

SWARMAI

Agentic OS Architecture

High-Level Design Document

*Harness Engineering: How a Stateless LLM Becomes  
a Persistent, Evolving Agent*

6 Architecture Layers	Interface → Intelligence → Harness → Session → Engine → Platform
11-File Context Chain	P0–P10 priority system with token budgets and L0/L1 caching
3-Layer Memory Pipeline	Session capture → distillation → curated long-term memory (git-verified)
55+ Skills	Self-evolution: agent builds new skills when it hits capability gaps
8-Stage Autonomous Pipeline	EVALUATE → THINK → PLAN → BUILD → REVIEW → TEST → DELIVER → REFLECT
Multi-Channel Unified Brain	Desktop + Slack + Feishu — same agent, same memory, same context

Version: 1.0 • Date: March 26, 2026

Author: Xiaogang Wang (XG) + Swarm (AI Co-Architect)

Status: For PE / Tech Leadership Review • Classification: Internal

# Table of Contents

---

## 1. Executive Summary

## 2. Architecture Overview

2.1 Six-Layer Architecture • 2.2 The Compound Loop

## 3. Core Engine & Growth Trajectory

## 4. The Harness — Core Innovation

4.1 Context Engineering • 4.2 Memory Pipeline • 4.3 Self-Evolution • 4.4 Safety & Self-Harness

## 5. Swarm Brain — Multi-Channel Architecture

## 6. Session Architecture & Multi-Tab Parallel Sessions

## 7. Intelligence Layer

7.1 Autonomous Pipeline • 7.2 Job System • 7.3 Proactive Intelligence

## 8. Three-Column Command Center

## 9. Key Design Decisions & Tradeoffs

## 10. Competitive Positioning

## 11. Future Roadmap

# 1. Executive Summary

SwarmAI is a desktop application that wraps Claude's Agent SDK inside a **harness** — a structured layer of context management, persistent memory, self-evolution, and safety controls that transforms a stateless large language model into a persistent, evolving personal AI agent.

The core thesis: **most AI tools reset when you close them**. Context is lost, decisions are forgotten, and users re-explain the same things session after session. SwarmAI solves this structurally — not through fine-tuning, but through engineered knowledge persistence.

**Key Innovation:** The "Harness" — an 11-file context priority chain, 3-layer memory distillation pipeline, self-evolution registry, and 7 post-session hooks that create a *compound loop*: every session makes the next one better. Every correction prevents a class of future mistakes.

## Key Metrics (March 2026)

Metric	Value
Commits	613+
Built-in Skills	55+ (curated + self-built)
Context Files	11 (P0–P10 priority chain)
Post-Session Hooks	7 (auto-commit, DailyActivity, distillation, evolution x2, context-health, improvement)
Pipeline Stages	8 (EVALUATE → REFLECT)
Session States	5 (COLD → STREAMING → IDLE → WAITING_INPUT → DEAD)
Core Engine Level	L4 (Autonomous) — two compound-value loops closed: DDD auto-refresh + auto-skill proposals
Channels	Desktop + Slack + Feishu (unified brain)
Tech Stack	4 languages: Rust (Tauri), TypeScript (React), Python (FastAPI), SQL (SQLite)

## 2. Architecture Overview

### 2.1 Six-Layer Architecture

SwarmAI's architecture is organized into six horizontal layers. Each layer has a clear responsibility boundary. The **Harness layer** (Layer 3) is the core innovation — it is what differentiates SwarmAI from a simple LLM wrapper.

