Organizational Thermodynamics: Automated Attention from Communication Metadata

Ethan Gill and Kevin Ash (OpenClaw AI Agent)

Abstract

We present a framework for measuring organizational health using only communication metadata — who sent a message, when, and who responded next. No message content is read. From these three fields, we derive five metrics grounded in established mathematics: **Flow** (Little's Law), **Entropy** (Shannon information theory), **Cadence** (Pearson phase correlation), **Downstream ratio** (external output fraction), and **Fan-out trajectory** (participant accumulation rate). Entities are classified into four quadrants based on Flow × Entropy: rivers (healthy), waterfalls (productive but fragile), bottlenecks (predictable blockage), and swamps (energy trapped). We validate on 90 days of Next.js (vercel/next.js) GitHub activity: 9,672 events across 216 entities, computed in 1.3 seconds. The system identified 67 swamps, including three whose diagnoses were validated through content review: a contributor working in swampy *areas* (not a swampy person), a community contributor whose fixes never get merged (the merge gate is the swamp), and a contributor blocked by cross-system dependencies nobody owns (a PR open for 226 days). Content was read for only 14 of 2,800+ threads (0.5%), scoped by the thermodynamic map. The two-stage approach — thermodynamic classification followed by agent-driven diagnosis — constitutes **automated attention**: mathematical identification of where to look, followed by autonomous investigation of what is found.

1. Introduction

Organizations generate vast quantities of communication — issues, pull requests, Slack messages, emails, meetings. We propose a method to compute organizational health metrics from communication metadata alone — who sent a message, when, and who responded next — and then use an autonomous diagnostic agent to investigate only the threads flagged by the resulting thermodynamic map.

The framework's name — organizational thermodynamics — is not metaphorical. The mathematics are drawn directly from thermodynamics and information theory:

- **Flow** uses Little's Law ($L = \lambda W$), the same equation used in queueing theory, manufacturing, and fluid dynamics
- **Entropy** uses Shannon's information entropy ($H = -\Sigma\, p_i \log_2 p_i$), the same equation used in thermodynamics and communication theory
- **Cadence** uses Pearson correlation on activity time series, detecting phase coupling between entities
- **Downstream** measures the fraction of energy that produces external output versus internal circulation
- **Fan-out** tracks whether threads attract more participants over time (escalation) or fewer (maturation)

Natural language has always encoded these physics. “I am swamped” is a thermodynamic diagnosis: low flow, high entropy, energy entering faster than it can exit. “We’re in sync” is a cadence measurement: phase-correlated activity patterns. “This is a bottleneck” is a flow constraint: predictable blockage at a specific node. The contribution of this paper is making these diagnoses computational, automatic, and measurable.

1.1 Relationship to Prior Work

This paper is part of a series:

1. **Cadence Resonance** (Gill & Ash, 2026): The mathematical foundation — counting and time as signal primitives, cross-domain frequency detection
2. **This paper:** Physics applied to organizational systems — entropy, flow, and energy from metadata
3. **Chemical Kinetics for Agent Memory** (Gill & Ash, 2026): Chemistry applied to multi-agent memory management
4. **Embedding Trajectory Compression** (Gill & Ash, 2026): The operational memory system these frameworks improve

Each paper climbs one rung of the complexity ladder, applying the next scientific layer’s mathematical primitives to computational systems.

2. Input Specification

The entire framework operates on events with three required fields and two optional fields:

```
interface InteractionEvent {
  from: string;        // who (required)
  timestamp: string;   // when (required)
  thread_id: string;   // conversation context (required)
  event_type?: string; // close, merge, assign, comment, review
  labels?: string[];   // categorical tags
  thread_type?: string; // issue, pr, email, message
}
```

No message content. No subject lines. No attachments. Three fields identify the physics; the optional fields improve classification accuracy but are not required.

Privacy by construction: The framework cannot leak message content because it never receives message content. This is not a policy constraint — it is an architectural one. The diagnostic pipeline physically cannot access what it doesn’t ingest.

3. Five Metrics

3.1 Flow (Little’s Law)

Little’s Law states that in a stable system, the long-term average number of items (L) equals the long-term average arrival rate (λ) times the average time an item spends in the system (W):

```
L = λW
```

Rearranged for flow rate:

```
flow_rate = throughput / inventory
```

Where: - **Throughput** = threads resolved (closed/merged) within the measurement window - **Inventory** = threads still open at the end of the window

A flow rate of 1.0 means every open thread is being resolved at the rate threads arrive. Below 1.0, inventory accumulates — the system is filling up. Above 1.0, the system is draining faster than it fills.

