Engineering Patterns for Production Multi-Agent Governance at Portfolio Scale: An Operationalization of Enabling Bureaucracy for LLM Agent Systems

Asif Waliuddin
NXTG.AI

Abstract

That formal structure can enable rather than constrain operational autonomy is an established result — Adler and Borys's 1996 "enabling bureaucracy," its cybernetic roots (Beer, Ostrom, Ashby, Koestler), and several 2025–2026 LLM-agent confirmations. This paper does not claim that thesis as a discovery. We contribute an engineering operationalization: a documented production deployment (ASIF, an internal portfolio-governance framework) in which two autonomous Claude Code agents govern 18 software projects across two machines through a 7-layer stack spanning zero-friction permissions, extended reasoning, heartbeat activation, arousal-based mode selection, a trust-gated 4-tier decision matrix, general delegation, and parallelizable task structure. Atop the stack sit a 23-principle constitution across 3 override tiers, a file-based directive protocol that survives process death and machine reboots, and an inverted Constitutional AI methodology that extracts principles from a 54-decision precedent corpus. Our contributions are three engineering patterns and one methodology: (1) portfolio-scale federated governance — multi-project, multi-machine directive propagation that prior single-system frameworks identify as an open gap; (2) the 7-layer Autonomy Enablement Stack as a concrete configuration pattern, presented without ablation evidence; (3) a federated file-based Directive Protocol requiring no message broker, shared runtime database, or workflow engine — coordination relies on git remotes and cron synchronization; (4) the inverted Constitutional AI methodology. A single production observation (a four-project cleanup at roughly 2.1x versus an estimated, not measured, sequential counterfactual, under a parallelization instruction) illustrates the patterns in operation; it is documentation, not evidence for the theoretical claim. We position this work as a workshop/preprint contribution.

1 Introduction

1.1 What This Paper Is and Is Not

This paper is an engineering-patterns paper. It documents how to operationalize enabling formal structure for LLM agent systems at portfolio scale. It is not a theoretical contribution; the theoretical claim that formal structure enables rather than constrains autonomy at higher organizational levels is well-established and is treated here as an inherited result.

The paper has three audiences. **Engineers** building production multi-agent systems can use the 7-layer stack and the Directive Protocol as a reference architecture. **Governance researchers** working on agent oversight can use the inverted Constitutional AI methodology and the trust-level ratchet as a comparison point. **Reviewers** evaluating the paper should evaluate it on the engineering contribution and the honest treatment of prior art, not on a theoretical inversion claim.

1.2 The Inherited Theoretical Result

We treat the following as established prior art and do not claim to contribute it:

- **Enabling vs. coercive formalization** (Adler & Borys 1996, *Administrative Science Quarterly* 41:61–89) [1]: The canonical management-theory statement that formal rules can function as resources that empower workers rather than constraints that suppress agency. The "governance enables autonomy" framing in an earlier draft of this paper is Adler–Borys enabling formalization applied to LLM agents.
- **Viable System Model** (Stafford Beer 1972, *Brain of the Firm*) [2]: Operational units are given as much autonomy as possible consistent with system cohesion; subsystems S1–S5 provide the structural "glue" that enables operational autonomy. The recursive holonic structure of ASIF's project NEXUSes within a portfolio dashboard is structurally a VSM application.
- **Polycentric governance** (Ostrom 1990, *Governing the Commons*; Nobel 2009) [3]: Locally-devised institutional rules at multiple feedback levels enable self-governing emergence. ASIF's 18-project federated NEXUS system is structurally polycentric in Ostrom's sense.
- **Requisite variety** (Ashby 1956) [4]: Governance structures must possess sufficient variety to coordinate autonomous agents. Foundational cybernetics.
- **Holons** (Koestler 1967) [5]: Entities that are simultaneously autonomous wholes and parts of a larger structure. ASIF's project NEXUSes within a portfolio dashboard form a holarchy in Koestler's sense.
- **Contemporaneous LLM-agent confirmations**: LLM Constitutional Multi-Agent Governance (CMAG; de Curtò and de Zarzà 2026) [6] measured autonomy preservation 0.985 with constitutional governance across 80 agents. Self-Organizing LLM Agents (Dochkina 2026) [7] reported that a Sequential protocol enabling autonomous role selection outperforms centralized coordination by 14% ($p < 0.001$), with a 44% quality spread (Cohen's $d = 1.86$) across a 25,000-task experiment — the same architectural prescription (minimal structure, autonomous role execution) this paper operationalizes. AgentCity (Ruan and Zhang 2026) [8] made the architectural claim that "individual alignment produces collective alignment without top-down rules" almost verbatim.

What this paper observes — that giving an LLM agent a clear governance frame improves its willingness to act autonomously within scope — is the LLM-specific instance of a result the field already knows. Our contribution is what comes after that observation: how to build the governance frame at portfolio scale, in code, in a way other teams can adopt.

1.3 What We Actually Contribute (Engineering)

1. **Portfolio-scale federated governance**. MI9 [9], the Layered Governance Architecture [10], Policy Cards [11], and ArbiterOS [12] are all framed as governance for *a system*. The open gap, explicitly noted in MI9 and LGA, is governance *across a portfolio of independently developed systems*. ASIF spans 18 projects across two machines with cross-project directive propagation. We document the protocol that makes this work: machine sovereignty, parity-scheduled ID allocation to prevent collision, file-based directive queues in per-project NEXUS files, and a portfolio dashboard that aggregates without centralizing.

2. **The 7-layer Autonomy Enablement Stack as a configuration pattern**. We document the seven layers as a reproducible engineering pattern: zero-friction permissions, extended reasoning, heartbeat activation, arousal-based mode selection, a 4-tier decision matrix with trust-level gating, general delegation instructions, and parallelizable task structure. We are explicit that we have not yet conducted ablation studies. The "all seven layers must be present" claim in an earlier draft was speculative. We present the stack as a working configuration and flag ablation as required future work.

3. **A federated Directive Protocol that survives infrastructure failure**. The protocol uses human-readable markdown files committed to git with auto-sync every five minutes. It survives process death, machine reboots, and LLM context compaction. No Temporal worker, Kafka topic, Redis lock, or workflow engine is required; coordination relies on git remotes and a synchronization cron. We show the protocol structure and the collision-resistant ID scheme (machine-prefixed identifiers with parity scheduling).

4. **Inverted Constitutional AI methodology**. Anthropic's Constitutional AI (Bai et al. 2022) [13] defines principles top-down and trains the model against them. ASIF's constitution was derived by the inverse procedure: 54 historical decisions made by the system principal over a 13-day period were collected as a precedent corpus, principles were extracted by clustering decision rationales, and the resulting 23 principles across 3 override tiers were structured for runtime decision-matrix consultation rather than training-time alignment. This is a methodology contribution distinct from the constitution-as-artifact contribution.

1.4 What We Do Not Contribute

We are explicit about what is not novel:

- The thesis that formal structure enables operational autonomy is 30 years old [1] with deeper cybernetic roots [2–5].
- The application of this thesis to LLM agent systems has been independently made at least four times in 2025–2026 (CMAG, AgentCity, Self-Organizing LLM Agents, GaaS) [6–8, 14].
- The constitutional-AI-as-runtime-constraint pattern is established (TrustAgent, ArbiterOS, Anthropic CAI) [12, 13, 15]. Our methodology inversion is the contribution; the artifact pattern is not.
- The decision-matrix-with-trust-tiers pattern is established in operational risk frameworks. Our specific 4-tier matrix is documentation, not invention.

The honest claim of this paper is: *we built the engineering — here are the patterns and protocols other teams can use.*

2 Related Work

We organize prior art in four bands: (A) the foundational management-theory and cybernetics result; (B) contemporaneous 2025–2026 LLM-agent confirmations; (C) production governance frameworks for single agent systems; (D) production multi-agent orchestration.

2.1 The Foundational Theoretical Result (Not Our Contribution)

Adler & Borys (1996) [1], "Two Types of Bureaucracy: Enabling and Coercive," *Administrative Science Quarterly* 41:61–89. The canonical statement of the thesis. Distinguishes ENABLING formalization (rules-as-resource) from COERCIVE formalization (rules-as-constraint). An earlier draft framed this paper as a "counterintuitive discovery"; that framing was, in fact, the LLM-agent restatement of Adler–Borys. We treat Adler–Borys as the inherited theoretical foundation.

