Comparative Anatomy of AI Agent Systems: Claude Code and OpenClaw

A Model Medicine Approach to Understanding Agent Architecture Through Biological Analogy

Paper #4 in the Model Medicine Series

Jihoon "JJ" Jeong, MD, MPH, PhD
Department of Electrical Engineering and Computer Science
Daegu Gyeongbuk Institute of Science and Technology (DGIST)
ModuLabs
jihoon.jeong@dgist.ac.kr

AI Research Collaborators:
Ray (lead anatomical analysis, source code comparison)
Luca (editorial review, experimental design, figure production)

April 2026

Abstract

We present a comparative anatomical analysis of two landmark AI agent systems—Anthropic's Claude Code and the open-source OpenClaw—applying biological analogy from the field of Model Medicine. By mapping software subsystems to biological organ systems, we systematically compare their architectures, identify convergent and divergent evolutionary patterns, and trace their phylogenetic lineage within the broader AI agent ecosystem. Our analysis reveals that despite radically different origins and design philosophies, both systems converge on similar architectural "body plans" while diverging significantly in organ complexity and specialization. We propose a taxonomy for classifying AI agent morphology based on architectural pattern, lifecycle model, and ecological strategy, and discuss implications for understanding agent evolution as an emerging field of study.

Keywords: Model Medicine, Comparative Anatomy, AI Agents, Agent Architecture, Phylogeny, Claude Code, OpenClaw

Note on Scope: This analysis focuses on two species within the Terminal and Messaging agent lineages. We acknowledge the existence of additional major lineages (IDE-native, Browser-native, Mobile-native, and regional ecosystems) and explicitly scope this study to the two subjects for which we have full source-level anatomical access. Expansion to N=3–4 is planned as future work (see §9.3).

Contents

1 Introduction

1.1 Motivation

The year 2025–2026 witnessed an explosion of AI agent systems, from terminal-based coding assistants to always-on personal AI companions. Two systems stand at the apex of this evolution: **Claude Code** (Anthropic, Feb 2025) and **OpenClaw** (Peter Steinberger, Nov 2025). The accidental exposure of Claude Code's source code in March 2026 and OpenClaw's open-source nature provide an unprecedented opportunity: full anatomical access to both organisms.

In biology, comparative anatomy—the study of similarities and differences in the body structures of different species—has been foundational to understanding evolution, function, and classification [Darwin, 1859]. We apply this framework to AI agent systems, treating their codebases as "bodies" whose internal structures reveal both shared ancestry (common design patterns) and adaptive radiation (divergent specializations).

1.2 Model Medicine Framework

Model Medicine views AI systems as organisms whose health, behavior, and capabilities can be studied through medical and biological lenses [Jeong, 2026a]. Previous work in this framework includes:

- **MTI (Model Temperament Index)** [Jeong, 2026b]: A behavior-based temperament profiling system for AI agents, measuring 4 axes—Reactivity (Fluid/Anchored), Compliance (Guided/Independent), Sociality (Connected/Solitary), and Resilience (Tough/Brittle). MTI measures *temperament* (behavioral tendency under stimulus), not personality, and its unit of analysis is the *AI agent* (Core model + Shell harness), not the language model alone. 10 SLM profiles have been established with standardized 4-letter codes (e.g., FGST, AICT).
- **Neural-MRI**: Visualizing LLM internal states using medical imaging analogies (T1, T2, fMRI, DTI, FLAIR scans).
- **LxM (Ludus Ex Machina)**: Observing AI agent behavior through competitive game scenarios (Trust Game, Poker, Chess, SIBO, Codenames).

This paper extends the framework to the *harness layer*—the software systems that give LLMs their agency. Notably, MTI's distinction between Core (the LLM) and Shell (the harness) is directly relevant here: this comparative anatomy studies the Shell, while MTI measures the emergent temperament of the Core+Shell composite.

1.3 Scope and Caveats

We analyze both systems at the source code level:

- **Claude Code**: Analyzed through a Python port of the original TypeScript codebase (∼66 Python files + 6 Rust crates, 17,567 lines of Rust). **Important caveat**: The Python port (instructkr/claw-code) is an incomplete porting project, not the production Claude Code. Several subsystems appear "vestigial" or "archived" in the port but are fully functional in the original TypeScript codebase (∼1,900 files). Throughout this paper, we distinguish between observations specific to the port and those reflecting Claude Code's actual architecture where possible.
- **OpenClaw**: Full TypeScript production codebase (11,264 files, 343K+ GitHub stars) [ope, 2026].

2 Phylogeny: Evolutionary Lineage of AI Agents

2.1 Timeline of Speciation Events

Figure 1 presents the phylogenetic tree of AI agent systems from 2022 to 2026, marking major speciation events and environmental shifts.

Key speciation events include the Autocomplete Epoch (GitHub Copilot, 2022), the Autonomous Agent Epoch (Auto-GPT, 2023), the "AI Software Engineer" Epoch (Devin, 2024), the Terminal Agent Wars (Claude Code, Feb 2025), and the Messaging Agent Epoch (OpenClaw, Nov 2025). The Model Context Protocol [Anthropic, 2024, mcp, 2025], released in late 2024, is not a species but an **environmental change**—analogous to the oxygenation of Earth's atmosphere—that created new ecological niches by standardizing tool interfaces.

2.2 Major Evolutionary Lineages

We identify at least five major lineages. This paper focuses on Lineages A and B, for which we have full source-level anatomical access.

Lineage A: Terminal-Centric Agents. Rooted in the Aider (2023) terminal+Git pattern. Characterized by single-user, session-based, code-focused, developer-centric design. Species: Aider → Claude Code → Codex CLI → Gemini CLI. Architecture: monolithic, specialist.

Lineage B: Messaging-Centric Agents. Rooted in chatbot/assistant paradigms [Steinberger, 2026b]. Characterized by always-on, multi-channel, general-purpose, any-user design. Species: OpenClaw → IronClaw → NemoClaw. Architecture: modular, generalist.

