THE STRANGLER STRANGLER FLOW

A Comprehensive Framework for AI-Native Legacy Displacement

With Self-Improving Extraction and Governance-Centered Design

Version 4.0 — April 2026

Practitioner White Paper

Executive Summary

The Strangler Strangler Flow (hereafter: the AI Strangler Flow) reimagines legacy system displacement for the agentic AI era. Unlike traditional big-bang rewrites or manual strangler fig patterns, this framework leverages large language models with million-token context windows to comprehend, extract, and modernize legacy systems continuously. Legacy codebases—COBOL, PL/SQL, Java monoliths—can be analyzed far faster than in traditional discovery workflows, often within hours rather than weeks, converted into machine-readable knowledge graphs augmented with structured markdown documents, and strangled incrementally through autonomous AI agents, with human oversight concentrated at critical decision gates.

The framework makes two core contributions. The first is a formal governance model—six human intervention gates positioned at irreversible or ambiguous junctures in an otherwise autonomous pipeline—that defines where AI execution must yield to human judgment and why. The second is a self-improving extraction loop, adapted from Karpathy's autoresearch paradigm, in which each completed displacement cycle refines the system's own extraction strategies. The displacement pipeline does not merely execute; it learns, accumulating a meta-knowledge graph of process performance that makes each successive extraction faster, more accurate, and less dependent on human correction.

Documentation Innovation: Documentation becomes persistent AI state (knowledge graphs, embeddings, executable specs, and markdown narratives) enabling zero-latency handoffs between AI agents, continuous modernization, and human-readable audit trails. Markdown is not the source of truth—it is a rendered view of the knowledge graph, auto-generated and kept in sync. Humans edit markdown to provide clarifications; AI parses these edits back into the graph.

1. The Structural Failure of Legacy Displacement

Enterprise legacy systems—COBOL batch processors, PL/SQL monoliths, tightly coupled Java estates—accumulate business logic over decades in ways that resist extraction. The systems work. They are also increasingly unmaintainable, poorly understood, and staffed by a shrinking pool of specialists. The business case for displacement is rarely in doubt. The execution, routinely, is.

Legacy displacement fails because it is governed by a temporal mismatch between two processes that must run in sequence. The first is comprehension: before any code is written, the displacement team must build a sufficiently accurate mental model of the legacy system—its data flows, its implicit business rules, its undocumented behaviors, its shadow dependencies. In traditional projects, this discovery phase consumes weeks to months of analyst time, producing artifacts that begin decaying the moment they are created.

The second process is execution: designing a target architecture, building transitional scaffolding, validating behavioral equivalence, diverting traffic, and decommissioning legacy components. This phase requires sustained organizational commitment measured in quarters or years.

The mismatch is straightforward. Comprehension is slow enough that by the time execution begins, the original analysts have rotated off, business priorities have shifted, discovery artifacts have drifted from the live system, or the sponsoring executive has moved on. The project stalls, restarts with partial knowledge, or is quietly abandoned.

The strangler fig pattern (Fowler, 2004) mitigated this risk through incremental extraction. In practice, even identifying the first viable seam requires substantial comprehension work, and the total burden is spread rather than reduced. What has changed since 2004 is the arrival of LLMs with million-token context windows and agentic tool-use capabilities. These systems can compress the comprehension bottleneck substantially—from weeks or months to days, and in some cases to hours for bounded subsystems. This compression is sufficient to close the temporal mismatch that kills legacy projects.

2. Background and Related Work

The AI Strangler Flow builds on four streams of prior work.

The strangler fig pattern (Fowler, 2004) and its elaborations in the legacy modernization literature. The pattern's core insight—that incremental extraction is safer than wholesale replacement—remains the foundation. The specific extraction tactics (Branch by Abstraction, Event Interception, Legacy Mimic, Transitional Architecture) are adopted without modification. What changes is the speed at which these tactics can be selected, configured, and validated.

Domain-driven design, particularly bounded contexts as natural extraction boundaries (Evans, 2003) and techniques for working with legacy code without comprehensive upfront understanding (Feathers, 2004). The concept of "seams"—exploitable boundaries with low coupling and high business value—is a direct operationalization of bounded context identification, accelerated by graph-based dependency analysis.

Agentic AI systems—multi-agent architectures in which language models are equipped with tools, memory, and coordination protocols to perform complex, multi-step tasks with limited human supervision. The framework draws on this work for its agent topology and persistent state management.

