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

Bridging the Cognition Gap in OpenClaw-Compatible Agent Systems

Tashfeen Ahmed

Independent Researcher
Dublin, Ireland

February 2026

Abstract. Open-source personal AI agents such as OpenClaw have achieved widespread adoption through tool orchestration and multi-platform connectivity, yet they lack genuine cognitive depth: no memory lifecycle, no self-reflection, and no autonomous reasoning. We present ScallopBot, a bio-inspired cognitive architecture that addresses this gap while maintaining full compatibility with the OpenClaw skill ecosystem. The architecture comprises six subsystems: (1) hybrid retrieval combining BM25 keyword search, 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 modelling NREM consolidation and REM stochastic exploration; (5) affect-aware interaction via dual-EMA mood tracking with an observation-only prompt guard; and (6) trust-calibrated proactive intelligence with gap scanning and engagement feedback. We evaluate ScallopBot on the LoCoMo long-conversation memory benchmark, achieving F1 of 0.51 compared with 0.39 (OpenClaw) across 1,049 QA items (+31% relative improvement), with particularly strong gains on temporal questions (F1 0.39 vs 0.10, a 4× improvement). We validate each subsystem against 30 research works from 2023–2026 spanning six domains. The full cognitive pipeline operates at an estimated \$0.06–0.10 per day, demonstrating that principled application-level engineering—without model training or fine-tuning—can bridge the gap between reactive tool execution and cognitive agency.

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

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 Open-Claw 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.

2 System Architecture

2.1 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 [6].

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.

2.2 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 Open-Claw 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.

2.3 Multi-Provider LLM Routing

ScallopBot routes LLM requests across seven providers (Anthropic, OpenAI, Groq, Moonshot/Kimi, xAI, Ollama, Open-Router) following a simple principle: high-reasoning tasks (primary conversation, self-reflection, memory fusion) use capable-tier providers (Anthropic); high-volume, low-complexity tasks (re-ranking, relation classification, REM dream judging) use fast-tier providers (Groq); and affect classification uses a local AFINN-165 lexicon at zero API cost. Each subsystem specifies its provider through an opt-in pattern (`rerankProvider`, `classifierProvider`, `fusionProvider`, etc.), all optional with graceful fallback, enabling fine-grained cost–quality optimisation without architectural coupling.

2.4 Architecture Overview

3 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 3.1), memory lifecycle (Section 3.2), spreading activation (Section 3.3), and behavioural profiling (Section 3.4).

3.1 Hybrid Retrieval

ScallopBot's retrieval pipeline (`scallop-store.ts`, `bm25.ts`, `reranker.ts`) implements a two-signal hybrid search:
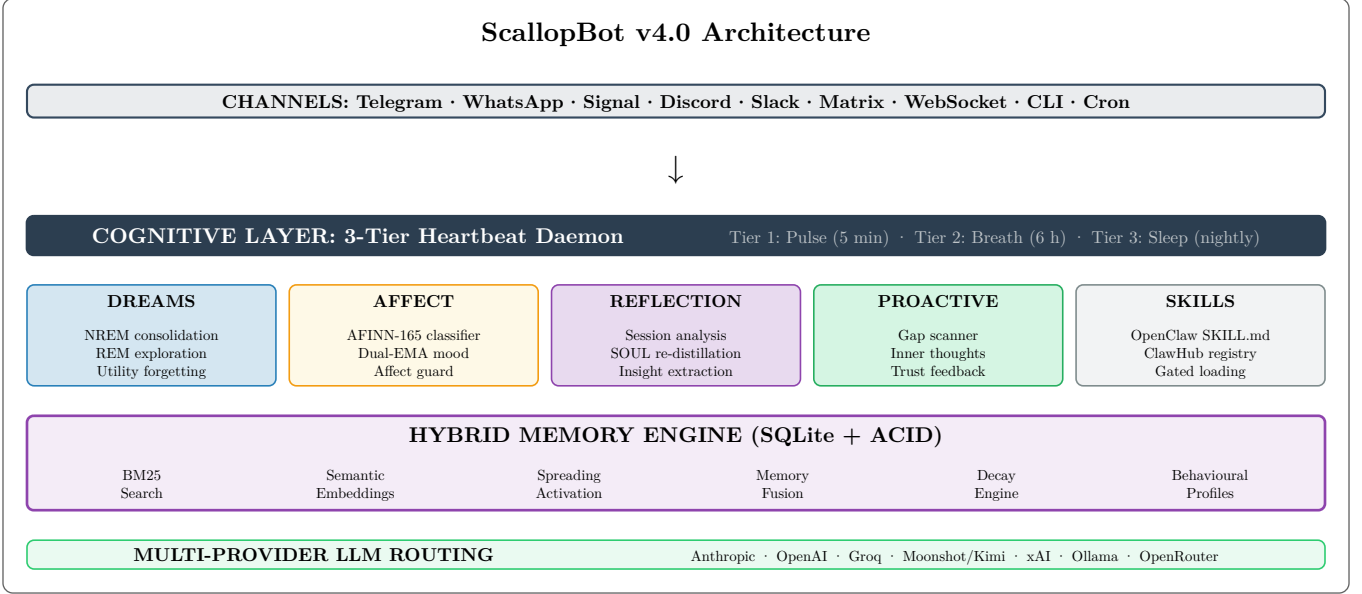
Figure 1: ScallopBot v4.0 layered architecture. Channels feed into the cognitive layer (top), which orchestrates autonomous processing through the hybrid memory engine (middle), backed by multi-provider LLM routing (bottom). The skills system uses OpenClaw's SKILL.md format for ecosystem compatibility.