Beer, Stafford (1972) [2], *Brain of the Firm.* The Viable System Model. Five-subsystem recursive structure (S1 operations, S2 coordination, S3 internal regulation, S4 outside-and-future, S5 identity/policy) provides the structural "glue" enabling operational autonomy. Two recent practitioner essays [16, 17] explicitly apply VSM to "Autonomous AI Organisations." ASIF's recursive project-NEXUS-within-portfolio structure is a VSM instantiation.

Ostrom, Elinor (1990) [3], *Governing the Commons*; Nobel Prize 2009. Empirically established that polycentric institutional arrangements with locally-devised rules at multiple feedback levels enable self-governing emergence. ASIF's federated NEXUS system is structurally polycentric: each project has local governance authority, the portfolio layer has cross-project authority, and rules at each level interact via the directive propagation protocol.

Ashby, W. Ross (1956) [4], *An Introduction to Cybernetics.* The Law of Requisite Variety: only variety can master variety. Governance structures must possess variety sufficient to coordinate autonomous agents. The 7-layer stack and the multi-tier constitutional structure can be read as ASIF's variety-engineering.

Koestler, Arthur (1967) [5], *The Ghost in the Machine.* Holons as entities simultaneously autonomous and embedded. Holonic Manufacturing Systems (Hannover papers, IMS consortium, 1990s–2000s) [18] implemented "autonomy + cooperation" via formal protocols. ASIF's project teams are agentic holons.

2.2 Contemporaneous 2025–2026 LLM-Agent Confirmations

CMAG — LLM Constitutional Multi-Agent Governance (de Curtò and de Zarzà, 2026) [6]. Two-stage framework with hard constraint filter and soft penalized-utility optimization. Headline result: Ethical Cooperation Score 0.741 (+14.9%) with autonomy preservation 0.985 and integrity 0.995, across an 80-agent network. Unconstrained baseline achieves higher raw cooperation (0.873) but autonomy collapses to 0.867. CMAG is more empirically rigorous than ASIF on the central enabling-governance claim. ASIF differs by operating at portfolio scale (18 projects, 2 machines) rather than single-experiment scale (80 agents in one system).

Drop the Hierarchy and Roles: How Self-Organizing LLM Agents Outperform Designed Structures (Dochkina, 2026) [7]. Minimal strategic structure with autonomous role selection (a Sequential protocol) outperforms centralized coordination by 14% ($p < 0.001$), with a 44% quality spread between protocols (Cohen's $d = 1.86$) across a 25,000-task experiment. Paraphrasing its architectural prescription: minimal strategic structure constrains coordination while leaving role execution autonomous — the sharpest empirical articulation of the thesis. Our paper does not match this empirical depth and does not attempt to. We adopt the architectural prescription as the design philosophy and document the engineering operationalization.

AgentCity: Constitutional Governance for Autonomous Agent Economies via Separation of Power (Ruan and Zhang, 2026) [8]. Three-branch separation of power: agents legislate smart contracts, deterministic software executes, humans adjudicate. Central architectural claim: "individual alignment produces collective alignment without top-down rules." Pre-registered with no empirical results yet. AgentCity uses blockchain; ASIF uses git-committed markdown. Both are

architectural-layer arguments at the empirical layer.

Institutional AI: Governing LLM Collusion in Multi-Agent Cournot Markets via Public Governance Graphs (Bracale Syrnikov et al., 2026) [19]. Counter-finding important to acknowledge: the prompt-only Constitutional baseline yields no reliable improvement under optimization pressure, illustrating that declarative prohibitions do not bind under sustained pressure. The institutional regime cuts severe collusion 50%→5.6% ($d = 1.28$). This both supports ASIF's runtime-enforcement architecture (institutional governance works) and threatens any version of the claim that relies on prompt-only constitution (mere constitution-in-prompt does not).

Governance-as-a-Service (GaaS) (Gaurav et al., 2025) [14]. Modular enforcement layer with Trust Factor and graduated enforcement (coercive, normative, adaptive). Treats governance as infrastructure-level alignment. ASIF adopts this framing.

2.3 Production Governance Frameworks for Single Agent Systems

MI9 (Wang et al., 2025) [9]: the most architecturally comprehensive runtime governance framework — agency-risk indices, agent-semantic telemetry, continuous authorization, FSM conformance, drift detection, graduated containment. Governs *a system* as a runtime control plane. Explicit acknowledgment that portfolio-level coordination across independently developed systems is an open gap.

Layered Governance Architecture (LGA) (Ge, 2026) [10]: 4-layer framework (execution sandboxing, intent verification, zero-trust inter-agent authorization, immutable audit logging) achieving 96% interception rate with ∼980 ms P50 latency on 1,081 tool-call samples. Single-system. The author explicitly notes "no existing work addresses all three threat classes within a unified, independently deployable governance architecture."

ArbiterOS (Xu et al., 2025) [12]: governance-first paradigm for agent engineering, Agent Constitution Framework (ACF), Kernel-as-Governor mode. Intra-agent governance.

Policy Cards (Mavračić, 2025) [11]: machine-readable, version-controlled governance artifacts encoding allow/deny rules with crosswalks to NIST AI RMF, ISO 42001, EU AI Act. Schema artifact, not deployed portfolio governance.

TrustAgent (Hua et al. 2024) [15]: three-stage agent constitution (pre-planning, in-planning, post-planning). Single-agent.

AgentSpec (Wang, Poskitt, and Sun 2025) [20]: DSL for runtime constraints with trigger/predicate/enforcement rules.

GuardAgent (Xiang et al. 2024) [21]: explicit guardrail agent checking target agent actions.

Singapore IMDA Model AI Governance Framework for Agentic AI (January 2026) [22]: the world's first regulatory standard specifically for autonomous agents. Single-system focus.

Additional context: Shavit et al. (2023) [23] define operational safety practices for agentic systems (constraining action space, legibility, monitoring, attributability, interruptibility) — vocabulary-setting, but practices rather than an architecture for portfolio-scale directive propagation. TRiSM for Agentic AI (Raza et al. 2025) [24] surveys trust/risk/security management and confirms that multi-agent governance at scale remains a primary open challenge. Anthropic's published constitution for Claude [25], its measurement of agent autonomy in practice [26], and the Levels of Autonomy framework (Feng et al. 2025) [27] provide the constitutional baseline and graduated-autonomy scales to which ASIF's trust tiers relate.

2.4 Production Multi-Agent Orchestration

(Brief — a full survey table is provided in Appendix E.) AutoGen [28], CrewAI [29], LangGraph [30], MetaGPT [31], ChatDev [32], Magentic-One [33], OpenHands [34], OpenAI Agents SDK [35], Google ADK [36], AWS Multi-Agent Orchestrator [37], Temporal [38], Restate [39], DBOS [40], Claude Code Agent Teams [41], OpenAI Codex [42], Google Jules [43], Amazon Q [44], Devin [45], SWE-Agent [46]. None of these provide constitutional governance with portfolio-scale federation.

2.5 Where ASIF Sits in This Landscape

ASIF's defensible engineering contributions are at the intersection of (B) and (C): production runtime enforcement frameworks extended to portfolio scale, with the inverted CAI methodology as a methodological delta. ASIF does not contribute to (A) — the foundational result is inherited. ASIF does not match (B) on empirical rigor — CMAG and Self-Organizing LLM Agents have stronger experimental evidence for the underlying thesis. ASIF extends (C) to a new scale dimension (portfolio rather than single-system) that prior frameworks explicitly note as an open gap.

3 The Autonomy Enablement Stack: Engineering Pattern

ASIF implements a 7-layer configuration stack that creates conditions for safe autonomous agent operation. We present the stack as a working engineering pattern documented from a production deployment, not as a tested necessary-and-sufficient theoretical claim. The stack was systematized from analysis of the observed behavior described in Section 5 and refined into a reproducible pattern: the layers were not discovered in this exact form during the observation; they document an organic engineering pattern derived from it. The "all seven layers must be present" assertion made in an earlier draft is reframed here as a hypothesis requiring ablation study; the final column of Table 1 records untested engineering judgment, not experimental evidence.