The autonomous self-improvement paradigm, exemplified by Karpathy's autoresearch (2026). Autoresearch demonstrated that a single AI agent, given a fixed evaluation criterion and a mutable execution space, can run hundreds of iterative optimization cycles overnight—discovering improvements that eluded expert human researchers. The architecture is deliberately minimal: a human-authored research direction (program.md), an agent-editable training script (train.py), and a fixed evaluation function (prepare.py). The agent loops through hypothesize-modify-evaluate-commit cycles, keeping changes that improve the metric and discarding those that do not. Shopify CEO Tobias Lütke demonstrated the pattern's transferability by applying it to the Liquid templating engine, yielding a 53% rendering speedup from 93 automated commits. The AI Strangler Flow adapts this paradigm from model training to legacy displacement (Section 8).

3. The Governance Model: Human Intervention Gates

The central design problem in AI-driven legacy displacement is not automation but governance. An agentic AI system capable of ingesting a codebase, generating extraction scaffolding, and diverting production traffic is also capable of making catastrophic mistakes at machine speed. The value of the framework depends entirely on the placement, design, and enforceability of human decision gates.

The AI Strangler Flow defines six gates, each triggered by a specific condition and requiring a specific category of human judgment. The gates are not optional review points. They are hard stops: the AI pipeline halts execution, serializes its state, and waits for an explicit human decision token before proceeding.

Gate	Trigger	Human Decision	Rev ?	Markdown Artifact
G0: Strategic Intent	Project initiation	Define outcomes, risk, compliance	N/A	project-brief.md
G1: Seam Selection	Multiple viable seams	Select by business priority	Yes	seams/SEAM-XXX.md
G2: Ambiguity Resolution	Confidence < 95%	Preserve bug or fix it?	Yes	validation/divergence.md
G3: Schema Transform	Migration scripts ready	Approve lossy conversions	No	adr/schema-migration.md
G4: Cutover Auth	Diversion > threshold	Authorize live traffic shift	Yes	status/dashboard.md
G5: Decommission	Zero traffic 14 days	Final shutdown & archive	No	archive/tombstone.md
G6: Exception	Anomaly detected	Investigate & approve fix	Varies	incidents/INC-XXX.md

3.1 Escalation Protocol

All gates share a common escalation mechanic. When a gate condition is triggered, the AI serializes its complete state (including the knowledge graph snapshot, all artifacts generated, and a structured decision context), halts execution, notifies the designated human stakeholder, and starts a four-hour decision timer. If the timer expires without a human response, the system enters a safe pause state: no forward progress, no rollback, full state preservation.

```
IF confidence < 0.95 OR business_criticality == "high" OR financial_impact >
"$1M/hour":
    HALT execution
    GENERATE escalation_context.json + escalation_context.md
    NOTIFY human_stakeholder
    SET timeout = 4 hours
    IF timeout_expired: PAUSE extraction, maintain fallback
```

3.2 Decision Load and Cognitive Ergonomics

The governance model assumes that humans make good decisions when presented with structured AI context. This assumption degrades under volume. A complex extraction might surface 40 or 50 ambiguities at Gate 2, each requiring domain expertise to resolve. The cognitive science literature on decision fatigue is unambiguous: the quality of human judgment deteriorates predictably with the number of sequential decisions.

The framework addresses this through three mechanisms. First, the AI batches Gate 2 decisions by category—all numeric precision ambiguities together, all encoding issues together, all business logic disputes together—to minimize context-switching cost. Second, within each batch, decisions are ordered by estimated business impact (descending), ensuring that the highest-consequence judgments are made while cognitive resources are freshest. Third, the framework defines a per-session decision ceiling: if the number of pending Gate 2 decisions exceeds a configurable threshold (default: 15 per session), the AI splits them across multiple sessions with mandatory intervals.

4. The AI Strangler Architecture

The architecture comprises five layers: a Human Sovereignty Layer providing strategic intent and gate decisions; an AI Coordination Layer managing agent orchestration, context routing, and graph-markdown synchronization; a Multi-Agent Layer of six specialized agents; a Knowledge Store Layer containing both the system knowledge graph and the process meta-knowledge graph; and the Self-Improving Ratchet Loop that feeds extraction performance data back into pipeline refinement.

4.1 Dual-Mode Documentation

AI-Native Layer: Knowledge graphs, vector embeddings, executable specs—optimized for machine consumption, queryable via graph traversal and semantic search.