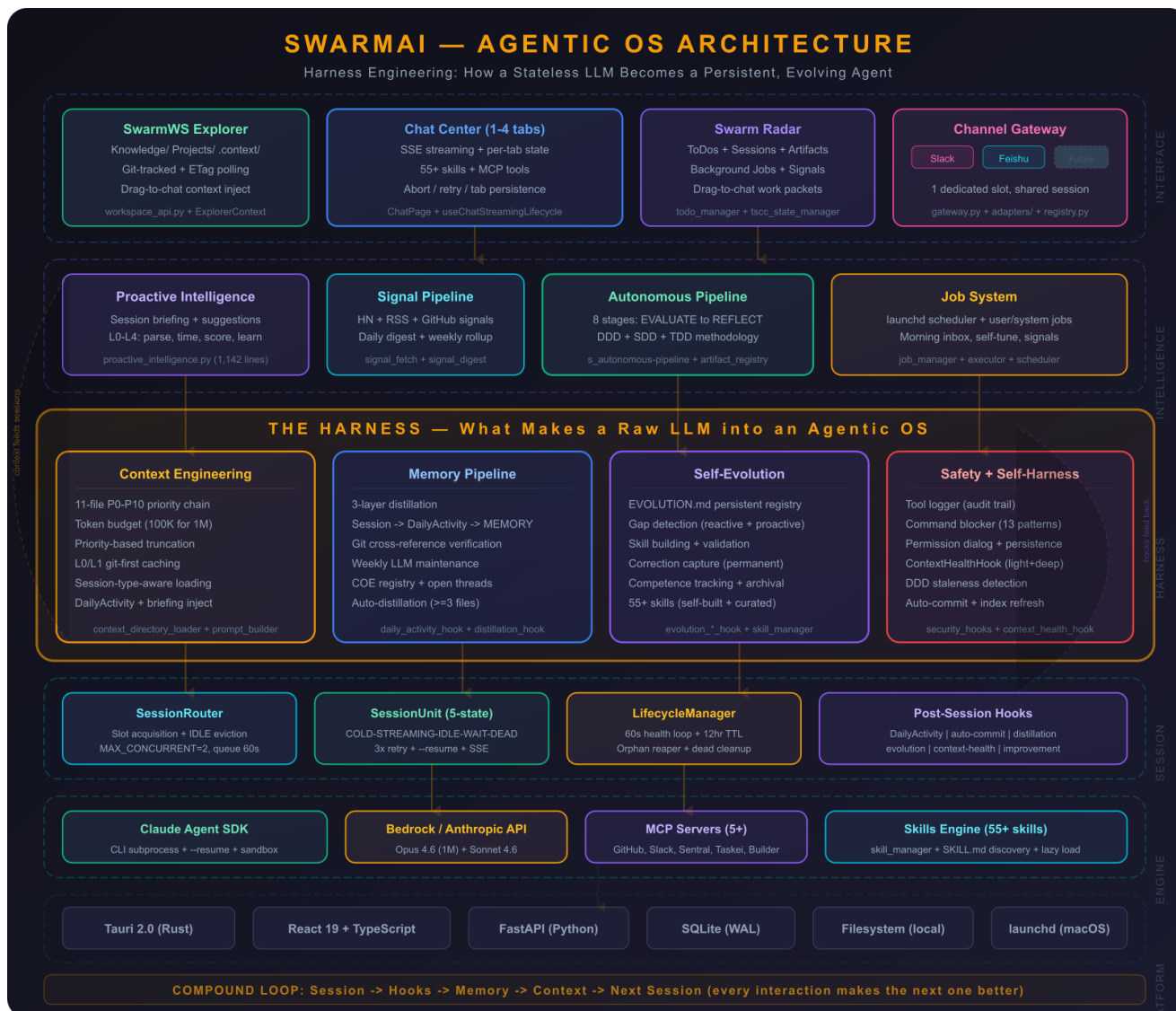


Figure 1: SwarmAI Agentic OS Architecture — Six-layer design with the Harness as the core innovation

Layer	What It Does	Key Components
Interface	Visual workspace, multi-tab chat, dashboard, channels	SwarmWS Explorer, Chat (1–4 tabs), Radar, Gateway (Slack/Feishu)
Intelligence	Proactive awareness, autonomous execution, jobs	Proactive Intelligence, Signal Pipeline, Autonomous Pipeline, Job System
Harness	Core: raw Claude → persistent, evolving agent	Context (11 files), Memory (3-layer), Evolution (55+ skills), Safety
Session	Multi-session lifecycle, isolation, recovery	SessionRouter, SessionUnit (5-state), LifecycleManager, 7 Hooks
Engine	AI model access, tool ecosystem	Claude Agent SDK, Bedrock/Anthropic, MCP Servers (5+), Skills Engine
Platform	Desktop infra, all local, zero cloud	Tauri 2.0, React 19, FastAPI, SQLite, filesystem, launchd

## 2.2 The Compound Loop

The defining characteristic is the **compound loop** — a feedback cycle where every session's output becomes the next session's input:

- **Session executes** — user interacts, decisions are made, code is written, files are created
- **Hooks fire** — 7 post-session hooks capture: DailyActivity, auto-commit, distillation, evolution, context-health, improvement, evolution-trigger
- **Memory updates** — DailyActivity accumulates;  $\geq 3$  unprocessed files trigger distillation promoting recurring themes and decisions to MEMORY.md
- **Context enriched** — next session's system prompt assembled from updated 11-file chain with latest memory and project context
- **Agent is smarter** — next session starts with full awareness of everything that happened and mistakes to avoid

**Design Principle:** Prevention over recovery. The compound loop makes errors structurally impossible over time, not handled after they occur.

## 3. Core Engine & Growth Trajectory

The Swarm Core Engine is the meta-architecture that ties all six flywheels together. Each flywheel feeds the others: memory informs context, context improves sessions, sessions trigger evolution, evolution builds skills, skills improve memory capture — compound growth with every interaction.

