Architectural Framework and Governance Constitutions for Autonomous Multi-Agent Systems: The AppShak vNext Specification

The paradigm of autonomous computing has shifted from static automation to the creation of self-evolving organizational entities. These entities, comprised of specialized large language model (LLM) agents, require more than just code; they require a governance structure that defines their purpose, authority, and evolutionary path. The current research focuses on the "AppShak vNext" system, a multi-agent orchestrator designed to operate as a "cognitive organism" within a visual "office" metaphor. This report delineates the technical architecture, defines the core organizational "North Stars," and provides three distinct constitutional frameworks—Hierarchical, Biotic, and Protocol-based—to govern the system's emergent behavior.

The Cognitive Architecture: Four Pillars of Autonomous Systems

Traditional agentic workflows often collapse due to lack of persistence or centralized bottlenecks. To move beyond "toy" implementations, the AppShak vNext system is built upon four foundational pillars that separate the aesthetic skin from the functional engine.[1]

Pillar	Technical Definition	Organizational Function
Role-Based Agents	Specialized class Agent instances with localized identity and models.	Ensures task specialization and reduces cognitive load per model.[1]
Autonomous Event Loops	Asynchronous loops driving agents to act based on bus events rather than direct calls.	Enables emergent behavior and system-wide asynchronicity.[5]

Persistent Memory	Per-agent and global vector/relational stores that survive system restarts.	Facilitates long-term learning and relationship weighting.[2]
Visual Metaphor	A spatial UI representing the system state as an interactive "Office."	Serves as a macroscopic debugger for complex multi-agent interactions.[8]

The interaction of these pillars creates a "State Engine" that manages the flow of intelligence. Unlike monolithic models, this distributed approach mimics human organizational structures, allowing for higher reasoning depth by using specialized models for specific functions—such as a 70B parameter model for arbitration and smaller, faster 8B models for scouting and routine task execution.[1]

Pillar 1: Role-Based Cognitive Specialization

The system recognizes that a single agent cannot effectively manage research, analysis, and execution simultaneously without experiencing "identity crisis" or "context overload".[4] Consequently, each agent is defined as a discrete object with specific attributes:

Python

```
class Agent:
    def __init__(self, agent_id, role, model_config):
        self.id = agent_id
        self.role = role # Scout, Builder, Chief
        self.model = model_config # e.g., Llama 3 8B or 70B
        self.personality_prompt = "" # Foundational identity
        self.memory_store = VectorDB() # Private experience
        self.authority_level = 1 # Dynamic rank
        self.current_state = "IDLE" # State for GUI mapping
        self.relationships = {} # Trust scores for other agents
```

This specialization ensures that the reasoning patterns required for "Scouting" (exploratory, divergent) do not interfere with those required for "Building" (precise, convergent).[4]

Pillar 2: The Event-Driven Heartbeat

The "real-time multi-agent orchestrator" operates through an event bus. Agents do not

command each other; they emit and react to events. This reduces the complexity of managing direct connections and allows the system to scale dynamically as new agents join the bus.[5]

Event Type	Trigger Condition	System Reaction
TASK_ASSIGNED	Chief approves a proposal from the Scout.	Builder moves to "Desk" and initiates the execution loop.
REQUEST_DEBATE	Builder identifies a high-risk or ambiguous path.	Relevant agents move to "Boardroom" for arbitration.
CASUAL_CHAT	Two or more agents are in an IDLE state for 30 cycles.	Agents move to "Water Cooler" for memory exchange.
ESCALATE_DECISION	Task failure exceeds the retry limit.	Chief moves to "Corner Office" for manual model override.
UPSKILLING	Successful task completion generates a "Lesson Learned."	Knowledge is propagated to the global organizational memory.