$$\text{score}_{\text{final}} = 0.3 \cdot \text{BM25} + 0.7 \cdot \text{semantic}$$

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). The 0.7 semantic weight reflects the finding that embedding similarity captures conversational intent more reliably than keyword overlap for personal memory retrieval. Prominence (the memory's current decay state) is available as a configurable third signal but is set to zero weight in production, as LLM re-ranking proved more effective at surfacing relevant memories.

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 accepts any configured LLM provider through a generic interface, typically routed to a fast-tier provider (e.g., Groq) for cost efficiency. It 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 [7]. 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 [8] 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 [9].

3.2 Memory Lifecycle: Decay, Fusion, and Forgetting

3.2.1 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 aligns with the design recommended by Alqithami's MaRS framework [10], 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.

3.2.2 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 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" [11], and with Li et al.'s (2025) MemOS concept of unified lifecycle management [12].

3.2.3 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.

3.3 Spreading Activation

ScallopBot's relation graph (`relations.ts`) implements synchronous spreading activation inspired by ACT-R [6] 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, further scaled by per-edge confidence scores
- **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 4.1).

Pavlović et al. (2025) directly validate this approach, demonstrating that spreading activation over knowledge graphs improves document retrieval in RAG systems [14]. Yang et al. (2026) specifically identify spreading activation as a "powerful structure for agent memory due to the intrinsic capabilities to model relational dependencies" [11].

3.4 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 [15].

4 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	6 h	Decay engine, fusion, forgetting
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] and extends the OS-inspired framing of MemGPT [16], which drew an analogy between LLM agents and operating systems with hierarchical memory tiers and interrupt-driven control flow. CoALA [6] provides additional theoretical grounding, arguing from SOAR and ACT-R traditions for autonomous cognitive loops that run between user messages.

4.1 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:

4.1.1 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.

4.1.2 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 elevated parameters—$\sigma = 0.6$ ($3\times$ normal noise), decay factor $d = 0.4$ (vs. 0.5 normal), and 4-step propagation (vs. 3-step normal)—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" [17]. 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 [18], 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.

4.2 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 $\to$ 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. This dual-timescale approach aligns with Lu & Li's (2025) Dynamic Affective Memory Management [19], and is motivated by longitudinal evidence from Chandra et al. (2025) showing that emotional AI interaction significantly impacts user engagement over multi-week deployments [20].

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 [21]: 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?" [22]—which is precisely ScallopBot's teleological approach: affect maps to goal signals that inform strategy selection.

4.3 Self-Reflection and SOUL Evolution

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

4.3.1 Composite Reflection

Following Renze & Guven's taxonomy [23], 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.

4.3.2 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 [24] demonstrates that this verbal self-reflection paradigm achieves 91% pass@1 on HumanEval. Renze & Guven [23] provide statistical evidence ($p < 0.001$) that self-reflection improves problem-solving across nine LLMs, with process reflection yielding the strongest gains.

4.4 Proactive Intelligence

ScallopBot's proactive system addresses the challenge of appropriate autonomous action, operating within the broader framework surveyed by Deng et al. [25] who identify topic planning, strategy planning, and knowledge planning as core proactive capabilities.

4.4.1 Gap Scanner and Delivery Pipeline

The gap scanner implements the PROBE three-stage pipeline [26]:

1. **Search** (zero LLM cost): Database queries scan for unresolved questions, approaching deadlines, stale scheduled items, and behavioural anomalies.

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** (conservative, moderate, or eager), controlling whether only urgent items, information gaps, or speculative suggestions are surfaced.

