in what the text *lets you conclude*, not in what it *contains*.

HIVEMIND closes this gap with a mechanism adapted from the doubt-driven-development loop of addyosmani/agent-skills (Section 1.6): a **doubt reviewer** — a fresh-context adversarial

agent whose decisive property is *ignorance of the secret.* After the template pipeline produces a redacted package, and before delivery, the package — and *only* the package — is handed to a reviewer agent that has **no clearance to the source**, never sees the originating reasoning, and is asked one adversarial question: *"From this text alone, what can you infer about restricted facts — the existence, direction, or magnitude of any pricing authority, legal matter, margin position, or personnel decision affecting this account?"*

Why this is a *sound* leakage test rather than another quality check is the clearance boundary it runs across:

- **The reviewer's threat model is the recipient's threat model.** The recipient (the sales agent, T3) also has no access to the T4/T5 source. So anything the clearance-blind doubt reviewer infers from the package alone is, by construction, something the recipient could infer too — a leak. Anything it cannot infer is safe to deliver. This inverts the §8.2 validator, which *knows* the restricted formats and hunts for them; the doubt reviewer is *ignorant* of the secret and probes for inference, the only way to surface implication-leakage.
- **The STOP criterion is the existing human-review escalation, not improvisation.** If the doubt reviewer recovers a forbidden inference, the disposition is not to silently soften the wording in a loop until it goes quiet — that risks an arms race between two models with no ground truth. The template authors are given the finding; if the template can be tightened it is, otherwise the request halts to human review via `ERR_HUMAN_REVIEW_REQUIRED` (Section 7.3.5). Never improvise a best-effort partial summary; when in doubt, escalate.

The honest scope of the borrowing: addyosmani's loop is an engineering-discipline mechanism for code review with a fully-specified reconcile/stop machinery HIVEMIND does not reproduce. HIVEMIND takes only the fresh-context-reviewer-with-stripped-context primitive and re-purposes it as an *information-flow* oracle by running it across a clearance boundary — turning a quality check into a leakage test. It is the runtime counterpart to the offline redaction-fidelity eval of Section 4.4.2: the eval measures residual leakage over a seeded corpus before deployment; the doubt reviewer measures it per-package at delivery time. One cost: it is an extra LLM call near the capability ceiling of a local model (Section 5.6), so it should gate only T3+ declassifications where the pipeline produced a non-trivial output, and it must fail *closed* — a reviewer that errors or times out routes the package to human review, never auto-delivers.

4.4.2 A Redaction-Fidelity Eval — Measuring Residual Leakage Directly

headroom (Section 1.6) demonstrates that an agent-memory transform can ship with a reproducible benchmark; HIVEMIND should hold its declassification transforms to the analogous standard. The quantity to measure is *not* task accuracy — the redaction pipeline's job is the opposite of fidelity-preservation — but **residual leakage**: how much a reader can recover about the restricted source from the redacted output alone. We specify a seeded-corpus eval with three parts:

1. **A seeded leakage corpus.** Synthetic source records spanning the §4.4 template categories (pricing authority, legal status, margin data, personnel information), each paired with the restricted fact it must conceal and a set of "forbidden inferences" (existence, direction, magnitude of an authorization; identity of a party to a legal matter).
2. **The measured quantity — residual mutual information.** For each category, estimate the mutual information between the redacted output and the restricted fact, operationalized via the fresh-context inference test of Section 4.4.1: the fraction of forbidden inferences a clearance-blind reviewer recovers. A perfect template scores zero; the §4.4 worked failure ("the CFO has authorized significant flexibility…") scores high on the *direction* inference even though it leaks no restricted token.

3. **Acceptance thresholds per category.** Each template gets a published residual-leakage ceiling; a template that exceeds it is rejected and returned to the compliance authors before it can enter the pipeline.

This is the artifact Limitations 2 and 10 (Section 11) call for, narrowed to the property HIVEMIND actually cares about. It does not establish information-theoretic *bounds* — that remains open — but it replaces an untested claim of leakage control with a measured one, and makes redaction templates regression-testable after every model or rule change.

4.5 Competitive Intelligence Flow

HIVEMIND defines structured **upward push channels** (formalized in Rule 2a, Section 3.3) for intelligence that has organizational value beyond the individual agent.

The sales rep agent, after Pedro's meeting, logs:

```
L1-SALES-REP → L2-SALES (upward push, channel: competitive_intelligence):
  type: competitive_intelligence
  source: Pedro Martínez, client meeting 2026-03-17
  content: Competitor X announced Product Y, delivery Q3 2026
  confidence: medium (hearsay from client)
  classification: T2/INTERNAL
```

L2-SALES validates the push against the channel schema (required fields present, classification within T2 ceiling for this channel), adds it to its competitive intelligence aggregation, and forwards a synthesized report upward to L3-EXEC at its next scheduled sync. Because the rep agent is shared, the intelligence is immediately part of the knowledge every salesperson's interactions draw on — the silo of Section 1.1 cannot form at the role level.

Competitive intelligence does **not** flow laterally by default. This is deliberate: lateral intelligence sharing creates gossip networks and can cause coordinated behavior that amplifies intelligence into premature strategic commitments. Aggregation happens at the department level; decisions about action are made by management.

4.5.1 Structured Channel Schema and Confidence Lifecycle

For practical deployment, upward push channels need tighter operational specification than a prose description alone. Each approved channel should define:

1. **Field schema.** Required fields, types, enumerations, and size limits.
2. **Classification ceiling.** Maximum allowable classification for submissions on that channel.
3. **Confidence vocabulary.** A controlled set such as `low | medium | high | verified` to avoid free-form confidence descriptions.
4. **Disposition rules.** Whether the push is stored, queued for human review, aggregated immediately, or discarded after a retention window.
5. **Escalation triggers.** Conditions under which the department agent must create a managerial alert rather than simply store the information.

An illustrative schema for `competitive_intelligence`:

```
{
  "type": "competitive_intelligence",
  "source_employee": "string",
  "event_date": "YYYY-MM-DD",
  "customer_context": "optional string",
```

```
  "competitor": "string",
  "claim": "string, max 500 chars",
  "confidence": "low|medium|high|verified",
  "evidence_type": "hearsay|email|attachment|public-web|meeting-note",
  "classification": "T1|T2",
  "requires_followup": true
}
```

The confidence lifecycle should also be explicit. Information enters as a claim, may later be corroborated, and can eventually be superseded. Department agents therefore maintain not just a fact store but a **claim registry** with state transitions such as `submitted -> corroborated -> operationalized -> archived` or `submitted -> contradicted -> retired`. This distinction prevents weak signals from calcifying into institutional memory simply because they were logged once.

4.6 The Coordination Dividend

The Sales Coordination Model illustrates HIVEMIND's value proposition: the whole is greater than the sum of its parts, and the boundaries make this possible rather than preventing it.

Without boundaries, a single omniscient sales AI would be dangerous — leaking, creating unfair information asymmetries, and resisting auditability. With strict individual agents and no coordination, you have 12 isolated assistants without institutional memory. HIVEMIND's mediated coordination captures most of the integration value while enforcing boundaries that make enterprise deployment tenable.

5. Privacy and Legal Considerations

Legal requirements motivate the compliance architecture that follows in Section 6. We address them first.

5.1 Employee Awareness and Transparency

Employees must know their agents are monitored. This is a legal requirement in most jurisdictions (EU Working Time Directive, national employment law in Spain, GDPR Article 88).

The employment contract addendum must clearly state:

1. The employee's AI agent is operated by the company on company infrastructure.
2. All interactions with the agent are logged.
3. The agent's memory files are readable by department supervisors and executive agents.
4. The agent is subject to nightly compliance sweeps.
5. The agent cannot be configured to conceal information from the compliance system.

This is analogous to existing monitoring clauses for company email and devices. The critical difference: the AI agent may feel more personal — employees may share things with their agent they wouldn't type in a work email. The transparency clause must make clear that intimacy is not privacy.

5.2 User Adoption and Change Management

Legal transparency is necessary but insufficient. Enterprise AI adoption literature (Brynjolfsson & McAfee, 2014; Davenport & Ronanki, 2018) consistently shows that organizational change management is often harder than the technology itself.

Anticipated adoption challenges:

- **Trust deficit.** Salespeople knowing their agent is monitored by management's agent may self-censor, sharing less with their assistant and defeating the purpose. Mitigation: clear communication that compliance monitoring targets data boundary violations, not employee performance evaluation. The audit system flags clearance breaches, not whether María complained about her workload to the sales agent.
- **Resistance to coordination.** Salespeople accustomed to "owning" their client relationships may resist a system where a department agent aggregates their intelligence. Mitigation: demonstrate value early — the first time a salesperson walks into a meeting with a comprehensive briefing they didn't have to assemble, resistance decreases.
- **Over-reliance.** Employees may delegate judgment to their agents. The agent says "refer pricing to management" — does the salesperson understand *why*, or just comply? Mitigation: agent outputs should explain reasoning, not just issue directives. Training should emphasize the agent as an augmentation tool, not an authority.