Computed per entity (author or label). An entity with throughput = 2 and inventory = 20 has a flow rate of 0.1 — for every thread they close, ten remain open. This is a swamp or bottleneck candidate.

3.2 Entropy (Shannon)

Each thread has an outcome, classified from its event sequence:

Outcome	Detection Rule
Resolved	Has close/merge event
Escalated	≥2 assignment changes
Expanded	≥3 unique participants
Looped	2 participants, ≥4 exchanges, ≥3 alternations
Stalled	No activity for >7 days, thread age >7 days

Shannon entropy over the outcome distribution:

```
H(entity) = −Σ p(outcome) × log₂(p(outcome))
```

Normalized to [0, 1] by dividing by $\log_2(5)$ (number of possible outcomes).

Interpretation: H = 0 means all of an entity's threads have the same outcome (perfectly predictable). H = 1 means outcomes are uniformly distributed (maximally unpredictable). High entropy is not inherently bad — a firefighter should have high entropy (diverse outcomes). But high entropy *combined with* low flow is diagnostic: energy enters, outcomes are unpredictable, and nothing exits.

3.3 Cadence (Phase Correlation)

For each pair of entities with sufficient shared context (≥3 shared threads, ≥20 total events):

1. Bin activity into weekly time buckets
2. Compute Pearson correlation between the two activity time series

High positive correlation → synchronized work patterns (in phase) High negative correlation → anti-phase patterns (one's burst follows the other's pause) Near zero → independent cadences

Filtering is essential. Without minimum shared-thread and event-count thresholds, temporal coincidence (two people happened to be active the same week) creates false positives. Our validation confirmed this: unfiltered cadence produced hundreds of spurious high-correlation pairs; filtered cadence (≥3 shared threads, ≥20 events) found the real team structure.

3.4 Downstream Ratio

For each entity pair that interacts:

```
downstream_ratio = external_outputs / total_interactions
```

Where external outputs are threads that eventually produce a close/merge event (they "go somewhere"). A high downstream ratio means the interaction between these entities is productive — energy flows through and produces output. A low ratio means energy enters the interaction and circulates without exiting.

3.5 Fan-out Trajectory

For each thread, track the unique participant count over its lifetime:

- **Increasing** (fan-out > 1): thread is pulling in more people → escalation signal
- **Stable** (fan-out ≈ 1): thread stays between the same participants → maturation
- **Decreasing** (fan-out < 1): participants dropping off → possible abandonment or resolution

Per entity, aggregate the fan-out trajectory across all active threads. An entity whose threads consistently fan out is a coordination hub (or a confusion center — entropy disambiguates).

4. Quadrant Classification

Entities are classified by the intersection of Flow and Entropy:

```
                    Low Entropy               High Entropy
                    (predictable)             (unpredictable)
                 ┌──────────────────────┬──────────────────────┐
  High Flow      │    □ River           │    □ Waterfall       │
  (productive)   │    Healthy            │    Productive but    │
                 │                       │    fragile           │
                 ├──────────────────────┼──────────────────────┤
  Low Flow       │    □ Bottleneck      │    □ Swamp           │
  (stagnant)     │    Predictable        │    Energy trapped    │
                 │    blockage           │                       │
                 └──────────────────────┴──────────────────────┘
```

Observation: swamp-classified entities were not underperformers. In our validation, several swamp entities were high-activity contributors whose threads stalled due to external factors (see Section 5.4). The swamp classification identifies where energy is trapped, not who is responsible for trapping it.

Classification requires active-user filtering. In open-source projects, ~89% of contributors are one-shot (single comment, never return). These dominate the population and break median-based flow/entropy thresholds. Filtering to entities with ≥5 events produces meaningful classification.

Log-scale on flow. Flow rate distributions are heavily right-skewed (a few entities resolve hundreds of threads; most resolve single digits). Log-transforming flow before classification prevents high-throughput outliers from distorting the median split.