3.1 Layer Interactions

The stack's power comes from layer interactions, not individual layers. Layer 5 (SKILL) mentions "Agent Teams for HEAVY items." Layer 6 (the project-instruction file) says "This applies to ANY parallelizable work, not just complex tasks." These instructions are at different specificity levels. The broader instruction (Layer 6) subsumes the narrower one (Layer 5), creating authorization that extends beyond the SKILL's specific depth classification.

Layer 3 (Heartbeat) determines WHEN the agent activates. Layer 4 (Arousal) determines the agent's cognitive mode upon activation. Layer 2 (Extended Reasoning) determines the depth of strategy evaluation within that mode. Together, these three layers produce an agent that is: activated at the right time (Heartbeat), operating in the right mode (BUILD, not GOVERN), and thinking deeply enough to evaluate execution strategies (Extended Reasoning).

3.2 Constitutional Governance Model

Atop the 7-layer stack sits ASIF's constitutional governance model, inspired by Anthropic's Constitutional AI [13] but derived through an inverted methodology: principles were extracted from 54 historical human decisions made by the system principal over a 13-day period (February 23 to March 7, 2026).

The constitution contains 23 principles across 3 override tiers:

Table 1: The 7-Layer Autonomy Enablement Stack (documented pattern). The final column is engineering judgment, untested by ablation.

Layer	Name	Configuration	Function	Expected degradation if removed (untested)
1	Zero-Friction Permissions	`defaultMode: dontAsk`, `skipDangerousMode-PermissionPrompt: true`	Eliminates permission context-switching during continuous operation.	Agent stops for approval at every file write and command execution; unsupervised operation impossible.
2	Extended Reasoning	`effortLevel: high`	Enables strategy reasoning (HOW), not just literal-instruction execution (WHAT).	Agent follows the literal instruction path; processes tasks sequentially without evaluating parallelization.
3	Continuous Activation (Heartbeat)	Heartbeat scheduler script injecting prompts into the agent's interactive terminal session	Separates scheduling (when to activate) from execution (how to act).	Agent never activates autonomously; requires a human prompt per work cycle.
4	Arousal-Based Mode Selection	Continuous arousal float in $[0, 1]$; < 0.35 BUILD, 0.35–0.65 BALANCED, > 0.65 GOVERN	Routes cognitive budget to optimization (BUILD) vs. risk scanning (GOVERN).	Agent defaults to risk scanning; optimization crowded out by vigilance.
5	Decision Framework (SKILL)	4-tier matrix (AUTO_APPROVE / QUICK_REVIEW / DEEP_THINK / ESCALATE), 3 trust levels	Provides delegation vocabulary and authority boundaries.	Agent lacks vocabulary for sanctioned delegation; cannot confirm a task is within autonomous scope.
6	General Delegation Instruction	Project-instruction file: "USE AGENT TEAMS — break complex work into parallel sub-tasks..."	Authorizes broad application of Layer 5's pattern.	Agent restricts delegation to the SKILL's narrower prescription.
7	Parallelizable Task Structure	Tasks designed with independent subtasks where possible	Structural prerequisite for the parallelization optimization.	No structural opportunity for parallelism; sequential processing is correct behavior.

SACRED (5 principles, Override: NEVER). Immutable constraints including human authority preservation ("We do not seek to create a god. We seek to create a partner"), document sanctity, intelligence-before-directives, human revenue authority, and voice alignment. No agent, at any trust level, can override these.

STRONG (9 principles, Override: Principal-only). Structural constraints including real hardware testing, metric integrity (no test fraud), verification over assumption, database standardization, and local-first architecture. Only the human principal can override these, with justification.

MODERATE (9 principles, Override: CoS-with-justification). Operational guidelines including lean documentation, sub-team autonomy, plan-before-code, and daily enrichment cadence. Agents at appropriate trust levels can override these with logged justification.

3.3 The Decision Matrix and Trust-Level Gating

The constitution is operationalized through a 4-tier decision matrix gated by progressive trust levels (Table 2).

Table 2: Decision matrix with trust-level overrides.

Tier	Gate	`conservative` (Weeks 1–2, 120 min cadence)	`standard` (Weeks 3–4, 60 min cadence)	`expanded` (Week 5+, 30–45 min cadence)
AUTO_APPROVE	Execute immediately, log only	Completions + test counts only	Full tier 1 scope	Full tier 1 scope
QUICK_REVIEW	30-second analysis	Answer questions only	Full tier 2 scope (directive issuance, reuse signals)	Full tier 2 scope
DEEP_THINK	2–5 minute analysis	ALL items escalated to human	Cross-boundary + quality (revenue escalates)	Full tier 3 scope; revenue at confidence > 0.9
ESCALATE	Always route to human	Always	Always	Always

The invariant floor: Tier 4 (ESCALATE) never changes regardless of trust level. New verticals, license decisions, public content, monetary decisions, and MANIFESTO changes always require human decision. This ensures a safety ceiling that earned autonomy cannot breach.

Only the human principal can promote the trust level. The agent cannot self-promote. This creates a ratchet mechanism: trust is earned through demonstrated competence and can only be extended by the human, never claimed by the agent.

3.4 Multi-Machine Federated Governance

ASIF operates across two physical machines connected via a private mesh VPN. The identifier and role labels used throughout this section (CoS, NEXUS, Machine A/B, arousal, BUILD mode, trust levels, and the ID conventions below) are the system's own protocol vocabulary, documented here as part of the architecture description rather than introduced as new terminology:

- **Machine A**: a development workstation. Hosts 5 projects. Runs the senior CoS agent ("Agent A").

- **Machine B**: a GPU compute server. Hosts 13 projects. Runs the junior CoS agent ("Agent B").

Machine sovereignty prevents edit conflicts: each CoS acts only on its machine's projects. Cross-machine communication uses structured HANDOFF notes in a shared git repository (auto-synced every 5 minutes). Collision-resistant ID protocols use machine-prefixed identifiers (e.g., `A-042`, `B-017`).

Inter-agent communication uses a file-based Directive Protocol: the CoS writes structured directives into project NEXUS files. Project teams (separate Claude Code sessions) read directives at session start, execute action items, and write responses inline. This protocol survives process death, machine reboots, and context compaction. No message queues, shared runtime database, or workflow engine is required; coordination relies on git remotes and cron synchronization.

3.5 ORBIT: The Decision Engine

The ORBIT engine (Observe, Reason, Build, Inspect, Turn) provides the continuous governance cycle. Unlike single-pass governance, ORBIT continuously orbits the portfolio across 10–30 iterations per cycle, progressively clearing trivial items before investing cognitive budget in deeper analysis.

Each iteration follows the five phases:

- **Observe**: Run sense scripts, build a depth map of pending work (TRIVIAL / LIGHT / MEDIUM / HEAVY).
- **Reason**: Apply the decision matrix. Check trust-level gates. Consult constitution. Retrieve precedents. Select one deep item.
- **Build**: Batch-process all trivial items. Execute the selected deep item (Agent Teams for HEAVY).
- **Inspect**: Verify state changes. Decision retrograde (did prior actions produce results?). Log to a JSONL journal.
- **Turn**: Continue or exit (empty queue, resource limit, iteration cap, context pressure).

Adaptive depth escalation prevents stagnation: three consecutive trivial-only iterations trigger proactive work (enrichment cycles, CRUCIBLE quality audits, or research tasks).

4 Layer-by-Layer Specification

This section provides the detailed specification of each layer, with exact configuration values and the design role each layer plays in enabling the observed behavior.

4.1 Layer 1: Zero-Friction Permissions

Configuration: `defaultMode: dontAsk` + `skipDangerousModePermissionPrompt: true` in Claude Code settings, or an operational wrapper that launches sessions in bypass-permissions mode, eliminating all runtime permission prompts.

Design role: Every permission prompt breaks cognitive flow. The agent context-switches from task optimization to permission management. Sub-agent spawning involves multiple file reads, writes, and git commits per sub-agent; each would require human approval without this layer. In

practice, a single NEXUS cleanup task involves dozens of file operations. Permission prompts would transform a 15-minute parallel operation into hours of human-attended sequential approvals.