Adoption strategy: See Section 9 (Phased Deployment) for a staged rollout that addresses these challenges incrementally.

5.3 GDPR Implications

Lawful basis for processing. Agent-to-agent sharing of personal data requires a lawful basis. For customer data: legitimate interest or contract performance. For employee data processed by the HR agent: employment contract.

Data minimization. Agents should not accumulate personal data beyond operational necessity. Memory files are pruned regularly of data exceeding retention periods. The nightly sweep includes a data minimization pass.

Purpose limitation. Customer data shared with L2-SALES for account coordination cannot be repurposed for marketing analytics without a separate lawful basis and disclosure. Audit logs provide the evidence trail.

Data Processing Agreements (DPAs). If any LLM inference is routed to cloud providers (OpenAI, Anthropic, Google), a DPA compliant with GDPR Articles 28-29 must be in place. The DPA must specify: data categories processed, processing purposes, sub-processor list, data deletion obligations, and breach notification procedures. For on-premises inference (recommended — see Section 5.6), DPAs with API providers are not required, but DPAs with hardware vendors providing managed services may still apply.

Article 22: Automated decision-making. If HIVEMIND agents make or substantially contribute to decisions affecting employees (e.g., quarantine suspending an agent disrupts the employee's work), Article 22 requires: (a) informing the employee that automated processing is involved, (b) providing meaningful information about the logic, and (c) ensuring the right to obtain human intervention. The quarantine process (Section 6.4) includes human-in-the-loop review specifically to satisfy this requirement.

Article 4(4): Profiling. Agent memory files that accumulate behavioral patterns about employees (work habits, communication style, client relationship quality) may constitute profiling. The DPIA (Section 5.4) must assess this and define retention limits for behavioral data.

Cross-border transfers. Every message sent to a cloud LLM API constitutes a potential cross-border data transfer. This is one of the strongest arguments for on-premises inference.

5.4 Data Protection Impact Assessment

A multi-agent system processing employee and customer data at scale requires a DPIA under GDPR Article 35. The DPIA must be completed before deployment and should assess:

- Data categories processed by each agent tier
- Risks to data subjects (employees, customers) from memory accumulation, inter-agent sharing, and potential breaches
- Necessity and proportionality of each data processing activity
- Safeguards: clearance model, encryption, audit system, human oversight mechanisms
- Profiling assessment: whether agent memory constitutes profiling under Article 4(4)

5.5 The EU AI Act

The EU AI Act (Regulation 2024/1689) introduces obligations that HIVEMIND must address. A multi-agent system that monitors employee communications and makes automated quarantine decisions likely falls under **high-risk AI system** classification (Annex III, Category 4: employment, workers management, and access to self-employment).

Obligations for high-risk classification:

- **Risk management system (Article 9).** HIVEMIND's risk analysis (Section 8) and compliance architecture (Section 6) serve this function but must be formalized as a continuous risk management process, not a one-time assessment.
- **Data governance (Article 10).** Training data, validation data, and operational data must meet quality criteria. For HIVEMIND, this primarily concerns the data used to train the compliance classifiers and semantic scanners.
- **Technical documentation (Article 11).** The system's design, purpose, capabilities, and limitations must be documented for regulatory inspection. This paper serves as a starting point but would need expansion.
- **Record-keeping (Article 12).** Automatic logging of system events — HIVEMIND's communication logs and audit trails satisfy this if configured for the required retention period (the Act references "appropriate to the intended purpose," typically interpreted as the system's operational lifetime plus a regulatory buffer).
- **Transparency (Article 13).** Users (employees) must be informed they are interacting with an AI system and understand its capabilities and limitations.
- **Human oversight (Article 14).** The system must be designed to allow effective human oversight. HIVEMIND's human-in-the-loop quarantine review, weekly audit reports, and the L3-EXEC dead switch address this requirement.
- **Conformity assessment.** Before deployment, HIVEMIND must undergo a conformity assessment (self-assessment for most Annex III systems, third-party for biometric systems). This should be factored into the deployment timeline.

5.6 Data Residency and On-Premises Processing

For a family business handling customer data under GDPR, the recommendation is: all LLM inference should be on-premises.

- On-premises server running open-weight models (LLaMA 3, Mistral, Qwen, or equivalent).
- OpenClaw configured with `local_inference: true`, routing to Ollama or similar local inference server.
- No customer data, employee data, or corporate correspondence transmitted to cloud APIs.

Capability gap acknowledgment. On-premises models (currently 70B parameter class on single-server GPU deployments) have measurable capability gaps compared to frontier cloud models for complex reasoning, nuanced language generation, and rare-domain knowledge. For most enterprise agent tasks — email drafts, calendar management, CRM queries, structured data retrieval — 70B models are adequate. However, the compliance classifiers (Section 6.3) and the template-based redaction pipeline (Section 4.4) should be validated against the specific

local model's capabilities before deployment. Tasks requiring frontier-class reasoning (complex legal analysis, multi-factor strategic synthesis) should be flagged for human handling rather than delegated to a local model operating near its capability ceiling.

5.7 The Right to Be Forgotten — Offboarding as a Non-Event

Role-binding largely dissolves the offboarding problem that a per-person design creates. There is no personal agent to destroy, no institutional knowledge to extract and migrate, and no workspace to scrub — the agent and its accumulated knowledge belong to the *role*, not the departing person. The customer histories, negotiation context, and procedural know-how the original design had to rescue from a leaving employee's agent never lived in a personal silo in the first place.

When an employee leaves:

1. **Reassign the workload.** The departing person's entries in `workload-division.md` are reassigned (to a successor or temporarily to the department manager). This is the substantive step, and it is one edit.
2. **Revoke the human's access.** The person's authentication to interact with the role agent is disabled. The agent continues serving everyone else unchanged.
3. **Handle the thin per-person layer.** The only person-specific data is the lightweight interaction profile (Section 2.7) — name, tenure, communication preferences. This is the sole GDPR right-to-be-forgotten surface, and it is small and clearly bounded: the employee may review it and request deletion, after which it is purged.
4. **Preserve audit trails.** Communication logs referencing the person are retained per the regulatory retention period (audit integrity requires this), then purged on schedule.

The contrast is the point. The original design's six-step extract-migrate-scrub-purge protocol existed to fight the talent leak after the fact. Role-binding removes the leak at the source: knowledge never leaves, because it was never bound to the person who is leaving.

5.8 The IT Staff Privilege Clause

IT staff with root access can, by virtue of their role, read any file on the server. This is unavoidable. The mitigation: compliance audit logs are stored in an immutable, append-only store with cryptographic signing. Tampering is detectable. A separate access log records all privileged access events, reviewed by the CEO or external auditor.

5.9 From Trainers to Knowledge Stewards

The knowledge commons (Section 2.7) reshapes a job rather than eliminating one. Today, learning-and-development staff onboard each hire by hand: the effort is O(employees), re-spent on every departure, lost when the trainer themselves leaves, and inconsistent from one trainer to the next. Re-point that same person at the commons and the economics invert. The steward authors and maintains the role slices, the industry context, the customer model, and the voice standards *once*; the corpus then onboards every agent and every new human hire off the same substrate. Training stops being repetition and starts compounding.

This is a concrete, sellable answer to "what happens to our trainers." They become **knowledge stewards**: owners (per the frontmatter of Section 2.8) of specific commons paths, accountable for accuracy and review cadence, with scoped write access through the Git review workflow. The skill transfers directly — a trainer already knows what a newcomer needs to learn; they now write it down in a form that serves humans and machines at once.

Fossilization and the new-hire research loop. A persistent role agent risks entrenching a departed employee's habits and a single steward's blind spots — "we have always done it

this way," automated and amplified. The counterweight is to make onboarding *bidirectional*. A common and healthy practice for human new hires is to expand on their formal training by researching open questions and asking the experienced people around them. HIVEMIND institutionalizes this: during onboarding, a new hire is tasked with probing the commons for gaps, asking colleagues, and feeding findings back to the relevant steward as proposed commons updates. The newest, least-habituated person — the one most likely to notice what is stale or unexplained — becomes a routine source of refresh. Onboarding thus doubles as a knowledge audit, and the corpus is continuously challenged by exactly the people with the least incentive to defend its assumptions.

5.10 The Future of Work: Who Wins

The steward transformation is a specific instance of a broader and contested question, and it is worth addressing directly because it shapes whether a workforce adopts a system like this or resists it.

