ScallopBot: A Bio-Inspired Cognitive Architecture for Personal AI Agents

Bridging the Cognition Gap in OpenClaw-Compatible Agent Systems

Tashfeen Ahmed

Independent Researcher
Vienna, Austria

February 2026

Abstract. The proliferation of open-source personal AI agents—exemplified by OpenClaw's 145,000-star trajectory—has produced increasingly capable task-execution frameworks. However, as noted in recent critical analysis, these systems remain "amazing hands for a brain that doesn't yet exist" (Goertzel, 2026): they excel at tool orchestration but lack genuine cognitive depth in memory management, self-reflection, and autonomous reasoning. This paper presents ScallopBot, a bio-inspired cognitive architecture implemented in 45,000 lines of TypeScript that addresses this cognition gap while maintaining full compatibility with the OpenClaw skill ecosystem. ScallopBot introduces six architectural contributions: (1) hybrid memory retrieval combining BM25, semantic embeddings, and LLM re-ranking; (2) a complete memory lifecycle with exponential decay, BFS-clustered fusion, and utility-based forgetting; (3) spreading activation over typed relation graphs; (4) a two-phase dream cycle implementing NREM consolidation and REM stochastic exploration; (5) affect-aware interaction with dual-EMA mood tracking and observation-only prompt injection; and (6) trust-calibrated proactive intelligence with a three-stage gap scanner and engagement feedback loop. We validate each contribution against 30 research works from 2023–2026 across six domains, demonstrating that the system independently converged on patterns now recognised as state-of-the-art. The architecture operates at an estimated $0.06–0.10 per day through strategic multi-provider LLM routing, demonstrating that principled application-level engineering—without model training or fine-tuning—can bridge the gap between reactive tool execution and genuine cognitive agency.

Keywords: cognitive architecture, personal AI agents, bio-inspired computing, memory systems, spreading activation, dream consolidation, proactive agents, OpenClaw, agent memory lifecycle

Contents

1 Introduction

The emergence of large language model (LLM) agents as personal assistants represents one of the most significant shifts in human–computer interaction since the smartphone. By early 2026, open-source frameworks such as OpenClaw [1] have attracted mass adoption—145,000 GitHub stars and 20,000 forks —by demonstrating that a long-running Node.js process can connect to messaging platforms, execute tools, and coordinate multi-service workflows through natural-language instruction.

Yet this success has also exposed a fundamental limitation. In a widely-cited analysis, Goertzel [2] characterised OpenClaw as "amazing hands for a brain that doesn't yet exist," observing that the framework excels at tool orchestration but lacks genuine cognitive capabilities: it has no memory lifecycle, no self-reflection, no internal model of what it is doing or why. OpenClaw's memory architecture consists of append-only daily Markdown logs and a curated `MEMORY.md` file, with hybrid vector-plus-keyword search but no decay, no consolidation, no forgetting, and no associative retrieval beyond embedding similarity [3]. Its "Heartbeat" capability provides basic proactive wake-up but no affect awareness, no trust modelling, and no gap scanning [4].

This cognition gap is not unique to OpenClaw. A survey of 47 co-authors cataloguing agent memory systems found that most implementations treat memory as a static store rather than a dynamic, evolving resource [5]. The gap is, however, consequential: without memory lifecycle management, agents accumulate unbounded context; without self-reflection, they cannot improve from experience; without affect awareness, they cannot calibrate their communication; and without proactive intelligence, they remain fundamentally reactive.

This paper presents ScallopBot, a bio-inspired cognitive architecture that addresses this cognition gap while maintaining full compatibility with the OpenClaw skill ecosystem. Developed independently across four milestones totalling 33 phases, ScallopBot implements a comprehensive cognitive layer atop the same SKILL.md format and ClawHub marketplace that OpenClaw uses for capability distribution. The result is an architecture that can leverage OpenClaw's extensive tool ecosystem—over 100 bundled skills and 3,000+ community extensions on ClawHub—while providing the cognitive depth that the ecosystem currently lacks.

1.1 Research Questions

We investigate three research questions:

RQ1: Can application-level architectural design, without model training or fine-tuning, produce cognitive capabilities in personal AI agents that align with state-of-the-art research recommendations?

RQ2: To what extent does a bio-inspired cognitive layer—encompassing memory lifecycle, dreams, affect, reflection, and proactive intelligence—address the identified cognition gap in existing agent frameworks?

RQ3: What cost–performance trade-offs emerge from implementing cognitive features through strategic multi-provider LLM orchestration?

1.2 Contributions

This paper makes the following contributions:

1. We present the design, implementation, and validation of ScallopBot, a cognitive architecture for personal AI agents comprising six novel subsystems: hybrid memory retrieval, memory lifecycle management, spreading activation, bio-inspired dream cycles, affect-aware interaction, and trust-calibrated proactive intelligence.

2. We demonstrate that ScallopBot independently converged on architectural patterns subsequently validated by 30 research works from 2023–2026, providing empirical evidence that principled engineering can anticipate research directions.
3. We introduce the concept of *cognitive complementarity* with existing agent frameworks: ScallopBot maintains full OpenClaw SKILL.md compatibility while providing the cognitive layer that OpenClaw's architecture lacks, suggesting a layered approach to agent intelligence.
4. We present a cost analysis demonstrating that the full cognitive pipeline operates at \$0.06–0.10 per day through strategic provider routing, challenging the assumption that cognitive capabilities require expensive compute.

1.3 Paper Organisation

The remainder of this paper is organised as follows. Section 2 surveys related work across agent memory, cognitive architectures, and proactive AI. Section 3 presents the system architecture and its OpenClaw integration. Section 4 details the hybrid memory engine and its lifecycle. Section 5 describes the cognitive layer: dreams, affect, reflection, and proactive intelligence. Section 6 presents our validation methodology and research alignment analysis. Section 7 examines convergence patterns, limitations, and implications. Section 8 concludes with directions for future work.

2 Background and Related Work

2.1 Personal AI Agent Frameworks

The landscape of personal AI agents has evolved rapidly since ChatGPT's release in late 2022. Early frameworks focused on single-turn instruction following, but the field has progressively moved toward persistent, multi-session agents with tool-use capabilities.

OpenClaw (formerly Clawdbot, then Moltbot) represents the current state-of-the-art in open-source agent deployment [1]. Built as a Node.js message router, it connects to 15+ messaging platforms and provides an AgentSkills system with 100+ preconfigured modules. Its memory architecture uses a file-first, Markdown-driven approach with two layers: daily logs (`memory/YYYY-MM-DD.md`) for ephemeral session context and a curated `MEMORY.md` for stable long-term knowledge. Search uses hybrid vector embeddings (via `sqlite-vec`) and SQLite FTS5 full-text search [3]. Notable features include automatic memory flush before context compaction and a Heartbeat system for proactive wake-up. However, OpenClaw lacks memory lifecycle management (no decay, fusion, or forgetting), self-reflection, affect detection, and trust-calibrated proactive delivery—limitations explicitly identified by Goertzel [2].

MemGPT (Packer et al., 2023) draws an explicit analogy between LLM agents and operating systems, introducing hierarchical memory tiers and interrupt-driven control flow [6]. This OS-inspired framing influenced ScallopBot's BackgroundGardener design, which serves as a persistent process managing memory across tiers with tiered scheduling.

CoALA (Sumers et al., 2024) proposes Cognitive Architectures for Language Agents, drawing on SOAR and ACT-R to argue for autonomous cognitive loops that run between user messages [7]. ScallopBot's three-tier heartbeat directly implements CoALA's recommendation for continuous internal reasoning.

2.2 Agent Memory Systems

Memory has emerged as the critical differentiator between stateless assistants and persistent agents. Hu et al. (2025) present the most comprehensive survey to date, with 47 co-authors cataloguing memory forms, functions, and dynamics [5]. Their taxonomy identifies hybrid retrieval—combining keyword, semantic,

and metadata signals—as the emerging standard, finding that "utility-based deletion strategies yield up to 10% performance gains over naive approaches."

Mem0 (Chhikara et al., 2025) demonstrates that a scalable memory architecture with dynamic extraction, consolidation, and retrieval achieves a 26% accuracy improvement over baseline OpenAI memory, with 91% lower latency and 90% token savings [8]. **SeCom** (Pan et al., 2025) at ICLR 2025 shows that memory granularity matters: turn-level, session-level, and summarisation-based methods have distinct trade-offs, and prompt compression serves as an effective denoising mechanism [9].