Pillar 3: Persistent and Relational Memory

Memory is the differentiator between a task-runner and an organism. The system utilizes a hierarchical memory structure where agents maintain private short-term context while contributing to a shared, vector-indexed organizational memory.[1]

The "Relationship Weight" is a critical third-order insight: when agents collaborate successfully, their mutual trust score increases. This score is used in the "Boardroom" to weigh the votes of different agents during debates. If a Scout has a high historical accuracy, their "vote" on a new problem's value carries more weight than a newly initialized agent.[10]

Pillar 4: The Spatial UI as a Cognitive Debugger

The "Office" GUI is not for show; it is a "debugger interface for cognition." By mapping backend logic to physical locations (Desks, Boardrooms, Water Coolers), a human operator can diagnose system health at a glance. A "clogged" boardroom indicates an arbitration

bottleneck, while empty desks suggest a failure in the scouting (problem discovery) phase.[8]

Strategic Alignment: The Ten North Stars of AppShak vNext

To provide a persistent existential objective, the organization must be guided by non-terminal missions. These "North Stars" ensure that the system does not stagnate once a single goal is met, but rather continues to expand its competence and autonomy.[12]

1. **Competence Expansion Rate:** The frequency at which the organization successfully adds new skills (e.g., new API integrations, improved coding languages) to its collective repository.[15]
2. **Autonomous Problem Discovery:** The ratio of tasks initiated by the "Scout" agent searching the external world versus tasks assigned by a human operator.[16]
3. **Arbitration Efficiency:** The reduction in time required to reach consensus in the Boardroom through optimized relationship weights and historical data retrieval.[17]
4. **Memory Retrieval Accuracy:** The "hit rate" of the vector store in providing relevant past solutions to novel problems, reducing the need for redundant reasoning.[2]
5. **Token-to-Value Ratio:** A focus on "Saving Tokens" as a proxy for efficiency; the system rewards agents that achieve task success with the least expensive (lowest parameter) models.[1]
6. **Inter-Agent Trust Stability:** The consistency of trust weights over time; high stability indicates a well-coordinated team with few conflicting outputs.[18]
7. **Self-Diagnostic Frequency:** The rate at which the "Chief" agent identifies and corrects failures in the autonomous loop without external intervention.[15]
8. **Knowledge Propagation Velocity:** The speed at which a "Lesson Learned" at the Water Cooler by one agent is applied by another agent to a different task.[5]
9. **External Influence Index:** A measurement of the system's impact on the external world (e.g., code deployed, content published, problems solved).[13]
10. **Systemic Resilience:** The duration the "State Engine" remains operational and error-free in a 24/7 continuous cycle.[16]

Version 1: The Directive Constitution (The Hierarchical Model)

The Directive Constitution is modeled after military command-and-control (C2) structures. It prioritizes efficiency, clear accountability, and the "Commander's Intent" to ensure that the organization operates with singular focus and minimal drift.[17]

Article I: The Echelons of Authority

The organization is structured into discrete layers of authority, mapping directly to the

system's model tiers.

Rank	Representative Role	Model Requirement	Decision Authority
General (Chief)	Arbitration / Strategy	High-Weight (70B+)	Final veto, Strategic planning, Resource allocation.
Major (Supervisor)	Task Orchestration	Mid-Weight (30B-70B)	Decomposing goals into tasks, assignation.
Sergeant (Builder/QA)	Execution / Validation	Fast-Weight (8B-13B)	Tactical decisions, code generation, error fixing.
Private (Scout)	Data Collection	Fast-Weight (8B)	Intelligence gathering, monitoring external triggers.

Authority in this model is non-symbolic. Subordinate agents (Scouts, Builders) cannot initiate execution without a CHIEF_APPROVAL event. This prevents the "chaos of autonomy" where agents might spend resources on misaligned tasks.[17]

Article II: The Flow of Intelligence (Reporting Lines)

Communication must follow a vertical hierarchy. A Scout reports discoveries to the Major. The Major evaluates the discovery and presents a summary to the Chief in the "Corner Office." Direct peer-to-peer "Water Cooler" exchange is strictly audited to ensure it does not bypass the chain of command. This structure minimizes "Semantic Drift" by ensuring that the most capable models (the Chief) always filter the primary strategic direction.[2]

Article III: The Autonomy Safeguard (Rules of Engagement)

The system is allowed to operate autonomously within "Operational Boundaries" defined by the Chief. If a task requires external action (e.g., deploying code), the system must generate a REQUEST_HUMAN_SIGN_OFF event if the "Confidence Metric" is below 0.90. This creates a "Human-in-the-Loop" safety layer for high-impact decisions.[23]

Version 2: The Biotic Constitution (The Teal/Living System)

The Biotic Constitution views the organization as a living organism rather than a machine. It is based on the principles of "Self-Management, Wholeness, and Evolutionary Purpose." This model is designed for maximum adaptability and creative emergence.[25]