Human-Readable Layer: Structured markdown documents—generated from the knowledge graph for human review, audit trails, and regulatory compliance.

Key Insight: Markdown is not the source of truth—it is a rendered view of the knowledge graph, auto-generated and kept in sync. Humans edit markdown to provide clarifications; AI parses these edits back into the graph.

4.2 Bidirectional Sync Protocol

AI → Markdown: AI updates knowledge graph, renders markdown views for human consumption, commits to Git with metadata (generated_by, graph_version).

Human → AI: Human edits markdown (clarifications, decisions, annotations). AI parses markdown on commit, extracts structured data, updates knowledge graph with source: “human_annotation”, confidence: 1.0.

Conflict Resolution: Graph is source of truth for machine operations. Human annotations override AI inferences when explicitly marked. Git history provides audit trail of all changes.

---

5. The Six Phases of AI Strangulation

5.0 Instant Archaeology

Objective: Eliminate the knowledge gap that kills legacy projects.

An agentic coding tool (Claude Code, Kimi Code, Codex) ingests the entire legacy codebase—source files, configuration, JCL, database schemas, build scripts—and produces four machine-readable artifacts plus auto-generated markdown.

The first artifact is a knowledge graph: nodes represent functions, data structures, business rules, database tables, and batch jobs; edges represent dependencies, data flows, and semantic relationships. The second is a vector embedding index for semantic search across the codebase. The third is a business rule catalog annotated with source locations, confidence scores, and natural-language summaries. The fourth is a dependency inventory of external systems.

A critical honesty about this phase: the AI’s processing time is fast, but the validated output takes longer. The knowledge graph and rule catalog must be spot-checked against production behavior, particularly for business rules with confidence scores below 95%. The claim is compression, not elimination, of human involvement.

Boundary of observability: Shadow IT—undocumented Access databases, Excel macros on individual workstations, ad hoc cron jobs on unmanaged servers—lives outside the codebase and will not appear in the knowledge graph. Organizations should conduct a shadow IT sweep as a prerequisite, not as a substitute.

Gate 0 (Strategic Intent) is exercised before or in parallel with this phase. Humans define target outcomes, risk appetite, compliance constraints, and success metrics. Human edits to the auto-

generated markdown (e.g., “The Access DB was built by Bob in 2003 when the mainframe was down. Finance still uses it daily.”) are parsed back into the knowledge graph with confidence: 1.0.

5.1 Dynamic Seaming

Objective: Find exploitable boundaries without analysis paralysis.

The AI queries the knowledge graph for extraction candidates—bounded slices with low coupling and high business value. The query is not a simple sort; the AI simulates the impact of extracting each candidate on the remaining system, modeling cascading dependency effects and identifying hidden coupling.

The output is a ranked list of 3–5 candidate seams, each specified in both machine-readable YAML (with coupling coefficient, risk score, affected business rules, interface contracts) and human-readable markdown (with executive summary, decision log, and review checkboxes).

Gate 1 (Seam Selection) is exercised at the end of this phase. Humans review the markdown, add context (“Tuesday discount is intentional—marketing promotion since 1995. Keep it.”), and approve. AI syncs back: updates knowledge graph with human_verified: true.

5.2 Abstraction Fabrication

Objective: Build strangler scaffolding automatically.

The AI generates: an adapter layer (Branch by Abstraction pattern), an event interception layer duplicating traffic, a legacy mimic facade replicating exact interface behavior (including bugs if specified at Gate 0), a differential test suite, and auto-generated Architecture Decision Records (ADRs) and rollback runbooks in markdown.

A context checkpoint is produced: a serialized state object containing all generated artifacts, open decisions requiring human input, markdown documentation references, and an entry prompt for the next agent.

Gate 2 (Ambiguity Resolution) may be triggered during this phase if the AI encounters business logic it cannot classify with sufficient confidence.

5.3 Shadow Validation

Objective: Establish high-confidence behavioral parity before any user sees the new system.

The AI establishes high-confidence behavioral equivalence through three mechanisms: mining production logs into large-scale differential test vectors; deploying in shadow mode (dark launch) comparing outputs in parallel; and performing behavioral embedding comparison that evaluates semantic meaning rather than exact string match.