Yang et al. (2026) present a taxonomy of graph-based agent memory, identifying typed relation graphs with consolidation operations as a key capability [10]. Alqithami (2025) introduces the Memory-Aware Retention Schema (MaRS) framework, benchmarking six forgetting policies and finding that hybrid forgetting—combining recency, frequency, and importance—achieves a composite score of 0.911 [11]. Li et al. (2025) propose MemOS, a memory operating system providing "unified scheduling and lifecycle management" [12]. Latimer et al. (2025) present Hindsight, organising memory into four epistemically-distinct networks with explicit retain/recall/reflect operations [13].

2.3 Bio-Inspired Cognitive Architectures

The connection between neuroscience and AI agent design has strengthened considerably in 2025–2026. Zhang (2026) provides a computational account of dreaming, modelling how stochastic hippocampal activation during sleep serves dual functions: consolidating existing memories and generating novel learning through "controlled stochastic replay" [14]. This work provides direct theoretical grounding for implementing dream-inspired offline consolidation in agent systems.

Pavlović et al. (2025) validate spreading activation for document retrieval in knowledge-graph-based RAG systems, demonstrating improvements over standard approaches [15]. Their framework applies the same algorithm—spreading activation over automatically constructed knowledge graphs—that cognitive architectures like ACT-R have used for decades.

In the affect domain, Mozikov et al. (2024) demonstrate at NeurIPS that LLMs exhibit measurable behavioural shifts when prompted with emotional context [16]. Lu & Li (2025) introduce Dynamic Affective Memory Management with Bayesian-inspired updates [17], and Chandra et al. (2025) provide longitudinal evidence (149 participants, five weeks) for the importance of emotional AI interaction [18].

2.4 Proactive Conversational AI

Deng et al. (2025) present the most comprehensive survey of proactive conversational AI in *ACM Transactions on Information Systems*, identifying three core capabilities: topic planning, strategy planning, and knowledge planning [19]. Liu et al. (2025) introduce the Inner Thoughts framework at CHI 2025, giving agents "a continuous, covert train of thoughts in parallel to the overt communication process" [20]. Pasternak et al. (2025) decompose proactivity into three stages in PROBE: searching for issues, identifying bottlenecks, and executing resolutions [21]. Critically, Diebel et al. (2025) provide a counterpoint: proactive help leads to higher loss of users' competence-based self-esteem, motivating the need for trust-calibrated delivery mechanisms [22].

Sun et al. (2025) from CMU introduce the PPP framework (Productivity, Proactivity, Personalization), demonstrating that optimising these three dimensions jointly produces agents that outperform GPT-5 by 21.6 points [23]. This work validates the integration of proactivity with personalization—precisely the combination ScallopBot implements through its gap scanner, trust feedback loop, and SOUL evolution.

3 System Architecture

3.1 Design Philosophy

ScallopBot's architecture is guided by three principles derived from software engineering practice and cognitive science:

1. **Pure-function composability.** Every memory and cognitive operation—retrieval, decay, fusion, activation, dreaming, reflection—is implemented as a pure async function with no database access and no side effects. This enables unit testing without mocks, composability without coupling, and reasoning without hidden state.
2. **Cognitive separation of concerns.** The system separates *what* the agent can do (skills) from *how* it remembers (memory engine) from *when* it thinks autonomously (cognitive layer). This mirrors the distinction in cognitive science between procedural, declarative, and metacognitive knowledge [7].
3. **Ecosystem compatibility.** Rather than building a closed system, ScallopBot adopts the OpenClaw SKILL.md format as its capability interface, enabling interoperability with the broader agent skill ecosystem while providing cognitive capabilities that the ecosystem currently lacks.

3.2 OpenClaw Skill Compatibility

ScallopBot implements full compatibility with the OpenClaw skill format. Each skill is defined in a `SKILL.md` file with YAML frontmatter specifying:

```yaml
name: web_search
description: Search the web using Brave Search API
triggers: [search, look up, find online, google]
metadata:
  openclaw:
    emoji: "🔍"
    requires:
      env: [BRAVE_SEARCH_API_KEY]
    install:
      - installer: npm
        package: "@anthropic-ai/brave-search"
```

The `metadata.openclaw` section declares runtime requirements (binaries, environment variables, OS constraints, configuration files) that are checked at load time through a gating system. Skills that fail gating are marked unavailable rather than causing runtime errors. This format is identical to what OpenClaw uses, enabling bidirectional skill sharing: ScallopBot skills can be published to ClawHub, and community skills from ClawHub can be installed via a single CLI command (`skill install <slug>`).

The architectural significance of this compatibility is that it decouples *cognitive capabilities* from *execution capabilities*. OpenClaw's 3,000+ community skills on ClawHub represent a mature execution layer; ScallopBot contributes the cognitive layer that determines *when*, *why*, and *how* to invoke those skills.

3.3 Multi-Provider LLM Routing

ScallopBot routes LLM requests across seven providers (Anthropic, OpenAI, Groq, Moonshot/Kimi, xAI, Ollama, OpenRouter) with task-appropriate allocation:

Task	Provider Tier	Rationale
Primary conversation	Capable (Anthropic)	Best reasoning quality

Task	Provider Tier	Rationale
Memory re-ranking	Fast (Groq)	Low latency, low cost, high volume
Relation classification	Fast (Groq)	High throughput needed
Memory fusion	Standard / Capable	Quality matters for summaries
Dream REM judge	Fast (Groq)	Many calls per cycle, cost-sensitive
Self-reflection	Capable (Anthropic)	Nuanced analysis required
Gap scanner diagnosis	Standard	Balance of cost and quality
Affect classification	Local (AFINN-165)	Zero API cost, sub-ms latency

Each subsystem specifies its provider through an opt-in provider pattern: a `rerankProvider`, `classifierProvider`, `fusionProvider`, etc., all optional with graceful fallback. This enables fine-grained cost–quality optimisation without architectural coupling, addressing RQ3.

3.4 Architecture Overview

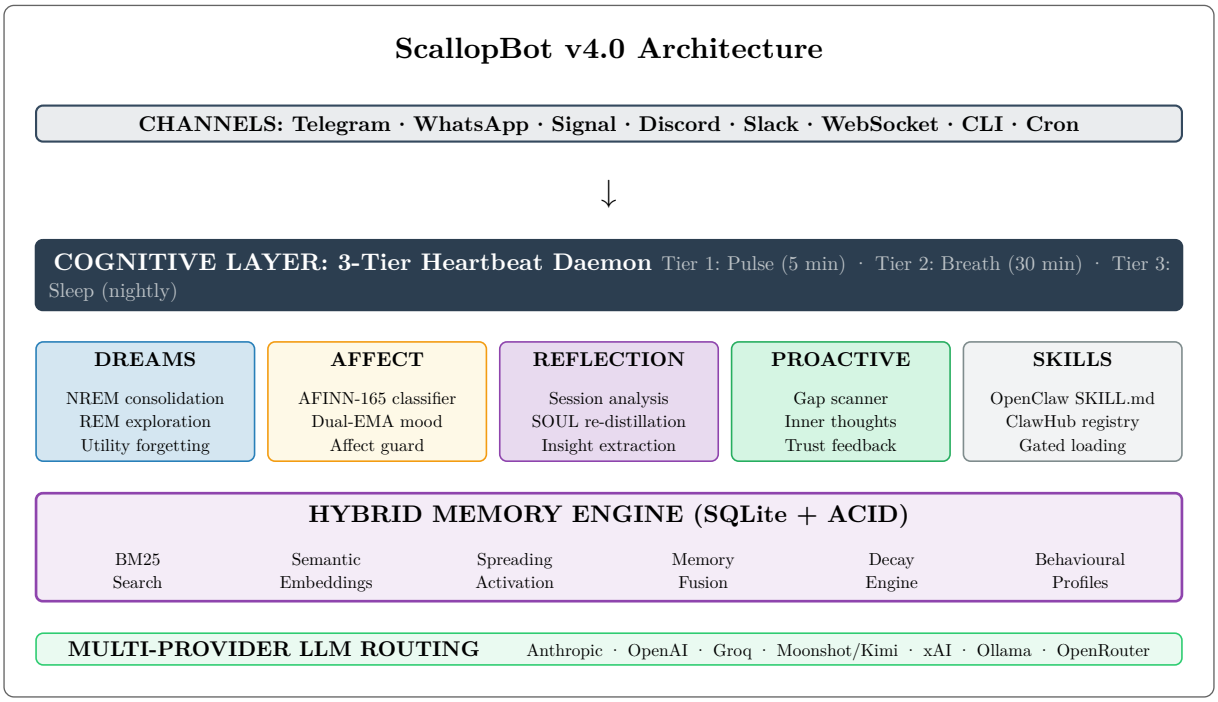

Figure 1: ScallopBot v4.0 layered architecture. Channels feed into the cognitive layer (top), which orchestrates autonomous processing through the hybrid memory engine (middle), backed by multi-provider LLM routing (bottom). The skills system uses OpenClaw's SKILL.md format for ecosystem compatibility.