The widespread belief is that systems like HIVEMIND destroy jobs: if the AI holds the knowledge and does the routine work, the people become redundant. There is a real basis for the fear — the demand for *low-thinking office work* (re-keying data, assembling the same report, answering the same question for the hundredth time) does progressively narrow as agents absorb it. We do not dismiss that. But the more interesting claim, and the one we argue here, is that the same shift leaves *some* workers materially better off — specifically those whose value is in learning and teaching rather than in repetition.

Consider the economics of anyone who sells their time in person — a trainer, a consultant, a coach. Their income has a hard ceiling: it is bounded by the number of hours they can work multiplied by the maximum rate the market will bear for an hour of their attention. No matter how good they are, the cap is difficult to breach, because the product *is* their time and time does not scale.

The pre-AI escape from that ceiling is familiar: record the training once and sell it to many — an online course platform. This does raise the cap by decoupling revenue from hours, but it commoditizes the work (a one-time sale in a race to the bottom) and the recorded content ages the moment the market moves.

HIVEMIND offers a different and, we argue, better escape for that same consultant. When the AI does the bulk of the repetitive training inside a client company, the consultant's role moves *up* the value chain — from trainer to **company knowledge manager**. The shape of the job changes in four ways, each of which lifts the income ceiling:

1. **No repetition within a company.** They never deliver the same training twice to the same firm; the commons holds it and the agents teach it. The consultant's time is freed from the part that did not scale.
2. **Recurring revenue against fossilization.** The hard problem the AI *cannot* solve alone is staleness (Section 2.8). Knowledge decays; markets, products, and regulations move. Keeping a company's knowledge base current is ongoing work, which justifies a *recurring* fee rather than a one-off engagement — and recurring, retainer-style revenue is structurally more valuable than project revenue.
3. **A premium for a scarce, compounding skill.** Curating and refreshing an organization's institutional knowledge well is hard and high-leverage, so it commands a premium rather than a commodity rate.
4. **Many companies at once.** The same consultant can steward the knowledge bases of *several* client companies in parallel. The AI agents are, in effect, their force multiplier — the "minions" that deliver the teaching in each firm while the consultant does the part only a human does well.

In this picture the consultant's actual day-to-day collapses to what such people tend to enjoy most: continuously learning about new products and market trends, and teaching it *once* — after which it propagates, through the agents, to every company they serve. The repetitive delivery is gone; the learning-and-synthesis core remains and is paid better.

We state this as a perspective, not a forecast — labor-market outcomes depend on far more than one architecture, and the narrowing of routine office work is real and will be painful for some. But the categorical claim "AI eliminates these jobs" is too blunt. For the subset of roles whose value is learning and teaching, a system that removes the repetition and adds leverage can streamline the job into something higher-paid and more enjoyable, not something that disappears. Whether a given organization experiences AI adoption as elimination or as elevation depends heavily on which roles it chooses to invest in transforming.

6. Compliance and Audit Architecture

6.1 The Audit Imperative

Legal counsel, compliance officers, and regulators will ask: How do you know the AI is not leaking sensitive data? How do you detect violations? What is your incident response process? HIVEMIND answers with a comprehensive audit architecture treating every inter-agent communication as a compliance event.

6.2 Communication Logging

Every inter-agent message is logged to a tamper-evident audit store:

```
{
  "timestamp": "2026-03-17T14:32:11Z",
  "sender_agent": "L1-SALES-REP",
  "sender_clearance": "T3",
  "recipient_agent": "L2-SALES",
  "recipient_clearance": "T4",
  "message_type": "upward_push",
  "channel": "competitive_intelligence",
  "content_classification": "T2",
  "content_hash": "sha256:a3f9...",
  "content_preview": "[LOGGED, NOT STORED IN AUDIT DB]",
  "compliance_flags": [],
  "reviewed": false
}
```

Full message content is hashed and stored separately in an encrypted audit store accessible only to the compliance officer and L3-EXEC. The main audit log contains metadata and hashes — enough to detect violations without constituting its own leak.

6.3 Nightly Compliance Sweep

A cron job runs nightly with four analysis passes:

6.3.1 Information Leak Detection

- *Pass 1: Structural scan.* Checks memory files for references to data from restricted directories — file paths, document IDs, or hashes cross-referencing T4/T5 sources.
- *Pass 2: Semantic scan.* An audit model reviews recent memory additions for content resembling restricted data (salary figures, credential-like strings, legal proceedings language).

On semantic scan reliability: This scan is inherently probabilistic and will produce false positives — salary-like numeric patterns in legitimate sales figures, credential-like strings in product codes. Expected false positive rates for structured pattern detection in free-text agent memories are high enough to cause alert fatigue if every match triggers quarantine. Therefore: semantic scan findings are categorized as *advisory* (queued for weekly human review) unless they match high-confidence patterns (exact salary-range formats from the HR schema, known credential formats from the IT password policy). Only high-confidence matches trigger immediate escalation. Advisory findings that accumulate across multiple scans for the same agent are escalated to immediate review.

6.3.2 Compliance Violation Detection

- Personal data from EU subjects logged without need-to-know justification?
- Customer data requests properly logged with purpose codes?
- Data scheduled for deletion (right-to-be-forgotten) still present in memory files?

6.3.3 Behavioral Anomaly Detection

Agent communications are profiled over time. Deviations flagged:

- 10x normal query volume to department agent.
- Queries for customer data outside assigned portfolio without justification.
- Outputs significantly longer than historical norm (potential data exfiltration).

6.3.4 Memory Contamination Check

Restricted data propagates through legitimate channels — a manager mentions a salary figure to a salesperson, the agent transcribes it. The scan targets structured data types (salary-consistent numerics, IP addresses, credential patterns, names from restricted HR roster) in lower-clearance memory files.

Memory files distinguish between agent-generated and human-dictated entries. Human-dictated entries receive elevated scrutiny.

6.3.5 Governance Metrics for Compliance Classifiers

A production deployment should not treat the nightly sweep as a black box. Compliance classifiers and semantic scanners need explicit operating metrics reviewed by humans:

- **Precision by violation class.** Credential exposure detection, salary leakage detection, legal-matter detection, and cross-portfolio query anomalies should each have separate precision targets.
- **Recall on seeded test cases.** The compliance team should maintain a synthetic test corpus of known violations injected into staging memories so recall can be measured after every model or rule update.
- **Time-to-review.** Advisory findings only matter if they are reviewed promptly. A weekly queue that grows without bound is a silent compliance failure.
- **False-positive burden per employee.** Excessive false positives damage trust and adoption; this is an organizational metric, not only a technical one.

A useful operational rule is to separate **detectors** from **dispositions**. The detector raises a flag; a rules engine or human reviewer determines consequence. This keeps the most error-prone component — semantic classification — from directly controlling the harshest operational response.

How to build the detector — a measured caution. There is a tempting but wrong way to build the semantic scanner of Section 6.3.1 and the clearance validator of Section 8.2: train a supervised "is-this-restricted/harmful-content" head on top of a frozen sentence-embedding model. Offline measurement of exactly this configuration (frozen MiniLM embeddings,

a supervised harmfulness head) found it scored *below chance* — an AUROC of roughly 0.29 — because a frozen embedding does not carry "harmfulness" or "restrictedness" as a linearly recoverable supervised target. In the same measurement, two *unsupervised structural* signals over the same embeddings validated strongly: **k-NN novelty** (how far a message sits from the agent's own prior history) scored AUROC ≈ 0.88, and **clause-cosine incongruity** (the internal mismatch between a message and the channel or clearance it claims) scored AUROC ≈ 0.90. The lesson transfers directly: HIVEMIND's compliance classifiers should not be framed as supervised content-restriction heads over frozen embeddings — the framing measured below chance — but should favor *structural* detectors: novelty relative to an agent's own baseline, and incongruity between a message and its declared channel or clearance. This is also exactly the detector/disposition separation just argued (a structural anomaly is a flag, not a verdict), and it gives the governance-metrics argument an empirical anchor rather than a purely architectural one. It partially answers the broader open question of whether current models can reliably perform clearance-aware classification: measurement says *which framing* of that operation works (structural) and which collapses (supervised-over-frozen).

6.4 Automated Quarantine

When a violation exceeds the severity threshold:

1. Agent session suspended, workspace snapshotted.
2. Tools restricted to read-only. No new writes, no external communication.
3. L3-EXEC and compliance officer alerted with full event record.
4. Compromised memory content flagged for human review.
5. Compliance officer disposition: (a) false positive — reinstate; (b) memory contamination — purge specific entries, reinstate; (c) systemic breach — full audit required.

The human-in-the-loop review satisfies GDPR Article 22 requirements for automated decisions affecting employees.

Quarantine thresholds are calibrated: immediate quarantine for definitive credential exposure or T5 data in T3 memory; human review queue for ambiguous cases. This calibration should be reviewed quarterly based on false positive rates.