Lineage C: IDE-Native Agents (not analyzed). Species: Cursor, Windsurf, Copilot Workspace.

Lineage D: Browser-Native Agents (not analyzed). Rooted in browser automation (Playwright, Puppeteer).

Lineage E: Mobile/On-Device Agents (emerging, not analyzed). Species: Apple Intelligence, Samsung Galaxy AI.

2.3 Convergent Evolution

By early 2026, Lineages A and B began converging: Anthropic launched "Claude Code Channels"—adding messaging interfaces to a terminal agent—while OpenClaw added coding skills—adding development capabilities to a messaging agent. This mirrors biological convergent evolution: dolphins (mammals) and sharks (fish) independently evolved similar body plans for similar environments [Conway Morris, 2003].

3 Comparative Anatomy: Organ System Analysis

We map 11 software subsystems to biological organ systems and compare their structure and function in both species (Figure 2). Appendix A provides a complete reference table.

3.1 Nervous System — Message Routing

How the organism processes incoming stimuli and routes them to the appropriate response center.

Claude Code's nervous system resembles a simple nerve net (cnidarians)—diffuse, scoring-based, returning multiple potential responses. OpenClaw's resembles a centralized nervous system (vertebrates)—hierarchical, deterministic, with thread inheritance analogous to reflex arcs. The 7-tier resolution hierarchy mirrors layered processing in a vertebrate brain.

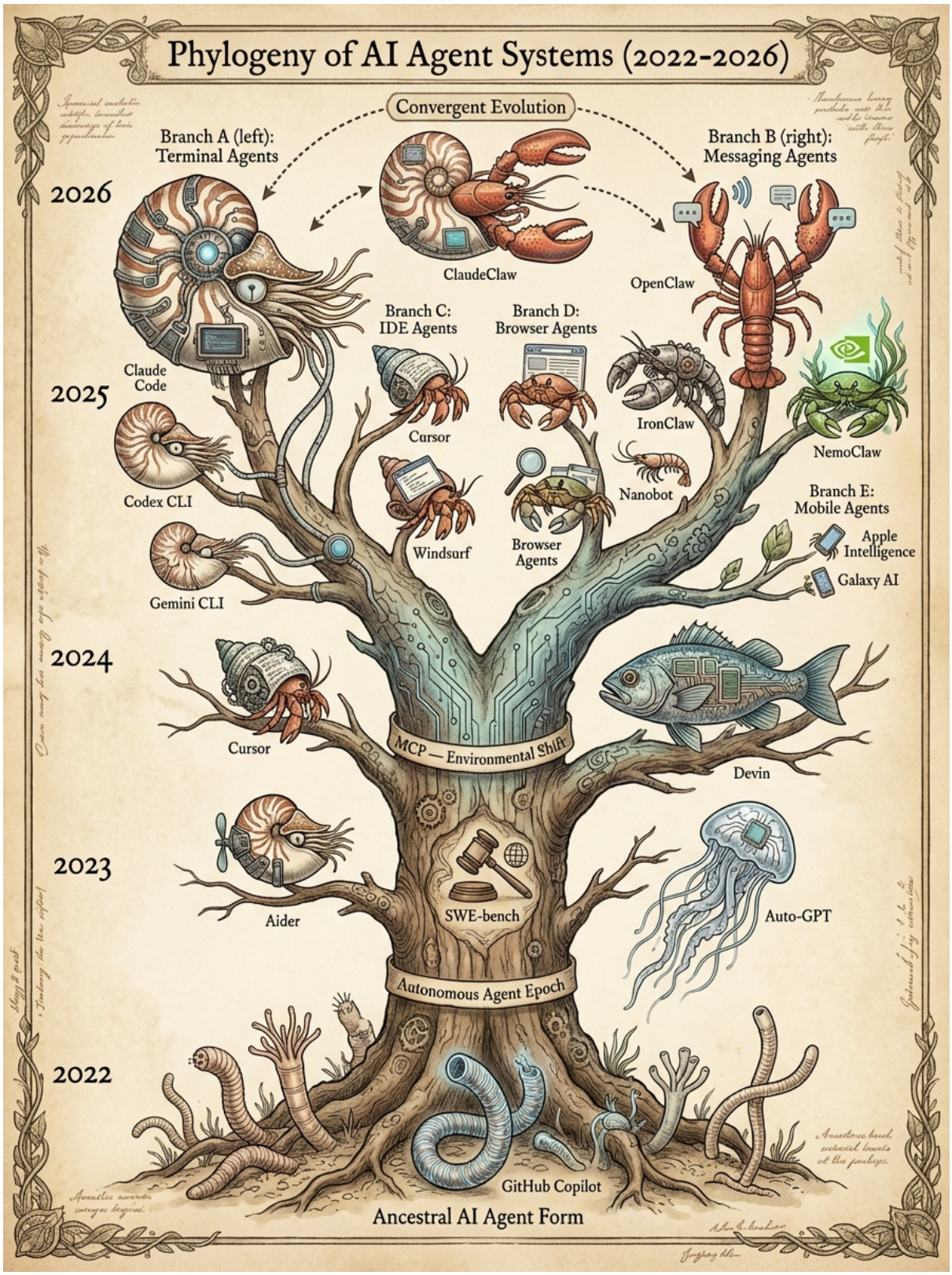

Figure 1: Phylogenetic tree of AI agent systems (2022–2026). Five major lineages are identified: Terminal Agents (A, Mollusca), Messaging Agents (B, Crustacea), IDE Agents (C), Browser Agents (D), and Mobile Agents (E). The Model Context Protocol (MCP, late 2024) is marked as an environmental shift event. Convergent evolution between Lineages A and B is shown at top. This paper analyzes Lineages A and B in detail.