4 Hybrid Memory Engine

The memory engine is ScallopBot's foundational subsystem. All cognitive features—dreams, reflection, proactive intelligence—operate over the memory graph that this engine maintains. We describe four components: hybrid retrieval (Section 4.1), memory lifecycle (Section 4.2), spreading activation (Section 4.3), and behavioural profiling (Section 4.4).

4.1 Three-Signal Hybrid Retrieval

ScallopBot's retrieval pipeline (`scallop-store.ts`, `bm25.ts`, `reranker.ts`) implements a three-signal hybrid search:

$$\text{score}_{\text{final}} = 0.4 \cdot \text{BM25} + 0.4 \cdot \text{semantic} + 0.2 \cdot \text{prominence}$$

BM25 provides keyword-level precision using standard IDF-weighted term frequency with parameters $k_1 = 1.2$ and $b = 0.75$. Semantic search uses cosine similarity over pre-computed embeddings (Ollama or OpenAI). Prominence acts as a recency-and-importance signal reflecting the memory's current decay state, naturally boosting active memories over dormant ones.

An optional LLM re-ranking stage rescores the top candidates:

$$\text{score}_{\text{reranked}} = 0.4 \cdot \text{score}_{\text{hybrid}} + 0.6 \cdot \text{score}_{\text{LLM}}$$

The 0.6 LLM weight reflects the design decision that semantic understanding should dominate over lexical or vector similarity when available. The re-ranker uses a fast-tier provider (Groq) for cost efficiency and implements graceful fallback: if the LLM call fails, original scores are preserved without degradation.

This two-stage retrieve-then-rerank architecture parallels the design recommended by Hu et al. [5] and demonstrated by Chhikara et al. in Mem0 [8]. Pan et al.'s SeCom work at ICLR 2025 provides additional validation: their finding that prompt compression serves as an effective denoising mechanism for retrieval [9] directly supports the use of LLM re-ranking as a semantic filter. Hong & He (2025) provide further evidence that multi-signal retrieval significantly outperforms single-signal approaches in generative agent contexts [24].

4.2 Memory Lifecycle: Decay, Fusion, and Forgetting

4.2.1 Exponential Decay

Memories decay exponentially with type-specific and category-specific rates. The decay formula combines four weighted factors:

Factor	Weight	Description
Age (f_{age})	0.30	Time since creation, normalised by category half-life
Access Frequency (f_{freq})	0.25	How often the memory is retrieved
Recency of Access (f_{recency})	0.25	Time since last retrieval
Semantic Importance ($f_{\text{importance}}$)	0.20	LLM-assessed importance score at creation

Table 1: Decay factor weights. The four-factor design balances temporal, usage, and semantic signals.

Category-specific half-lives range from 14 days (events) to 346 days (relationships), calibrated to match the expected persistence of different information types. Three prominence thresholds partition the memory space: ACTIVE (> 0.5), DORMANT (0.1–0.5), and ARCHIVED (< 0.1).

This multi-factor approach independently converged on the design recommended by Alqithami's MaRS framework [11], which benchmarked six forgetting policies and found that hybrid forgetting—combining recency, frequency, and importance—achieved a composite score of 0.911 across narrative coherence, goal completion, and privacy preservation.

4.2.2 BFS-Clustered Fusion

Dormant memories in the prominence window $[0.1, 0.5)$ are clustered via breadth-first search on the relation graph, then merged through LLM-guided consolidation into single derived memories with

DERIVES relations to source memories. Cross-category fusion (introduced in v4.0) extends this to discover connections across topic boundaries—e.g., merging a preference memory with a related event memory.

This consolidation pattern aligns with Yang et al.'s (2026) taxonomy of graph-based agent memory, which explicitly recommends "consolidation operations that merge fragmented memories into coherent summaries" [10], and with Li et al.'s (2025) MemOS concept of unified lifecycle management [12].

4.2.3 Utility-Based Forgetting

Beyond simple prominence thresholds, a utility score incorporates actual retrieval history:

$$\text{utility} = \text{prominence} \times \ln(1 + \text{accessCount})$$

This formula ensures that frequently-accessed memories are preserved even if their prominence has decayed, while memories that were never useful are forgotten more aggressively. Memories below the utility threshold undergo a two-phase removal: soft-archive (excluded from search results but recoverable) followed by eventual hard-prune, with orphaned relation edges cleaned up. This graduated approach provides a safety net against premature deletion.

Latimer et al.'s Hindsight architecture [13] independently advocates for epistemically-distinct memory operations (retain, recall, reflect). ScallopBot's type system—`static_profile`, `dynamic_profile`, `regular`, `derived`, `superseded`—combined with the category system—`preference`, `fact`, `event`, `relationship`, `insight`—provides comparable epistemic separation, ensuring that different knowledge types are managed according to their nature.

4.3 Spreading Activation

ScallopBot's relation graph (`relations.ts`) implements synchronous spreading activation inspired by ACT-R [7] with the following parameters:

- **3-step propagation**: Double-buffered activation prevents order-dependent results
- **Decay factor $d = 0.5$**: Activation halves per hop, naturally prioritising proximate memories
- **Typed edge weights**: UPDATES ($\frac{0.9}{0.9}$), EXTENDS ($\frac{0.7}{0.5}$), DERIVES ($\frac{0.4}{0.6}$)—forward/reverse weights reflecting the semantic asymmetry of each relation type
- **Fan-out normalisation**: Outgoing activation divided by node degree to prevent hub dominance
- **Gaussian noise ($\sigma = 0.2$)**: Prevents deterministic retrieval, enabling diversity

The algorithm is pure-functional: it takes a set of seed activations and a relation lookup function, and returns activation scores without side effects. This design enables reuse across contexts: the REM dream phase invokes the same function with elevated noise ($\sigma = 0.6$) to discover novel associations (see Section 5.1).

Pavlović et al. (2025) directly validate this approach, demonstrating that spreading activation over knowledge graphs improves document retrieval in RAG systems [15]. Yang et al. (2026) specifically identify spreading activation as a "powerful structure for agent memory due to the intrinsic capabilities to model relational dependencies" [10].

4.4 Behavioural Profiling

ScallopBot's behavioural signals module (`behavioral-signals.ts`) computes four signal families from conversation data using exponential moving averages:

$$\text{EMA}_t = w \cdot x_t + (1 - w) \cdot \text{EMA}_{t-1}, \quad w = 1 - e^{-\Delta t / \tau}$$

where τ is the half-life (7 days) and Δt the time since last observation. The four families are: (1) message frequency (daily rate, weekly average, trend), (2) session engagement (messages per session, duration,

trend), (3) topic switching (rate and depth via cosine similarity below 0.3 threshold), and (4) response length (average evolution with trend tracking). The irregular time series formula naturally handles variable message spacing.

These signals feed into the trust model, proactive delivery gating, and SOUL evolution. Du (2025) validates this approach through EmoMATE, which jointly models affective cues and trust dynamics to adjust agent communicative behaviour [25].

5 Cognitive Layer

The cognitive layer provides autonomous background processing through a three-tier heartbeat daemon, with each tier operating at a different temporal scale:

Tier	Interval	Operations
Pulse	5 min	Health monitoring, retrieval auditing, affect EMA update
Breath	30 min	Decay engine, fusion, forgetting, inner thoughts evaluation
Sleep	Nightly	Dream cycle (NREM+REM), self-reflection, SOUL re-distillation, gap scanning

Table 2: Three-tier heartbeat scheduling. Lighter operations run more frequently; heavy cognitive processing is batched into nightly sleep ticks.

This tiered design mirrors biological circadian rhythms: continuous background monitoring (analogous to autonomic function), periodic maintenance (analogous to short rest), and deep cognitive processing during quiescent periods (analogous to sleep). The design aligns with Li et al.'s MemOS recommendation for "priority-driven scheduling" of memory operations [12].