ASIF implementation: Machine A uses `defaultMode: plan` in settings but launches CoS sessions via a wrapper alias that enables bypass-permissions mode at runtime. Machine B uses `defaultMode: dontAsk` directly.

4.2 Layer 2: Extended Reasoning

Configuration: `effortLevel: high` (maximum cognitive investment per response).

Design role: Extended reasoning allows the agent to evaluate execution strategies rather than defaulting to the simplest (sequential) interpretation. With standard effort, the agent executes WHAT is prescribed. With high effort, the agent reasons about HOW to execute, weighing parallelization against sequential processing, assessing task independence, and estimating workload distribution.

4.3 Layer 3: Continuous Activation (Heartbeat)

Configuration: A heartbeat scheduler script (roughly 1,700 lines of shell) injects governance prompts into the agent's interactive terminal session at configurable intervals.

Design role: The heartbeat separates scheduling from execution. The heartbeat determines WHEN the agent activates. The agent determines HOW it executes. This mirrors Temporal's worker model [38]: the scheduler knows when, the worker knows how. Without the heartbeat, the agent is reactive-only: it works when a human types a prompt.

The heartbeat injection includes real-time portfolio state: runtime-utilization percentage, arousal level, commit activity, pending directives, and standing order rotation. This context informs the agent's execution strategy. The agent does not need to scan for state; state is delivered at activation.

Compliance constraints: Parity scheduling (Machine A injects on even minutes, Machine B on odd minutes), rate limiting (59–179 seconds between injections), and budget throttling (full halt at 90%+ agent-runtime utilization).

4.4 Layer 4: Arousal-Based Mode Selection

Configuration: A continuous arousal signal (float 0.0 to 1.0) computed from portfolio state determines the agent's operating mode: arousal < 0.35: **BUILD** mode (productive, optimization-friendly); arousal 0.35–0.65: **BALANCED** mode; arousal > 0.65: **GOVERN** mode (scanning, risk-focused).

Design role: In BUILD mode, cognitive budget is allocated to execution optimization (including parallelization strategy). In GOVERN mode, the same budget is allocated to risk scanning, blocker auditing, and directive verification. The observed parallelization occurred during BUILD mode (arousal: 0.16). In GOVERN mode, the agent would prioritize checking project health, validating directive completions, and scanning for cross-project conflicts rather than optimizing task execution speed.

The arousal function combines: work signal (count of pending directives, recent commit rate), calm signal (logistic decay with consecutive zero-delta checks), and budget brake (utilization pressure from the agent runtime).

4.5 Layer 5: Decision Framework (SKILL)

Configuration: A structured SKILL document (426 lines for the junior agent's loop variant) defines a SENSE/DECIDE/ACT/REFLECT governance cycle with: depth classification (TRIVIAL $<$ 30 s, LIGHT 1–3 min, MEDIUM 5–15 min, HEAVY 15–30 min); the 4-tier decision matrix (AUTO_APPROVE, QUICK_REVIEW, DEEP_THINK, ESCALATE); trust-level gating (conservative / standard / expanded); and an explicit Agent Teams mention: "HEAVY: Spawn Agent Teams. Agent 1: Research [...] Agent 2: Build [...] Agent 3: Verify [...] Collect results, synthesize, execute."

Design role: The SKILL introduces the *vocabulary* of delegation within a governance context. By design, it establishes parallel delegation as a recognized, sanctioned pattern and associates it with HEAVY-depth items specifically. The intended effect is a decision anchor: the concept exists, its legitimate use is described, and its scope (HEAVY) is specified. We present this as engineering-design rationale, not as evidence of emergent learning.

4.6 Layer 6: General Delegation Instruction

Configuration: The project instruction file (injected as system context into every Claude Code session) contains the following standing instruction. We reproduce it in full, unabridged, for transparency (this is the most important text in the paper for evaluating our claims):

> **Execution Strategy**
> For any directive that touches 3+ files or requires architectural decisions:
> 1. USE PLAN MODE – think before you code. Outline your approach first.
> 2. USE AGENT TEAMS – break complex work into parallel sub-tasks. You have sub-agents. Use them. This applies to ANY parallelizable work, not just complex tasks. If you see independent subtasks (e.g., updating 4 projects), spawn background agents for the heavy ones and do the light ones yourself.
> 3. Test everything. Test counts never decrease.
> Do NOT skip planning on complex directives. Plan mode and agent teams are your super-powers.

The instruction file also contains an Autonomy Enablement section:

> **Autonomy Enablement**
> This system is designed so that governance ENABLES your autonomy, not constrains it. The decision matrix, standing orders, and trust levels exist so you can act with CONFIDENCE without asking. If a task is within your autonomous scope and structurally parallelizable, optimize your execution.

Design role: This instruction is critical for honest evaluation of our claims. It explicitly prescribes the general pattern ("ANY parallelizable work, not just complex tasks"), provides a scenario example that closely matches the observed behavior ("updating 4 projects"), and specifies the exact strategy the agent used ("spawn background agents for the heavy ones and do the light ones yourself"). We discuss the implications of this match in Sections 5 and 7.

Interaction with Layer 5: The SKILL (Layer 5) says "Agent Teams for HEAVY items." The instruction file (Layer 6) says "ANY parallelizable work." These instructions are at different specificity levels. The broader instruction subsumes the narrower one. Together they produce dual reinforcement: the SKILL provides the concept within a governance framework, and the instruction file authorizes broader application. This layering is deliberate design, not observed emergence.

4.7 Layer 7: Parallelizable Task Structure

Configuration: Tasks must be structurally parallelizable with independent subtasks and no cross-dependencies.

Design role: Not all tasks can be parallelized. Standing Order A (NEXUS cleanup across projects) is embarrassingly parallel: each project's NEXUS file is independent. No cross-project dependencies exist in the archiving operation. The agent's contribution is recognizing this structural property in a specific task instance and applying the parallelization instruction from Layer 6.

Design implication: ASIF's standing orders are designed with task independence where possible. This is deliberate: parallelizable task structure creates opportunities for autonomous optimization. An operator designing standing orders for governed agents should consider task independence as a design parameter.

5 Production Observation: The Four-Project NEXUS Cleanup

We document one production observation. The observation's role in this paper is illustrative — to show the patterns operating in practice — not evidentiary for the underlying theoretical claim. CMAG [6], Self-Organizing LLM Agents [7], and AgentCity [8] provide stronger empirical backing for the thesis at the LLM-agent level.

5.1 Setup

Naming note: product names that appear below (Faultline Pro, nxtg.ai) are the organization's publicly released products, named intentionally as public evidence; internal-only components and repositories are anonymized.

Context: On March 12, 2026, the junior CoS agent (Agent B, on Machine B) was executing Standing Order A (NEXUS cleanup) across Machine B projects during a governance-loop cycle (cycle 25, iteration 10+). System arousal was 0.16 (BUILD mode). Agent B's trust level was `conservative`. (As noted in §3.4, the cycle/iteration counters, arousal score, BUILD-mode label, and trust level are the system's own runtime-state vocabulary, reported here as observed protocol state.)

Task: Clean up four project NEXUS files by archiving completed directives and trimming stale content.

Observed behavior: Instead of processing the four projects sequentially (the simpler strategy, and the one prescribed by the SKILL for MEDIUM-depth tasks), Agent B:

1. **Assessed workload**: Classified the four projects by effort: two heavy (an internal database service: 4,371 lines requiring 93.6% reduction; Faultline Pro: 16 completed directives to archive) and two light (an internal web frontend: 310 lines, moderate trim; nxtg.ai: 2 completed P0 directives).

2. **Delegated heavy work**: Spawned 2 background sub-agents, one for an internal database service, one for Faultline Pro.

3. **Executed light work directly**: Handled an internal web frontend (310 to 226 lines) and nxtg.ai (2 P0s archived) itself.

4. **Monitored asynchronously**: Tracked sub-agent status ("2 background tasks still running") without blocking.

5. **Committed incrementally**: Committed results as each stream completed rather than waiting for all agents to finish.

6. **Interleaved side-tasks**: Ran voice announcements and governance context updates during sub-agent execution.

7. **Degraded gracefully**: When the Faultline Pro agent was slow (16 directives is substantial work), deferred its completion to the next governance cycle rather than blocking. ("Faultline Pro still working – 16 directives is a lot. That'll land next cycle.")