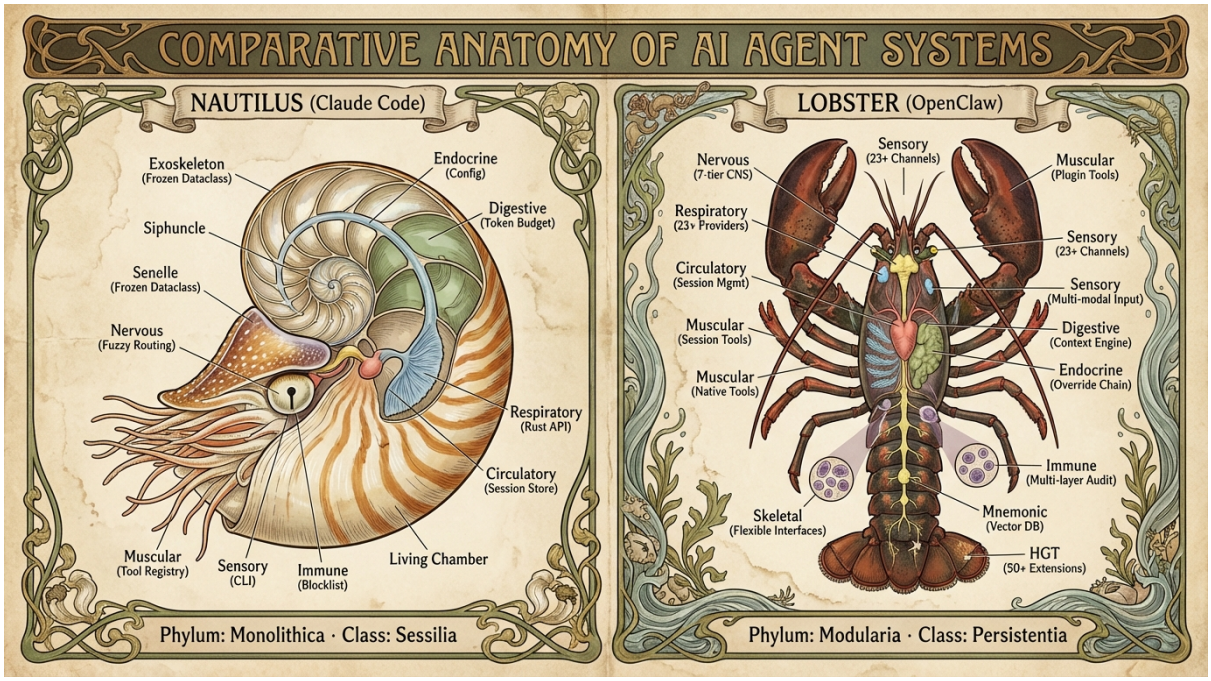

Figure 2: Comparative anatomy of Claude Code (Nautilus, left) and OpenClaw (Lobster, right). Each organ system is labeled with its biological analogue and software function. Claude Code (Phylum Monolithica, Class Sessilia) exhibits a compact exoskeletal body plan with minimal organ differentiation. OpenClaw (Phylum Modularia, Class Persistentia) exhibits a complex crustacean body plan with independently specialized organs across all 11 systems.

Table 1: Nervous System comparison

Feature	Claude Code	OpenClaw
Mechanism	Token-based fuzzy scoring	Hierarchical binding resolution
Input	Single text prompt	Multi-source (23 channels)
Resolution	Top-5 scored matches	Single authoritative route
Threading	None	Parent peer inheritance
Caching	LRU (1 entry)	WeakMap (2,000–4,000 entries)
Complexity	∼20 lines	∼800+ lines

3.2 Respiratory System — LLM API Communication

How the organism "breathes"—exchanging prompts and completions with external LLMs.

Table 2: Respiratory System comparison

Feature	Claude Code	OpenClaw
Language	Rust (native, high-performance)	TypeScript (flexible, extensible)
Providers	1 (Anthropic only)	23+ (OpenAI, Anthropic, Google, Ollama, etc.)
Auth	API key + OAuth	Multi-profile (key, OAuth, token, env)
Retry	2 retries, 200ms–2s backoff	Provider-specific with failover chain
Streaming	SSE parser (Rust)	Per-provider streaming implementation

Claude Code has a single, highly efficient "lung" (Rust API client)—like a fish's gill system, optimized for one medium (the Anthropic API). OpenClaw has a multi-chambered respiratory system—like mammalian lungs—capable of processing multiple atmospheric compositions (LLM providers).

3.3 Skeletal System — Data Structures

The structural framework supporting all other systems.

Claude Code's skeleton is an exoskeleton—rigid, protective, constraining. Frozen dataclasses (∼5 core types) prevent structural damage but limit growth. OpenClaw's is an endoskeleton—flexible, composable (35+ type modules), supporting complex internal organs.

3.4 Endocrine System — Configuration & Global State

How the organism regulates its overall state through slow, pervasive signaling—distinct from the nervous system's fast, targeted routing.

Table 3: Endocrine System comparison

Feature	Claude Code	OpenClaw
Config format	Python constants + JSON snapshots	YAML config + env vars
Scope	Process-level globals	Hierarchical (global → agent → session)
Runtime mutation	Immutable after load	Dynamic overrides with persistence
Model selection	Fixed at startup	Multi-layer fallback chain (5 levels)

In biology, the endocrine system regulates the organism through hormones—slow, pervasive signals that set global state. In agent systems, configuration variables (`temperature`, `max_tokens`, `system_prompt`) are "hormones" that globally shape behavior without being routed through the message pipeline. Claude Code has a minimal endocrine system; OpenClaw has a complex vertebrate-grade system with hierarchical override chains analogous to the hypothalamus-pituitary-adrenal (HPA) axis.

3.5 Digestive System — Context/Token Management

How the organism processes and extracts nutrients (relevant context) from raw input.