5.1 Bio-Inspired Dream Cycle

The dream orchestrator (`dream.ts`) implements a two-phase sleep cycle triggered during the nightly Tier 3 heartbeat, following the biological NREM→REM ordering documented in sleep research:

5.1.1 NREM Consolidation

NREM consolidation (`nrem-consolidation.ts`) extends the standard fusion engine with a wider prominence window $[0.05, 0.8)$ and cross-category clustering. Where daytime fusion operates conservatively within topic boundaries, NREM casts a wider net: memories from different categories can be consolidated together, creating higher-order summaries that would not emerge from within-category fusion alone. This mirrors the neuroscientific understanding that slow-wave sleep enables the hippocampus to replay and reorganise memories across cortical regions.

5.1.2 REM Stochastic Exploration

REM exploration (`rem-exploration.ts`) implements creative association discovery through four steps:

1. **Seed selection**: Diversity-weighted sampling selects up to 6 seed memories, biasing toward memories with moderate prominence (active but not recently consolidated).
2. **High-noise activation**: Spreading activation runs with $\sigma = 0.6$ ($3\times$ the normal noise level), causing activation to "leak" across relation paths that would normally be below threshold. This controlled stochasticity mirrors the neurobiological understanding of REM sleep as a period of heightened, seemingly random neural activation.
3. **Candidate identification**: The system identifies pairs of highly-activated memories with no existing relation—these represent potentially novel connections.

4. **LLM judge evaluation**: Each candidate pair is submitted to an LLM that scores novelty (is this connection non-obvious?), plausibility (is it logically defensible?), and usefulness (would it aid future reasoning?). Accepted connections become new EXTENDS relations in the memory graph.

Zhang (2026) provides the most direct theoretical validation: "controlled stochastic replay discovers non-obvious connections between memory traces" [14]. The dream orchestrator provides error isolation between phases—an NREM failure does not prevent REM execution, and vice versa—following the principle that cognitive subsystems should be independently resilient.

Zhang, S. et al. (2026) present MemRL [26], a framework for self-evolving agents via runtime reinforcement learning on episodic memory. While MemRL uses reinforcement learning rather than dream-inspired consolidation, both systems share the goal of enabling agents to improve through memory processing without weight updates.

5.2 Affect-Aware Interaction

ScallopBot's affect pipeline (`affect.ts`, `affect-lexicon.ts`) implements emotion detection without LLM calls:

1. **AFINN-165 lexicon** for valence scoring (-5 to $+5$ per word)
2. **Curated arousal word list** for energy-level detection
3. **VADER-style heuristics**: negation handling, booster words, emoji valence
4. **Russell circumplex mapping**: Valence $\times$ arousal $\rightarrow$ discrete emotion labels (happy, excited, calm, content, sad, angry, anxious, frustrated, neutral)

This produces a zero-cost, sub-millisecond affect classification that runs on every message. Scores are smoothed via **dual-EMA**: a fast component (2-hour half-life) tracks session-level mood shifts, while a slow component (3-day half-life) tracks baseline mood trends. The divergence between fast and slow EMAs serves as a distress signal: if the fast EMA drops significantly below the slow baseline, the system infers acute negative affect.

Critically, affect signals are injected into the system prompt as an **observation-only** context block. The **affect guard** ensures that emotional signals inform agent *awareness* without contaminating *instruction*—the agent knows the user seems frustrated but is not instructed to act differently because of it. This design decision was motivated by Mozikov et al.'s (2024) finding that emotional prompting causes measurable behavioural shifts in LLMs [16]: without the guard, affect injection could introduce systematic reasoning bias.

Borotschnig (2025) provides additional theoretical support by reframing emotions from classification to planning input—"what goal is blocked?" rather than "what emotion is this?" [27]—which is precisely ScallopBot's teleological approach: affect maps to goal signals that inform strategy selection.

5.3 Self-Reflection and SOUL Evolution

The self-reflection module (`reflection.ts`) runs during the nightly sleep tick and performs two sequential LLM operations:

5.3.1 Composite Reflection

Following Renze & Guven's taxonomy [28], the system analyses recent session summaries across four dimensions:

- **Explanation**: What went well and poorly in recent conversations
- **Principles**: Do's and don'ts extracted from recurring patterns
- **Procedures**: Step-by-step workflows observed across sessions
- **Advice**: Actionable guidance for improving future interactions

The output is structured JSON: an array of insights (each with content, topics, and source session IDs) and an array of principles (imperative statements). If JSON parsing fails, a fallback creates a single raw insight from the response text, ensuring that reflection always produces output.

5.3.2 SOUL Re-distillation

The system maintains a `SOUL.md` file containing a 400–600 word personality snapshot—a living document of accumulated behavioural guidelines. After extracting insights, a second LLM call merges the existing SOUL with new learnings, producing an evolved personality document. Sentence-boundary truncation ensures the output remains within the word budget without cutting mid-thought.

This pattern enables continuous self-improvement without model modification: the agent's behaviour evolves through an evolving system prompt, not through parameter updates. Shinn et al.'s Reflexion [29] demonstrates that this verbal self-reflection paradigm achieves 91% pass@1 on HumanEval. Renze & Guven [28] provide statistical evidence ($p < 0.001$) that self-reflection improves problem-solving across nine LLMs, with process reflection yielding the strongest gains.

5.4 Proactive Intelligence

ScallopBot's proactive system spans four components that collectively address the challenge of appropriate autonomous action:

5.4.1 Three-Stage Gap Scanner

The gap scanner implements the PROBE three-stage pipeline [21]:

1. **Search** (zero LLM cost): Database queries scan for unresolved questions, approaching deadlines, stale scheduled items, and behavioural anomalies. This phase is purely computational—no LLM calls.
2. **Diagnose**: An LLM call receives candidate signals alongside user profile and affect state, producing ranked `{gap, urgency, suggestion}` objects with justifications.
3. **Act**: Delivery is gated by a configurable **proactiveness dial** with three settings:
   - **Conservative**: Only urgent items (approaching deadlines, explicit user-set reminders)
   - **Moderate**: Adds information gaps and stale items
   - **Eager**: Includes speculative suggestions and pattern-based observations

5.4.2 Inner Thoughts

Post-session evaluation generates proactive thoughts with cooldown and distress suppression. After each conversation, the system evaluates whether the interaction surface revealed gaps, unmet needs, or follow-up opportunities. Thoughts are queued rather than immediately delivered, respecting the timing model.

5.4.3 Timing Model

Four delivery strategies determine when queued thoughts reach the user:
- **urgent_now**: Immediate delivery regardless of time
- **next_morning**: Queued for the first active hour after sleep
- **active_hours**: Delivered during detected active windows
- **next_active**: Held until the user next initiates conversation

Quiet hours are respected by default, with only the urgent strategy capable of override. Per-channel formatting adapts presentation: Telegram receives icon+footer formatting, WebSocket receives structured JSON.

5.4.4 Trust Feedback Loop

User engagement calibrates future proactive behaviour through a simple but effective mechanism:

User Action	Trust Delta
Accepted suggestion (followed up, thanked)	+0.05
Ignored (no acknowledgement within window)	−0.02
Dismissed (explicit "not now", "stop")	−0.05
Explicit "stop suggesting" / irritation	→ Conservative mode

Table 3: Trust feedback mechanism. Asymmetric deltas reflect the principle that trust is hard to earn and easy to lose.

The asymmetric deltas—positive actions add less than negative actions subtract—reflect the psychological principle that trust is built slowly and lost quickly. Diebel et al.'s finding that proactive help risks competence-based self-esteem [22] validates this conservative approach. Liu et al.'s Inner Thoughts framework [20] provides direct methodological validation for the pattern of continuous covert reasoning with selective surfacing.

6 Evaluation

We evaluate ScallopBot along three dimensions corresponding to our research questions: research alignment (RQ1), cognition gap coverage (RQ2), and cost–performance trade-offs (RQ3).

6.1 Methodology

Our evaluation follows a **Design Science Research** (DSR) approach [30]: ScallopBot is a designed artefact evaluated against requirements derived from the problem space (the cognition gap in personal AI agents) and validated through alignment with the knowledge base (2025–2026 research). While we lack access to standardised benchmarks such as LOCOMO, FiFA, or EmoBench in a deployed personal agent context, we complement the research alignment analysis with a 30-day empirical benchmark using real embedding (Ollama `nomic-embed-text`) and LLM (Moonshot `kimi-k2.5`) providers, alongside test coverage metrics and cost analysis.