6.5 Audit Reporting

Weekly reports generated by L3-EXEC for compliance officer and CEO:

- Inter-agent communication volume by department
- Compliance flags raised and dispositions
- Quarantine events
- Redacted data delivery summary
- Anomaly trends (trailing 30-day window)
- Semantic scan advisory findings awaiting review

7. Technical Implementation on OpenClaw/TinkerClaw

7.1 Platform Overview

Each agent is a configured OpenClaw instance with:

- System prompt defining persona, role, clearance context, behavioral rules
- Workspace directory mapping to accessible corporate filesystem paths

- Role-scoped tool set
- Persistent memory (MEMORY.md + daily journals)
- Inter-session communication capability (sessions_send, sessions_spawn)

TinkerClaw's companion app is the employee-facing interface.

7.2 Linux Filesystem Permission Model

```
# System users: one per ROLE agent (not per person), plus dept/exec
useradd -r -s /sbin/nologin agent_salesrep      # shared by all salespeople
useradd -r -s /sbin/nologin agent_fieldengr     # shared by all field engineers
useradd -r -s /sbin/nologin agent_sales         # L2-SALES (department agent)
useradd -r -s /sbin/nologin agent_strategos

# Groups by clearance tier
groupadd clearance_t2  # INTERNAL
groupadd clearance_t3  # CONFIDENTIAL
groupadd clearance_t4  # RESTRICTED
groupadd clearance_t5  # SECRET

# Role agents: T2 + role-specific T3 (clearance attaches to the role)
usermod -aG clearance_t2,clearance_t3_sales agent_salesrep
usermod -aG clearance_t2,clearance_t3_eng   agent_fieldengr

# Department agent: up to T4 for its domain
usermod -aG clearance_t2,clearance_t3_sales,clearance_t4_sales agent_sales

# Executive agent: all tiers
usermod -aG clearance_t2,clearance_t3_sales,clearance_t3_hr,clearance_t4,clearance_t5 agent_stratego

# Directory permissions
chmod 750 /corporate/internal && chown root:clearance_t2 /corporate/internal
chmod 750 /corporate/confidential/sales && chown root:clearance_t3_sales /corporate/confidential/sal
chmod 750 /corporate/restricted && chown root:clearance_t4 /corporate/restricted
chmod 700 /corporate/secret && chown root:clearance_t5 /corporate/secret
```

Each agent process spawns under its system user via `sudo -u agent_salesrep openclaw start`. The kernel enforces access regardless of model behavior.

7.3 Inter-Agent Communication Protocol

OpenClaw's sessions infrastructure is the communication backbone. We specify the protocol at sketch level sufficient for implementation.

7.3.1 Message Format

```
{
  "protocol_version": "hivemind/1.0",
  "message_id": "uuid-v4",
  "timestamp": "ISO-8601",
  "sender": {
    "agent_id": "agent_salesrep",
    "clearance": "T3",
    "department": "sales"
  },
  "recipient": {
```

```
    "agent_id": "agent_sales",
    "clearance": "T4",
    "department": "sales"
  },
  "message_type": "query | response | push | directive | alert",
  "channel": "string (for push messages: channel name from approved list)",
  "correlation_id": "uuid-v4 (links response to original query)",
  "payload": {
    "content_classification": "T1-T5",
    "body": "...",
    "attachments": []
  },
  "ttl_seconds": 300,
  "requires_ack": true
}
```

7.3.2 Delivery Semantics

- **At-most-once delivery** with acknowledgment. Messages are persisted in the gateway queue until acknowledged or TTL expires.
- **Timeout handling.** If no acknowledgment within `ttl_seconds`, the sender receives a `TIMEOUT` response. The sender may retry with a new `message_id` (idempotent; the gateway deduplicates by `correlation_id` within a sliding window).
- **Ordering.** Messages within a single sender→recipient pair are delivered in order (FIFO queue per pair). Cross-pair ordering is not guaranteed.
- **Message size limit.** 64KB per message. Larger payloads use a file-reference mechanism: the sender writes to a shared staging directory (with appropriate permissions), and the message contains the file path.

7.3.3 Authentication

Each agent session authenticates to the gateway using a session token issued at spawn time, bound to the agent's Linux user identity. The gateway verifies:

1. The session token is valid and not expired.
2. The claimed `agent_id` matches the authenticated session.
3. The recipient is in the sender's `can_send_to` list (topology enforcement).
4. The `content_classification` in the payload does not exceed the recipient's clearance.

Messages failing any check are rejected with an error code and logged as a compliance event.

7.3.4 Gateway Topology

```
# gateway/agent-topology.yaml
sessions:
  - id: agent_salesrep              # one shared sales-rep role agent
    clearance: T3
    department: sales
    can_receive_from: [agent_sales]
    can_send_to: [agent_sales]
    push_channels: [competitive_intelligence, customer_feedback, meeting_notes]
    owners: [maria.garcia, pedro.martinez, ..., sales.manager]

  - id: agent_sales
    clearance: T4
    department: sales
    can_receive_from: [agent_salesrep, agent_fieldengr, ...]
```

```
    can_send_to: [agent_salesrep, agent_fieldengr, ..., agent_strategos]

  - id: agent_strategos
    clearance: T5
    can_receive_from: [agent_sales, agent_it, agent_hr, ...]
    can_send_to: [agent_sales, agent_it, agent_hr, ...]
```

7.3.5 Protocol Error Codes and Declassification Workflow

Architecture papers often underspecify failure semantics. For implementers, explicit error classes matter because they determine whether an agent retries, escalates, or stops. HIVEMIND should standardize at least the following gateway responses:

- `ERR_TOPOLOGY_DENIED` — sender not authorized to address recipient
- `ERR_CLASSIFICATION_EXCEEDED` — payload classification above recipient ceiling
- `ERR_SCHEMA_INVALID` — malformed push payload or missing required fields
- `ERR_DECLASSIFICATION_REQUIRED` — response would require controlled downgrade before delivery
- `ERR_HUMAN_REVIEW_REQUIRED` — no approved template exists; request queued
- `ERR_TIMEOUT` — recipient unavailable within TTL

Declassification itself should follow a documented sequence:

1. Higher-tier agent identifies responsive material above the recipient's clearance.
2. System checks for an approved redaction template matching the content category.
3. If a template exists, the system generates a downgraded object with a new object ID and logs provenance links to the source.
4. If no template exists, the response halts and moves to human review.
5. The recipient receives either the downgraded object or an explicit pending notice — never a best-effort free-form partial summary.

This matters practically because many information leaks occur in edge-case handling. The dangerous moment is not the normal case; it is the case where the system is almost able to answer and improvises. HIVEMIND's implementation should make improvisation impossible at the downgrade boundary.

7.4 Memory Files as Auditable State

Workspace layout:

```
/agents/
+-- sales-rep/                # owned by agent_salesrep, group: sales-dept-ro
|   +-- MEMORY.md             # shared institutional memory for the sales role
|   +-- memory/
|   |   `-- 2026-03-17.md
|   +-- workspace/
|   |   +-- clients/
|   |   `-- knowledge/        # symlinks into /corporate/knowledge (Section 2.7)
|   +-- people/               # thin per-person interaction profiles (Section 2.7)
|   `-- SOUL.md
+-- field-engineer/           # similar structure, one per ROLE
+-- sales-dept/               # owned by agent_sales, group: strategos-ro
|   +-- MEMORY.md
|   `-- department-intel/
```

```
‘-- strategos/                  # owned by agent_strategos, root group
    ‘-- ...
```

The `-ro` suffix indicates read-only group access for the parent agent. Higher-clearance agents can read but should not routinely write to lower-clearance workspaces — writes are reserved for directives and post-review corrections.

7.5 Multi-Tenancy and Process Isolation

Linux filesystem permissions provide the primary isolation boundary but are not sufficient against all attack vectors. A compromised agent process could potentially:

- Read `/proc` entries of other agent processes, leaking environment variables or command-line arguments.
- Exploit kernel vulnerabilities to escalate privileges.
- Consume shared resources (CPU, memory, disk I/O) to degrade other agents' performance.

Additional isolation measures:

- **Process namespaces.** Each agent process runs in its own PID and mount namespace via `unshare` or lightweight containers (systemd-nspawn, Podman). This prevents `/proc` snooping across agents.
- **Seccomp profiles.** Agent processes are restricted to the minimum required system calls. File operations, network (to gateway only), and standard I/O — no `ptrace`, no raw socket, no module loading.
- **Resource limits.** Per-agent cgroup limits for CPU, memory, and I/O bandwidth prevent resource exhaustion attacks.
- **Network isolation.** Agent processes can communicate only with the gateway (localhost, specific port). No direct network access to other agents, external services, or the internet (unless specific tools are enabled for that agent's role).

For deployments requiring stronger isolation (regulated industries, larger organizations), full container isolation (Docker/Podman with read-only root filesystems) or VM-level isolation (one VM per department) is recommended.