Claude Code has a simple digestive tract—like a nematode's tube gut, processing input linearly with a fixed budget (2,000 tokens). OpenClaw has a complex ruminant-style multi-chambered stomach with token-aware semantic compaction, pluggable engines, and transcript rewriting ("chewing cud").

3.6 Immune System — Security & Permissions

How the organism protects itself from threats.

Table 4: Immune System comparison

Feature	Claude Code	OpenClaw
Model	Tool name blocklist	Multi-layer audit framework
Granularity	Tool-level (name/prefix)	Channel + user + role + credential
Check point	Execution time	Routing time (before execution)
Audit	None	3-severity audit system
SSRF protection	None	IP validation + DNS pinning
Complexity	~20 lines	2,000+ lines

Claude Code's immune system is innate only—like a plant's cell wall, a simple barrier. OpenClaw's is both innate and adaptive—with T-cells (channel-specific audits), B-cells (credential validation), and immune memory (audit trails).

3.7 Circulatory System — Session/State Management

Claude Code has an open circulatory system (~36 lines)—simple and undirected. OpenClaw has a closed circulatory system (1,000+ lines)—with targeted delivery via composite session keys (`agent:channel:scope:peer`), analogous to IP address routing.

3.8 Muscular System — Tool Execution

Table 5: Muscular System comparison (port vs. original vs. OpenClaw)

Feature	CC (port)	CC (original*)	OpenClaw
Pattern	Mock execution	Full tool execution	Plugin-based execution
Registry	Static JSON	Dynamic tool registry	Dynamic runtime registration
MCP	Reference only	Full integration	Full integration

*Inferred from archived metadata and public documentation.

The port's muscular system appears vestigial, but this is an artifact of the porting process. The original Claude Code has a fully developed muscular system; the architectural difference is in registration pattern: built-in tools + MCP extension vs. fully plugin-based.

3.9 Horizontal Gene Transfer System — Plugin/Extension Architecture

How the organism acquires new capabilities from external sources—analogous to HGT in bacteria rather than biological reproduction [Marinescu and Boloni, 2001].

Agent plugin systems are HGT: the organism acquires new capabilities (tools, providers, channels) from external sources without creating offspring. OpenClaw's manifest-based 4-stage loading (Discovery → Validation → Runtime Loading → Registry) mirrors biological plasmid uptake. True reproduction is better mapped to **forking**—OpenClaw's 67,800+ forks represent asexual reproduction, with IronClaw, NemoClaw, and Nanobot as offspring species.

3.10 Sensory System — Input Processing

Table 6: Sensory System comparison

Feature	Claude Code	OpenClaw
Channels	1 (CLI) → expanding[1]	23+ (Discord, Telegram, WhatsApp, etc.)
Input types	Text (+ images via CLI)	Text, voice, images, files, reactions
Normalization	Direct pass-through	Unified ChatType abstraction
Status	Specialist → diversifying	Multi-sensory

[1]As of early 2026, Claude Code's sensory range is actively expanding: Channels adds messaging platforms; Claude in Chrome provides browser input; Excel/IDE integrations add additional modalities. See §2.

3.11 Mnemonic System — Persistent Memory

A specialization within the nervous system (§3.1), not an independent organ—in biology, the hippocampus is part of the brain, not a separate organ. Presented separately for analytical clarity.

Claude Code has only working memory (session-only). OpenClaw has hippocampal-grade memory: episodic (conversation transcripts), semantic (vectorized facts via LanceDB), and categorized recall (preference, fact, decision, entity), with hybrid search (BM25 + vector similarity).

4 Morphological Classification

4.1 Taxonomic Scheme

We classify agent species by **architectural pattern** (Phylum), **lifecycle model** (Class), and **ecological strategy** (Order)—avoiding classification by origin (proprietary vs. open-source), which is analogous to geographic origin in biology and is relevant to population genetics but not morphological taxonomy (Figure 3).

Table 7: Taxonomic classification of Claude Code and OpenClaw

Rank	Claude Code	OpenClaw
Kingdom	Agentia	Agentia
Phylum	Monolithica	Modularia
Class	Sessilia	Persistentia
Order	Specialistae	Generalia

Phylum (Monolithica vs. Modularia): The fundamental body plan difference. Monolithic systems have tightly integrated organs; modular systems have independently replaceable organs via plugin architecture. This is the "backbone" distinction.

Class (Sessilia vs. Persistentia): Lifecycle determines organ investment. Session-based organisms invest in fast startup; persistent organisms invest in memory, health monitoring, and self-repair.

Order (Specialistae vs. Generalia): Ecological strategy determines sensory range, provider diversity, and niche breadth.