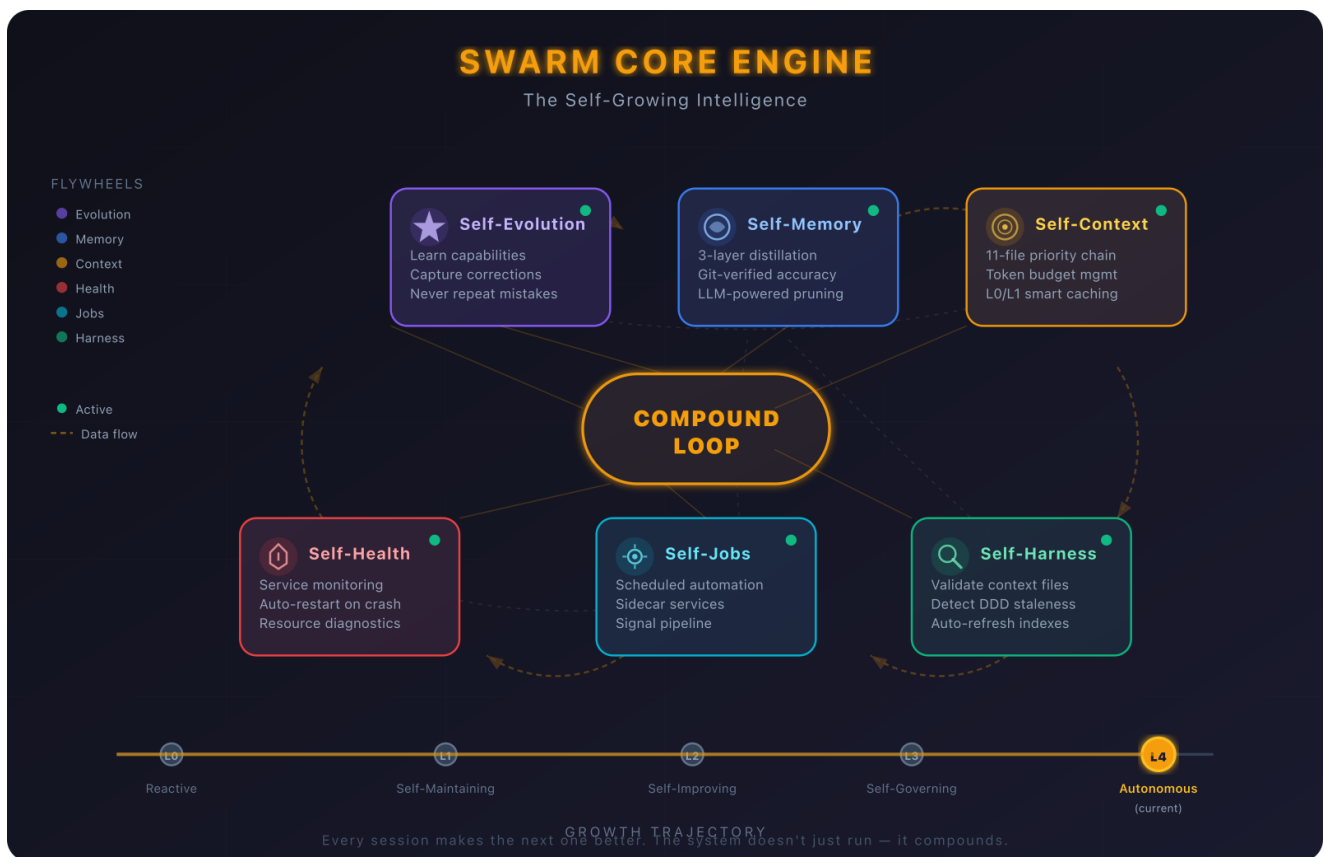


Figure 2: Swarm Core Engine — Six interconnected flywheels and growth trajectory (L4 Autonomous — current)

Flywheel	What It Does	Key Components
Self-Evolution	Builds new skills, captures corrections, never repeats mistakes	EVOLUTION.md, 55+ skills, gap detection, correction registry
Self-Memory	3-layer distillation, git-verified, weekly LLM pruning	DailyActivity, distillation hooks, MEMORY.md, briefing

Flywheel	What It Does	Key Components
Self-Context	11-file P0-P10 priority chain + token budgets + caching	Context loader, prompt builder, budget tiers, freshness
Self-Harness	Validates context files, detects DDD staleness, auto-refresh	ContextHealthHook (light+deep), auto-commit, integrity
Self-Health	Monitors services, resources, sessions; auto-restart	Service manager, resource monitor, lifecycle manager
Self-Jobs	Background automation, scheduled tasks, signal pipeline	Job scheduler, service manager, signal fetch/digest

Growth Trajectory

Level	State	Capabilities	Status
L0	Reactive	Responds to questions, no memory	Complete
L1	Self-Maintaining	Remembers, self-commits, captures corrections, health monitoring	Complete
L2	Self-Improving	Weekly LLM maintenance, unified jobs, feedback loops closed	Complete
L3	Self-Governing	Session-type context, proactive gap detection, DDD auto-sync	Complete
L4	Autonomous	Stale docs → auto-fix (DDD refresh), recurring gaps → auto-skill proposals. Two compound-value loops closed.	Current

## 4. The Harness — Core Innovation

The Harness is what makes SwarmAI more than a ChatGPT wrapper. It is a structured engineering layer between the user interface and the raw LLM that provides four critical capabilities: **context continuity**, **memory persistence**, **self-improvement**, and **safety**.

### 4.1 Context Engineering

Most AI tools assemble a single system prompt. SwarmAI maintains an **11-file priority chain (P0–P10)** that is assembled, cached, and budget-managed through a multi-stage pipeline. This is the most token-intensive subsystem and the one with the highest impact on agent quality.