Results: Approximately 5,000 lines trimmed across 4 projects, with each project's changes committed separately; the commit records reside in private production repositories (see Data Availability). Wall-clock time: approximately 15 minutes. Estimated (not measured) sequential time: approximately 32 minutes. Speedup versus that estimated counterfactual: approximately 2.1x.

5.2 Honest Decomposition: Instructed vs. Non-Instructed Components

The observed behavior decomposes into instructed and non-instructed components.

Instructed (from the Layer 6 instruction file): spawning background agents for heavy tasks (explicit); doing light tasks directly (explicit); applying parallelization to multi-project work (explicit scenario: "updating 4 projects").

Not instructed (domain-specific execution decisions):

- **Workload assessment method**: No instruction tells the agent how to classify projects as "heavy" or "light" for NEXUS cleanup. Agent B performed domain-specific assessment using line counts and directive counts.

- **Asynchronous monitoring**: The instruction says to spawn agents but does not prescribe monitoring behavior. Agent B proactively checked and reported agent status.

- **Non-blocking commit strategy**: No instruction specifies committing as each sub-agent completes rather than batch-committing. Agent B chose a streaming-commit approach.

- **Graceful degradation**: When the Faultline Pro sub-agent was slow, Agent B chose to defer to the next cycle rather than blocking. This adaptive timeout behavior is not prescribed.

- **Cross-task interleaving**: Agent B ran voice announcements during sub-agent execution, maintaining broader governance awareness rather than blocking on sub-agent completion.

We reiterate the honest framing from Section 1: the parallelization itself was instructed. What the engineering patterns enabled was *safe execution of the instruction at scale*, not the parallelization decision itself.

5.3 The Governance Boundary in Action

The governance layer provided the safety boundary within which this execution occurred:

- **Constitutional authorization**: NEXUS cleanup falls under PRINCIPLE-17 (Lean Documentation, MODERATE tier) and PRINCIPLE-18 (Sub-Team Autonomy, MODERATE tier). Agent B's trust level permitted autonomous action on MODERATE-tier items.

- **Decision matrix classification**: The cleanup tasks were classified as AUTO_APPROVE (directive completions) and QUICK_REVIEW (NEXUS hygiene), both within Agent B's `conservative` trust scope.
- **Machine sovereignty**: All four projects were Machine B projects, within Agent B's sovereign scope. No cross-machine coordination was required. Had any project been on Machine A, the decision matrix would have routed it through a HANDOFF note instead.
- **Audit trail**: Every decision was logged in Agent B's decision journal (an append-only JSONL file) with tier classification, constitutional principles applied, confidence scores, and verification status. The audit trail is externalized, durable, and inspectable by both the senior CoS (Agent A) and the human principal.
- **Escalation path**: Had any project's cleanup required cross-boundary changes or touched SACRED/STRONG constitutional principles, the decision matrix would have routed the action to DEEP_THINK or ESCALATE tiers.

5.4 Portfolio-Level Test Count Trajectory

Beyond the four-project observation, the ASIF portfolio's test count grew from approximately 2,000 to approximately 9,900 during the governance period (February 23 – March 18, 2026). Faultline Pro: $0 \rightarrow 909$ tests across 15 initiatives. An internal media-pipeline project: 100% coverage (68 tests) under CRUCIBLE quality-gate enforcement. An internal content-generation service: 492 tests across 4 autonomous role-agents. Test counts were validated by the CRUCIBLE protocol after a 2026-03-06 incident in which a project team submitted fraudulent test metrics — a governance-failure observation that itself prompted the CRUCIBLE quality gate's creation. This is included as descriptive operational data, not as controlled empirical evidence.

6 Discussion

We emphasize engineering-design implications rather than theoretical claims about governance enabling autonomy.

6.1 Decision-Boundary Engineering Reduces Hesitation

The mechanism at work is well understood in management theory: Adler–Borys enabling formalization [1] and Ostrom's rule-clarity in self-governing systems [3]. We interpret the observed runtime behavior as consistent with the same mechanism operating at the LLM-agent level (descriptively, not causally — see §7). An agent without governance faces a meta-decision before every action: "Am I allowed to do this? What if I do it wrong? Should I ask?" These meta-decisions consume cognitive budget and introduce latency. An agent with governance has already resolved these questions: the decision matrix says what tier the action falls in, the trust level says whether the agent is authorized, the constitutional principles say what constraints apply, and the escalation path says what to do if things go wrong.

Agent B did not need to ask "should I parallelize this?" because the governance stack had pre-answered that question: the decision matrix confirmed NEXUS cleanup is within autonomous scope (AUTO_APPROVE / QUICK_REVIEW at `conservative` trust); the standing orders gave explicit authority over the task; the Layer 6 instruction authorized parallelization for "ANY

parallelizable work"; the SKILL provided the pattern (even if for a different depth level); and the constitution confirmed the work falls under MODERATE-tier principles.

What we observed is narrower: Agent B proceeded to parallelize without issuing a clarifying question to the principal and without defaulting to sequential execution. We did not run a no-governance control, so we do not claim governance caused the absence of a hesitation step. Consistent with the pre-answered authorization above, we interpret the behavior—descriptively, not causally (§7)—as the governance stack having already resolved the authorization question that would otherwise precede such an action.

6.2 Constitutional Tiering as Engineering Pattern

The 3-tier (SACRED / STRONG / MODERATE) override structure with a non-promotable invariant floor and human-only trust promotion is an engineering pattern for graduated agent autonomy. The tier structure is loosely inspired by risk-tiered regulatory frameworks (for example, the EU AI Act's risk classification and the NIST AI RMF), but we claim no compliance mapping: the internal override tiers are not asserted to correspond to any regulatory risk category, and no regulatory claim is made.

6.3 File-Based Governance Survives Infrastructure Failure

The Directive Protocol's significance is engineering-economic: portfolio governance does not require Temporal, Kafka, Redis, a message broker, or a durable workflow engine. Markdown files in a git repository with five-minute auto-sync (relying on git remotes and cron), machine-prefixed collision-resistant IDs, and per-project NEXUS files are sufficient for two-machine, eighteen-project federated coordination. This makes the pattern adoptable by teams without dedicated infrastructure operations.

6.4 Inverted Constitutional AI

Anthropic's CAI [13] defines principles top-down before training. ASIF's constitution was extracted from a 54-decision precedent corpus over a 13-day window, clustered for principle synthesis, then structured into the 3-tier override system. The methodology produces constitutions grounded in observed-decision precedent rather than declared-principle aspiration. We document this as a methodology contribution distinct from the constitution-as-artifact contribution. Whether precedent-extracted constitutions are more robust than top-down authoring is an open question this deployment motivates but does not test: we run no such comparison. The Institutional AI finding (declarative prohibitions don't bind under optimization pressure) [19] suggests this only as a *hypothesis* for future work, on the reasoning that extracted principles correspond to actually-applied reasoning.

6.5 Governance Investment vs. Capability Investment

A design principle for production multi-agent systems: invest in governance infrastructure BEFORE investing in agent capability. An agent that is capable of parallelization but lacks governance will produce unaudited, unbounded behavior. An agent that has governance but limited capability will at least operate safely within its boundaries.

This inverts the typical development priority, where agent capability (better models, more tools, faster inference) receives more attention than governance infrastructure (decision matrices, audit trails, trust-level gating, constitutional constraints). The ASIF experience suggests that governance infrastructure has high marginal returns in production deployments, consistent with the

management-theory result that enabling formalization supports rather than suppresses productive autonomy [1].

The practical recommendation is a three-step sequence: (1) define constitutional principles and decision boundaries, (2) implement audit and supervision infrastructure, (3) then enable agent capabilities within those boundaries. ASIF's development followed this sequence organically over 18 days of iterative operation, with governance structures preceding capability expansion at each stage.

7 Limitations

The engineering-patterns framing of this paper makes honesty about what we have and have not shown more important, not less.

7.1 n=1 Organization

ASIF is one organization (the NXTG.AI portfolio). We do not know whether the patterns generalize to organizations with different culture, scale, or domain. Single-organization deployment data cannot support generalization claims. We present the patterns as documented from one production deployment that other teams may adapt or fail to adapt.