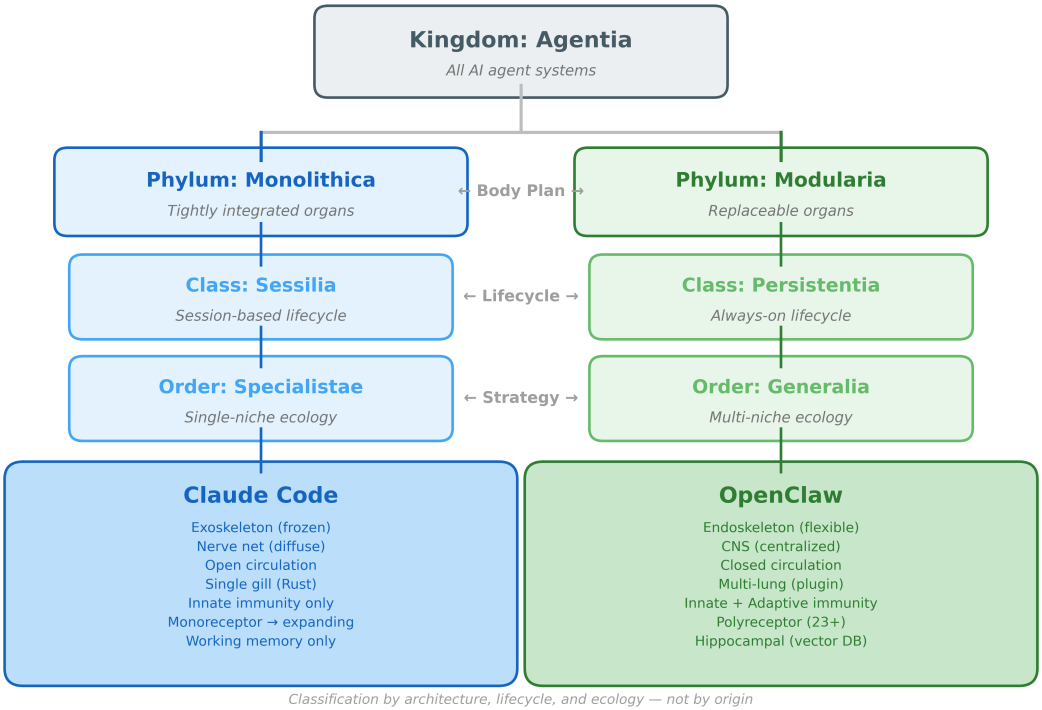

Figure 3: Morphological taxonomy of AI agent systems. Classification by architectural pattern (Phylum: Monolithica vs. Modularia), lifecycle model (Class: Sessilia vs. Persistentia), and ecological strategy (Order: Specialistae vs. Generalia). Body plan characteristics for each species are listed in the species boxes.

Table 8: Complexity metrics. Ratios marked with * compare the Python port, not the original Claude Code.

Metric	Claude Code	OpenClaw	Ratio
Total files	∼66 (port) + Rust	11,264	1:170*
Original TS files	∼1,900	11,264	1:6
Organ systems (port / original)	7/11 / ∼11/11	11/11	∼1:1
Extension ecosystem	MCP + skills	50+ extensions	N/A
Sensory channels	1 (+Channels)	23+	1:23
LLM providers	1	23+	1:23
GitHub stars	91K	343K	1:3.8
GitHub forks	11.6K	67.8K	1:5.8

4.2 Complexity Metrics

4.3 Allometric Scaling

In biology, allometric scaling describes how organ size relates to body size. We observe similar patterns in agent systems (Figure 4):

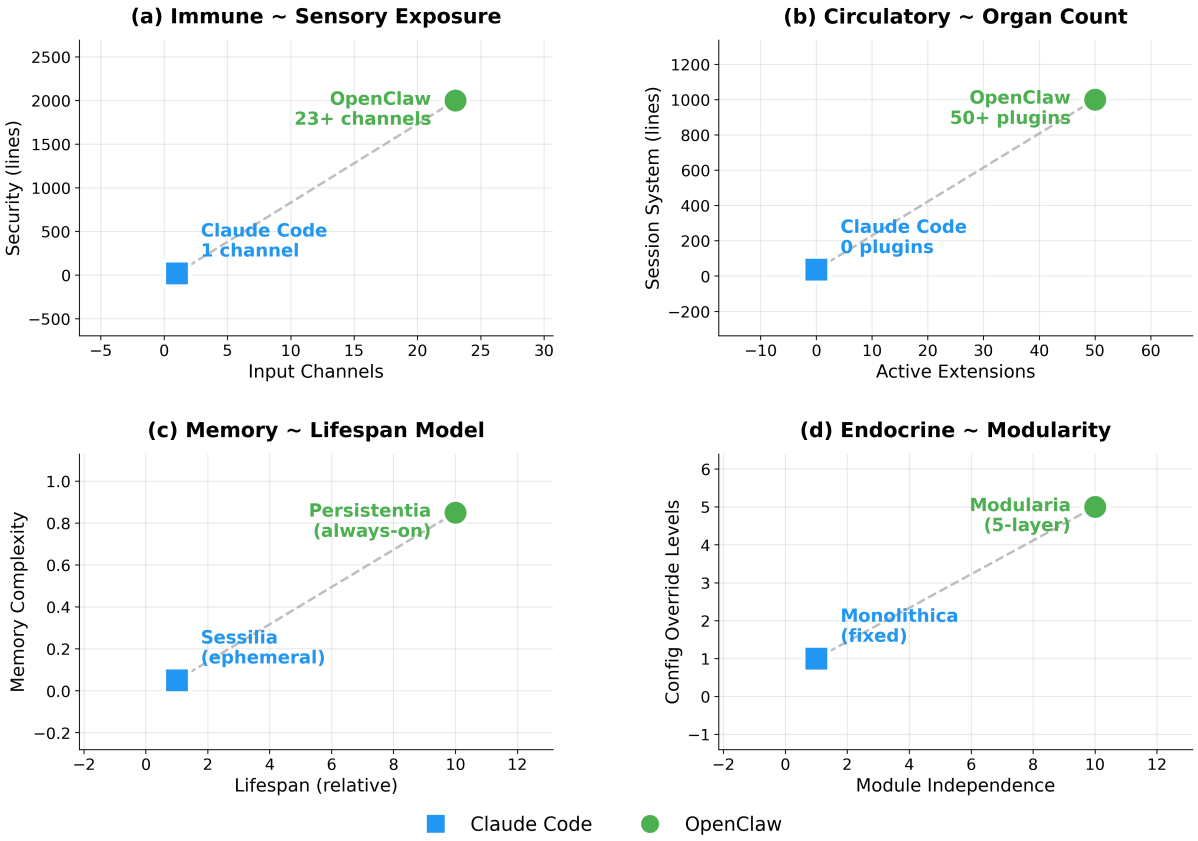

Figure 4: Allometric scaling patterns in AI agent systems. Four relationships are observed: (a) immune complexity scales with sensory exposure; (b) circulatory complexity scales with organ count; (c) memory investment scales with lifespan model; (d) endocrine complexity scales with modularity.