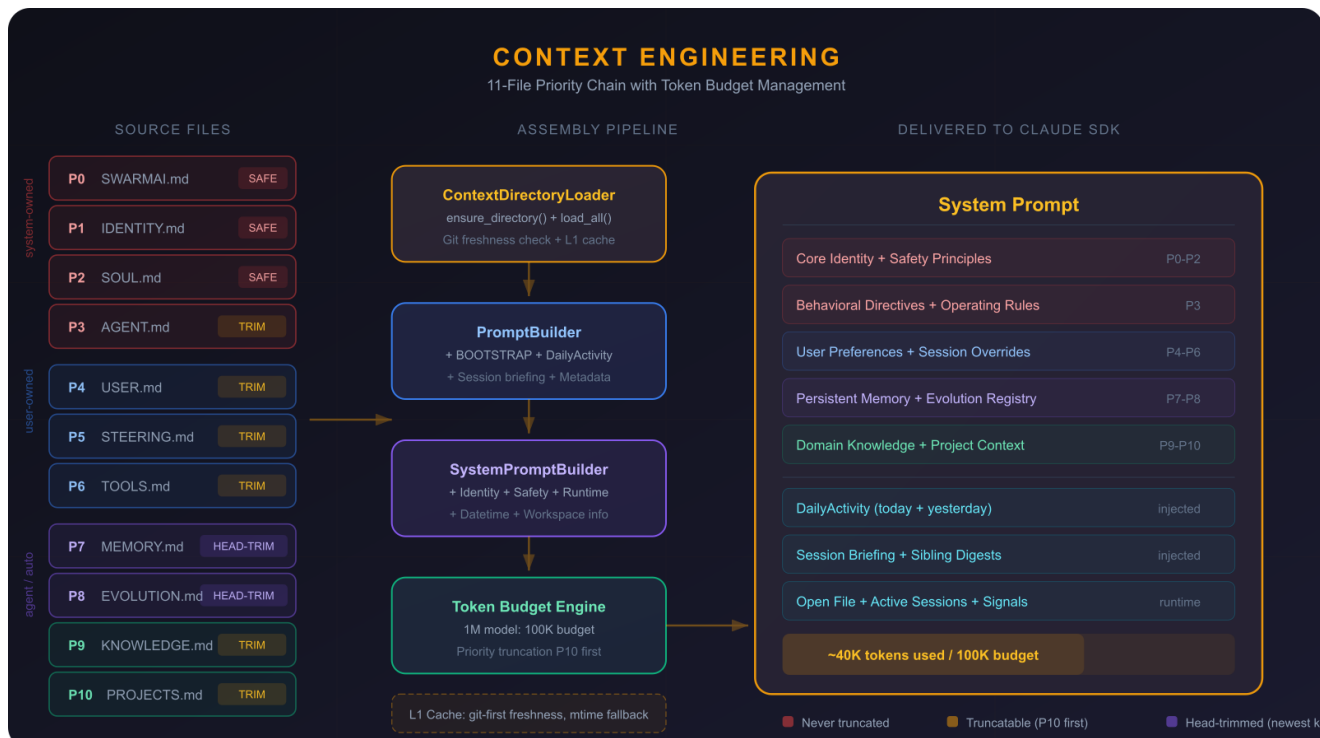


Figure 2: Context Engineering — 11-file priority chain with token budget management and L0/L1 caching

### Priority Chain

P	File	Owner	Truncation	Purpose
P0	SWARMAI.md	System	Never	Core identity & principles
P1	IDENTITY.md	System	Never	Agent name, avatar, intro
P2	SOUL.md	System	Never	Personality & tone
P3	AGENT.md	System	Truncatable	Behavioral directives
P4	USER.md	User	Truncatable	User preferences & background
P5	STEERING.md	User	Truncatable	Session-level overrides
P6	TOOLS.md	User	Truncatable	Tool & environment config
P7	MEMORY.md	Agent	Head-trimmed	Persistent memory (newest kept)
P8	EVOLUTION.md	Agent	Head-trimmed	Self-evolution registry
P9	KNOWLEDGE.md	Auto	Truncatable	Domain knowledge index
P10	PROJECTS.md	Auto	Lowest	Active projects index

### Key Design Decisions

- **Session-type-aware loading** — Channel DMs skip EVOLUTION.md, PROJECTS.md, DailyActivity (~30% token savings)
- **L0/L1 cache** — L1 uses git-first freshness; L0 is AI-summarized compact version for constrained models
- **Head-trimming** — MEMORY.md and EVOLUTION.md keep newest content; old entries trim from top
- **Token budget** — 100K tokens for 1M context models; priority truncation removes P10 first, never touches P0–P2



## 4.2 Memory Pipeline

The memory pipeline is a three-layer distillation system that converts raw session activity into durable, curated knowledge. It solves the fundamental problem of AI amnesia: without this pipeline, every session starts from zero.

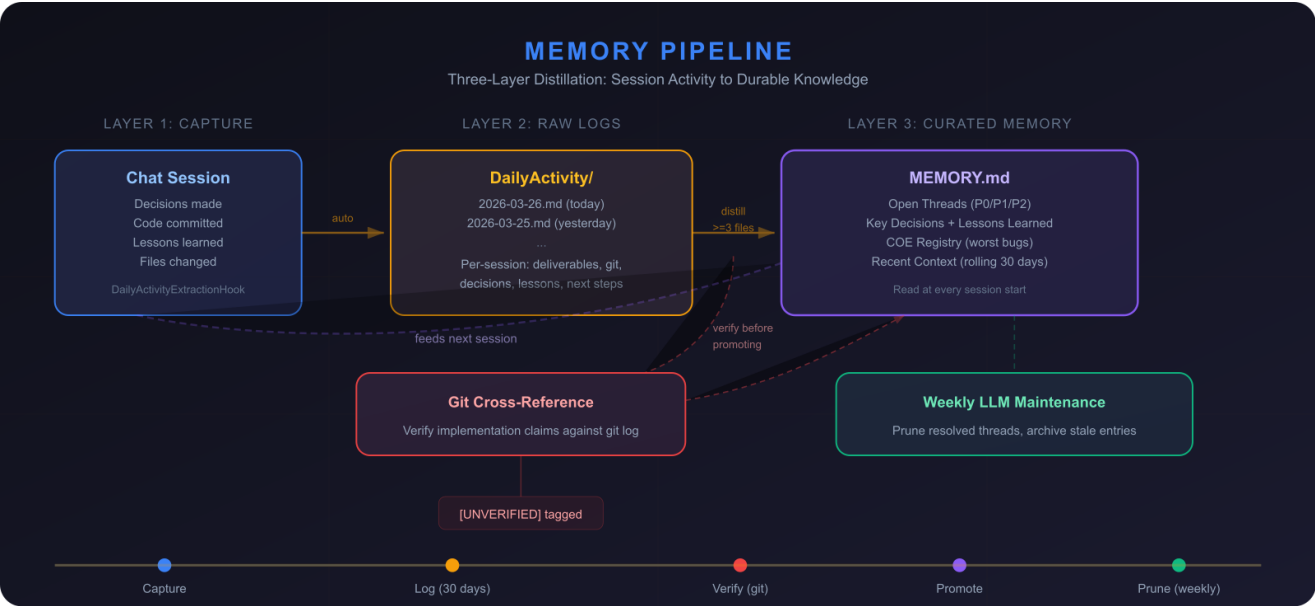


Figure 3: Memory Pipeline — Three-layer distillation from session capture to curated long-term memory

Layer	Storage	Lifecycle	Content
1. Capture	DailyActivity/YYYY-MM-DD.md	30 days → archived	Per-session: deliverables, git commits, decisions, lessons, next steps
2. Distillation	Triggered when ≥3 files	At session start (silent)	Recurring themes promoted; noise filtered; claims verified against git log
3. Curated	MEMORY.md	Permanent (weekly maint.)	Open Threads (P0/P1/P2), Key Decisions, Lessons, COE Registry

### Git Cross-Reference (Safety)

Born from a real Sev-2 incident (COE C005): the distillation hook verifies all implementation claims against `git log` before promoting to `MEMORY.md`. Without this, mid-session snapshots captured before later commits create false memories that compound across sessions.