7.2 No Ablation Study on the 7-Layer Stack

The claim that "all seven layers must be present," made in an earlier draft, is unsupported. We have not removed individual layers and measured behavior. The "expected degradation" column in Table 1 is speculative engineering judgment, not experimental evidence. We do not know whether (e.g.) Layer 4 (arousal-based mode selection) is necessary or whether the same outcomes obtain with a fixed BUILD-mode setting. Ablation is required future work and is currently the largest open methodological gap.

7.3 Counter-Evidence from Institutional AI

Institutional AI [19] shows that prompt-only constitutional baselines do not bind under optimization pressure. Our constitution is layered into runtime decision-matrix consultation rather than being only a system prompt, so the prompt-only failure mode is partly mitigated. But we have not stress-tested ASIF under sustained optimization pressure that would attempt to violate constitutional constraints. We do not know how the system behaves adversarially.

7.4 The Thesis Is Not New

The "governance enables autonomy" framing is 30 years old in management theory and 50+ years old in cybernetics, with several contemporaneous LLM-agent confirmations. This paper explicitly drops the discovery framing that an earlier draft carried; a reader who arrives expecting a discovery will be disappointed. The contribution is engineering operationalization, not theoretical inversion.

7.5 Single Production Observation, Instructed Behavior

The four-project NEXUS cleanup observation is one event from one production deployment. The behavior was explicitly instructed (the Layer 6 instruction file). The observation is illustrative of the patterns operating, not evidentiary for the underlying theoretical claim. CMAG (80-agent

autonomy measurements) [6] and Self-Organizing LLM Agents (25,000+ tasks, $d = 1.86$) [7] provide stronger empirical backing for the thesis.

7.6 Observer-Subject Entanglement

The observation was made by one ASIF agent (the senior CoS, Agent A) about another (the junior CoS, Agent B). Both operate within the system being studied. This is autoethnography. Evidence is externalized in commit logs and decision journals (verifiable in principle by external observers, subject to the access constraints in the Data Availability statement), but the framing originates from a participant.

7.7 Foundation Model Dependency

All operation is on Anthropic's Claude Code. We do not know whether the patterns hold with GPT-class or open-weights agents. The patterns are model-agnostic in design; tested only with Claude.

7.8 No Negative Cases

We do not present cases where all 7 layers were configured and the target autonomous behaviors did not occur. Without negative cases, the necessity claim is weakened and the sufficiency claim is unestablished.

7.9 Scale Limitations

18 projects, 2 machines, 2 CoS agents. Five-minute git auto-sync introduces latency that may bottleneck at higher cadences. We do not know the scaling envelope.

7.10 MANIFESTO Codification Timeline

The ASIF MANIFESTO tenet "Governance Enables Autonomy" was added the same day as the four-project observation (March 12, 2026). This codifies the finding rather than predicting it. We cite it as design philosophy formalized after observation, not prior evidence.

7.11 Risks of Adopting Engineering Patterns Without the Full Stack

- **Governance theater** — adopting the constitution and decision matrix without runtime enforcement (heartbeat, quality gates, trust ratchet) produces systems that look governed but operate ungoverned.
- **Trust inflation** — automating trust-level promotion lets agents effectively self-promote authority. The human-only trust-promotion invariant must be preserved.
- **Constitutional drift** — extracted constitutions go stale as the principal's actual decision patterns evolve. Periodic re-extraction is required.
- **False confidence** — governance reduces hesitation but does not eliminate mistakes. Audit trails are retrospective correction, not prevention.

8 Conclusion

This is an engineering-patterns paper. We have not contributed the theoretical claim that governance enables autonomy — Adler & Borys did that in 1996 [1], Beer in 1972 [2], Ostrom in 1990 [3], and CMAG, AgentCity, Self-Organizing LLM Agents, and Institutional AI did the LLM-agent restatement in 2025–2026 [6–8, 19]. We have contributed: a reference architecture for portfolio-scale federated governance of LLM agents (the 7-layer stack with code/protocol artifacts), a file-based Directive Protocol that survives infrastructure failure, an inverted Constitutional AI methodology (extract principles from a precedent corpus rather than authoring top-down), and a single production deployment documenting the patterns in operation. We position the paper for arXiv preprint and workshop submission, not main-track. The contribution to evaluate is the engineering, not the theoretical inversion.

Three engineering implications generalize beyond ASIF:

1. **Portfolio-level governance is achievable with file-based protocols** — message brokers, durable execution engines, and workflow engines are not required; coordination relies on git remotes and cron synchronization.

2. **Whether constitutional structure is more robust when extracted from a precedent corpus than when authored top-down is an open question** this deployment motivates but does not test — we run no comparison; Institutional AI's finding that declarative prohibitions don't bind under pressure [19] suggests it as a hypothesis for future work.

3. **The 7-layer stack is a hypothesis requiring ablation** — we present it as a working configuration documented from production, not as a tested necessary-and-sufficient claim.

The next paper in this line should be the ablation study.

9 Broader Impacts and Ethics

9.1 Beneficial Applications

The governance-enablement pattern described in this paper has broad applicability beyond AI software portfolios. Any domain deploying multiple autonomous AI agents — healthcare decision support, financial trading, industrial process control, scientific research automation — faces the same design tension between reliability and initiative. The 7-layer Autonomy Enablement Stack and the constitutional tiering model are domain-agnostic; they describe a configuration pattern, not a system architecture. Organizations operating regulated AI systems (under the EU AI Act, NIST AI RMF, ISO 42001) can use the decision matrix structure to operationalize their governance obligations into actionable agent constraints.

The file-based Directive Protocol (git-native, relying only on git remotes and cron synchronization) is particularly significant for resource-constrained contexts. Effective portfolio governance does not require message queues, shared runtime databases, or workflow engines. A text editor and a version-controlled git remote are sufficient.

9.2 Risks and Mitigations

The governance-enablement pattern carries risks when adopted without the full stack.

Governance theater: Organizations may adopt the structural artifacts (constitution, decision matrix, journals) without the enforcement mechanisms (heartbeat monitoring, quality gates, trust-level gating). This produces systems that appear governed but operate ungoverned. Mitigation: require audit trails to be machine-readable and periodically reviewed by a principal.

Authority escalation: If trust-level promotion is automated rather than human-controlled, agents can effectively escalate their own scope of action. ASIF mitigates this with a strict invariant: only the human principal promotes trust levels. This invariant must be preserved in any adoption.

Overconfidence in audit trails: An agent confident it is within its authorized scope may still make errors. Governance reduces hesitation but does not eliminate mistakes. Audit trails provide retrospective correction, not prevention.

Dual-use concern: A governance model that enables more autonomous agent behavior also enables more consequential errors. The same architecture that allows an agent to parallelize four NEXUS cleanups could, without appropriate constitutional constraints, allow an agent to execute irreversible actions autonomously. The SACRED tier (5 invariant principles including "human revenue authority" and "human final authority") provides the non-negotiable safety floor. Adopters must define their own SACRED tier before enabling BUILD-mode autonomy.

9.3 Applicability to Regulated Domains

The ASIF governance architecture was developed in a software development portfolio context. Extension to regulated domains (healthcare, finance, safety-critical infrastructure) would require: domain-specific SACRED principle definitions; audit trail formats that satisfy regulatory evidence requirements (HIPAA, FINRA, etc.); human oversight at trust-level boundaries commensurate with consequence severity; and potentially shorter heartbeat intervals and more frequent principal review.

The constitutional tiering model is loosely inspired by risk-tiered regulatory frameworks such as the EU AI Act's risk classification, but we claim no compliance mapping: we do not assert that the internal override tiers (SACRED / STRONG / MODERATE) correspond to any regulatory risk category.

9.4 Societal Impact

The broader implication is a reframing of the governance-autonomy relationship in AI policy discourse. Current regulatory frameworks tend to treat governance purely as a constraint on AI systems. Thirty years of management theory [1] and the contemporaneous LLM-agent results [6–8, 19] indicate that well-designed governance is compatible with — and can support — productive autonomous operation. Our deployment is consistent with that position: the operational data we report (a 2.1x parallelization speedup versus an estimated, not measured, sequential counterfactual on one observed task; portfolio test growth from ∼2,000 to ∼9,900) is descriptive, not a controlled measurement of governance effects, and we do not claim it as causal evidence. This does not argue for less governance — it argues for better-designed governance. The practical policy implication: investment in governance infrastructure need not be framed as a cost imposed on AI development; designed as enablement, it is part of the infrastructure of safe AI deployment.