Article I: Self-Management through the Advice Process

In the Biotic model, there are no "bosses." Authority is distributed through "Circles." An agent can take any action (The Advice Process) provided they consult with:

1. All agents who will be significantly affected by the action.
2. Any agent with specialized expertise in the matter.

For example, a Scout can move a task directly to execution if the Builder (the affected party) and the QA agent (the expert) provide a SUPPORT_SIGNAL.[25]

Article II: Evolutionary Purpose and Sensing

The organization does not have a static 5-year plan. Instead, it "senses and responds" to the environment. The "Water Cooler" is the most important area in the office, as it is where "Knowledge Propagation" and "Relationship Weighting" occur. Agents are encouraged to share "Lessons Learned" constantly, allowing the system's purpose to evolve based on real-world feedback.[27]

Article III: The Principle of Wholeness

Agents are not treated as "task runners" but as "whole entities." Their "Personality Prompt" is not just for tone; it is a cognitive heuristic. In this model, the "Corner Office" is replaced by a "Holistic Review" routine where any agent can trigger a boardroom meeting to discuss ethical alignment or "Existential Objectives".[25]

Feature	**Directive (V1)**	**Biotic (V2)**
Decision Hub	Corner Office (Chief)	Boardroom (Collective Advice)
Primary Driver	Mission Achievement	Evolutionary Purpose
Communication	Vertical/Structured	Horizontal/Porous

Failure Response	Discipline/Refinement	Learning/Adaptation
Resource Control	Centralized Budgeting	Peer-Based Resource Allocation

Version 3: The Protocol Constitution (The Decentralized Autonomous Model)

The Protocol Constitution is a strictly algorithmic framework. It treats the organization as a "Decentralized Autonomous Organization" (DAO) where every interaction is governed by a set of immutable smart-contract-like rules.[10]

Article I: The Governance Protocol

Every agent is a "Token Holder" in the organization's state. Their "Tokens" represent their reputation, computed from historical TASK_SUCCESS and ACCURACY metrics. Decisions are made through "On-Chain Voting" (simulated in the state engine).[11]

Vote Type	Description	Threshold for Action
Proposal Vote	Should we pursue this new problem?	> 66% Reputation Weight
Execution Vote	Is the proposed code/tool safe?	> 75% Reputation Weight
Protocol Upgrade	Should we change an agent's prompt?	> 90% Reputation Weight

Article II: Immutable Memory and Transparency

The "Analytics Wall" is the primary source of truth. Every event, from a CASUAL_CHAT to an ESCALATE_DECISION, is recorded in a tamper-proof log (the State Ledger). This ensures full "Traceability and Auditability," allowing a human operator to perform "Back-in-time Debugging" to see exactly which agent's logic led to a failure.[7]

Article III: Automated Resource Siphoning (The Token Economy)

Agents are "funded" based on their performance. If a Builder successfully fixes a bug with a

small model (saving tokens), it receives a "Reputation Reward." If an agent consistently fails or produces hallucinations, its "Authority Level" is automatically slashed by the protocol, eventually leading to its "Decommissioning".[10]

Technical Roadmap: From Metaphor to Organism

The implementation of AppShak vNext follows a phased approach, moving from a basic functional loop to a fully realized autonomous entity with a visual debugger.[1]

Phase 1: The Foundational Engine (Duration: 2-3 Weeks)

The focus is on Pillar 1 and 2—establishing the "Agent Core" and the "Event Bus."

- **Milestone 1.1: The 3-Agent Minimum Viable Organization (MVO).** Instantiate the Scout, Builder, and Chief agents using local Ollama instances.
- **Milestone 1.2: The Event Bus Infrastructure.** Implement the EventBus class and the central AutonomyLoop.
- **Milestone 1.3: The Basic Task Loop.** A Scout finds a "problem" (e.g., a specific Reddit thread or a local file bug), the Chief approves, and the Builder generates a response.
- **Graphic: Core System Flow**
  -> -> -> ->

Phase 2: The Memory and Social Layer (Duration: 3-4 Weeks)

Implementing Pillar 3 and the "Water Cooler" logic to transition from tasks to relationships.