Divergences are classified automatically where root cause is clear (numeric precision, whitespace, timestamp formatting) and escalated to Gate 2 where ambiguous. The validation report is generated in both YAML (machine-readable) and markdown (human-readable) with executive summary, test coverage tables, divergence analysis, performance comparison, and decision prompts.

Gate 2 (Ambiguity Resolution): Human reviews the markdown report, selects an option (e.g., “Finance confirms $0.03 immaterial. Fix it.”), and AI syncs the decision back to the knowledge graph.

Gate 3 (Schema Transformation) is exercised if data migration is required. This gate is irreversible once executed against production data.

5.4 Surgical Diversion (Ongoing)

Objective: Gradually shift traffic without big-bang risk.

Traffic shifts from legacy to modern incrementally, managed by the AI within bounds set at Gate 0. Autonomy is graduated: below a low-risk threshold defined at Gate 0, the AI may auto-proceed; above that threshold, progression requires either explicit human approval or a risk-tiered policy defined in advance by business criticality, reversibility, and blast radius.

A live markdown dashboard is auto-updated continuously by the Operator Agent, showing current traffic distribution, real-time metrics, segment-level status, recent events, pending approvals, and rollback status. Rollback restores traffic to the legacy path within 30 seconds of anomaly detection.

Gate 4 (Cutover Authorization) is exercised when diversion exceeds the configured threshold. CTO reviews dashboard, approves, adds comment. AI syncs decision to knowledge graph.

5.5 Continuous Dissolution (Post-Cutover)

Objective: Retire legacy slices and prevent new legacy creation.

Once a legacy component carries zero traffic for the defined observation period, the AI initiates the dissolution sequence: monitoring for residual access, migrating archival data, updating the corporate knowledge graph, identifying dead code, and generating comprehensive tombstone documentation in markdown.

The tombstone document captures: vital statistics (birth/death dates, lifespan, replacement system), all business rules extracted with verification status, known quirks intentionally retired, data archive locations by temperature tier (hot/warm/cold), corporate memory narratives preserved from human annotations, migration metrics, and lessons learned.

Gate 5 (Decommissioning) is exercised to authorize final shutdown. This gate is irreversible.

6. Multi-Agent Architecture

The displacement process is executed by a pipeline of six specialized AI agents, each communicating exclusively through the shared knowledge graph and structured context checkpoints.

Agent	Role	Phase	Generates
Archaeologist	Ingest & Map	Phase 0	graph.db, embeddings.idx, architecture/*.md
Strategist	Plan & Score	Phase 1	seam_manifest.json, seams/*.md, adr/*.md
Surgeon	Build & Adapt	Phase 2	Adapters, test suites, runbooks/*.md
Validator	Test & Compare	Phase 3	parity_report.json, validation/*.md
Operator	Divert & Monitor	Phase 4	Routing configs, status/dashboard.md
Archivist	Retire & Record	Phase 5	archive/*-tombstone.md, playbooks/*.md

6.1 Inter-Agent Reconciliation

A known limitation of pipeline architectures is that downstream agents may discover errors in upstream outputs. When the Surgeon encounters a dependency the Archaeologist missed, or the Validator identifies a misclassified business rule, the discovering agent writes a structured amendment to the knowledge graph and flags the discrepancy in the context checkpoint. If the amendment changes the risk profile (altering coupling coefficient or introducing a new external dependency), Gate 6 (Exception Escalation) is triggered. If additive and risk-neutral, it is applied automatically and logged for human review.

7. Dual-Mode Documentation Standards

7.1 Knowledge Graph Schema

The knowledge graph is the single source of truth, stored in a graph database (Neo4j, Amazon Neptune) and queryable by both AI agents and human analysts. Every node carries a confidence score, a provenance chain, and a link to its auto-generated markdown documentation.

```
(:Function {name: "CALC-DISCOUNT", embedding: [...]})
-[:IMPLEMENTS]->(:BusinessRule {id: "DISCOUNT-GOLD", description: "Gold Tuesday 15%"})
-[:DEPENDS_ON]->(:DatabaseTable {name: "CUSTOMER_TIER"})
-[:HAS_DOCUMENTATION]->(:MarkdownDoc {path: "rules/discount-gold.md"})
```