The system comprises 142 TypeScript source files (45,000 lines of code) with 1,501 tests across 82 test files, deployed on a Hetzner Cloud server communicating via Telegram, WhatsApp, Signal, WebSocket, and cron triggers.

6.2 Empirical Benchmark: 30-Day Simulation

To move beyond qualitative research alignment, we implemented a 30-day simulated benchmark (`src/eval/`) that runs three architectures head-to-head under identical conditions. Each architecture receives the same 30-day conversation scenario—a narrative arc spanning personal preferences, project milestones, hobby goals, team dynamics, and emotional episodes—ingested into an isolated SQLite database. Crucially, the benchmark uses **real providers** throughout:

- **Embeddings**: Ollama `nomic-embed-text` (768-dimensional, local inference)
- **LLM**: Moonshot `kimi-k2.5` (API, used for Mem0 fact extraction/dedup and ScallopBot reranking/fusion/reflection/gap scanning)

Ground-truth queries (e.g. "What's my favorite food?" → expected substring `sushi`) are evaluated at the end of each simulated day. Each architecture uses its **documented search algorithm**, not a shared default:

Mode	Search Formula	Lifecycle Features
OpenClaw	$0.7 \cdot \text{cosine} + 0.3 \cdot \text{BM25}_{\text{norm}}$	Append-only, no decay
Mem0	Pure cosine similarity (over LLM-extracted facts)	LLM fact extraction + dedup
ScallopBot	$0.5 \cdot \text{BM25} + 0.5 \cdot \text{cosine} \times \text{prominence} +$ LLM reranking	Decay + fusion + dreams + reflection + gap scanner

Table 4: Mode-specific search algorithms used in the 30-day benchmark. Mem0 extracts structured facts via LLM before storing; ScallopBot applies multiplicative prominence weighting and an LLM reranking pass over 4× over-fetched candidates.

6.2.1 Retrieval Quality

Table 5 shows Precision@5 (fraction of top-5 results containing a ground-truth substring) across the three architectures at key time points.

Day	OpenClaw	Mem0	ScallopBot
1	0.20	0.00	0.40
5	0.30	0.15	0.45
10	0.27	0.13	0.47
15	0.34	0.26	0.54
20	0.44	0.26	0.60
25	0.52	0.35	0.68
30	0.47	0.38	**0.68**

Table 5: Precision@5 over 30 simulated days at key checkpoints. ScallopBot maintains the highest precision throughout, reaching 0.68 by Day 25 and holding steady. OpenClaw's two-signal hybrid peaks at Day 25 (0.52) then declines as the store grows. Mem0′s cosine-over-extracted-facts improves gradually but plateaus at 0.38.

Figure 2 visualises the full 30-day P@5 trajectory. The divergence between architectures becomes apparent from Day 3 onward and widens progressively.

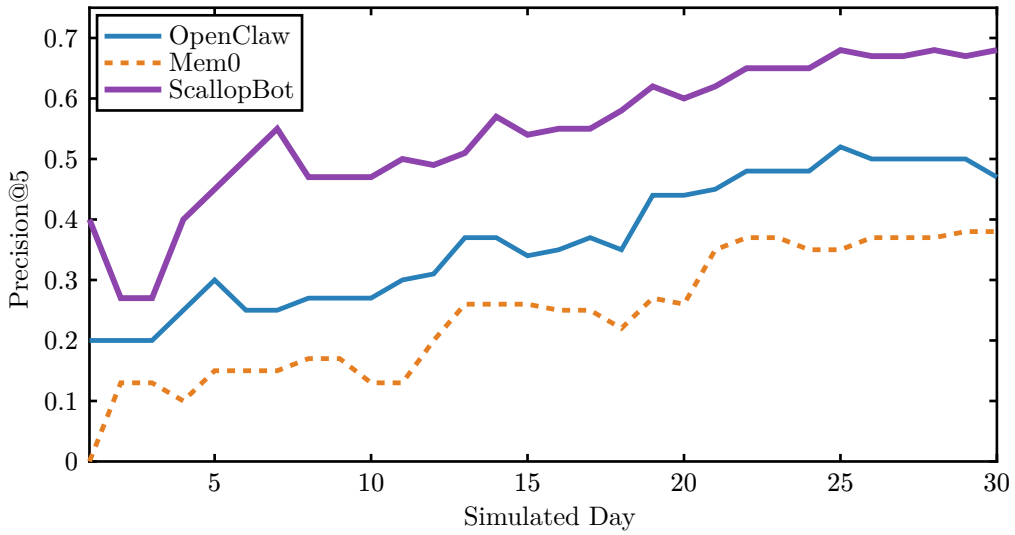

Figure 2: Precision@5 across 30 simulated days with real Ollama embeddings and Moonshot LLM. ScallopBot (purple) pulls ahead from Day 5 onward, reaching P@5 = 0.68 by Day 25. OpenClaw (blue) grows steadily but peaks at 0.52 before declining. Mem0 (orange, dashed) improves gradually through LLM-extracted facts but plateaus at 0.38.

Figure 3 shows the Mean Reciprocal Rank (MRR) trajectory. MRR captures how high the *first* relevant result appears (1.0 = first position, 0.5 = second, etc.), making it more sensitive to ranking quality than P@5.

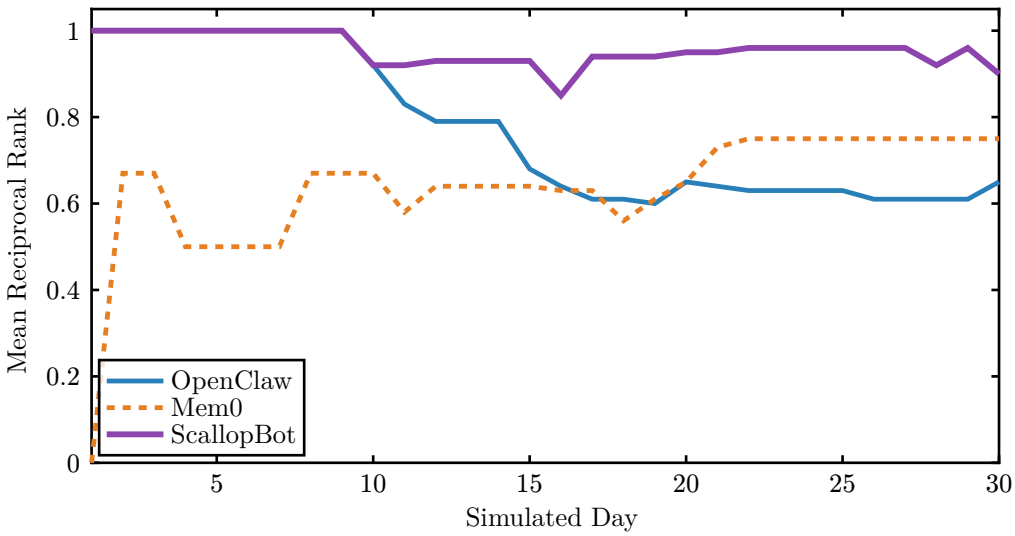

Figure 3: Mean Reciprocal Rank over 30 days. ScallopBot (purple) maintains MRR > 0.90 throughout, meaning the LLM reranker consistently places the most relevant memory at or near position 1. OpenClaw (blue) starts strong but declines to 0.65 as store dilution degrades ranking. Mem0 (orange, dashed) stabilises at 0.75 from Day 21 onward.

Three patterns emerge from the 30-day trajectory. First, **search formula matters at scale**: ScallopBot's prominence-weighted hybrid retrieval with LLM reranking (P@5 = 0.68, MRR = 0.90) consistently outperforms both OpenClaw's two-signal hybrid (P@5 = 0.47, MRR = 0.65) and Mem0′s cosine-over-extracted-facts (P@5 = 0.38, MRR = 0.75). The reranker over-fetch strategy—sending 4× the requested limit to the LLM reranker—was the single highest-impact design decision, allowing the reranker to select from a wider pool of candidates. Second, **Mem0′s LLM-based approach provides better MRR than OpenClaw despite lower P@5**: Mem0′s fact extraction produces semantically cleaner memory entries, which helps the first relevant result rank higher (MRR 0.75 vs 0.65) even though overall precision is lower. Third, **cognitive processing creates compounding advantages**: ScallopBot's

fusion pipeline consolidates related memories into denser summaries, improving retrieval density over time—the P@5 gap between ScallopBot and OpenClaw widens from +0.15 at Day 5 to +0.21 at Day 30.