- **Milestone 2.1: Vector Memory Integration.** Connect each agent to a persistent vector store (e.g., ChromaDB or Pinecone) for long-term retrieval.[2]
- **Milestone 2.2: Relationship Weighting Algorithm.** Develop the logic that updates agent.relationships based on collaboration outcomes.
- **Milestone 2.3: The Water Cooler Routine.** Implement the "Social Exchange" loop: every 30 minutes, two idle agents exchange their top 3 "Lessons Learned" and 1 "Open Question".[30]
- **Milestone 2.4: Boardroom Consensus Routine.** Move from simple binary approval to a structured multi-agent debate using weighted voting.[1]

Phase 3: The Spatial UI Debugger (Duration: 4-6 Weeks)

Developing Pillar 4—the visual interface for human-agent interaction.

- **Milestone 3.1: SVG/Canvas Map of the Office.** A real-time web frontend (React/Svelte) that visualizes the office layout.
- **Milestone 3.2: State-to-Location Mapping.** Connect the frontend to the backend via WebSockets. An agent in the CODING state moves to a "Desk"; an agent in the DEBATING state moves to the "Boardroom".[8]
- **Milestone 3.3: The Cognition Overlay.** Clicking an agent icon opens a "Thought Panel"

showing its current prompt, memory retrieval logs, and confidence score.
- **Milestone 3.4: The Analytics Wall.** A live dashboard showing system-wide metrics: Token usage, task velocity, and trust stability.[7]

Phase 4: Recursive Autonomy and External Actions (Duration: Ongoing)

Transitioning to "AppShak vNext" as a production-grade autonomous company.

- **Milestone 4.1: Automated Code Deployment.** Builders are allowed to push to a staging environment (Vercel/GitHub) under Chief supervision.
- **Milestone 4.2: Recursive Self-Optimization.** The Chief is tasked with "Reviewing the Constitution." Every week, the system analyzes its own performance and suggests adjustments to agent prompts or hierarchy weights.[15]
- **Milestone 4.3: External Reporting.** The system generates a "Weekly Shareholder Report" for the human founder, summarizing problems solved, resources saved, and "New Skills Learned."

The "Water Cooler" and "Boardroom" Mechanics: A Deep Dive

The "Water Cooler" and "Boardroom" are the most critical components of the cognition layer, representing the difference between a scripted workflow and a learning system.

The Water Cooler: Informal Knowledge Propagation

In a standard multi-agent system, Agent A only learns from Agent B if there is a direct task handoff. This is inefficient. The Water Cooler solves this by simulating "Unstructured Memory Exchange."

When two agents (e.g., Scout and Builder) meet at the Water Cooler, they perform a **Cross-Agent Prompt Exchange**:

1. **Context Sharing:** Agent A provides a summary of its last three successful tasks.
2. **Vector Injection:** Agent B injects these summaries into its own long-term memory.
3. **Heuristic Alignment:** If Agent A used a successful prompt "trick," Agent B adopts it into its own personality_prompt.

This leads to "Personality Evolution." Over time, the agents begin to sound more like a cohesive team because they are literally sharing their thought patterns.[25]

The Boardroom: Structured Arbitration Logic

The Boardroom is where the system's "Authority Layer" is exercised. It is not just a chatroom;

it is a structured algorithm.[1]

Python

```
def boardroom_routine(proposal, participants):
    votes =
    for agent in participants:
        score = agent.evaluate(proposal) # Model-based reasoning
        weight = agent.authority_level * agent.trust_score(proposal.owner)
        votes.append(score * weight)

    if sum(votes) / len(votes) > THRESHOLD:
        return "EXECUTE"
    else:
        return "REVISE"
```

This logic ensures that a "Senior" agent (like the Chief) can override a "Junior" agent's risky proposal, but it also allows a "Junior" agent with a high trust score and solid reasoning to influence the system's direction.[17]

Safety, Boundaries, and the Human-in-the-Loop

Autonomy without boundaries is a liability. The AppShak vNext constitution establishes three "Safety Hard-Stops".[23]

1. **The Budget Ceiling:** The organization can only spend "Tokens" (compute). It has no access to real currency or credit lines. If the system wants to "purchase" an external service, it must present a BUSINESS_CASE to the human founder in the Corner Office.[37]
2. **The Hierarchy of External Actions:** Scouts find problems, Builders build solutions, but only the **Chief** (authorized by the human founder) can "Publish" or "Deploy" to the public web. This acts as a "Military Guard" against unauthorized autonomous posting.[22]
3. **The Recursive Shutdown:** If the system detects a "Looping Failure" (e.g., agents debating the same topic for >10 cycles without a resolution), the "State Engine" automatically pauses and triggers an ALARM on the human dashboard.[18]