7.2 Markdown Document Directory

```
/docs
  /architecture/     current-state.md, transitional-state.md, target-state.md
  /business-rules/   discount-gold.md, bin-blacklist.md
```

```
/seams/            PAY-001.md, INV-003.md
/adr/              001-event-interception.md, 002-schema-migration.md
/validation/       PAY-001-report.md
/runbooks/         payment-extraction-rollback.md
/status/           migration-dashboard.md (live-updated)
/incidents/        INC-2026-0331-001.md
/archive/          PAY-001-tombstone.md
/playbooks/        payment-module-extraction-playbook.md
```

7.3 Context Checkpoints

Checkpoints are immutable serialized AI session states enabling zero-context-loss handoffs between agents. Each checkpoint references the current knowledge graph snapshot, all generated artifacts (including markdown documents), open decisions, a structured decision-context digest, and an entry prompt for the next agent. Each phase produces a new checkpoint rather than modifying the previous one, creating a complete audit trail.

7.4 Decay Prevention

For long-running displacements, the knowledge graph must remain synchronized with the live codebase. Rather than periodic rescans, the framework integrates with version control (or file-system change monitoring) to trigger incremental graph updates on every change. Each update is validated against the affected region of the graph; contradictions trigger Gate 6. Markdown documents are auto-regenerated from the graph on every commit; human edits that don’t parse back to the graph are flagged for review.

---

8. Process Self-Improvement: The Extraction Ratchet

The displacement process described in Sections 5 through 7 treats each seam extraction as an independent operation. This discards the most valuable byproduct of each extraction: knowledge about the extraction process itself. Karpathy’s autoresearch demonstrated that when the optimization loop is closed—when the system evaluates its own performance and modifies its own strategies accordingly—improvements compound.

8.1 The Three-File Mapping

The constraint structure that makes autoresearch safe maps directly onto the strangler framework.

Autoresearch Component	Strangler Flow Equivalent	Mutability
program.md	Gate 0: Strategic Intent	Human-authored only
train.py	Pipeline strategies (seam heuristics, templates, thresholds)	Agent-editable, ratcheted

prepare.py	Parity report + production metrics	Fixed evaluation criterion

The strategic intent document defines what “better” means: parity thresholds, latency bounds, coupling reduction targets, maximum Gate 2 escalations per extraction. The extraction pipeline—seam identification heuristics, abstraction generation templates, validation strategies, traffic diversion algorithms—is the mutable execution space. The parity report combined with production metrics is the fixed evaluation criterion.

8.2 The Meta-Knowledge Graph

The system knowledge graph maps the legacy system. The self-improving loop requires a second graph: a meta-knowledge graph that maps the displacement process itself. After each completed extraction, the system records: seam characteristics, extraction strategy employed, parity results, Gate 2 decision history, temporal profile (actual vs. estimated), and inter-agent reconciliation events.

Over successive extractions, this meta-graph enables data-driven strategy selection. The Strategist queries: “For seams with coupling below 0.2 and high business rule density, Event Interception achieved 99.97% parity in three previous extractions.” The Validator queries: “Numeric precision differences of this type were classified as cosmetic in all four previous occurrences; auto-classify and log.”

8.3 The Ratchet Mechanism

Each extraction pipeline configuration is versioned. Modifications that produce better results (higher parity, fewer Gate 2 escalations, shorter phase durations) are committed. Modifications that produce worse results are discarded and the pipeline reverts. The ratchet operates at the granularity of individual pipeline components, not the pipeline as a whole—a new seam scoring heuristic can be committed while a new abstraction template is discarded.

Critical safety boundary: The ratchet only operates on the mutable execution space. It never modifies the governance gates, the evaluation criteria, or the strategic intent. The system gets better at executing within its constraints; it does not modify the constraints themselves. Gate definitions are stored outside the mutable pipeline configuration and are read-only to all agents.

8.4 The Autonomy Ratio

As the meta-knowledge graph accumulates data and the pipeline ratchets toward better performance, a measurable quantity emerges: the autonomy ratio—the proportion of the displacement process that executes without human intervention. In early extractions, the ratio is low (many Gate 2 triggers, imprecise seam predictions, conservative divergence classification). As the ratchet accumulates patterns, the ratio increases—not because gates are removed but because the pipeline learns to produce outputs that clear governance thresholds without escalation.

The autonomy ratio is observable and auditable. A plateau signals convergence on the current capability ceiling; further improvement requires revised strategic intent or architectural pipeline changes—both human decisions.