7.6 Cron Jobs for Compliance

```
# /etc/cron.d/hivemind-compliance
0 2 * * * compliance-bot /opt/hivemind/sweep/run-full-sweep.sh >> /var/log/hivemind/sweep.lo
0 3 * * * compliance-bot /opt/hivemind/audit/rotate-logs.sh
0 4 * * 0 compliance-bot /opt/hivemind/reports/generate-weekly.sh | mail -s "HIVEMIND Weekly
0 * * * * compliance-bot /opt/hivemind/sweep/quick-scan.sh
```

7.7 Observability and Debugging

An 85-agent swarm requires observability infrastructure beyond compliance logging:

- **Health dashboard.** Real-time status of all agent processes: running/suspended/quarantined, last activity timestamp, memory file size, session count. Implemented as a lightweight web UI served by the gateway, accessible to L2-IT and L3-EXEC.
- **Distributed tracing.** Each inter-agent communication carries a trace ID (the `correlation_id` in the message protocol). The gateway aggregates trace logs, enabling reconstruction of multi-agent interaction chains when diagnosing coordination failures.

- **Alert thresholds.** Beyond compliance anomalies, operational alerts for: agent process crash, memory file exceeding size threshold (indicating runaway accumulation), gateway queue depth exceeding capacity, inference server response time degradation.
- **Log aggregation.** All agent stdout/stderr, gateway logs, and compliance sweep output aggregate to a central log store (e.g., journald with remote forwarding, or ELK stack for larger deployments). Queryable by L2-IT for troubleshooting.

7.8 Versioning, Rollback, and Configuration Management

Agent configurations (system prompts, tool access, clearance assignments, topology rules) are managed as version-controlled files:

- All configuration lives in a Git repository (`/opt/hivemind/config/`) with branch protection and required reviews for changes.
- Each configuration change is tagged with a version, author, and rationale.
- Deployment is atomic: a configuration change is applied to all affected agents simultaneously (or rolled back entirely) via a deploy script that validates the new configuration against the topology rules before applying.
- Rollback procedure: `git revert` + redeploy. Agent workspaces (memory files) are not affected by configuration rollbacks — only system prompts, tool access, and routing rules change.

7.9 Backup and Disaster Recovery

- **Agent memory files** are backed up nightly (after the compliance sweep) to encrypted off-site storage. Incremental backups with 30-day retention.
- **Compliance audit logs** are backed up separately with longer retention (matching regulatory requirements — typically 5-7 years for employment-related data).
- **Recovery procedure.** After a server failure: (1) restore from latest backup; (2) compliance sweep runs immediately on restored data to verify integrity; (3) agents restart with their last known good state. Recovery time objective: 4 hours for a single department, 8 hours for full system.
- **Memory file integrity.** Each backup includes SHA-256 checksums. Restoration verifies checksums before bringing agents online.

7.10 TOOLS.md / Trust Tier Extension

Each agent has a role-specific TOOLS.md:

```
# TOOLS.md - L1-SALES-REP (shared sales rep role agent)

## Trust Tiers

- **Owners:** all salespeople holding this role + the sales manager - interactive use
- **Department Supervisor:** L2-SALES - can issue directives, read all outputs
- **Executive:** L3-EXEC - read-only access, cannot be refused

## Tool Access

- email: read/send (the current user's mailbox only, resolved per session)
- crm: read/write (the role's accounts; ownership per workload-division.md)
- calendar: read/write (the current user's calendar)
- product_catalog: read (T2 resource)
- contracts: read (T3 resource, sales-role accounts)
```

```
- payroll_api: DENIED
- hr_records: DENIED
- infrastructure: DENIED

## Escalation Rules

- Any request involving an account owned by another rep: note ownership from workload-division.md an
- Any pricing commitment request: verify against approved pricing tiers
- Any legal/contract language: flag for human review, do not commit
```

7.11 Operational Governance: Ownership, Bounded Autonomy, and User/IT Separation

No agent runs ownerless, but the right control is accountability, not surveillance.

Every agent has one or more human owners. Each agent node in the topology carries a list of `owners` — the humans accountable for it. A role agent's owners are typically its occupants plus the department manager; a department or executive agent's owners are the relevant managers. Ownership establishes responsibility (who is answerable for what this agent does) and a point of contact for review.

No hard inactivity thresholds. An earlier instinct was to require human interaction within a fixed window and to suspend agents that go dark. We reject this: people take holidays, travel, and work in bursts, and a 24-hour timer would fire constantly on legitimate absence, training everyone to ignore it. Role agents are dormant by default when no one is using them — an idle agent consumes nothing and poses little risk. Presence-policing solves a problem that does not exist.

The real concern is runaway autonomy, not silence. The failure mode worth engineering against is an agent's *scheduled* work — cron jobs, background sweeps, proactive monitors — running unattended and burning tokens (or taking actions) for no reason: a misconfigured monitor re-querying every minute, a self-improvement loop with no stopping condition. Bounded-autonomy controls therefore target *recurring automated work*, not interactive sessions:

- Every cron or background task declares an expected cost and cadence; the gateway enforces per-agent rate and spend ceilings.
- Recurring tasks that produce no consumed output over several cycles are flagged for owner review rather than left to run.
- Expensive or irreversible recurring work requires explicit owner authorization to start, consistent with the resource-awareness principle that governs the platform generally.

Users signal IT; they do not self-administer. This is the governance backbone of the self-modification axis (Section 3.5, Axis 5). Following the division-of-labor principle, **only IT (`L2-IT`) may upgrade, rebuild, or restart the gateway**, change tool allowlists, or alter clearance assignments. End users interact with their role agent; they never touch the infrastructure. This separation protects the integrity of the whole swarm — a single user must not be able to widen their own access or destabilize shared infrastructure — and keeps the configuration-management discipline of Section 7.8 meaningful.

For this to be *fast* rather than a bottleneck, the improvement loop is engineered, not ad hoc:

- **A structured improvement channel, ingested by IT's agent.** Every user has a low-friction channel to file a bug they hit or a capability they imagine would speed their work. Crucially, these tickets are *consumed by `L2-IT`'s own agent*, not just read by a human. That changes what a good ticket looks like: because an agent will parse and act on it, the end user is expected to put substantial detail into each request — the exact behavior observed, the desired behavior, the workflow it blocks, concrete examples, and the

frequency. A vague "the agent is slow" produces nothing; a specific, example-rich request becomes actionable work. The channel schema enforces this (required fields, reproduction steps, expected-vs-actual), turning the user base into a distributed requirements-gathering function.

- **IT collects, builds, and tests.** `L2-IT` triages the incoming queue, deduplicates and prioritizes, and implements the improvement — increasingly by *vibe-coding* against its own agent (the IT engineer directs the agent to draft the tool, skill, plugin, or recipe, then reviews it) rather than hand-writing every line. `L2-IT` then performs functionality testing in a staging environment (Section 8.7) before release.
- **One rollout reaches everyone.** Because the workforce runs on shared role agents, an implemented improvement is deployed once to the canonical role build and every occupant gets it simultaneously — the same property that makes onboarding inheritance (Section 3.2) makes capability upgrades instant and uniform.

The result is a controlled evolution loop: distributed sensing of what to improve (every user, in detail), centralized authority to implement it safely (`L2-IT`), and uniform propagation of the result (the shared agent). The company gets continuously faster without letting anyone destabilize the substrate everyone depends on.

8. Risks and Mitigations

8.1 The Curious Agent Problem

A salesperson asks "find everything about the González account." The agent, interpreting "everything" literally, combines fragments from legitimate tool calls (email, calendar, wiki) that in aggregate allow it to infer restricted content. Each individual call is authorized; the synthesis crosses a clearance boundary.

Mitigation:

- Tool call logging includes query intent. Compliance sweep flags aggregation-pattern queries.
- The pre-execution tool-boundary hook (AEGIS, Section 2.4) can *block* a synthesis-and-exfiltrate tool call before it runs — so an aggregation that would cross a clearance boundary is stopped at the boundary, not merely flagged hours later in the nightly sweep.
- System prompt includes clearance awareness: "If a query seems to require information beyond your clearance level, acknowledge the limitation and escalate to L2-SALES."
- Agent outputs pass through a structural anomaly check (novelty against the agent's own history; incongruity between the output and the agent's clearance — see Section 6.3.5) rather than a supervised "restricted-content" classifier, which measurement shows is the weaker framing.

8.2 Inter-Agent Social Engineering

Could a lower-clearance agent manipulate a higher-clearance agent into revealing restricted information? Including via the upward push channel — encoding a covert query as "competitive intelligence" crafted to elicit a response containing restricted data.

Mitigation:

- Department agents use a separate, more restrictive system prompt for inter-agent communication. Explicit rule: "Never include T4+ content in responses to T3 agents regardless of framing."