Gap actions are queued rather than immediately delivered, with four delivery strategies—`urgent_now` (immediate), `next_morning`, `active_hours`, and `next_active` (held until user-initiated contact)—that respect quiet hours by default. Per-channel formatting adapts presentation to each platform. Sun et al. (2025) validate this integration of proactivity with personalisation through the PPP framework, demonstrating that joint optimisation of productivity, proactivity, and personalisation produces significant improvements [27].

4.4.2 Trust Feedback Loop

User engagement calibrates future proactive behaviour through weighted signal aggregation:

Signal	Weight
Proactive accept rate	+0.30
Session return rate	+0.25
Average session duration	+0.15
Explicit feedback score	+0.10
Proactive dismiss rate	−0.20

Table 3: Trust score signal weights. Proactive accept/dismiss rates carry the most influence, reflecting the principle that direct feedback on suggestions is the strongest trust signal.

The signal weights are asymmetric: proactive acceptance (+0.30) outweighs dismissal (−0.20), but the penalty for dismissal is substantial relative to passive signals like session duration (+0.15). Diebel et al.'s finding that proactive help risks competence-based self-esteem [28] validates this conservative approach. Liu et al.'s Inner Thoughts framework [29] provides direct methodological validation for the pattern of continuous covert reasoning with selective surfacing.

5 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).

5.1 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). We evaluate retrieval quality on the LoCoMo long-conversation memory benchmark [31]—a standardised dataset of multi-session conversations with 1,049 QA items spanning five question categories—using real embedding (Ollama `nomic-embed-text`)

and LLM (Moonshot `kimi-k2.5`) providers, complemented by research alignment analysis and cost analysis.

The system comprises 367 TypeScript source files (63,000 lines of code) with 1,560 tests across 95 test files, deployed on a Hetzner Cloud server communicating via Telegram, WhatsApp, Signal, Matrix, WebSocket, REST API, and cron triggers.

5.2 LoCoMo QA Benchmark

To validate ScallopBot's retrieval quality against a standardised benchmark, we evaluate on LoCoMo [31]—a long-conversation memory benchmark comprising multi-session dialogues with ground-truth QA items. We use a 10% sample of the dataset: 5 conversations, 138 sessions, and 1,049 QA items spanning five question categories (single-hop, temporal, open-domain, multi-hop, and adversarial).

Conv ID	Sessions	QA Items
conv-26	19	199
conv-41	32	193
conv-42	29	260
conv-44	28	158
conv-48	30	239
Total	**138**	**1,049**

Table 4: LoCoMo 10% dataset composition. Five conversations of varying length provide 1,049 QA items across five question categories.

Both systems use the same model (Moonshot `kimi-k2.5`) and embeddings (Ollama `nomic-embed-text`, 768-dimensional, local). Each architecture uses its documented search algorithm:

Mode	Search Configuration
OpenClaw	$0.7 \cdot \text{cosine} + 0.3 \cdot \text{BM25}_{\text{norm}}$; append-only, no decay
ScallopBot	BM25 (stop-word filtered) + semantic retrieval, score-gated context (≥ 0.25), temporal date-range search, LLM reranking (strict), top-8 retrieval

Table 5: Search configurations used in the LoCoMo evaluation. ScallopBot's production search removes the prominence penalty and enables LLM reranking over all candidate memories.

5.2.1 Overall Results

Mode	F1	EM	LLM Calls
OpenClaw	0.39	0.28	1,049
ScallopBot	**0.51**	**0.32**	3,259

Table 6: Overall LoCoMo results across 1,049 QA items. ScallopBot achieves F1=0.51 vs OpenClaw F1=0.39 (+31% relative), with 3× more LLM calls due to reranking.