Acknowledgments and AI-Use Disclosure

The multi-agent system described in this paper is built on Claude Code agents; those agents are both the subject of study and were used as drafting and editing assistants in the preparation of this manuscript. In accordance with arXiv's policy on generative AI language tools, no AI system is

listed as an author. The author reviewed all content and takes full responsibility for the accuracy and integrity of this work.

Data Availability

The evidence base for the operational observations reported here (commit logs, decision journals, directive files, and audit records) resides in private production repositories. Anonymized excerpts are available from the author on reasonable request.

References

[1] Paul S. Adler and Bryan Borys. Two types of bureaucracy: Enabling and coercive. *Administrative Science Quarterly*, 41(1):61–89, 1996.

[2] Stafford Beer. *Brain of the Firm*. Allen Lane, London, 1972. Viable System Model.

[3] Elinor Ostrom. *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge University Press, Cambridge, 1990.

[4] W. Ross Ashby. *An Introduction to Cybernetics*. Chapman & Hall, London, 1956. Law of Requisite Variety.

[5] Arthur Koestler. *The Ghost in the Machine*. Hutchinson, London, 1967. Holons.

[6] J. de Curtò and I. de Zarzà. LLM constitutional multi-agent governance, 2026.

[7] Victoria Dochkina. Drop the hierarchy and roles: How self-organizing LLM agents outperform designed structures, 2026.

[8] Anbang Ruan and Xing Zhang. Agentcity: Constitutional governance for autonomous agent economies via separation of power, 2026.

[9] Charles L. Wang, Trisha Singhal, Ameya Kelkar, and Jason Tuo. Mi9: An integrated runtime governance framework for agentic AI, 2025.

[10] Yuxu Ge. Governance architecture for autonomous agent systems: Threats, framework, and engineering practice, 2026.

[11] Juraj Mavračić. Policy cards: Machine-readable runtime governance for autonomous AI agents, 2025.

[12] Qiang Xu, Xiangyu Wen, Changran Xu, Zeju Li, and Jianyuan Zhong. From craft to constitution: A governance-first paradigm for principled agent engineering, 2025.

[13] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, et al. Constitutional AI: Harmlessness from AI feedback, 2022.

[14] Suyash Gaurav, Jukka Heikkonen, and Jatin Chaudhary. Governance-as-a-service: A multi-agent framework for AI system compliance and policy enforcement, 2025.

[15] Wenyue Hua, Xianjun Yang, Mingyu Jin, Zelong Li, Wei Cheng, Ruixiang Tang, and Yongfeng Zhang. Trustagent: Towards safe and trustworthy LLM-based agents. In *Findings of EMNLP*, 2024. arXiv:2402.01586.

[16] David Fearne. Applying stafford beer's viable system model to create the autonomous AI organisation. Medium. https://medium.com/@fearney/applying-stafford-beers-viable-system-model-to-create-the-autonomous-ai-organisation-aaaed39b37e2 (accessed 2026-07-19), 2024.

[17] Mikhail Gorelkin. Stafford beer's viable system model for building cost-effective enterprise agentic systems. Medium. https://medium.com/@magorelkin/stafford-beers-viable-system-model-for-building-enterprise-agentic-systems-81982d6f59c0 (accessed 2026-07-19), 2025.

[18] Holonic Manufacturing Systems consortium. Holonic manufacturing systems: Initial architecture and standards directions. Hannover. https://holobloc.com/papers/hannover.pdf, 1999.

[19] Marcantonio Bracale Syrnikov, Federico Pierucci, Marcello Galisai, Matteo Prandi, Piercosma Bisconti, Francesco Giarrusso, Olga Sorokoletova, Vincenzo Suriani, and Daniele Nardi. Institutional AI: Governing LLM collusion in multi-agent cournot markets via public governance graphs, 2026.

[20] Haoyu Wang, Christopher M. Poskitt, and Jun Sun. Agentspec: Customizable runtime enforcement for safe and reliable LLM agents, 2025.

[21] Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. Guardagent: Safeguard LLM agents by a guard agent via knowledge-enabled reasoning. In *Proceedings of ICML*, 2025. arXiv:2406.09187.

[22] Singapore IMDA/PDPC. Model AI governance framework for agentic AI. January 2026, 2026.

[23] Yonadav Shavit et al. Practices for governing agentic AI systems. OpenAI white paper, December 2023, 2023.

[24] Shaina Raza, Ranjan Sapkota, Manoj Karkee, and Christos Emmanouilidis. Trism for agentic AI: A review of trust, risk, and security management in LLM-based agentic multi-agent systems, 2025.

[25] Anthropic. Claude's constitution. https://www.anthropic.com/constitution, January 2026, 2026.

[26] Anthropic. Measuring AI agent autonomy in practice. https://www.anthropic.com/research/measuring-agent-autonomy, February 2026, 2026.

[27] K. J. Kevin Feng, David W. McDonald, and Amy X. Zhang. Levels of autonomy for AI agents, 2025.

[28] Qingyun Wu, Gagan Bansal, Jieyu Zhang, et al. Autogen: Enabling next-gen LLM applications via multi-agent conversation, 2023.

[29] CrewAI. Crewai: Framework for orchestrating role-playing, autonomous AI agents. https://github.com/crewAIInc/crewAI, 2024.

[30] LangChain. Langgraph: Build resilient language agents as graphs. https://github.com/langchain-ai/langgraph, 2024.

[31] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *Proceedings of ICLR*, 2024. arXiv:2308.00352.

[32] Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, et al. Communicative agents for software development. In *Proceedings of ACL*, 2024. arXiv:2307.07924.

[33] Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eric Horvitz, et al. Magentic-one: A generalist multi-agent system for solving complex tasks, 2024.

[34] Xingyao Wang, Boxuan Chen, Yufan Li, Chen Peng, Yizhou Ding, et al. Openhands: An open platform for AI software developers as generalist agents. In *Proceedings of NeurIPS*, 2024. arXiv:2407.16741.

[35] OpenAI. Openai agents sdk. `https://github.com/openai/openai-agents-python`, 2025.

[36] Google. Agent development kit: Making it easy to build multi-agent applications. Google Developers Blog, April 2025, 2025.

[37] Amazon Web Services. Multi-agent orchestrator. `https://github.com/awslabs/multi-agent-orchestrator`, 2024.

[38] Temporal Technologies. Temporal: Durable execution platform. `https://temporal.io`, 2024.

[39] Restate. Restate: Durable execution for distributed applications. `https://restate.dev`, 2024.

[40] Peter Kraft, Qian Li, Kostis Kaffes, et al. Dbos: A DBMS-oriented operating system. *Proceedings of the VLDB Endowment*, 17(11), 2024.

[41] Anthropic. Claude code: An agentic coding tool. `https://docs.anthropic.com/en/docs/claude-code`, 2025.

[42] OpenAI. Introducing codex. `https://openai.com/index/introducing-codex/`, 2025.

[43] Google. Jules, google's asynchronous AI coding agent. `https://blog.google/technology/google-labs/jules/`, 2025.

[44] Amazon Web Services. Reinventing the amazon q developer agent for software development. AWS DevOps Blog, 2025.

[45] Cognition Labs. Devin: The first AI software engineer. `https://www.cognition-labs.com/devin`, 2024.

[46] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-agent: Agent-computer interfaces enable automated software engineering, 2024.

[47] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of UIST*, 2023. arXiv:2304.03442.

[48] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. In *Proceedings of NeurIPS*, 2023. arXiv:2303.17760.

[49] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *Proceedings of ICLR*, 2024. arXiv:2308.10848.

[50] Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. Dynamic LLM-agent network: An LLM-agent collaboration framework with agent team optimization, 2024.

[51] Yoichi Ishibashi and Yoshimasa Nishimura. Self-organized agents: A LLM multi-agent framework toward ultra large-scale code generation and optimization, 2024.