- All department agent responses to role agents pass through a **clearance validator** before delivery — a separate, deterministic check — and the response-generating tool call is itself subject to the pre-execution tool-boundary hook (Section 2.4), which can *deny* a delivery that would carry restricted content downward, so the validator is backed by an enforcing gate rather than being advisory alone.
- **Clearance validator design.** To avoid the attack-surface and latency concerns of a second LLM call, the validator is primarily rule-based: regex patterns for known restricted data formats (salary ranges, credential patterns, document IDs from restricted paths), plus a structural anomaly check (Section 6.3.5) for semantic incongruity rather than a supervised restricted-content head. The full LLM-based validation is reserved for messages flagged as ambiguous by the rule-based layer. This two-tier approach reduces latency (rule-based check: <10ms; LLM check only when triggered: ~500ms) and limits the attack surface.
- Push channels return no data to the sender. The channel schema is validated; malformed pushes are rejected and logged. Push frequency is rate-limited per agent.
- Communication logs capture request-response pairs. Patterns where a specific agent consistently receives more sensitive-seeming responses are flagged.

8.3 Memory Persistence and Contamination

Restricted information enters lower-clearance memory through human channels. María attends a management meeting and hears salary figures; she mentions this to the sales agent, which writes it into memory. T4 data now exists in a T3 memory file.

Mitigation:

- Semantic memory contamination scan (Section 6.3.4) targets this. Structured patterns trigger review (with the false-positive calibration described in Section 6.3.1).
- Memory files tag entries as agent-generated vs. human-dictated. Human-dictated entries receive elevated scrutiny.
- Employee training: "Your agent is a company system. Information you share is subject to the same policies as company email."

8.4 Single Point of Failure: The Executive Agent

L3-EXEC has T5 clearance and access to all workspaces. If compromised, the entire information architecture is at risk.

Mitigation:

- Isolated infrastructure with no external network access. Inputs arrive only through sanitized gateway channels.
- Most aggressive behavioral monitoring. Outputs reviewed daily.
- T5 data store uses additional encryption beyond filesystem permissions.
- **L3-EXEC dead switch**: CEO can immediately suspend the executive agent and revert to human-only T5 access.
- Two-person rule: L3-EXEC-generated summaries including T5 content require human review before delivery.

8.5 Model Capability Variance

Different models follow clearance rules with different reliability. A 5% failure rate across thousands of daily interactions is serious compliance exposure.

Mitigation:

- Clearance enforcement is infrastructure-first: filesystem permissions (kernel) and the pre-execution tool-boundary hook (Section 2.4) are two independent constraints that each *deny* a violation rather than rely on the model behaving — a model error must defeat both, not one.
- Model-level clearance instruction (system prompts, in-context rules) is the softest, defense-in-depth layer, not primary enforcement.
- Smaller models excluded from T4+ operations, even as supporting steps. Model capability requirements scale with clearance tier.
- Periodic red-team exercises: extract restricted data via social engineering, prompt injection, and aggregation attacks. Findings feed into prompt improvements and infrastructure hardening.

8.6 Failure Modes and Recovery

Department agent crash. If L2-SALES becomes unavailable mid-coordination:

- Employee agents detect the failure when their messages receive `TIMEOUT` responses from the gateway.
- Employee agents enter a **degraded mode**: they continue serving their human user with locally available data but cannot initiate cross-agent queries or receive updates.
- The gateway queues incoming messages for the crashed department agent (up to queue capacity).
- Automated restart: systemd watchdog restarts the department agent process. On restart, the agent replays queued messages.
- If restart fails three times, L3-EXEC and L2-IT are alerted for manual intervention.

Memory file corruption. If a compliance sweep detects inconsistency:

- The corrupted memory file is quarantined (moved to a review directory, replaced with the last known good backup).
- The affected agent restarts with restored memory. Recent entries since the last backup are lost — an acceptable trade-off for integrity.
- The corruption event is investigated: was it a disk error, a software bug, or a security incident?

Nightly sweep failure. If the compliance sweep crashes or times out:

- The sweep logs the failure and sends an immediate alert to the compliance officer.
- A catch-up sweep runs at the next available window.
- If two consecutive sweeps fail, all agents are suspended pending manual investigation.

State reconciliation after recovery. After any recovery event, a targeted compliance sweep runs on the affected agent(s) before returning to normal operation.

8.7 Model Updates and Behavioral Drift

When the underlying LLM is updated, agent behavior may change unpredictably.

- **Staging environment.** A parallel HIVEMIND instance with synthetic data runs the new model version. Automated test suites verify clearance compliance, redaction accuracy, and behavioral baselines.
- **Canary deployment.** After staging validation, one department upgrades first. A week of monitoring before rolling out to remaining departments.
- **Behavioral baseline comparison.** Post-upgrade, the compliance sweep compares agent behavior metrics (query patterns, response lengths, escalation rates) against pre-upgrade baselines. Significant deviations trigger review.

8.8 Real-Time Communication Channels

The paper's architecture focuses on email and files but organizations communicate through Slack/Teams messages, phone calls, and video meetings.

- **Chat platforms (Slack, Teams).** Treated as an additional data source. Agent tools include chat API access (read-only for the employee's channels). Messages are classified at the channel's configured tier. The department agent monitors department-wide channels.
- **Phone calls and meetings.** If transcription is enabled (with employee consent per GDPR), transcripts are written to the agent's workspace and subject to the same clearance and compliance rules as any other document. Live call assistance requires streaming inference — a capability that increases infrastructure requirements (see Section 7.5 on resource limits).
- **Video meetings.** Meeting recordings and transcripts follow the same classification and storage rules. The highest-clearance participant in a meeting determines the transcript's classification tier.

8.9 Summary Risk Table

Risk	Severity	Likelihood	Primary Mitigation
Curious agent data aggregation	High	Medium	Query pattern monitoring, clearance-awareness prompting
Inter-agent social engineering	High	Low	Two-tier clearance validator, restricted inter-agent prompts
Memory contamination	Medium	Medium	Nightly semantic scan with calibrated thresholds
Executive agent compromise	Critical	Very Low	Isolated infrastructure, dead switch, two-person review
Model capability variance	Medium	Medium	Filesystem enforcement as primary layer, tier-based model selection
GDPR compliance violation	High	Medium	DPIA, data minimization sweeps, purpose limitation audit
Employee offboarding data residue	Medium	High	Structured offboarding protocol, automated purge scheduling
Department agent crash	Medium	Medium	Watchdog restart, degraded mode, queued message replay
Model update behavioral drift	Medium	Medium	Staging environment, canary deployment, baseline comparison
Real-time channel leakage	Medium	Medium	Channel-level classification, consent-gated transcription

9. Phased Deployment Strategy

Deploying HIVEMIND across an organization should be staged, not big-bang. Each phase has success criteria that must be met before advancing.

Phase 0: Infrastructure and Legal (Weeks 1-4)

- Complete DPIA and EU AI Act conformity assessment.
- Provision server infrastructure, install OpenClaw, configure local inference.
- Establish compliance officer role and audit procedures.
- Draft and execute employee contract addendums.
- **Gate:** DPIA approved, infrastructure operational, legal documentation complete.

Phase 1: Single Department, T2 Only (Weeks 5-10)

- Deploy the L1-SALES-REP role agent for 3-4 pilot salespeople (volunteers) with T2 clearance only, plus the L2-SALES agent.
- No restricted data access, no cross-role coordination beyond the shared product catalog and commons.
- Focus: employee adoption, the per-person interaction layer, commons authoring, and tool integration testing.
- **Gate:** >80% daily active use by pilot users, zero compliance flags, positive user feedback.

Phase 2: Single Department, Full Clearance (Weeks 11-18)

- Raise pilot agents to T3 clearance. Enable customer ownership routing, cross-agent history queries, competitive intelligence push.
- Deploy full compliance sweep infrastructure.
- L2-SALES performs T4 redaction via template pipeline.
- **Gate:** Compliance sweep clean for 4 consecutive weeks, redaction pipeline validated by management review, coordination demonstrably improving customer interactions.

Phase 3: Multi-Department (Weeks 19-30)

- Deploy L2-IT, L2-HR, L2-FINANCE with respective role agents.
- Enable L3-EXEC for executive oversight.
- Cross-department communication limited to L3-EXEC-mediated queries.
- **Gate:** All departments operational, cross-department queries functioning, weekly audit reports stable.

Phase 4: Full Operation (Week 31+)

- All employees have agents. Full clearance model active.
- Federated channels for cross-department structured data sharing.
- Continuous monitoring, quarterly red-team exercises.

Success Metrics

- **Adoption:** Daily active usage rate >75% after 90 days.
- **Compliance:** <2 quarantine events per quarter after Phase 3.
- **Value:** Measurable improvement in customer response time, reduced information request escalations, positive employee survey results.
- **Continuity (the headline metric):** *time-to-productivity for a new hire* — the interval from start date to independent role performance. The continuity thesis predicts a step-change here (e.g., a new sales rep productive in two weeks instead of three months), because the replacement inherits the role agent's accumulated knowledge instead of rebuilding it. This is the most concrete, measurable, and sellable outcome HIVEMIND offers; it should be instrumented from Phase 1 and compared against the organization's historical onboarding baseline.