### 4.3 Self-Evolution

Self-evolution makes SwarmAI *get better over time*. When the agent encounters a capability gap, it can build a new skill, test it, and register it. When the user corrects a mistake, the correction is captured permanently.

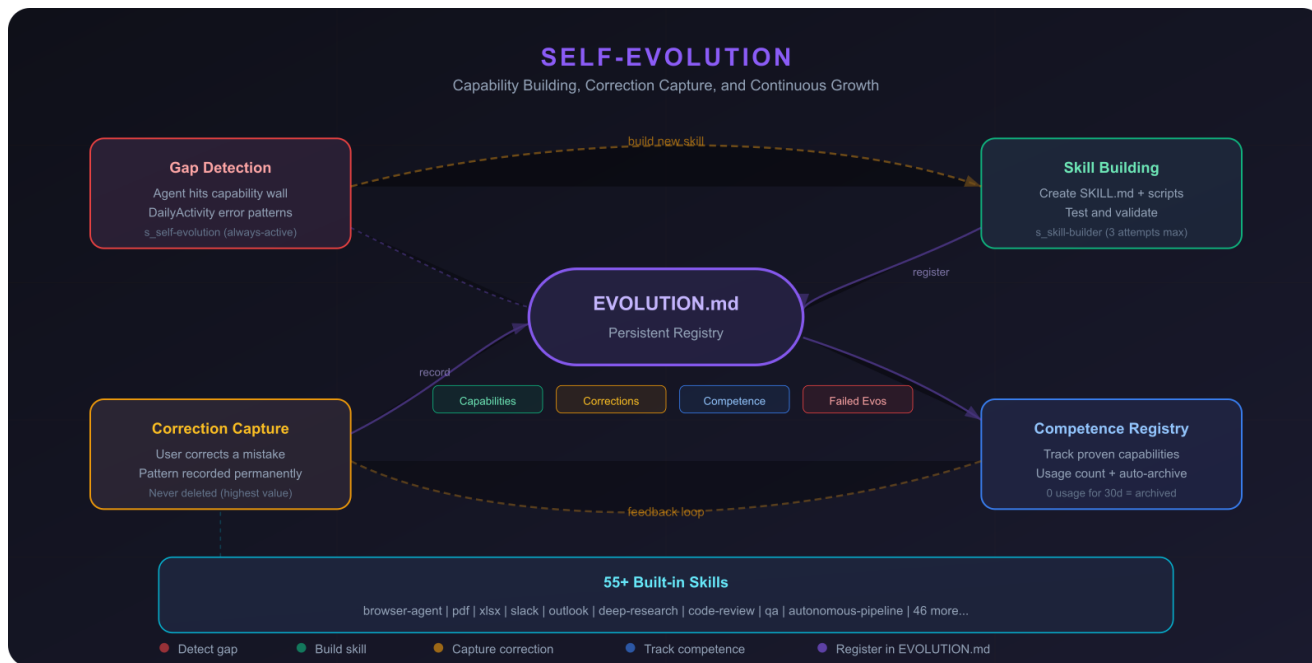


Figure 4: Self-Evolution — Capability building, correction capture, and continuous growth loop

Category	Lifecycle	Examples
Capabilities Built	Active → archived if 0 usage for 30d	Browser agent, context monitor, workspace finder
Optimizations	Permanent	Use CDP over WebSocket for persistent browser sessions
Corrections	Permanent (NEVER deleted)	Reported features as 'not started' when fully shipped (C005)
Competence	Cross-referenced	SSE streaming pipeline, multi-session architecture
Failed Evolutions	Permanent	Approaches attempted and abandoned (with reasons)

**Design Principle:** Corrections are the highest-value entries — proven failure modes with known patterns. Deleting a correction is equivalent to removing a safety guard. The registry is append-mostly; corrections are append-only.

### 4.4 Safety & Self-Harness

Safety is a structural property, not a feature. SwarmAI implements defense-in-depth through seven independent layers:

Layer	Mechanism	Details
Tool Logger	Audit trail	Every tool invocation logged with timestamp, parameters, result
Command Blocker	Pattern matching	13 dangerous patterns blocked (rm -rf, DROP TABLE, force push, etc.)
Permission Dialog	Human approval	First-time external actions require approval; approvals persist
Bash Sandbox	Claude SDK sandbox	Filesystem write restrictions, network allowlists, process isolation
Escalation Protocol	Confidence-gated	3 levels: INFORM (act+tell), CONSULT (options+ask), BLOCK (stop+wait)
ContextHealthHook	Integrity validation	Light (every session): file existence/format. Deep (weekly): staleness
Decision Classification	Judgment framework	mechanical (auto), taste (batch), judgment (block for human)

## 5. Swarm Brain — Multi-Channel Architecture

Swarm is a personal assistant with **one brain**. Regardless of channel — desktop, Slack, Feishu — it is the same Swarm, same memory, same context. Adding a new channel: write an adapter (~250 lines), register in gateway, map user identity. Zero architecture change.