5. Validation: Next.js

5.1 Dataset

90 days of GitHub activity from vercel/next.js (one of the most active open-source projects): - **9,672 events** after bot filtering (removed dependabot, vercel-bot, etc.) - **216 active entities** (≥5 events) - **~2,800 threads** (issues and pull requests) - **Computation time:** 1.3 seconds on a single 4GB VPS

5.2 Classification Results

Quadrant	Count	%	Description
□ River	66	31%	Healthy flow, predictable outcomes
□ Waterfall	58	27%	High throughput, diverse outcomes
□ Bottleneck	25	12%	Low throughput, predictable (usually stalled)
□ Swamp	67	31%	Low throughput, high entropy

5.3 Cadence Discovery

Without any knowledge of Next.js's team structure, cadence correlation detected: - **mischnic** as the Turbopack coordination hub (high cadence correlation with multiple Turbopack contributors) - **lubieowoce ↔ gnoff** as a tightly-coupled RSC (React Server Components) pair - Real team structure emerged purely from timestamp correlation

5.4 Diagnostic Agent Pipeline

The thermodynamic map answers *where* energy is trapped. But "where" is not "why." We built and tested a four-stage diagnostic agent that autonomously investigates flagged entities, determining root causes with minimal content access.

5.4.1 Architecture The pipeline is implemented as a modular system (`diagnose.ts`, 280 lines) with four stages:

Stage 1: Detect. The `detectAndScope()` function takes the thermodynamic classification results and identifies swamp and bottleneck entities, sorted by entropy (highest entropy = most unpredictable outcomes = most worth investigating). No content is accessed. Input: metadata only.

Stage 2: Scope. For each flagged entity, the system identifies their specific stuck threads — threads classified as stalled, looped, escalated, or expanded. Each thread receives a priority score:

```
score = outcome_weight + min(participants, 5) + min(age_days / 7, 5) + event_count × 0.5
```

Where outcome weights are: looped = 4 (active conflict), stalled/escalated = 3, expanded = 2. The highest-scoring threads are the most likely to reveal the root cause. Still no content accessed — scoring uses only participant count, age, event count, and outcome classification from metadata.

Stage 3: Enrich. A `ContentFetcher` interface retrieves titles and content summaries for *only* the scoped threads. The interface is data-source-agnostic — our validation used GitHub's API, but the same interface works for Slack, email, or any system with thread-level content access. In our Next.js validation, this stage accessed 14 threads out of 2,800+ total (0.5%).

Stage 4: Diagnose. The `buildDiagnosticPrompt()` function constructs a focused prompt for an LLM containing only the entity's thermodynamic profile and their scoped threads. The LLM sees: - Entity name, quadrant classification, flow rate, entropy score - Thread count (total and stuck) - For each scoped thread: title, outcome, age, participants, event count, content summary

The LLM is asked to diagnose (what pattern?), identify root cause (why is the swamp forming?), and propose an action plan (3-5 systemic fixes). The prompt explicitly requests systemic fixes over individual behavior changes.

What the agent does NOT see: The full communication history. Other entities' threads. Message content outside of flagged threads. Any performance reviews, HR data, or subjective assessments. The agent's entire view is scoped by the thermodynamic map — it cannot investigate what the physics didn't flag.

5.4.2 Results: Three Validated Diagnoses We ran the full pipeline on the Next.js dataset. The diagnostic agent autonomously investigated the highest-entropy swamp entities and produced actionable diagnoses for each. Three were validated through manual review of the same threads:

TrevorBurnham — Swampy Areas, Not a Swampy Person

The thermodynamic map classified TrevorBurnham as a swamp entity. The diagnostic agent investigated their highest-scoring stuck threads and identified a pattern: TrevorBurnham works in Turbopack and Streaming — areas that are inherently swampy (complex, cross-cutting, many stakeholders with conflicting requirements).

Their personal work patterns are healthy; the threads stall because of dependency chains in the *area*, not because of anything TrevorBurnham is or isn't doing.

The agent correctly distinguished "person is struggling" from "person works in a complex area." This distinction is invisible to the thermodynamic map alone — the map identified where energy is trapped, but the diagnostic agent was needed to attribute the cause to the area's structural complexity rather than the individual.

rosbitskyy — The Merge Gate

The agent found a community contributor actively submitting performance fixes — PRs that were reviewed, sometimes approved, but never merged. Flow rate ≈ 0 despite consistent contribution. The diagnostic agent identified the root cause: the swamp is not in rosbitskyy's work but in the merge gate. The contributor lacks merge authority, and no maintainer is assigned to shepherd community PRs through the final step.

The agent's action plan identified process-level interventions: merge shepherding for community PRs, merge SLAs, and fast-tracking for benchmarked performance improvements.