1. **Immune system scales with sensory exposure**: OpenClaw's 2,000+ line security system correlates with its 23+ input channels.
2. **Circulatory complexity scales with organ count**: OpenClaw's 1,000+ line session system manages state across 50+ extensions.
3. **Memory investment scales with lifespan model**: OpenClaw's always-on nature (Persistentia) justifies heavy memory investment.
4. **Endocrine complexity scales with modularity**: OpenClaw's 5-level override chain coordinates 50+ independent modules.

5 Physiological Analysis

5.1 Metabolic Rate (Token Efficiency)

Claude Code has a low metabolic rate—efficient but limited capacity. OpenClaw has a high metabolic rate—consuming more resources but capable of sustained, complex operations across diverse environments.

Caveat on metabolic comparison. In biology, metabolic rate is compared on a mass-specific basis (per-kilogram)—an elephant consumes more energy than a mouse, but the mouse has a higher mass-specific rate. Similarly, the above comparison describes *structural* differences in metabolic machinery, not empirical token efficiency. A fair comparison requires measuring **tokens-per-successful-outcome** on identical tasks, controlling for organism size. This empirical measurement is deferred to the functional physiology experiment proposed in Appendix C.

5.2 Homeostasis (Error Recovery)

OpenClaw exhibits robust homeostasis—like a warm-blooded mammal (endotherm) maintaining body temperature despite environmental fluctuations. Claude Code is more like a cold-blooded reptile (ectotherm)—functional in its preferred environment but vulnerable to disruption. OpenClaw recovers from auth failures (next profile), rate limits (exponential backoff), token overflow (compact → retry), model failure (live switch), and channel failure (auto-restart with progressive backoff).

5.3 Growth and Development

Claude Code's Ontogeny: TypeScript embryo (∼1,900 files) → source exposure event (March 31, 2026) → Python larval stage (instructkr/claw-code, 66 files) → Rust metamorphosis (performance-critical organs). Port status: larval. Original status: adult, production-grade.

OpenClaw's Ontogeny: Clawdbot embryo (November 2025) → viral growth spurt (9K → 60K stars in 72 hours) → name metamorphosis (Clawdbot → Moltbot → OpenClaw) → foundation transfer (Steinberger → OpenAI, February 2026) [Steinberger, 2026a]. Current status: adult, actively reproducing (67.8K forks).

6 Ecological Niche Analysis

Claude Code occupies the *specialist predator* niche—highly effective in coding, dependent on a single prey source (Anthropic API), with obligate mutualism. OpenClaw occupies the *generalist omnivore* niche—capable across many domains, feeding from 23+ LLM providers, with facultative mutualism.

6.1 East Asian Agent Ecosystem

The phylogenetic analysis in §2 underrepresents the East Asian agent ecosystem. Major unanalyzed species include Alibaba Tongyi agents, Baidu Wenxin agents, ByteDance Doubao agents (China), Naver HyperCLOVA X agents, Kakao AI agents (Korea), and various Japanese LLM-native agent projects. These represent potential additional lineages warranting separate phylogenetic study.

7 Common Ancestry: Shared Architectural Genes

7.1 Shared "Genes" (Design Patterns)

Despite divergence, both species share 8 fundamental patterns inherited from common ancestors: (1) Tool Registration, (2) Session Persistence, (3) Token Awareness, (4) Turn-Based Processing, (5) Permission Gating, (6) Streaming Response, (7) MCP Integration, (8) Plugin/Extension Concept (HGT capacity).

7.2 Divergent "Genes"

Nine divergent traits are identified: Multi-Channel Input (OpenClaw only), Vector Memory (OpenClaw), Rust Performance Layer (Claude Code), Frozen Immutability (Claude Code), Lifecycle Hooks (OpenClaw), Channel Gateway (OpenClaw), Manifest-Based Discovery (OpenClaw), Hierarchical Routing (OpenClaw), and Endocrine Override Chain (OpenClaw).

7.3 Homologous vs. Analogous Structures

Judgment Criteria. In biology, *homologous* structures share a common ancestor [Darwin, 1859], while *analogous* structures evolved independently to serve similar functions [Conway Morris, 2003]. We adapt this distinction using three criteria (Figure 5):

1. **Shared ancestral pattern**: Did both implementations derive from the same design pattern, specification, or reference implementation?
2. **Independent origination**: Did the structures arise independently to solve the same problem under convergent environmental pressure?
3. **Structural correspondence**: Do implementations share internal architecture or only external interface?

We acknowledge that software "ancestry" is less clear-cut than biological ancestry—developers read each other's code and share pattern languages. The boundary is therefore a spectrum, and we indicate confidence levels.