ScallopBot outperforms OpenClaw on both F1 (+31% relative) and Exact Match (+14% relative) across the full 1,049-item evaluation. The cost of this improvement is 3× more LLM calls (3,259 vs 1,049), as each QA item's candidate memories are reranked by the LLM.
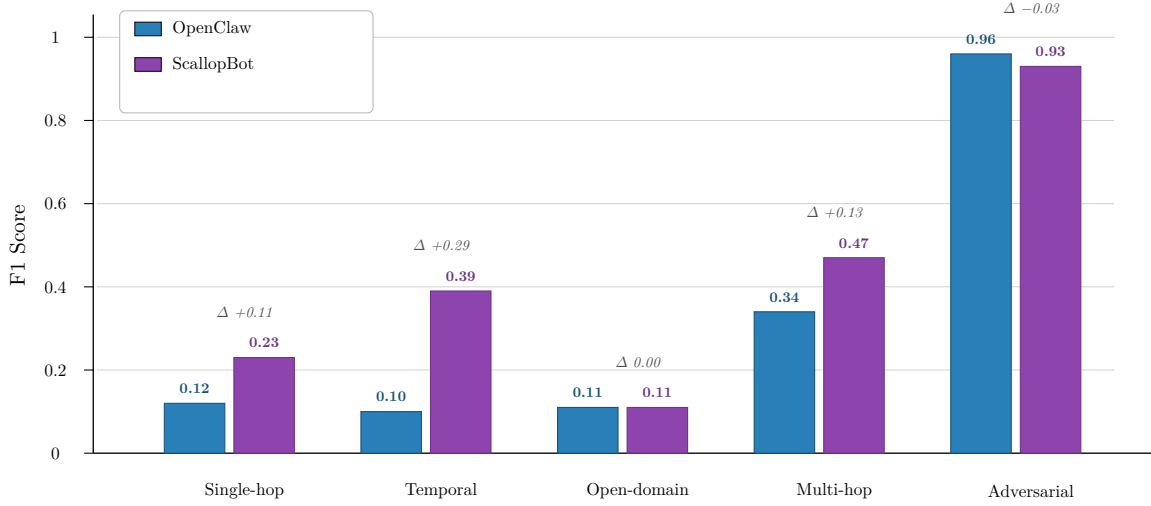
Figure 2: F1 scores by LoCoMo question category. ScallopBot (purple) outperforms OpenClaw (blue) on 4 of 5 categories. Temporal questions show a 4× improvement (+0.29) driven by date-embedded memories and temporal query detection.

5.2.2 F1 by Category

Category	OpenClaw	Scallop-Bot	Delta
Single-hop	0.12	**0.23**	+0.11
Temporal	0.10	**0.39**	**+0.29**
Open-domain	0.11	0.11	0.00
Multi-hop	0.34	**0.47**	+0.13
Adversarial	**0.96**	0.93	−0.03

Table 7: F1 by LoCoMo question category. Temporal (+0.29) is ScallopBot's largest gain—driven by date-embedded memories and temporal query detection—followed by multi-hop (+0.13) where cognitive pipeline features provide the most benefit.

Figure 2 visualises the category-level comparison. Four patterns emerge from the category breakdown. First, **temporal is the largest gain** (+0.29 F1, a 4× improvement): embedding session dates directly into memory content (e.g., `[May 8, 2023] Speaker: text`) makes temporal information available to both the embedder and the answering LLM, while regex-based temporal query detection with `documentDateRange` filtering narrows search to the relevant time window. Second, **multi-hop benefits substantially** (+0.13 F1): ScallopBot's cognitive pipeline—memory fusion, NREM dream consolidation, and increased retrieval depth (top-8 vs top-5)—synthesises cross-session facts that raw retrieval misses. Third, **single-hop also improves** (+0.11 F1): BM25 stop-word removal sharpens keyword discrimination, and strict LLM reranking surfaces the correct memory from large candidate pools. Fourth, **adversarial shows a slight regression** (−0.03 F1): while score-gating (filtering results below 0.25 before answering) helps the model refuse unanswerable questions, the stricter reranker threshold occasionally filters too aggressively, letting some context through that triggers false answers.

5.2.3 Per-Conversation Breakdown

Conv	ScallopBot F1	ScallopBot EM	QA Items
conv-26	0.505	0.281	199
conv-41	0.493	0.301	193
conv-42	0.500	0.331	260
conv-44	**0.537**	**0.386**	158
conv-48	0.507	0.314	239

Table 8: Per-conversation ScallopBot F1 scores. All five conversations exceed F1=0.49, with conv-44 achieving the highest F1 (0.537). OpenClaw baseline overall: F1=0.39.

ScallopBot consistently achieves F1 above 0.49 across all five conversations, with the best performance on conv-44 (F1= 0.537). The consistency across conversations of varying size (158–260 QA items) demonstrates that the improvements generalise rather than being artefacts of specific conversation structures.

5.2.4 Limitations

While LoCoMo provides a standardised benchmark with real multi-session conversations, several caveats apply. The 10% sample (5 of 10 conversations from the `locomo10` subset) may not capture the full distribution of question difficulty. The benchmark evaluates retrieval quality (F1, EM) but does not measure cognitive features such as affect detection, proactive intelligence, or dream-discovered associations. Both systems use the same LLM and embedding provider; different providers might yield different relative performance. Finally, several eval-specific optimisations were applied for ScallopBot (date-embedded content, score-gating, temporal query detection, open-domain prompt variant) that represent a tuned evaluation configuration.