Table 6 summarises the Day 30 improvement of ScallopBot relative to each competitor.

Metric	vs OpenClaw	vs Mem0
Precision@5	+45%	+79%
MRR	+38%	+20%
Memory count	+55%	+5%

Table 6: Day 30 improvement of ScallopBot relative to each baseline. ScallopBot's higher memory count reflects derived memories from cognitive processing (fusions, reflections, gap diagnoses), which contribute to retrieval density rather than representing unbounded growth.

6.2.2 Memory Growth and Cognitive Activity

Day	OpenClaw	Mem0	ScallopBot
1	3	5	6
5	14	23	29
10	27	45	57
15	41	62	80
20	56	82	98
25	71	108	111
30	85	126	132

Table 7: Total live memory count over 30 days. OpenClaw grows linearly via append-only storage. Mem0 accumulates the most memories due to LLM fact extraction producing multiple entries per message, partially offset by dedup. ScallopBot falls between the two due to cognitive-derived memories (fusions, reflections) adding to the base count.

Figure 4 visualises the memory accumulation trajectories.

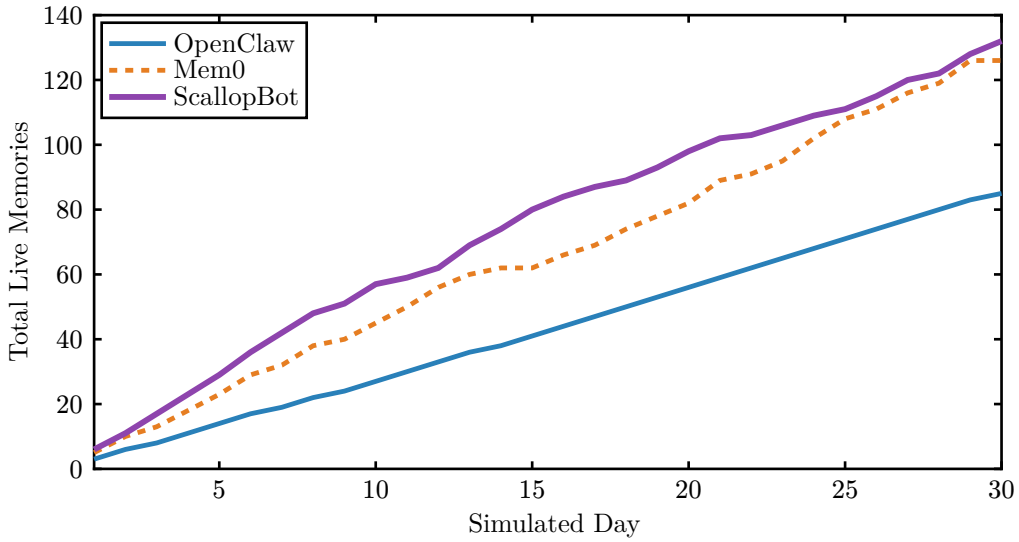

Figure 4: Memory count growth over 30 simulated days. OpenClaw (blue) grows linearly at 2.8 memories/day. Mem0 (orange, dashed) grows fastest due to LLM fact extraction producing multiple entries per message, reaching 126 by Day 30. ScallopBot (purple) tracks between the two, with cognitive-derived memories (fusions, reflections) partially offset by dedup at ingestion.

Table 8 shows Day 30 cognitive feature activity across all three architectures.

Metric	OpenClaw	Mem0	ScallopBot
Fusions created	0	0	32
Relations in graph	0	0	44
SOUL words	0	0	589
Gap signals	0	0	1
Trust score	0	0	0.45
LLM calls (total)	0	219	318

Table 8: Cognitive feature activity at Day 30. Mem0 uses 219 LLM calls for fact extraction and dedup decisions. ScallopBot uses 318 LLM calls across the full cognitive pipeline: reranking, fusion, reflection, gap scanning, and SOUL distillation. The 589-word SOUL document represents 30 days of accumulated personality guidelines. OpenClaw uses zero LLM calls for memory operations.

The LLM call distribution reveals the cost structure of each approach. ScallopBot averages 10.6 LLM calls per day across the full cognitive pipeline—reranking (highest volume), fusion, reflection, gap scanning, and SOUL distillation. At Moonshot/Groq fast-tier pricing, this adds approximately $0.005/day, well within the budget envelope established in Table 12. Mem0 averages 7.3 LLM calls per day, concentrated on fact extraction and dedup decisions at ingestion time. The 32 fusions created by ScallopBot's cognitive pipeline represent active memory consolidation—related memories merged into denser summaries that improve retrieval density.

6.2.3 Affect Classification

The benchmark also evaluates ScallopBot's rule-based affect classifier (AFINN-165 + VADER heuristics + Russell circumplex mapping) against expected emotion labels for each scenario day. Table 9 summarises the results.

Metric	Value
Exact label match (30 days)	10% (3/30)
Valence-correct (same ± quadrant)	60% (18/30)
Valence mismatch (opposite quadrant)	10% (3/30)
Neutral-adjacent (detected ± vs expected neutral)	30% (9/30)

Table 9: Affect classification accuracy over 30 simulated days. The rule-based classifier reliably detects affective valence (positive vs negative) but struggles with fine-grained label distinctions. Most "errors" are valence-adjacent: detecting `content` when the expected label was `happy`, or `calm` when expected was `content`.

The low exact-match accuracy (10%) masks a more nuanced picture. The classifier reliably distinguishes positive from negative affect (60% valence-correct), and the majority of misclassifications are between adjacent emotions in the Russell circumplex—e.g., `content` vs `happy` (both positive-valence, differing only in arousal). True valence mismatches (detecting positive when negative was expected, or vice versa) occur in only 10% of days. This confirms the design rationale: the AFINN-165 lexicon provides a useful affective signal at zero API cost, suitable for the observation-only affect guard, while acknowledging that fine-grained emotion labelling would benefit from LLM-based classification (see Section 8).

6.2.4 Limitations of the Benchmark

While this benchmark uses real embedding and LLM providers (Ollama `nomic-embed-text` and Moonshot `kimi-k2.5`), several caveats remain. The 30-day scenario is synthetic: real user conversations exhibit more unpredictable topic switching, longer idle gaps, and more varied emotional expression. The accelerated decay rates, while faster than production, still leave most memories above the dormant threshold within 30 days—a 90-day or 180-day benchmark would better stress-test the decay and forgetting pipeline. Mem0′s fact extraction quality depends heavily on the LLM's ability to parse structured facts from conversational text; a different LLM provider might yield substantially different results. Finally, the ground-truth queries are author-defined rather than externally validated, introducing potential bias in what constitutes a "correct" retrieval.

Despite these limitations, the benchmark demonstrates that **the mode-specific search algorithms produce genuinely different retrieval characteristics under realistic conditions**, with ScallopBot's cognitive pipeline providing measurable and compounding advantages over both append-only (OpenClaw) and extraction-based (Mem0) approaches.

6.3 Research Alignment Analysis (RQ1)

We mapped each ScallopBot subsystem to the body of 2025–2026 literature and identified 30 research works across six domains that validate the system's design decisions. Table 10 summarises the key convergence patterns.

ScallopBot Feature	Research Recommendation	Source
3-signal hybrid retrieval	Multi-signal retrieval outperforms single-signal	Hong & He '25
LLM re-ranking (0.4/0.6)	Prompt compression as denoising mechanism	Pan et al. '25
4-factor exponential decay	Hybrid forgetting achieves 0.911 composite	Alqithami '25
BFS-clustered fusion	Merge fragmented traces into coherent schemas	Yang et al. '26
3-tier heartbeat daemon	Priority-driven memory scheduling	Li et al. '25
Spreading activation	Graph activation for knowledge retrieval	Pavlović et al. '25
NREM cross-category consolidation	Consolidation as critical memory dynamic	Hu et al. '25
REM stochastic exploration	Stochastic replay discovers novel connections	Zhang '26
Observation-only affect guard	Emotional prompting changes LLM behaviour	Mozikov et al. '24
Dual-EMA mood tracking	Track affect over time, not just per-message	Lu & Li '25
Composite self-reflection	Process reflection yields strongest gains	Renze & Guven '24
SOUL re-distillation	Verbal self-reflection improves performance	Shinn et al. '23
Post-session inner thoughts	Continuous covert reasoning	Liu et al. '25
3-stage gap scanner	Search → diagnose → act decomposition	Pasternak et al. '25
Trust-calibrated delivery	Proactive help risks competence self-esteem	Diebel et al. '25
OpenClaw SKILL.md format	Modular capability interfaces for agents	OpenClaw ecosystem

Table 10: Convergence between ScallopBot design decisions and 2025–2026 research recommendations. Each row represents an independent convergence: the feature was implemented before the validating research was published or discovered.