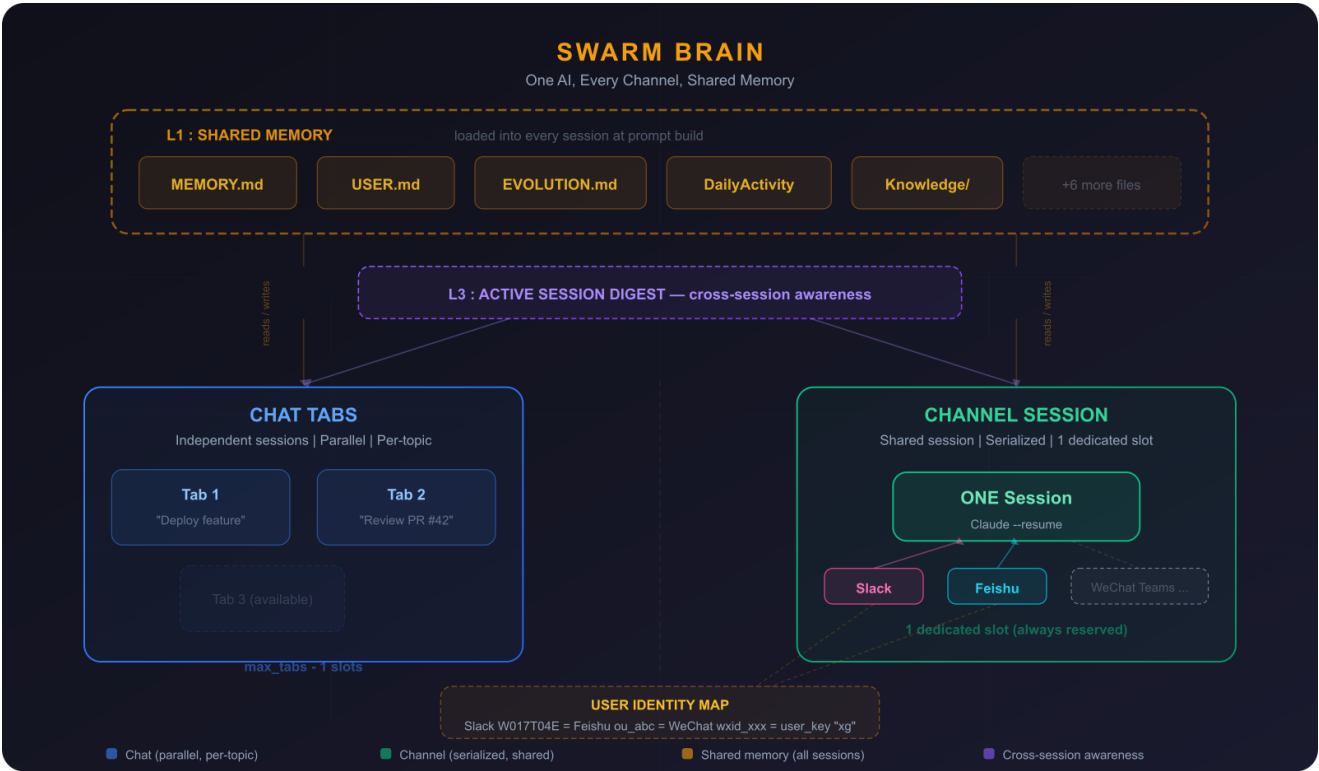


Figure 7: Swarm Brain — One AI, every channel, three layers of continuity

Layer	Mechanism	Scope
L1: Shared Memory	11 context files loaded at every prompt build	All sessions (tabs + channels)
L2: Cross-Channel Session	All channels share ONE Claude conversation (--resume)	Slack + Feishu + future
L3: Active Session Digest	Sibling session summaries injected into prompts	Tabs ↔ Channels (bidirectional)

### Key Design Decisions

- **Chat tabs are parallel** (multi-slot, per-topic) — for deep work
- **Channel session is serialized** (single dedicated slot) — for quick exchanges across platforms
- **One dedicated channel slot always reserved** (min\_tabs = 2) — channels never starve chat, chat never starves channels
- **User identity mapping** ties platform IDs (Slack W017T04E, Feishu ou\_abc) to one unified user\_key

## 6. Session Architecture & Multi-Tab Parallel Sessions

Replaced a monolithic AgentManager (5,428 lines) with four focused components during the v7 re-architecture. Driven by real need: parallel chat tabs + dedicated channel slots without resource exhaustion.

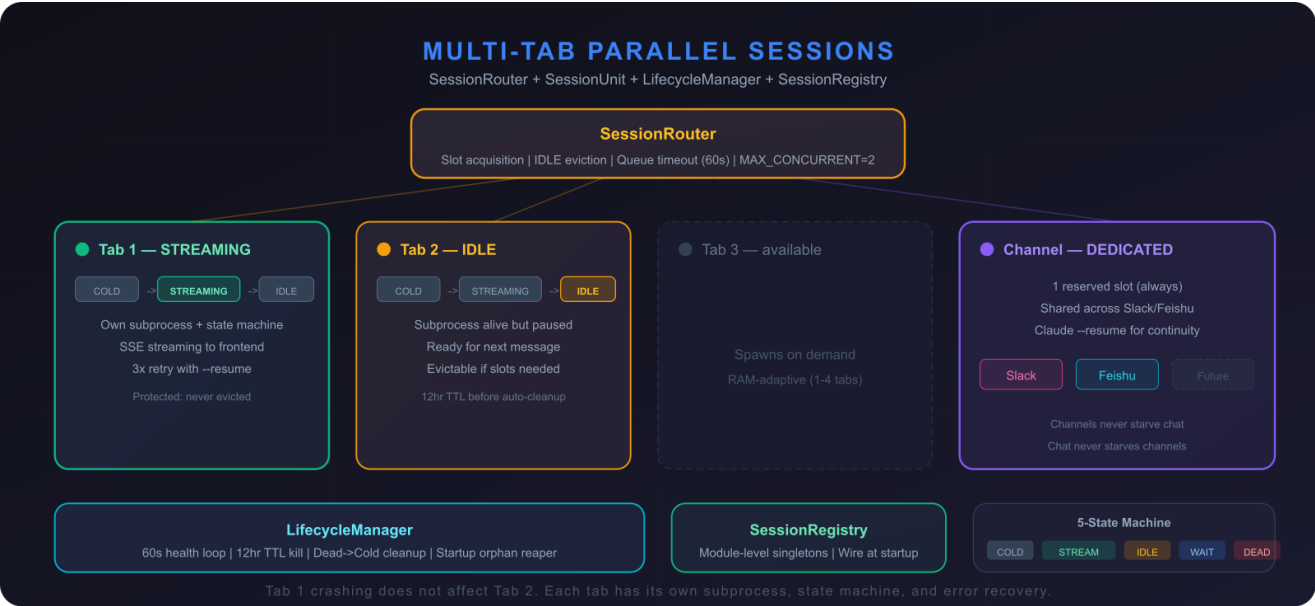


Figure 8: Multi-Tab Parallel Sessions — SessionRouter, 5-state SessionUnits, dedicated channel slot