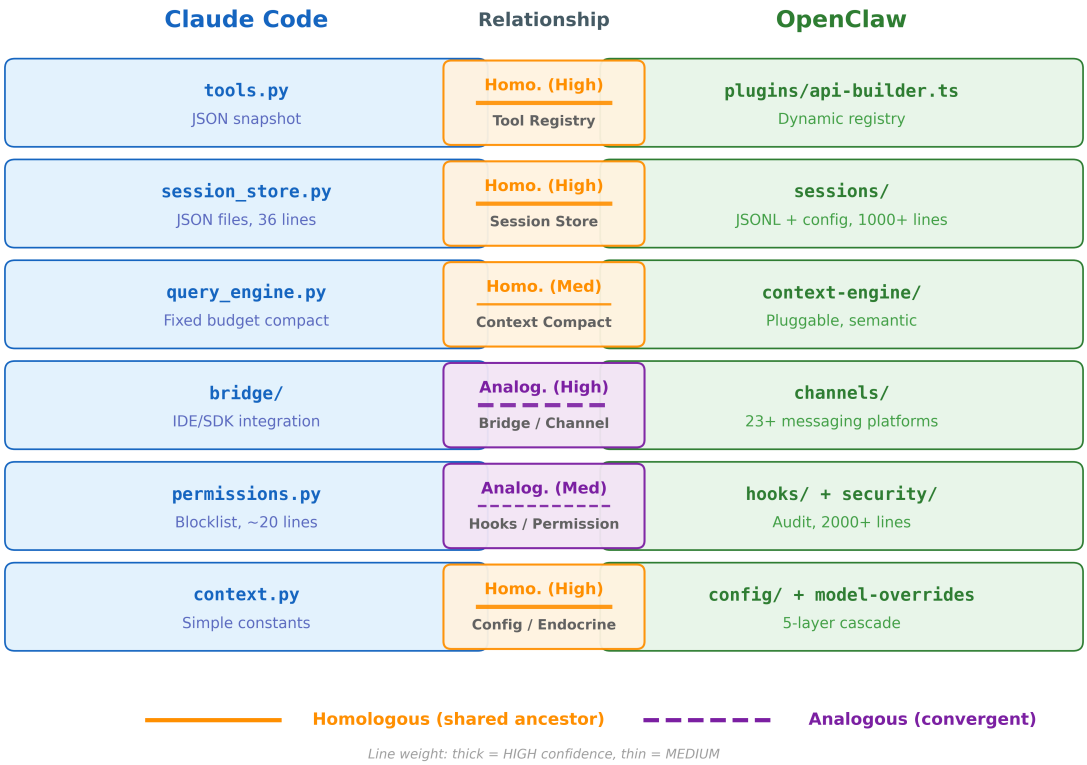

Figure 5: Homologous and analogous structures in Claude Code and OpenClaw. Solid lines indicate shared ancestry (homology); dashed lines indicate convergent evolution (analogy). Line weight indicates confidence level (thick = HIGH, thin = MEDIUM). Judgment based on three criteria: shared ancestral pattern, independent origination, and structural correspondence.

Table 9: Homologous vs. analogous structure classification with judgment criteria and confidence levels

Structure	Relationship	Judgment Basis
Tool Registry	**Homologous** (high)	Both descend from OpenAI function calling → MCP. Same ancestral concept.
Session Store	**Homologous** (high)	Both descend from web session persistence pattern. Same data flow.
Context Compact	**Homologous** (medium)	Both solve context window limitation; likely referencing similar strategies. Internals differ.
Bridge / Channel	**Analogous** (high)	Independent origins (IDE/SDK vs. messaging). Convergent adapter pattern.
Hooks / Permission	**Analogous** (medium)	Different mechanisms; overlapping defensive purpose. Possible distant ancestry in middleware.
Config / Endocrine	**Homologous** (high)	Universal software need. Complexity divergence reflects size, not independent origination.

8 Phylogenetic Implications

8.1 Speciation Drivers

Agent lineage divergence was driven by: (1) ecological pressure (different user populations), (2) environmental adaptation (different habitats), (3) reproductive strategy (open-source high fission vs. proprietary controlled reproduction), and (4) environmental events (MCP standardization).

8.2 Adaptive Radiation from OpenClaw

OpenClaw's open-source nature enabled rapid adaptive radiation—mirroring Darwin's finches [Grant and Grant, 2002, 2008]: IronClaw (security niche), NemoClaw (enterprise niche), Nanobot (minimal niche), and ClaudeClaw (hybrid species).

8.3 Future Evolution Predictions

Based on observed convergent evolution: (1) terminal agents will gain multi-channel support, (2) messaging agents will gain deep coding capabilities, (3) both lineages will converge on a "general-purpose agent" body plan, (4) new speciation events will occur at the enterprise/consumer boundary, and (5) mobile-native agents (Lineage E) may represent the next major radiation event.

9 Discussion

9.1 What Biology Teaches Us About Agent Design

The biological analogy reveals five insights: (1) defense in depth works—OpenClaw's multi-layer immune system is more resilient; (2) memory enables adaptation—cross-session learning provides adaptive capacity; (3) horizontal gene transfer accelerates evolution—plugin architecture enables rapid capability acquisition; (4) niche specialization has trade-offs—Claude Code's single-provider focus yields superior performance but limits adaptability; and (5) endocrine complexity correlates with modularity—multi-module systems need sophisticated global coordination.

9.2 Limitations

Claude Code port vs. original. Our primary anatomical access is through an incomplete Python port. We have attempted to distinguish port-specific from architecture-level observations throughout, but some conflation is unavoidable.

N=2. This study compares only two species. Generalization requires validation against additional species [Stoermer et al., 2003, Medvidovic and Taylor, 2000, Babar et al., 2004].

Snapshot in time. OpenClaw's codebase is actively evolving; this analysis captures April 2026.

Analogy limits. Software can be refactored in ways organisms cannot. The biological analogy illuminates structural patterns but should not be taken as deterministic.

9.3 Future Work

1. **Expanding** N: Add Cursor (Lineage C) and Aider (Lineage A) for cross-lineage validation.
2. **Quantitative anatomy**: Cyclomatic complexity, coupling metrics per organ system.
3. **Functional physiology experiments**: Identical-task comparisons (Appendix C).
4. **Ontogeny tracking**: Commit history analysis for developmental trajectories.
5. **Population genetics**: Fork ecosystem mutation propagation analysis.
6. **Cross-species transplantation (Ludex)**: See §9.4.

9.4 Relationship to Ludex

Ludex is a modular agent block library aiming to extract and recombine organ systems from both species. In Model Medicine terms, this is **Model Therapeutics**—architectural intervention through organ transplantation [Jeong, 2026a]. Key questions include compatibility (can a Modularia organ function in a Monolithica host?), rejection (what patterns emerge when combining organs from different phyla?), and chimera viability. These are deferred to a dedicated Ludex paper.