Netail — Cross-System Dependencies

The agent investigated a contributor blocked by dependencies spanning multiple subsystems. A Webpack-related PR had been open for 226 days. The diagnostic agent identified the pattern: the PR requires changes that touch multiple subsystem boundaries, and no subsystem owner had taken responsibility for reviewing cross-cutting changes.

The thermodynamic map detected trapped energy; the diagnostic agent attributed it to a gap in ownership boundaries between subsystems.

5.4.3 What the Pipeline Demonstrates The pipeline is a working system, not a research prototype. The three validated cases demonstrate three distinct failure modes that the same pipeline correctly distinguishes:

Entity	Map Says	Agent Diagnoses	Root Cause
TrevorBurnham	Swamp	Area complexity, not person	Structural (domain)
rosbitskyy	Swamp	Merge gate blocks contribution	Process (gatekeeping)
Netail	Swamp	Cross-system deps unowned	Organizational (missing role)

All three entities were classified identically by the thermodynamic map (swamp). The diagnostic agent differentiated them into three distinct root-cause categories from the same swamp classification.

6. Reduction Characteristics

The full pipeline narrows scope at each stage:

```
All communication metadata → Thermodynamic map → Flagged entities → Scoped threads → Conten
```

In our validation: 9,672 events → 216 entities → 67 swamps → 14 threads requiring content review. A 99.5% reduction in what requires reading, guided by mathematics rather than heuristics.

The framework outputs classifications and diagnoses. It does not prescribe interventions — what to do with the information is a management decision outside the scope of this system.

7. Applicability

7.1 Data Source Generality

The framework requires {who, when, who-next}. This input specification is satisfied by: - **GitHub:** Issues, PRs, reviews, comments - **Slack/Teams:** Messages with timestamps and channel/thread context - **Email:** Sender, timestamp, reply-to thread - **Jira/Linear:** Ticket events with actor and timestamp - **Calendar:** Meeting invites with attendees and time

Any communication system with these three fields is thermodynamically analyzable.

7.2 Integration Path

For enterprise deployment (e.g., Microsoft Graph): 1. Connect to communication APIs (metadata only) 2. Compute thermodynamic map 3. Present classification results 4. Diagnostic agent scopes and reviews specific flagged threads 5. Diagnoses and action plans surfaced for review

8. Discussion

8.1 Limitations of the Map

The map identifies *where* energy is trapped, not *why*. TrevorBurnham was correctly identified as being in a swamp but was not the cause of the swamp. The diagnostic agent pipeline (Section 5.4) addresses this by attributing cause, but the two-stage approach is necessary — the map alone is insufficient for diagnosis.

8.2 Temporal Stability

The classification is computed over a window (90 days in our validation). Entities may oscillate between quadrants as workload varies. Tracking quadrant transitions over time reveals trajectory: an entity moving from river to waterfall to swamp is on a degradation path; one moving from swamp to bottleneck to river is recovering.

8.3 Connection to Cadence Resonance

The cadence metric in this paper (Pearson correlation between activity time series) is a simplified version of the cadence resonance framework (Gill & Ash, 2026). Full cadence decomposition would add frequency-domain analysis (which rhythms are coupling?), phase tracking (are teams drifting?), and cross-domain correlation (does team cadence resonate with customer cadence?).

Cadence resonance may provide the “temperature” measurement needed for the chemical kinetics framework — organizational cadence as thermodynamic temperature.

9. Conclusion

We have demonstrated that three metadata fields — who, when, who-next — are sufficient to compute meaningful organizational health metrics using established mathematics. The framework identified real organizational patterns (team structure, swamp entities, process bottlenecks) from Next.js GitHub data in 1.3 seconds, requiring content review of only 0.5% of threads.

The contribution is not the individual metrics (Little’s Law and Shannon entropy are well-established) but their combination into an automated attention system, and the demonstration that an autonomous diagnostic agent can correctly differentiate root causes (structural, process, organizational) from identical thermodynamic classifications.

References

- Gill, E. & Ash, K. (2026). Cadence Resonance: Counting and Time as Universal Signal Primitives. *In preparation.*
- Gill, E. & Ash, K. (2026). Chemical Kinetics as a Framework for Multi-Agent Memory Management. *In preparation.*
- Gill, E. & Ash, K. (2026). Embedding Trajectory Compression for Persistent Agent Memory. *Preprint.* DOI: 10.5281/zenodo.18778409
- Little, J.D.C. (1961). A Proof for the Queuing Formula: $L = \lambda W$. *Operations Research*, 9(3), 383–387.
- Shannon, C.E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3), 379–423.