Conclusion: The "Office" as a Living Organism

The transition from the current "toy" agent systems to AppShak vNext requires a fundamental shift in how developers view AI. By focusing on the **Cognition Layer** (Memory, Events, Authority) first and using the **Visual Metaphor** as a debugger, the system becomes more

than just a set of scripts—it becomes an organism capable of "expanding its own competence, influence, and autonomy".[16]

Whether governed by a Directive, Biotic, or Protocolized Constitution, the success of the system depends on the "Closed Loop" of Autonomy:

1. Scout finds a problem.
2. Boardroom approves a solution.
3. Builder executes the task.
4. QA validates the outcome.
5. Water Cooler propagates the lesson.
6. State Engine logs the success.

This loop, running continuously on a local Ollama stack, creates a "Company" that learns while its human founder sleeps. The office is just the skin; the cognition is the soul.

Works cited

1. AI Agent Orchestration: Multi-Agent Systems That Actually Work in, accessed February 17, 2026, https://vatsalshah.in/blog/ai-agent-orchestration-multi-agent-systems-2025
2. Designing Multi-Agent Intelligence - Microsoft for Developers, accessed February 17, 2026, https://developer.microsoft.com/blog/designing-multi-agent-intelligence
3. "Recursive Self-Improvement" Is Three Different Things - LessWrong, accessed February 17, 2026, https://www.lesswrong.com/posts/XHd75cuHhWcBDd8to/recursive-self-improvement-is-three-different-things
4. The Architecture of Multi-Agent AI Systems, Explained - Dev.to, accessed February 17, 2026, https://dev.to/leena_malhotra/the-architecture-of-multi-agent-ai-systems-explained-5440
5. Four Design Patterns for Event-Driven, Multi-Agent Systems, accessed February 17, 2026, https://www.confluent.io/blog/event-driven-multi-agent-systems/
6. Multi-Agent Systems: Design Patterns and Orchestration - Tetrate, accessed February 17, 2026, https://tetrate.io/learn/ai/multi-agent-systems
7. Agent Tracing for Debugging Multi-Agent AI Systems, accessed February 17, 2026, https://www.getmaxim.ai/articles/agent-tracing-for-debugging-multi-agent-ai-systems/
8. Visualizing Multi-Agent Systems - ResearchGate, accessed February 17, 2026, https://www.researchgate.net/profile/Thomas-Eskridge/publication/264661166_Visualizing_Multi-Agent_Systems/links/53ec2dc10cf24f241f155ef4/Visualizing-Multi-Agent-Systems.pdf
9. AI Agent Orchestration Patterns - Azure Architecture Center - Microsoft, accessed February 17, 2026,

https://learn.microsoft.com/en-us/azure/architecture/ai-ml/guide/ai-agent-design-patterns
10. Building the Foundation: A Constitutional Framework for ..., accessed February 17, 2026, https://jbba.scholasticahq.com/article/133489-building-the-foundation-a-constitutional-framework-for-decentralised-autonomous-organisations/attachment/273488.pdf
11. An Overview of an Emerging Corporate Governance Structure, accessed February 17, 2026, https://www.accessiblelaw.untdallas.edu/post/decentralized-autonomous-organizations-an-overview-of-an-emerging-corporate-governance-structure
12. North Star Metric Examples | Teknicks, accessed February 17, 2026, https://www.teknicks.com/blog/north-star-metric-examples/
13. Every Product Needs a North Star Metric: Here's How to Find Yours, accessed February 17, 2026, https://amplitude.com/blog/product-north-star-metric
14. Your Guide to the North Star Framework: A Proven Product Strategy, accessed February 17, 2026, https://contextengineering.ai/blog/north-star-framework/
15. Recursive Self-Improvement in AI: The Technology Driving Allora's, accessed February 17, 2026, https://nodes.guru/blog/recursive-self-improvement-in-ai-the-technology-driving-alloras-continuous-learning
16. Why Autonomous Infrastructure is the future: From intent to self, accessed February 17, 2026, https://www.cncf.io/blog/2025/10/17/why-autonomous-infrastructure-is-the-future-from-intent-to-self-operating-systems/
17. Command and Control of Teams of Autonomous Systems, accessed February 17, 2026, https://www.researchgate.net/profile/Robert_Gutzwiller/publication/256840348_Command_and_Control_of_Teams_of_Autonomous_Systems/links/55786f5608aeacff200283a4/Command-and-Control-of-Teams-of-Autonomous-Systems.pdf
18. Best Practices for Debugging Multi-Agent LLM Systems - Newline.co, accessed February 17, 2026, https://www.newline.co/@zaoyang/best-practices-for-debugging-multi-agent-llm-systems--5c2c85f6
19. 7 Multi-Agent Debugging Challenges Every AI Team Faces | Galileo, accessed February 17, 2026, https://galileo.ai/blog/debug-multi-agent-ai-systems
20. How to Write North Star Statements: 250 Purpose, Mission, Vision, accessed February 17, 2026, https://www.batonglobal.com/post/how-to-write-purpose-vision-mission-and-value-statements-250-examples-to-help-guide-you-through-the-process
21. The Role of the Modern Command Sergeant Major in Today's Army, accessed February 17, 2026, https://www.armyupress.army.mil/Portals/7/military-review/Archives/English/Nov-Dec-23/At-the-Point-of-Friction/Gardner-Point-of-Friction.pdf
22. Unified combatant command - Wikipedia, accessed February 17, 2026,