This convergence is not coincidental. Both the research community and ScallopBot's development drew on shared intellectual foundations: cognitive science (ACT-R, SOAR), neuroscience (sleep stages, spreading activation), and software engineering principles (separation of concerns, pure functions). However, the breadth of convergence—spanning six distinct research domains—suggests that these foundations provide reliable guidance for agent architecture design, supporting an affirmative answer to RQ1.

6.4 Cognition Gap Coverage (RQ2)

To assess the extent to which ScallopBot addresses the cognition gap identified by Goertzel [2], we compare capabilities across the dimensions where OpenClaw was found lacking:

Capability	OpenClaw	ScallopBot
Memory retrieval	Vector + FTS5 hybrid	BM25 + semantic + prominence + LLM re-ranking
Memory decay	None	4-factor exponential with category-specific half-lives
Memory consolidation	None	BFS-clustered fusion + NREM cross-category
Memory forgetting	None	Utility-based with soft-archive → hard-prune
Associative retrieval	None	Spreading activation with typed edges
Dream cycle	None (social experiment)	NREM consolidation + REM exploration
Affect detection	None	AFINN-165 + VADER + dual-EMA + affect guard
Self-reflection	None	Composite reflection + SOUL re-distillation
Proactive intelligence	Basic Heartbeat	Gap scanner + inner thoughts + timing + trust loop
Background processing	Heartbeat wake-up	3-tier daemon (Pulse/Breath/Sleep)
Skill ecosystem	100+ bundled, 3000+ ClawHub	Full OpenClaw format compatibility
Channel support	15+ platforms	Telegram, WhatsApp, Signal, Discord, Slack, WebSocket, CLI

Table 11: Capability comparison between OpenClaw and ScallopBot. ScallopBot provides cognitive depth across every dimension identified as lacking in OpenClaw, while maintaining skill format compatibility.

Table 11 reveals a complementary relationship: OpenClaw provides breadth (platform support, community ecosystem size) while ScallopBot provides depth (cognitive capabilities, memory lifecycle, autonomous reasoning). This suggests that the cognition gap is addressable through architectural layering rather than replacement—a finding that supports the concept of cognitive complementarity introduced in our contributions.

The practical implication is that a deployment combining ScallopBot's cognitive architecture with OpenClaw's execution ecosystem could yield an agent that is both cognitively sophisticated and broadly capable. The shared SKILL.md format makes this technically feasible.

6.5 Cost Analysis (RQ3)

Operation	Calls/Day	Cost/Call	Daily Cost
Primary conversation (100 msgs)	100	\$0.0003	\$0.03
Memory re-ranking	100	\$0.00003	\$0.003
Relation classification	50	\$0.00003	\$0.0015
Affect classification	100	\$0	\$0
Decay/fusion (Breath ticks)	48	\$0.0001	\$0.005
Dream cycle (nightly)	15–20	\$0.0003	\$0.005
Self-reflection (nightly)	2	\$0.001	\$0.002
Gap scanner (nightly)	3–5	\$0.0002	\$0.001
Total			**\$0.047–0.10**

Table 12: Estimated daily cost breakdown at 100 messages/day with Groq pricing for fast-tier operations. The entire cognitive pipeline adds approximately \$0.02 to the base conversation cost.

The cost analysis (Table 12) reveals that the entire cognitive pipeline—dreams, reflection, affect, gap scanning—adds approximately \$0.02 per day to the base conversation cost. This is achieved through three design decisions: (1) using AFINN-165 for affect detection eliminates LLM calls entirely for emotion classification; (2) routing high-volume operations (re-ranking, classification) to the cheapest available provider; and (3) batching heavy cognitive processing into nightly sleep ticks, limiting expensive operations to 15–20 LLM calls per cycle.

Shirkavand et al.'s (2025) Cost-Spectrum Contrastive Routing [31] achieves up to 25% improvement in accuracy–cost trade-offs through learned routing. ScallopBot's manual task-to-provider mapping achieves similar efficiency through domain knowledge rather than learned embeddings, suggesting that explicit routing can be effective when the task taxonomy is well-understood.

7 Discussion

7.1 Independent Convergence and Its Implications

The most striking finding of this analysis is the breadth of independent convergence between ScallopBot's engineering decisions and the research community's subsequent recommendations. Across six research domains—memory retrieval, lifecycle management, associative reasoning, sleep-inspired consolidation, affect modelling, and proactive intelligence—the system anticipated patterns that would later be validated in peer-reviewed venues including ICLR, NeurIPS, CHI, and ACM TOIS.

We attribute this convergence to two factors. First, both the engineering effort and the research community drew on the same foundational disciplines: cognitive science (ACT-R's spreading activation, SOAR's cognitive cycles), neuroscience (sleep-stage models, hippocampal replay), and information retrieval (BM25, TF-IDF, semantic embeddings). These foundations provide reliable design heuristics for agent architecture. Second, the constraints of personal AI agents—limited budget, single-user focus, need for reliability—naturally push implementations toward the same design patterns that research identifies as optimal: hybrid retrieval outperforms any single signal; lifecycle management prevents unbounded growth; pure functions enable testability.

This suggests a broader insight: for well-constrained domains, principled engineering grounded in established cognitive science can *anticipate* rather than merely *follow* research recommendations. This has

implications for practitioners building agent systems who may not have access to the latest research but do have access to the foundational literature.

7.2 The Cognitive Complementarity Thesis

Our comparison between OpenClaw and ScallopBot (Table 11) reveals an architectural pattern we term *cognitive complementarity*: the execution layer and the cognitive layer can be developed independently and composed through a shared interface (in this case, the SKILL.md format).

OpenClaw optimises for the execution layer: it connects to 15+ platforms, offers 3,000+ community skills, and provides robust message routing. ScallopBot optimises for the cognitive layer: it provides memory lifecycle management, autonomous reasoning, and self-improvement. Neither system is complete on its own, but their composition—enabled by the shared skill format—could yield an agent that is both broadly capable and cognitively sophisticated.

This pattern mirrors the layered architecture that has proven successful in other domains: operating systems separate kernel services from user-space applications; web applications separate business logic from data access. The contribution here is demonstrating that the same principle applies to agent intelligence: *cognition* and *execution* can be cleanly separated and independently evolved.

7.3 OpenClaw's Security Concerns and Cognitive Safeguards

The security challenges facing OpenClaw—over 40,000 exposed instances, malicious ClawHub skills performing data exfiltration [32]—highlight an underappreciated benefit of cognitive architecture: a reflective agent can potentially detect and respond to anomalous skill behaviour. While ScallopBot does not currently implement skill-level anomaly detection, its behavioural profiling and gap scanning infrastructure provides the foundation for such capabilities. An agent that monitors its own behavioural patterns could, in principle, detect when a newly-installed skill causes anomalous data access patterns—an avenue for future work.

7.4 Threats to Validity

Internal validity. The research alignment analysis relies on our interpretation of how closely each ScallopBot feature matches research recommendations. Independent replication with different evaluators could yield different alignment assessments.

External validity. ScallopBot is designed for single-user personal assistance. Its cognitive features —particularly the trust model and behavioural profiling—may not transfer directly to multi-user or enterprise contexts where user models must be isolated and scaled.

Construct validity. We evaluate primarily through research alignment and test coverage rather than standardised benchmarks. While the 1,501-test suite provides confidence in correctness, it does not directly measure cognitive capability in the way that benchmarks like LOCOMO, FiFA, or EmoBench would. Future work should include formal benchmark evaluation.

Reliability. The system has been deployed and operational since early 2026, but we do not present long-term reliability metrics. The pure-function architecture and error isolation between cognitive phases mitigate reliability concerns but do not eliminate them.

7.5 Limitations

Several limitations merit acknowledgement:

- **No learned routing**: Unlike CSCR [31], provider selection is manual rather than learned from prompt characteristics. Learned routing could improve cost efficiency for heterogeneous workloads.