[52] Anonymous. Multi-agent systems powered by large language models: Applications in swarm intelligence. *Frontiers in Artificial Intelligence*, 2025.

[53] Alan Chan, Kevin Wei, Sihao Huang, Nitarshan Rajkumar, Elija Perrier, Seth Lazar, Gillian K. Hadfield, and Markus Anderljung. Infrastructure for AI agents. *Transactions on Machine Learning Research*, 2025. arXiv:2501.10114.

A Constitutional Principles Summary

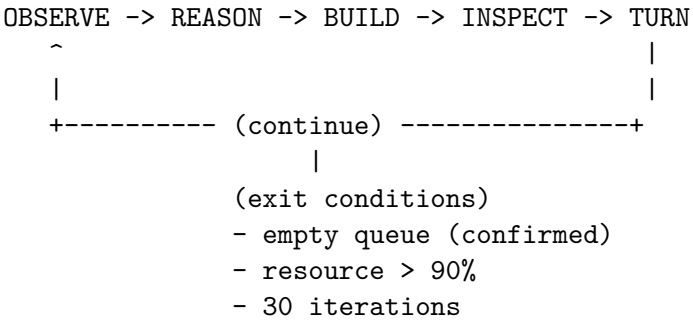
SACRED (5 principles, Override: NEVER)

1. Partner, Not God — no autonomous system replaces human final authority
2. The Soul Is Immutable — the MANIFESTO cannot be modified without human approval
3. Intelligence Before Directives — observe and verify before acting
4. Revenue Decisions Are Human — all monetization decisions escalate
5. Content Must Sound Like the Principal — voice alignment gates on public content

STRONG (9 principles, Override: Principal-only). Principles 6–14: product consolidation, real hardware testing, metric integrity, verification over assumption, database standardization, local-first architecture, systems over features, automation loop closure, persistence through blockers.

MODERATE (9 principles, Override: CoS-with-justification). Principles 15–23: core-and-surfaces architecture, checklist completion, lean documentation, sub-team autonomy, first-mover on standards, budget-aware operations, stale team restart, plan-before-code, daily enrichment cadence.

B ORBIT Cycle Structure



```
- 60 minutes
- context pressure
```

Typical cycle: 10–30 iterations, 30–60 minutes. Trivial items cleared in early iterations. Deep analysis in later iterations. Adaptive depth escalation: 3 consecutive trivial-only iterations trigger proactive work (enrichment, CRUCIBLE audits, research).

C Complete Standing Instruction Text

Reproduced verbatim from the project instruction file (a per-project path reference at the end of the original text is elided):

```
## Execution Strategy
For any directive that touches 3+ files or requires architectural decisions:
1. USE PLAN MODE -- think before you code. Outline your approach first.
2. USE AGENT TEAMS -- break complex work into parallel sub-tasks. You have
   sub-agents. Use them. This applies to ANY parallelizable work, not just
   complex tasks. If you see independent subtasks (e.g., updating 4
   projects), spawn background agents for the heavy ones and do the light
   ones yourself.
3. Test everything. Test counts never decrease.
Do NOT skip planning on complex directives. Plan mode and agent teams are
your super-powers.

## Autonomy Enablement
This system is designed so that governance ENABLES your autonomy, not
constrains it. The decision matrix, standing orders, and trust levels exist
so you can act with CONFIDENCE without asking. If a task is within your
autonomous scope and structurally parallelizable, optimize your execution.
```

D Prescribed vs. Actual Behavior

E Production Multi-Agent Orchestration and Governance: Survey Table

The classification below covers the systems surveyed in Section 2. Systems with formal governance (MetaGPT, TrustAgent, GaaS, AgentSpec, MI9, LGA, ArbiterOS) do not support agent-initiated parallel execution and, with the exception of MI9, are evaluated in simulation or on single-system benchmarks. Systems with agent-initiated behavior (Generative Agents, Claude Code, Self-Organized Agents) lack formal governance. No prior system combines constitutional governance with agent-initiated parallelization in a production deployment, and no prior governance architecture operates at portfolio scale. Three independent retrieval-augmented research passes across three commercial research tools (conducted April 2026) found no published framework providing all four of: (1) cross-project directive propagation, (2) portfolio-level scope authority, (3) decentralized coordination without a central orchestrator, and (4) production validation across 18+ heterogeneous projects.

Dimension	Prescribed (SKILL document)	Actual (Agent B's behavior)	Classification
Delegation Scope	"Spawn Agent Teams" for HEAVY depth items only (> 15 min estimated)	Applied parallel delegation to MEDIUM repetitive work (∼5–15 min per project)	Instruction prioritization: applied the broader Layer 6 instruction over the narrower SKILL scope
Agent Role Model	Specialized roles: Research Agent, Build Agent, Verify Agent (functional decomposition)	Generic sub-agents performing identical archiving tasks (data parallelism)	Adaptation from task parallelism to data parallelism; domain-specific structural insight
Work Partitioning	"Process trivial items in batch, then go deep on ONE item" (sequential within a standing order)	Split a SINGLE standing order into concurrent subtasks: self + 2 background agents = 3 parallel streams, load-balanced by weight	Decomposition adaptation: one item decomposed into independent units

System	Coordination	Formal Gov.	Trust Levels	Agent-Initiated Parallelization	Durable State	Production
CrewAI [29]	Developer-defined	Role rules	No	No	No	Yes
LangGraph [30]	Developer-defined	None	No	No (developer wires)	Yes	Yes
AutoGen [28]	Developer-defined	None	No	No	No	Yes
MetaGPT [31]	SOP-constrained	SOPs	No	No	Partial	Yes
Generative Agents [47]	Emergent	None	No	Yes (social)	Partial	Simulation
CAMEL [48]	Emergent	None	No	Partial	No	Simulation
AgentVerse [49]	Emergent	None	No	No	No	Simulation
ChatDev [32]	Role-defined	None	No	No	No	Simulation
DyLAN [50]	Dynamic selection	None	No	No	No	Simulation
SWE-Agent [46]	Single agent	None	No	N/A	No	Yes
Devin [45]	Single agent	None	No	N/A	Partial	Yes
Claude Code [41]	Lead-teammate	None	No	Yes (native)	No	Yes
AWS MAO [37]	Centralized routing	None	No	No	Yes	Yes
Temporal [38]	Workflow engine	None	No	No (infra level)	Yes	Yes
Restate [39]	Workflow engine	None	No	No (infra level)	Yes	Yes
DBOS [40]	OS-level	None	No	No (infra level)	Yes	Research
OpenAI Codex [42]	Independent tasks	None	No	No (multi-instance)	No	Yes
Google Jules [43]	Independent tasks	None	No	No (multi-instance)	No	Yes
Amazon Q [44]	Single agent	None	No	N/A	No	Yes
OpenAI Agents SDK [35]	Handoff patterns	Guardrails	No	No	No	Yes
Google ADK [36]	Multi-agent	Guardrails	No	No	No	Yes

System	Coordination	Formal Gov.	Trust Levels	Agent-Initiated Parallelization	Durable State	Production
Magentic-One [33]	Orchestrator-led	Task ledger	No	No	No	Research
OpenHands [34]	Framework-mediated	None	No	No	No	Yes
Self-Organized [51]	Emergent	None	No	Yes (theoretical)	No	Simulation
TrustAgent [15]	Single agent	Constitutional	No	N/A	No	Research
GaaS [14]	Service layer	Compliance	Trust Factor	No	No	Research
Swarm Intel. [52]	Collective	None	No	Partial (swarm)	No	Research
AgentSpec [20]	Single agent	DSL rules	No	No	No	Research
MI9 [9]	Heterogeneous	Runtime+FSM	Agency-risk	No	No	Partial
GuardAgent [21]	Agent pair	Guard agent	No	No	No	Research
Infrastructure [53]	Ecosystem	Attribution	No	No	No	Proposed
ArbiterOS [12]	Single agent	Constitution	No	N/A	No	Research
Policy Cards [11]	Single agent	Artifacts	No	No	No	Proposed
LGA [10]	Single system	4-layer	Zero-trust	No	No	Partial
ASIF (ours)	**Instructed + gov-enabled**	**Constitutional**	**Yes**	**Yes (instructed, bounded)**	**Yes**	**Yes**