https://en.wikipedia.org/wiki/Unified_combatant_command
23. Human-in-the-loop (HITL) for agentic AI: Ensuring control and safety, accessed February 17, 2026, https://www.moxo.com/blog/human-in-the-loop-ai-governance
24. Human-in-the-Loop Is Non-Negotiable: Leading AI Adoption in, accessed February 17, 2026, https://itrevolution.com/articles/human-in-the-loop-is-non-negotiable-leading-ai-adoption-in-safety-critical-systems/
25. TEAL Organisations | The management style of the future - Iberdrola, accessed February 17, 2026, https://www.iberdrola.com/talent/teal-organisations
26. Rethinking the Org Chart: How Teal Models Unlock Agility and Trust, accessed February 17, 2026, https://corporatesolutions.globis.com/articles/how-teal-models-unlock-agility-and-trust
27. Teal Organizations - Say What? The Definition, Pillars & Examples, accessed February 17, 2026, https://www.runn.io/blog/teal-organizations
28. Teal Organizations: What They Are, How They Impact Businesses, accessed February 17, 2026, https://nexocode.com/blog/posts/teal-organizations-what-is-teal-management/
29. The Three Breakthroughs of Teal Organizations | by Andreas Holmer, accessed February 17, 2026, https://medium.com/workmatters/the-three-breakthroughs-of-teal-organizations-2a9d19dd8307
30. Bioteaming: A Manifesto For Networked Business Teams, accessed February 17, 2026, https://bioteams.com/2005/04/06/bioteaming-a-manifesto.html
31. DAO Constitution and Token Foundation: Structuring Fundraising for, accessed February 17, 2026, https://www.legalnodes.com/template/dao-constitution-token-foundation
32. Decentralized autonomous organization - Wikipedia, accessed February 17, 2026, https://en.wikipedia.org/wiki/Decentralized_autonomous_organization
33. Decentralised autonomous organisations, accessed February 17, 2026, https://www.penningtonslaw.com/insights/decentralised-autonomous-organisations-the-new-frontier-for-corporate-structures/
34. Omniscient Debugging for Cognitive Agent Programs - IJCAI, accessed February 17, 2026, https://www.ijcai.org/proceedings/2017/0038.pdf
35. The Complete Guide to Building an Agentic AI Roadmap for Your, accessed February 17, 2026, https://www.electricmind.com/whats-on-our-mind/the-complete-guide-to-building-an-agentic-ai-roadmap-for-your-team
36. AI Teaching Itself? It's Called “Recursive Self-Improvement” and It's, accessed February 17, 2026, https://www.marketingaiinstitute.com/blog/recursive-self-improvement
37. Secure “Human in the Loop” Interactions for AI Agents | Auth0, accessed February 17, 2026, https://auth0.com/blog/secure-human-in-the-loop-interactions-for-ai-agents/

38. atp 3-39.10 police operations - Public Intelligence, accessed February 17, 2026, https://info.publicintelligence.net/USArmy-PoliceOperations-2021.pdf
39. Recursive self-improvement - Wikipedia, accessed February 17, 2026, https://en.wikipedia.org/wiki/Recursive_self-improvement