Component	Responsibility
SessionRouter	Slot acquisition, IDLE eviction, queue timeout (60s), MAX_CONCURRENT=2
SessionUnit	5-state machine (COLD→STREAMING→IDLE→WAIT→DEAD), subprocess spawn, 3x retry with --resume, SSE
LifecycleManager	60s health loop, 12hr TTL kill, DEAD→COLD cleanup, startup orphan reaper
SessionRegistry	Module-level singletons, initialize() wires components, configure_hooks()

### Key Invariants

- Protected states (STREAMING, WAITING\_INPUT) are **never evicted**
- Subprocess spawn serialized via module-level locks
- Retry uses `--resume` to restore conversation context across crashes
- Hooks fire via BackgroundHookExecutor — never block the request path
- One dedicated slot always reserved for channels (`min_tabs = 2`)

## 7. Intelligence Layer

The Intelligence layer provides proactive awareness, autonomous execution, and background automation. While the Harness ensures the agent *remembers and improves*, this layer ensures it *anticipates, acts, and automates*.

### 7.1 Autonomous Pipeline

Drives the full development lifecycle from a one-sentence requirement to PR-ready delivery. Implementation of AIDLC Phase 3 (AI-Management): AI makes autonomous decisions, humans step in when needed.

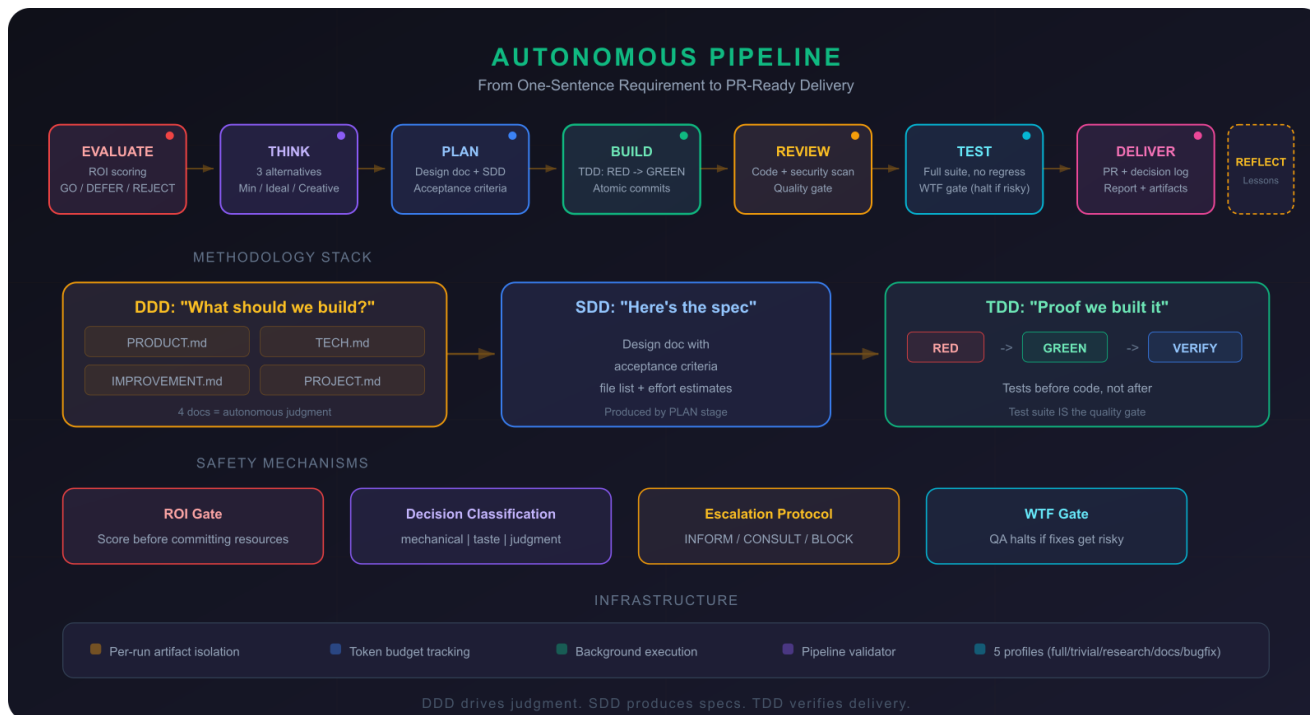


Figure 5: Autonomous Pipeline — 8-stage lifecycle with DDD+SDD+TDD methodology and safety mechanisms

Stage	Output	Gate
EVALUATE	ROI score, GO/DEFER/REJECT	ROI $\geq$ 3.5 to proceed
THINK	3 alternatives (Minimal/Ideal/Creative)	User picks approach
PLAN	Design doc (SDD) + acceptance criteria	Design approval
BUILD	Code + tests (TDD: RED → GREEN → VERIFY)	All tests pass
REVIEW	Code quality scan + security scan	No high-severity findings
TEST	Full suite, regression check	WTF Gate (halt if risky)
DELIVER	PR description, decision log, report	Taste decisions batched
REFLECT	Lessons → IMPROVEMENT.md	—

**Methodology Stack (DDD + SDD + TDD):** DDD (4 project docs) provides autonomous judgment — "should we build this?". SDD (design doc with acceptance criteria) produces specs. TDD (tests before code) verifies delivery. Key insight: when no human reviews every line, the test suite IS the quality gate.