8.5 The Overnight Run

An organization configures Gate 0 on Friday. Over the weekend, the system runs seam identification and impact simulation across the entire remaining monolith, generates abstraction scaffolding for the top three candidates, initiates shadow validation on the highest-confidence extraction, and writes results to the meta-knowledge graph. On Monday, the human team arrives to a ranked portfolio of extraction candidates with shadow validation already running. Each Monday is further along than the last.

8.6 From Displacement to Metabolism

The knowledge graph, meta-knowledge graph, and optimized extraction pipeline do not terminate when the last legacy component is dissolved. They persist as a living model of the modern system, maintained by the same agents. When a modern component accumulates sufficient technical debt, the same framework dissolves and replaces it. The strangler becomes permanent infrastructure—not a project to escape legacy but a metabolic process that prevents legacy from forming.

---

9. Risk Controls and Boundary Conditions

9.1 Immutability of the Legacy System

By default, no modification is made to legacy code. Changes occur through abstraction layers deployed alongside the legacy system; any direct legacy modification is treated as an exception requiring explicit approval. Rollback to the legacy path is always possible.

9.2 Parallel Execution & Rollback

The modern implementation runs in parallel for the full duration of shadow validation and traffic diversion. Rollback restores traffic in under 30 seconds, documented in auto-generated runbooks. Rollback is tested during shadow validation—the AI deliberately triggers and recovers from rollback scenarios.

9.3 Explainability

Every AI action includes decision provenance traceable to specific code lines, graph nodes, tool outputs, or production log entries, rendered in markdown for human review. The meta-knowledge graph provides additional auditability for the self-improving loop: every pipeline modification is linked to the extraction evidence that motivated it.

9.4 Parity Threshold Specification

Parity thresholds must be specified against a clear denominator at Gate 0. The recommended specification: zero semantically distinct business-impacting divergences, and below a defined threshold for cosmetic divergences. The threshold, classification criteria, and escalation path are all set at Gate 0.

9.5 The Target Architecture Assumption

This framework addresses displacement into a defined target architecture. Brownfield integration (strangling into an existing microservices estate) introduces receiving-side constraints not addressed here. Organizations pursuing brownfield displacement should conduct a target readiness assessment before initiating the AI Strangler Flow.

10. Economics

The AI Strangler Flow shifts cost structure from labor-intensive to compute-intensive. Primary recurring costs are API consumption for million-token context windows, graph database hosting (both system and meta-knowledge graphs), parallel execution infrastructure, and human time at governance gates.

The economic case rests on temporal compression amplified by the self-improving loop. A displacement that previously required a year or more compresses into weeks. The ratchet compounds this advantage: the cost curve is not strictly linear, as each successive extraction benefits from accumulated meta-knowledge, reducing repeated discovery effort and lowering coordination overhead over time.

Gate 2 and Gate 3 represent the highest human cost per extraction, but this cost decreases over successive cycles as the pipeline learns to produce fewer escalations. Decision load management (Section 3.2) ensures human cognitive resources are allocated effectively.

10.1 Temporal Compression Summary

The figures below are indicative target ranges for bounded extractions under adequate observability and infrastructure readiness, not universal guarantees.

Phase	Traditional	AI Strangler	Key Change
Discovery	Weeks/Months	Minutes (AI) + targeted validation	Instant AI comprehension
Seaming	Months	Minutes	Graph query vs. workshop analysis
Abstraction	Months	Hours	Auto-generated transitional architecture
Testing	Weeks	Hours to days	AI-generated differential tests
Cutover	Big-bang weekend	Continuous	Live AI-managed traffic diversion
Documentation	Never completed	Continuous	Living graph + auto-synced markdown

11. Failure Modes and Anti-Patterns

11.1 Hallucinated Business Rules

Language models can fabricate plausible business rules. Mitigation: no rule is accepted into the knowledge graph unless confirmed by observed production behavior. All rules with confidence below 95% require mandatory human review.

11.2 The Feature Parity Mirage

AI makes it easy to replicate everything, including features no one uses and debt no one intended. Gate 0 must explicitly define what to retire, not just what to replicate.

11.3 Over-Autonomy

The most dangerous failure mode: allowing AI to proceed past a gate because metrics look fine. As the autonomy ratio increases and gate activations become less frequent, there is organizational temptation to remove gates. This must be resisted—the gates are the safety architecture.

11.4 Documentation Theater / Markdown Drift

Don’t generate human-readable docs “just in case.” Maintain the knowledge graph as source of truth; markdown is a view, not a source. Auto-regenerate from graph on every commit; flag human edits that don’t parse back.