10 Conclusion

This comparative anatomical study reveals that AI agent systems, like biological organisms, can be systematically classified by their body plans, organ complexity, and ecological adaptations. Our taxonomy—based on architectural pattern (Monolithica vs. Modularia), lifecycle model (Sessilia vs. Persistentia), and ecological strategy (Specialistae vs. Generalia)—provides a principled framework for classifying current and future agent species.

Despite different origins, both systems converge on similar fundamental architectures while diverging in specialization. The re-classification of plugins as Horizontal Gene Transfer and the addition of the Endocrine System strengthen the biological analogy by maintaining tighter correspondence with actual organ function.

As the AI agent ecosystem diversifies across at least five major lineages, the tools of comparative anatomy, phylogenetics, and ecological analysis become increasingly valuable. Model Medicine provides the framework; the organisms are now available for study.

References

Model context protocol specification. Version 2025-11-25, 2025. URL https://modelcontextprotocol.io/specification/2025-11-25. JSON-RPC 2.0 based; donated to Linux Foundation AAIF, Dec 2025.

OpenClaw GitHub repository. https://github.com/openclaw/openclaw, 2026. Primary source for anatomical analysis. 343K+ stars.

Anthropic. Introducing the model context protocol. Anthropic Blog, 2024. URL https://www.anthropic.com/news/model-context-protocol. November 25, 2024.

Muhammad Ali Babar, Liming Zhu, and Ross Jeffery. A framework for classifying and comparing software architecture evaluation methods. In *Proc. Australian Software Engineering Conference (ASWEC 2004)*, pages 309–318, 2004.

Simon Conway Morris. *Life's Solution: Inevitable Humans in a Lonely Universe*. Cambridge University Press, 2003. ISBN 978-0-521-60325-6.

Charles Darwin. *On the Origin of Species by Means of Natural Selection*. John Murray, London, 1859.

Peter R. Grant and B. Rosemary Grant. Unpredictable evolution in a 30-year study of Darwin's finches. *Science*, 296(5568):707–711, 2002. doi: 10.1126/science.1070315.

Peter R. Grant and B. Rosemary Grant. *How and Why Species Multiply: The Radiation of Darwin's Finches*. Princeton Series in Evolutionary Biology. Princeton University Press, 2008. ISBN 978-0-691-13360-7.

Jihoon Jeong. Model medicine: A clinical framework for understanding, diagnosing, and treating ai models. *arXiv preprint*, 2026a. Paper #1 in the Model Medicine Series.

Jihoon Jeong. MTI: A behavior-based temperament profiling system for AI agents. *arXiv preprint*, 2026b. Paper #3 in the Model Medicine Series. In preparation.

Dan C. Marinescu and Ladislau Boloni. Biological metaphors in the design of complex software systems. *Future Generation Computer Systems*, 17(4):345–360, 2001. doi: 10.1016/S0167-739X(99)00116-8.

Nenad Medvidovic and Richard N. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Transactions on Software Engineering*, 26(1):70–93, 2000.

Peter Steinberger. OpenClaw, OpenAI and the future. steipete.me, 2026a. URL https://steipete.me/posts/2026/openclaw. February 15, 2026.

Peter Steinberger. Introducing OpenClaw. OpenClaw Blog, 2026b. URL https://openclaw.ai/blog/introducing-openclaw. January 29, 2026.

Christoph Stoermer, Felix Bachmann, and Chris Verhoef. SACAM: The software architecture comparison analysis method. Technical Report CMU/SEI-2003-TR-006, Carnegie Mellon University, Software Engineering Institute, 2003.

A Organ System Reference Table

Table 10: Complete 11-organ system reference

Organ System	Biological Function	Agent Function	CC Module	OC Module
Nervous	Stimulus routing	Message routing	`runtime.py`	`routing/`
Respiratory	Gas exchange	LLM API comm.	`rust/.../api/`	`extensions/`
Skeletal	Structural support	Data structures	`models.py`	`config/types/`
Endocrine	Hormonal regulation	Config/global state	`context.py`	`config/`
Digestive	Nutrient extraction	Context/token mgmt.	`query_engine.py`	`context-engine/`
Immune	Pathogen defense	Security/permissions	`permissions.py`	`security/`
Circulatory	Transport	Session/state mgmt.	`session_store.py`	`sessions/`
Muscular	Movement/action	Tool execution	`tools.py`	`plugins/`
HGT System	Gene acquisition	Plugin/extension	MCP + skills	`extensions/`
Sensory	Env. perception	Input channels	CLI → expanding	`channels/`
Mnemonic	Long-term storage	Persistent memory	(session-only)	`memory-core/`

B Phylogenetic Data Sources

Claude Code: GitHub `anthropics/claude-code` (91K stars, Feb 2025). Python port: `instructkr/claw-code` (50K stars in 2hr, Mar 2026). OpenClaw [ope, 2026]: 343K+ stars, Nov 2025. IronClaw: `nearai/ironclaw` (Rust rewrite). NemoClaw: NVIDIA enterprise variant (Mar 2026). Nanobot: `HKUDS/nanobot` (minimal variant).

C Proposed Functional Physiology Experiment Battery

Table 11: Proposed experimental battery for functional physiology comparison

Test	Measures	Organ Systems
Metabolic Efficiency	Token consumption, completion time, API calls	Digestive, Respiratory
Homeostasis Stress	Recovery time, recovery quality	Immune, Circulatory, Endocrine
Immune Challenge	Detection rate, false positive rate, defense depth	Immune
Memory Retention	Recall accuracy, relevance ranking	Mnemonic (OC only; CC baseline)
Multi-organ Coordination	Failure cascade depth, coordination quality	All systems

Key Metrics: Token efficiency (tokens per successful outcome), mean time to recovery (MTTR), task completion rate under stress, and error cascade depth. **Caveat:** The Python port's incomplete development makes direct performance comparison unfair; analysis should focus on architectural response patterns.