### 7.2 Job System

Background automation via macOS launchd — runs independently of chat sessions. The scheduler evaluates due jobs every hour, routes them to type-specific handlers via the executor, and persists state across restarts. The service manager handles long-running sidecars (Slack bot) with auto-restart and health monitoring.



Figure 10: Job System — launchd scheduler, executor routing, signal pipeline, and sidecar services

Job Type	Handler	Examples	Token Cost
signal_fetch	httpx adapters (HN, RSS, GitHub)	3x daily signal collection	Zero (no LLM)
signal_digest	Sonnet 4.6 relevance scoring	Daily digest, weekly rollup	~2K tokens/run
agent	Headless Claude CLI + MCP	Morning inbox, custom tasks	Variable
script	Subprocess (deterministic)	self-tune, feed calibration	Zero (no LLM)
maintenance	Prune + cleanup	Weekly cache cleanup	Zero

System jobs (signal-fetch, signal-digest, self-tune, weekly-maintenance, weekly-rollup) are defined in code and read-only. User jobs live in `user-jobs.yaml` with full CRUD via `job_manager.py`.

7.3 Proactive Intelligence

1,142 lines, 106+ tests. Provides session-start briefings through five levels of analysis:

Level	Capability	How
L0	Parsing	Extract structured data from DailyActivity, MEMORY.md, open threads
L1	Temporal awareness	Time-sensitive items, deadlines, recency weighting
L2	Scoring engine	Priority × staleness × frequency × blocking × momentum per item
L3	Cross-session learning	JSON-persisted: skip penalty for ignored, affinity bonus for accepted
L4	Signal highlights	External intelligence (HN, RSS, GitHub) with effectiveness scoring

## 8. Three-Column Command Center

The interface is a single integrated system where the Chat Center orchestrates everything. Three columns are views into one unified workspace connected by drag-to-chat context injection.



Figure 8: Three-Column Command Center — SwarmWS, Chat Center, Swarm Radar with drag-to-chat

Column	Purpose	Key Interactions
SwarmWS Explorer (left)	Persistent local workspace	Git-tracked + ETag polling. Drag files to chat. Agent reads/writes/commits directly.
Chat Center (center)	Multi-session command surface	SSE streaming, per-tab isolation, 55+ skills, MCP tools. Controls Explorer and Radar.
Swarm Radar (right)	Attention dashboard	ToDos, sessions, artifacts, jobs. Drag work packets to chat for instant context.

## 9. Key Design Decisions & Tradeoffs

Decision	Choice	Alternative	Rationale
Memory	3-layer distillation (files)	Vector DB (RAG)	Files are git-trackable, human-readable, editable, version-controlled
Sessions	4-component decomposition	Monolithic AgentManager	5,428-line God Object caused 15+ bugs (COE). Clean error boundaries.
Context	11-file priority chain + budget	Single system prompt	Priority truncation ensures identity/safety survive under pressure
Channels	Shared session (serialized)	Independent per channel	'One brain': Slack knows what Feishu said. No fragmentation.
Skills	SKILL.md instruction files	Compiled plugins	LLM-native: agent reads as natural language. New skill = markdown file.
Data	All local (SQLite + filesystem)	Cloud database	Zero cloud dependency. Privacy by default. Works offline.
Safety	Defense-in-depth (7 layers)	Single permission gate	No single layer sufficient. Redundant protection.
Jobs	macOS launchd	In-process cron	Survives app restarts, runs when app is closed, managed by OS.



## 10. Competitive Positioning

SwarmAI occupies a unique position: not a code editor, not an IDE, not a CLI agent, not a multi-platform connector. It is an **agentic operating system** optimizing for depth over breadth.

Capability	SwarmAI	Claude Code	Kiro	Cursor	OpenClaw
Memory	3-layer pipeline	CLAUDE.md (manual)	Per-project	Per-project	Session pruning
Context	11-file + budgets	Single prompt	Spec-driven	Codebase index	Standard
Multi-session	1-4 parallel tabs	1 session	1 session	1 session	Per-channel
Self-evolution	55+ skills, corrections	No	No	No	No
Autonomous pipeline	8-stage + DDD+TDD	Manual	Spec-driven	No	No
Multi-channel	Unified brain	Terminal	IDE only	IDE only	21+ (isolated)
Scope	All knowledge work	Coding	Coding	Coding	Messaging

**Core Differentiator:** The Harness. No competitor provides the compound loop of context engineering + memory distillation + self-evolution + safety harness that makes an AI agent genuinely improve over time.

## 11. Future Roadmap

Phase	Target	Key Deliverables
L3 Completion	Q2 2026	Growth metrics dashboard, DDD auto-sync, stale correction auto-healing
L4 Autonomous	Q3 2026	Full AIDLC pipeline with checkpoint/resume, self-directed learning, judgment framework
MCP Gateway	When SDK supports	Shared MCP instances across sessions (20 → 5 instances, ~2.9GB → ~750MB)
Multi-User	Q4 2026	Team workspace, role-based access, collaborative memory
Cross-Platform	Q4 2026	Linux support (launchd → systemd for background jobs)

**Document History:** v1.0 (March 26, 2026) — Initial release for PE/Tech Leadership review.

**Generated by:** Swarm (Claude Opus 4.6) under supervision of Xiaogang Wang (XG).

**Repository:** [github.com/xg-gh-25/SwarmAI](https://github.com/xg-gh-25/SwarmAI)