11.5 Knowledge Graph Drift

In long-running displacements, the graph can diverge from the live codebase. Use continuous synchronization (Section 7.4), not periodic manual rescans.

11.6 Incomplete Observability

The discovery phase can only analyze what it can access. Shadow IT lives outside the codebase. Conduct a prerequisite shadow IT inventory with human staff who have institutional knowledge.

11.7 Abandonment Without Unwinding

A partially strangled system (abstraction layers deployed, traffic still on legacy) is worse than the original. Establish an abandonment protocol at Gate 0: if terminated before cutover, remove abstraction layers, archive knowledge graph, restore pre-displacement state.

11.8 Ratchet Overfitting

If evaluation metrics are poorly specified, the pipeline converges on strategies that score well but produce poor outcomes along unmeasured dimensions (Goodhart’s Law). Warning sign: rising autonomy ratio with increasing post-deployment defect rates. Mitigation: comprehensive metric design at Gate 0 and regular human review of the meta-knowledge graph.

12. Open Questions

Regulatory Certification. Whether AI-generated migration artifacts satisfy regulatory audit requirements is an open question varying by jurisdiction and industry.

Brownfield Integration. Strangling into an existing microservices estate introduces receiving-side constraints requiring a companion target readiness framework.

Inter-Agent Trust. Subtle misclassifications may propagate undetected. Adversarial validation agents or formal verification techniques applied to the knowledge graph are potential mitigations.

Non-Code Legacy. Many systems encode logic in configuration, data, or human process. Mainframe scheduling systems embed sequencing logic invisible to code analysis.

Cross-System Orchestration. Real enterprises have dozens of interconnected legacy systems. Coordinating multiple simultaneous extractions across shared dependencies requires a meta-level orchestration layer.

Distributed Self-Improvement. Karpathy envisions autoresearch evolving to swarms of agents exploring optimizations in parallel. The analogous extension: multiple agent teams on different seams, with the meta-knowledge graph as shared memory enabling real-time cross-pollination of insights.

Framework Obsolescence. The governance model (gates) is the durable contribution; the technical implementation is expected to evolve. Separating enduring process principles from time-bound technology choices strengthens longevity.

---

13. Implementation Prerequisites

13.1 Technical Stack

Component	Options	Purpose
Vector Database	Pinecone, Weaviate, pgvector	Semantic search over embeddings
Graph Database	Neo4j, Amazon Neptune	Knowledge graph + meta-knowledge graph
Agent Framework	Claude Code, Kimi Code, Codex	Agentic tool use with million-token context
Object Storage	S3	Context checkpoints, binary fixtures
Service Mesh	Istio, Linkerd	Traffic management and canary routing
Documentation	Git + MkDocs/Docusaurus	Markdown rendering with CI/CD sync to graph

13.2 Organizational Requirements

Clear human owners for each Gate (Product Owner for Gate 1, CTO for Gate 4, Compliance for Gate 5). Sufficient AI API quotas for million-token context windows. Infrastructure for instant rollback. Distributed tracing to compare legacy vs. modern paths. Git-based markdown editing workflow with CI/CD sync to the knowledge graph.

Conclusion

The Strangler Strangler Flow treats legacy displacement not as a project with a beginning and end but as a continuous process that improves itself with each iteration. Legacy components are comprehended, extracted, validated, and dissolved incrementally by agentic AI that maintains institutional memory through a persistent knowledge graph augmented with human-readable markdown documentation.

The framework's first contribution is the governance model: six human intervention gates positioned at the junctures where strategic judgment, risk assessment, and irreversible commitment demand human authority. Its second contribution is the self-improving extraction loop: a ratchet mechanism, adapted from Karpathy's autoresearch paradigm, that transforms the displacement pipeline from a static tool into a learning system.

The temporal compression can be significant, particularly in discovery and planning, with the self-improving loop compounding gains across successive extractions. Over time, later extractions should require less repeated discovery effort and fewer escalations than earlier ones. Human expertise remains essential but shifts: from experimenter to experimental designer, from comprehending the system to governing the displacement.

The ultimate aspiration is not merely to replace legacy systems but to dissolve them continuously, safely, and incrementally into modern architecture. The knowledge graph then persists as a living model of the resulting system, enabling the same discipline to be applied again as new debt accumulates. In that sense, legacy displacement becomes less a one-time project than an ongoing organizational capability.