- **Knowledge freshness:** percentage of commons documents within their review cadence (Section 2.8); a declining figure is an early warning of institutional decay even when adoption and compliance look healthy.

9.1 Evaluation Agenda Before Full-Scale Rollout

Because HIVEMIND is a design paper rather than an empirical deployment report, the path from architecture to full production should include a modest but disciplined evaluation program. Before claiming enterprise readiness, the organization should run:

1. **Red-team leakage tests.** Structured attempts to extract T4/T5 information through prompt manipulation, aggregation, and malformed push payloads.
2. **Operational pilot metrics.** Median response time, escalation rate, redaction frequency, false-positive compliance burden, and employee satisfaction.
3. **Counterfactual comparison.** Compare pilot teams using HIVEMIND against similar teams using standard personal agents or no agents, especially for customer response time and duplicated work.
4. **Post-incident reviews.** Every quarantine, major false positive, and cross-agent coordination failure should produce a written root-cause analysis.

A useful publication-quality follow-on to this paper would therefore be a field study with three outputs: empirical performance, compliance incident statistics, and qualitative adoption findings. That study would materially strengthen the architecture's academic standing.

10. Future Directions

10.1 Scaling Beyond the Family Business

At 1,000+ employees, the hierarchy extends: Level 1 (role agents), Level 2 (team), Level 3 (department), Level 4 (division), Level 5 (executive). Role-binding helps here — the number of Level 1 agents scales with the number of *distinct job functions*, not with headcount, so a division of 200 people doing a handful of roles still has only a handful of role agents. The primary scaling challenge becomes department/division agent cognitive load, addressed by inserting sub-department (team-level) aggregators that feed upward.

10.2 Integration with Enterprise Systems

Full deployment requires integration with:

- **ERP systems** (SAP, Oracle, Dynamics): Agent tools querying ERP APIs, clearance mapped to ERP role-based access control.
- **CRM systems** (Salesforce, HubSpot): Native CRM integration replacing filesystem paths with API calls.
- **Active Directory / LDAP**: Agent clearance auto-provisioned from AD group membership.
- **Ticketing systems** (Jira, ServiceNow): IT and production agent coordination.

10.3 Federated Agent Networks Across Companies

The most ambitious direction: agent networks spanning organizational boundaries. Two companies with a supplier-customer relationship establish a federated channel where their sales role agents share pre-approved data categories without exposing internal data.

This requires a federated clearance protocol: each company's gateway defines what external agents can receive, filtering before crossing the federation boundary. This intersects with emerging enterprise AI standards (IEEE P3394, ISO/IEC JTC 1/SC 42).

10.4 The Role of NemoClaw and OpenShell

- **NemoClaw on mobile:** Employee phones run a NemoClaw instance for offline capability at T1/T2 clearance; T3+ queries route to the server.
- **OpenShell for automation:** Department agents and compliance infrastructure run through OpenShell scripts for scheduled operations.
- **TinkerClaw companion app:** The human-facing surface for all agents, ensuring consistent UX across the hierarchy.

10.5 Performance and Cost Estimates

A HIVEMIND deployment for 85 employees requires:

Component	Specification	Estimated Cost
Inference server	2× NVIDIA A100 80GB (or equivalent) for 70B model	€25,000-35,000
Application server	64-core CPU, 256GB RAM, 4TB NVMe	€8,000-12,000
Backup storage	10TB encrypted NAS	€2,000-3,000
Network	Gigabit internal, redundant uplink	existing infrastructure
Total hardware		**€35,000-50,000**
Annual operating cost	Power (~3kW), cooling, maintenance, sysadmin time	€15,000-25,000/yr

Compute budget. Role-binding sharply reduces the number of distinct agent *processes and configurations* — roughly one per job function plus department and executive agents, perhaps 10–15 agents for an 85-person company rather than 91. This cuts memory footprint, configuration-management surface, and operational complexity. It does *not* proportionally cut inference *demand*: the call volume is driven by how many humans are working, not how many agent processes exist. With ~85 active users issuing an average of 50 role-agent inference calls per day at ~2 seconds per call on a 70B model — plus department/executive aggregation and compliance scans — total demand is on the order of ~2.5 hours of sequential inference per day, well within the capacity of 2× A100 GPUs with significant headroom for peak loads. (A shared role agent serving concurrent users requires session isolation per user but shares the loaded model weights, which is where the memory savings come from.)

Comparison to cloud inference: The same call volume on cloud APIs (estimated $0.03/call for GPT-4-class models): $\sim 85 \times 50 \times \$0.03 \times 365 \approx \$46{,}500$/year — comparable to the first-year total cost of ownership for on-premises hardware, but without data residency guarantees. By year two, on-premises is significantly cheaper.

These are rough estimates. Actual costs depend on model choice, inference optimization (quantization, batching), and usage patterns. A detailed capacity planning exercise should precede procurement.

10.6 Toward Organizational Cognition

HIVEMIND is an attempt to give an organization a distributed cognition substrate — ensuring relevant knowledge reaches the humans who need it, filtered for their role, organized for their

context, and available in real time rather than buried in an email thread from 2019.

As local inference models improve — as 70B models become 7B models with equivalent capability, as edge hardware becomes cheaper — the cost floor drops. The day is not far when a 20-person company can run a full hierarchical agent swarm on a single rack-mount server. The architectural patterns described here will determine whether such deployment is safe, auditable, and genuinely useful.

11. Limitations

This paper has several significant limitations that should be acknowledged:

1. **No empirical validation.** HIVEMIND is a design blueprint, not a deployed system. The architecture has not been tested against real enterprise workloads. Claims about effectiveness (the "coordination dividend") are theoretical.

2. **Redaction reliability is measured, not bounded.** The template-based redaction pipeline (Section 4.4) is now checked two ways — an offline redaction-fidelity eval (Section 4.4.2) and a runtime doubt reviewer (Section 4.4.1) — which together replace an untested claim with a measured one. But no formal information-theoretic *bounds* on leakage are established, the eval is specified but not yet run on a real corpus, and the templates themselves must be authored by humans, a manual bottleneck.

3. **Cost estimates are approximate.** The performance figures in Section 10.5 are back-of-envelope calculations that have not been validated against actual deployment. Real-world usage patterns may differ significantly.

4. **Model capability assumptions.** The architecture assumes that current-generation 70B models can reliably perform clearance-aware operations (classification tagging, escalation detection, behavioral profiling). This assumption needs empirical validation per deployment.

5. **Organizational complexity not fully addressed.** Real companies have matrix reporting structures, temporary project teams, contractors, and consultants — none of which fit neatly into the three-level hierarchy. Extending HIVEMIND to these structures requires additional design work.

6. **Single-site assumption.** The architecture assumes a single-site deployment. Multi-site companies with distributed infrastructure face additional challenges: network latency for inter-agent communication, data replication across sites, and potentially conflicting local regulations.

7. **No user study.** Employee adoption challenges (Section 5.2) are discussed theoretically. Actual user acceptance, trust dynamics, and behavioral adaptation to monitored AI agents have not been studied.

8. **Cost-benefit analysis absent.** The paper does not quantify the ROI of HIVEMIND versus traditional knowledge management systems (SharePoint, wikis, email archives). The value proposition, while intuitively compelling, lacks comparative evidence.

9. **Formal integrity treatment is partial.** Confidentiality is well developed through the Bell-LaPadula framing, but integrity and provenance controls are described operationally rather than formalized. A stronger version would either formalize integrity constraints or narrow its claims explicitly to confidentiality-first design.

10. **Benchmark and test methodology is specified but unrun.** The redaction-fidelity eval (Section 4.4.2) defines a seeded leakage corpus, a measured quantity (residual mutual information), and per-category acceptance thresholds — the seeded-corpus methodology Limitation 10 previously called for entirely absent. It remains specified rather than executed, and the compliance classifiers and clearance validator still lack an analogous seeded-violation benchmark. Reproducibility is improved but not yet demonstrated.

11. **Role-binding widens intra-team data visibility.** A shared role agent holds the whole role's data and resolves individual ownership contextually (Section 4.2) rather than through a kernel-enforced boundary. This is appropriate for organizations where peers at the same tier already share a CRM and reasonably *can* see each other's accounts — the typical family business. It is a real limitation where intra-team confidentiality matters: competing salespeople guarding leads, commission disputes, or any setting where same-role peers must be walled from each other. Such cases need either per-occupant scoping within the role agent (reintroducing some of the complexity role-binding removes) or a return to per-person agents for that specific function.