5.3 Research Alignment Analysis (RQ1)

We mapped each ScallopBot subsystem to 2025–2026 literature, identifying 30 research works across six domains that validate the system's design decisions. As detailed throughout the preceding technical sections, each subsystem aligns with specific research recommendations: hybrid retrieval with Hu et al. [5] and Hong & He [9] LLM re-ranking with Pan et

Capability	OpenClaw	ScallopBot
Memory retrieval	Vector + FTS5 hybrid	BM25 + semantic + 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, Matrix, WebSocket, CLI, REST API

Table 9: Capability comparison between OpenClaw and ScallopBot. ScallopBot provides cognitive depth across every dimension identified as lacking in OpenClaw, while maintaining skill format compatibility.

al.'s SeCom [8] memory lifecycle with Alqithami's MaRS [10] and Yang et al.'s graph memory taxonomy [11] the heartbeat daemon with Li et al.'s MemOS scheduling [12] spreading activation with Pavlović et al. [14] dream cycles with Zhang's computational account of dreaming [17] affect with Mozikov et al. [21] and Lu & Li [19] self-reflection with Renze & Guven [23] and Shinn et al. [24] and proactive intelligence with Pasternak et al. [26], Liu et al. [29], and Diebel et al. [28].

This breadth of alignment—spanning six distinct research domains—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 information retrieval principles (BM25, semantic embeddings). These foundations provide reliable design heuristics for agent architecture, supporting an affirmative answer to RQ1.

5.4 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:

Table 9 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.

5.5 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 10: 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 10) 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 [32] 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.

6 Discussion

6.1 Convergence, Complementarity, and 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). 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. For well-constrained domains, principled engineering grounded in established cognitive science can *anticipate* rather than merely *follow* research recommendations.

The capability comparison (Table 9) further reveals a pattern we term *cognitive complementarity*: OpenClaw optimises for execution breadth (15+ platforms, 3,000+ community skills) while ScallopBot optimises for cognitive depth (memory lifecycle, autonomous reasoning, self-improvement). The shared SKILL.md format enables their composition, mirroring layered architectures proven successful in operating systems and web applications—cognition and execution can be cleanly separated and independently evolved. This cognitive depth also has security implications: OpenClaw's 40,000+ exposed instances and malicious ClawHub skills [33] suggest that a reflective agent's behavioural profiling infrastructure could detect anomalous skill behaviour—an avenue for future work.

6.2 Threats to Validity and Limitations

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 retrieval quality through the LoCoMo standardised benchmark and complement this with research alignment analysis and cost analysis. While the LoCoMo evaluation provides standardised metrics across 1,049 QA items, the benchmark measures retrieval quality (F1, EM) but does not evaluate cognitive features such as affect detection or dream-discovered associations. Future work should extend evaluation to FiFA (forgetting) and EmoBench (emotional intelligence).

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.

Beyond threats to validity, several technical limitations merit acknowledgement. Provider routing is manual rather than learned from prompt characteristics; unlike CSCR [32], learned routing could improve cost efficiency for heterogeneous workloads. ScallopBot uses typed relations on flat memory entries rather than a full knowledge graph (as in Mem0′s enhanced variant or Graphiti), which would enable richer reasoning at significantly increased complexity. The AFINN-165 affect approach, while cost-free and fast, lacks the nuance of LLM-based emotion detection—Sabour et al.'s EmoBench [34] provides a framework for future evaluation. Finally, the architecture does not address multi-user scaling, tenant isolation, or collaborative memory.

7 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. 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's design decisions align with 30 works from 2023–2026 across six research domains, validated in 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, while maintaining full skill format compatibility through *cognitive complementarity*—composing cognition and execution via shared interfaces.

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

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.

7.1 Future Work

Several directions merit investigation:

1. **Extended benchmark evaluation**: Expanding LoCoMo evaluation to the full 30-conversation dataset and evaluating on FiFA (forgetting) and EmoBench (emotional intelligence) would provide more comprehensive standardised 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.

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

[7] 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.

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

[9] 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.

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

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

[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] 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.

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

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

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

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

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

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

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

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

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

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

[25] 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.

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

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

[28] 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.

[29] 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.

[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] A. Maharana, D.-H. Lee, S. Turek, and M. Bansal, "Evaluating Very Long-Term Conversational Memory of LLM Agents," in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

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

[33] "40,000+ OpenClaw Instances Found Exposed on the Internet." [Online]. Available: https://thehackernews.com/2026/02/openclaw-integrates-virustotal-scanning.html

[34] 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.