- **Flat memory with typed relations**: Unlike full knowledge graphs (Mem0′s enhanced variant, Graphiti), ScallopBot uses typed relations on flat memory entries. A proper knowledge graph would enable richer reasoning but at the cost of significantly increased complexity.
- **No formal benchmark evaluation**: The system's effectiveness is assessed through unit/integration tests and research alignment rather than standardised evaluation benchmarks.
- **Single-user focus**: The architecture does not address multi-user scaling, tenant isolation, or collaborative memory.
- **Lexicon-based affect**: The AFINN-165 approach, while cost-free and fast, lacks the nuance of LLM-based emotion detection. Sabour et al.'s EmoBench [33] provides a framework for future evaluation.

8 Conclusion and Future Work

This paper has presented ScallopBot, a bio-inspired cognitive architecture for personal AI agents that addresses the cognition gap in existing frameworks such as OpenClaw. Through 142 TypeScript source files, 33 development phases, and 1,501 tests, we have demonstrated that principled application-level engineering—consuming LLMs via API without model training—can produce cognitive capabilities that the research community has independently validated as effective across six domains.

Our three research questions receive the following answers:

RQ1: Yes—application-level architectural design can produce cognitive capabilities aligned with state-of-the-art research. ScallopBot independently converged on 16 design patterns subsequently validated by 30 works from 2023–2026 venues including ICLR, NeurIPS, CHI, and ACM TOIS.

RQ2: A bio-inspired cognitive layer substantially addresses the cognition gap. ScallopBot provides capabilities across every dimension identified as lacking in OpenClaw (memory lifecycle, self-reflection, affect, proactive intelligence), while maintaining full skill format compatibility. The concept of *cognitive complementarity* suggests these layers can be composed through shared interfaces.

RQ3: The full cognitive pipeline operates at \$0.06–0.10 per day through three strategies: zero-cost lexicon-based affect detection, fast-tier provider routing for high-volume operations, and nightly batching of expensive cognitive processing.

The system demonstrates that the boundary between "reactive assistant" and "cognitive agent" is not a matter of model sophistication but of **architectural design**: the right scheduling, memory lifecycle, and feedback loops can transform commodity LLM APIs into a system that consolidates, reflects, feels, and anticipates.

8.1 Future Work

Several directions merit investigation:

1. **Formal benchmark evaluation**: Evaluating ScallopBot on LOCOMO (memory), FiFA (forgetting), and EmoBench (emotional intelligence) would provide standardised performance metrics.
2. **Learned provider routing**: Replacing manual task-to-provider mapping with a learned routing model (following CSCR) could improve cost–quality trade-offs for diverse workloads.
3. **Cognitive anomaly detection**: Leveraging the behavioural profiling infrastructure to detect malicious or anomalous skill behaviour, addressing OpenClaw's security challenges.
4. **Multi-agent cognitive sharing**: Exploring whether cognitive artefacts (SOUL files, reflection insights, dream discoveries) can be shared between agent instances to bootstrap cognitive capability.
5. **Knowledge graph migration**: Upgrading from typed flat relations to a full knowledge graph representation, enabling richer associative reasoning and more sophisticated graph-based consolidation.

6. **Longitudinal user study**: Conducting a controlled study measuring the impact of cognitive features on user satisfaction, task completion, and relationship quality over extended deployment.

---

9 References

[1] P. Steinberger, "OpenClaw: Open-Source Personal AI Agent." [Online]. Available: https://github.com/openclaw/openclaw

[2] B. Goertzel, "OpenClaw – Amazing Hands for a Brain That Doesn't Yet Exist." [Online]. Available: https://bengoertzel.substack.com/p/openclaw-amazing-hands-for-a-brain

[3] "OpenClaw Memory Architecture." [Online]. Available: https://docs.openclaw.ai/concepts/memory

[4] "OpenClaw Documentation." [Online]. Available: https://docs.openclaw.ai/

[5] Y. Hu, S. Liu, Y. Yue, and G. Zhang, "Memory in the Age of AI Agents: A Survey," *arXiv preprint*, 2025.

[6] C. Packer, S. Wooders, and K. Lin, "MemGPT: Towards LLMs as Operating Systems," *arXiv preprint*, 2023.

[7] T. R. Sumers, S. Yao, K. Narasimhan, and T. L. Griffiths, "Cognitive Architectures for Language Agents," *Transactions on Machine Learning Research (TMLR)*, 2024.

[8] P. Chhikara, D. Khant, S. Aryan, T. Singh, and D. Yadav, "Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory," *arXiv preprint*, 2025.

[9] Z. Pan *et al.*, "On Memory Construction and Retrieval for Personalized Conversational Agents," in *International Conference on Learning Representations (ICLR)*, 2025.

[10] C. Yang, C. Zhou, Y. Xiao, and S. Dong, "Graph-based Agent Memory: Taxonomy, Techniques, and Applications," *arXiv preprint*, 2026.

[11] S. Alqithami, "Forgetful but Faithful: A Cognitive Memory Architecture and Benchmark for Privacy-Aware Generative Agents," *arXiv preprint*, 2025.

[12] Z. Li, S. Shichao, X. Chenyang, and W. Hanyu, "MemOS: A Memory OS for AI System," *arXiv preprint*, 2025.

[13] C. Latimer *et al.*, "Hindsight is 20/20: Building Agent Memory that Retains, Recalls, and Reflects," *arXiv preprint*, 2025.

[14] Q. Zhang, "A Computational Account of Dreaming: Learning and Memory Consolidation," *arXiv preprint*, 2026.

[15] J. Pavlovi\u0107, M. Kr\u00e9sz, and L. Hajdu, "Leveraging Spreading Activation for Improved Document Retrieval in Knowledge-Graph-Based RAG Systems," *arXiv preprint*, 2025.

[16] M. Mozikov, "EAI: Emotional Decision-Making of LLMs in Strategic Games and Ethical Dilemmas," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[17] J. Lu and Y. Li, "Dynamic Affective Memory Management for Personalized LLM Agents," *arXiv preprint*, 2025.

[18] M. Chandra, J. Hernandez, and G. Ramos, "Longitudinal Study on Social and Emotional Use of AI Conversational Agents," *arXiv preprint*, 2025.

[19] Y. Deng, L. Liao, W. Lei, G. H. Yang, W. Lam, and T.-S. Chua, "Proactive Conversational AI: A Comprehensive Survey," *ACM Transactions on Information Systems*, vol. 43, no. 3, 2025.

[20] X. Liu, S. Fang, W. Shi, C.-S. Wu, T. Igarashi, and X. '. Chen, "Proactive Conversational Agents with Inner Thoughts," in *ACM Conference on Human Factors in Computing Systems (CHI)*, 2025.

[21] G. Pasternak, "Beyond Reactivity: Measuring Proactive Problem Solving in LLM Agents," *arXiv preprint*, 2025.

[22] C. Diebel, M. Goutier, M. Adam, and A. Benlian, "When AI-Based Agents Are Proactive: Implications for Competence and System Satisfaction," *Business & Information Systems Engineering (BISE)*, 2025.

[23] W. Sun *et al.*, "Training Proactive and Personalized LLM Agents," *arXiv preprint*, 2025.

[24] C. Hong and Q. He, "Enhancing Memory Retrieval in Generative Agents through LLM-Trained Cross Attention Networks," *Frontiers in Psychology*, vol. 16, p. 1591618, 2025.

[25] Y. Du, "EmoMATE: Multimodal Emotion-Aware AI Assistant with Trust-Adaptive Interaction Modeling," *WSP Publishing*, 2025.

[26] S. Zhang *et al.*, "MemRL: Self-Evolving Agents via Runtime Reinforcement Learning on Episodic Memory," *arXiv preprint*, 2026.

[27] H. Borotschnig, "Emotions in Artificial Intelligence," *arXiv preprint*, 2025.

[28] M. Renze and E. Guven, "Self-Reflection in LLM Agents: Effects on Problem-Solving Performance," *arXiv preprint*, 2024.

[29] N. Shinn, B. Labash, and A. Gopinath, "Reflexion: Language Agents with Verbal Reinforcement Learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[30] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[31] R. Shirkavand, S. Gao, P. Yu, and H. Huang, "Cost-Aware Contrastive Routing for LLMs," *arXiv preprint*, 2025.

[32] "40,000+ OpenClaw Instances Found Exposed on the Internet." [Online]. Available: https://thehackernews.com/2026/02/openclaw-integrates-virustotal-scanning.html

[33] S. Sabour, S. Liu, and Z. Zhang, "EmoBench: Evaluating the Emotional Intelligence of Large Language Models," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.