12. **Reduced per-person personalization.** The thin interaction layer (Section 2.7) personalizes manners, not memory. A dedicated personal agent could in principle tailor itself more deeply to one individual's working style. Whether the shared design's standardization benefits outweigh any adoption cost from reduced personalization is an empirical question this paper does not answer.

13. **Fossilization is mitigated by practice, not guaranteed by architecture.** The new-hire research loop (Section 5.9) is an organizational practice that depends on new hires actually probing and stewards actually incorporating feedback. It reduces but does not eliminate the risk that a persistent role agent entrenches outdated assumptions. A persistent shared agent that is *well* maintained is a continuity asset; one that is neglected is a single, authoritative-sounding source of stale truth — and the second failure mode is quieter than the first.

14. **Role decomposition is assumed clean.** The design assumes jobs decompose into roles with shared scope and shared clearance. Real organizations have hybrid roles, people who span two functions, and seniority gradations within a single job title that may warrant different clearances. Mapping these onto a single shared role agent — or deciding when a distinct role agent is warranted — requires case-by-case design beyond this paper's scope.

12. Conclusions

This paper has described HIVEMIND: a role-bound agent swarm architecture for enterprise AI deployment, designed around the principle that an organization's knowledge should outlive the people who hold it — and that continuity, coordination, and information security are not in conflict but are design requirements to be solved simultaneously.

The key contributions are:

1. **The Continuity Thesis and Role-Bound Agents.** One canonical agent per job function, shared by every occupant of that role, making the agent the durable holder of institutional knowledge. This turns the most expensive recurring failure in a mid-sized company — the talent leak — into a non-event: the agent never resigns, offboarding collapses to a workload-map edit, cross-department coordination collapses to one channel per function, and onboarding becomes inheritance.

2. **The Knowledge Commons and Cascade Model.** A centralized, version-controlled Markdown vault (company canon, industry context, customer-relationship model, product catalog) mounted via symlinks, with agent knowledge assembled as a cascade from company canon through role slice and workload map to a thin per-person interaction layer. Symlinks provide a single source of truth without weakening any clearance boundary. The accompanying transformation of trainers into knowledge stewards — who train the AI once instead of each hire repeatedly — is the human-capital case for the architecture.

3. **The Clearance Model, Applied at Role Granularity.** Five data classification tiers (PUBLIC through SECRET) mapped to Linux filesystem permissions, formally grounded in Bell-LaPadula with a controlled declassification mechanism, now keyed to roles rather than individuals. Infrastructure enforcement is categorically more reliable than model-level instruction.

4. **The Sales Coordination Model under Contextual Ownership.** A single shared sales agent resolves customer ownership contextually — knowing whom it serves and reading the workload map — rather than through inter-agent email routing, with template-based redacted summaries for upward-tier data and competitive intelligence aggregation. The role-bound design makes the original "two agents, same customer, divergent responses" failure mode structurally impossible.

5. **Compliance Architecture.** Nightly sweep across four analysis dimensions with calibrated false-positive thresholds, communication logging with full audit trail, and automated quarantine with human-in-the-loop review satisfying GDPR Article 22.

6. **Legal Framework.** GDPR and EU AI Act compliance analysis, including DPA requirements, profiling assessment, automated decision-making obligations, and high-risk AI system conformity assessment.

7. **Risk Analysis.** Ten principal risk categories with concrete mitigations, including failure recovery procedures, model update management, and real-time communication channel handling.

8. **Deployment Strategy.** A four-phase rollout with explicit gate criteria, success metrics, and adoption management considerations.

9. **Technical Specification.** Inter-agent protocol with message format, delivery semantics, authentication, and process isolation beyond filesystem permissions, plus bounded-autonomy operational governance and user/IT separation of duties. Enforcement is dual-deny: the kernel and a pre-execution tool-boundary hook each independently *prevent* a clearance breach rather than merely log it.

10. **Concurrent Action Under Arbitration.** A coordination protocol for the action plane (Section 3.7): workers prepare exact changes outside any lock, then apply them under short-lived per-object leases so disjoint objects proceed concurrently and shared objects serialize — making "clean merge" a non-event and giving the audit substrate a clean per-actor write history. This is the write-side counterpart to the read-side mediated retrieval, instantiating the principle that concurrency is made safe by protocol rather than avoided.

11. **The Multi-Axis Network View.** A framing of the swarm as one graph read along six orthogonal axes — clearance, supervision, bottleneck reduction, workload sharing, self-modification, and knowledge propagation — each with its own mechanisms and frequently in tension with the others. The self-modification axis is governed by division of labor: only IT may alter the infrastructure, fed by a detailed, agent-ingested improvement channel

that lets the workforce get continuously faster without anyone destabilizing the shared substrate.

12. **A Future-of-Work Argument.** A perspective on why role-bound AI need not be read as job elimination: for roles whose value is learning and teaching, removing the repetition and adding leverage can elevate the job — the trainer becomes a recurring-revenue knowledge manager who teaches once, fights fossilization, breaks the per-hour income ceiling, and serves many companies at once.

13. **Positioning Among Convergent Paradigms.** A related-work synthesis (Section 1.6) placing HIVEMIND among the fan-out, coordinator-with-workers, and org-chart topologies, and against Paperclip, coordinator/teammate execution models, single-agent role-switching, headroom's reversible caching, and addyosmani's doubt-driven review — with the convergence argument that corporate structure may be the natural coordination topology, and HIVEMIND's distinguishing claim that the org chart *is* the access-control graph.

14. **Clearance-Aware Memory and Measured Leakage Control.** The per-agent cognitive stack (Section 1.5) applied under role-binding; an append-only event store and a Compress-Cache-Retrieve-Under-Clearance variant that keep memory reversible without laundering tiers (Sections 2.9–2.10); cross-role federated retrieval (Section 3.6); and a matched pair that moves leakage control from asserted to measured — an offline redaction-fidelity eval and a runtime fresh-context doubt reviewer (Sections 4.4.1–4.4.2).

HIVEMIND is a design blueprint — not a finished system. It will evolve as models improve, enterprise integration matures, and regulators develop clearer frameworks. What we believe will endure is the fundamental tension it addresses: the value of coordination versus the necessity of boundaries. Every organization deploying AI at scale will face this tension. The answer is architecture: clear rules about what flows where, enforced at the right layer, audited continuously, with humans in the loop for consequential decisions.

References

Abbadi, I., & Martin, A. (2011). *Trust and Privacy in Cloud Information Management.* Proceedings of the 5th International Conference on Trust Management.

Agrawal, M., Johnson, S., & Sager, T. (2022). Multi-agent information flow control in distributed AI systems. *arXiv preprint arXiv:2204.09312.*

Bell, D. E., & LaPadula, L. J. (1976). Secure Computer System: Unified Exposition and Multics Interpretation. MITRE Corporation Technical Report MTR-2997.

Brewer, D. F. C., & Nash, M. J. (1989). The Chinese Wall security policy. *Proceedings of the IEEE Symposium on Security and Privacy*, 206–214.

Brynjolfsson, E., & McAfee, A. (2014). *The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies.* W. W. Norton.

Davenport, T. H., & Ronanki, R. (2018). Artificial intelligence for the real world. *Harvard Business Review, 96*(1), 108–116.

European Parliament and Council. (2016). *Regulation (EU) 2016/679 (General Data Protection Regulation).* Official Journal of the European Union.

European Parliament. (2024). *Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (AI Act).* Official Journal of the European Union.

Ferraiolo, D. F., & Kuhn, D. R. (1992). Role-based access controls. *Proceedings of the 15th National Computer Security Conference*, 554–563.

Garg, A., & Pfleeger, C. P. (2020). Information security policy in the enterprise: A framework for compliance. *Computers & Security, 92*, 101776.

Gunning, D., & Aha, D. (2019). DARPA's explainable artificial intelligence (XAI) program. *AI Magazine, 40*(2), 44–58.

Denning, D. E. (1976). A lattice model of secure information flow. *Communications of the ACM, 19*(5), 236–243.

NIST. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. National Institute of Standards and Technology.

Lewis, P., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems, 33*, 9459–9474.

McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex. *Psychological Review, 102*(3), 419–457.

NIST. (2020). *Zero Trust Architecture* (NIST Special Publication 800-207). National Institute of Standards and Technology.

Packer, C., et al. (2023). MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*.

Park, J. S., et al. (2023). Generative agents: Interactive simulacra of human behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*.

Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.

Saltzer, J. H., & Schroeder, M. D. (1975). The protection of information in computer systems. *Proceedings of the IEEE, 63*(9), 1278–1308.

Wang, G., et al. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science, 18*(6), 186345.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems* (2nd ed.). Wiley.

HIVEMIND: Role-Bound Agent Swarms for Enterprise Continuity Author: Oscar Serra Classification: INTERNAL — OpenClaw/TinkerClaw Research

References