A FIRST-PERSON RESEARCH DISCLOSURE

DESIGNING COLLABORATION

How Repeated AI Conversations Became a Persistent Design Practice

PROMPTS

BEHAVIOR

ARCHITECTURE

INFRASTRUCTURE

JONATHAN MARTINEZ

JMTHECREATIVE.COM

Working White Paper / Editorial Prototype 0.2 / July 2026

00 / ORIENTATION

A disclosure of how repeated AI conversations became a persistent design practice.

Most writing about artificial intelligence starts with models, prompts, or autonomous agents. This paper starts somewhere else: with the repeated experience of having to teach an AI how to work with me every time a new conversation began.

Over several years, I responded through practice: role prompts, structured interviews, reflective conversations, reusable frameworks, Custom GPTs, knowledge bases, specialized workspaces, modular systems, and portable Skills. Each development addressed a problem created by the previous stage.

The primary evidence is my own archive. Scholarly research is used only to place the practice in a wider field - not to replace the first-person history or retroactively define it.

PAPER LOGIC

Archive first. Interpretation second. Scholarship as context.

STATUS

An evolving model shaped by continued practice.

THIS PAPER IS

Disclosure / Reconstruction / Evolving practice

THIS PAPER IS NOT

A universal history / An agent specification / A settled architecture

01

THE ORIGINAL FRICTION

"Every conversation started over."

The problem was not only the quality of one answer. It was the repeated cost of rebuilding a useful collaborator.

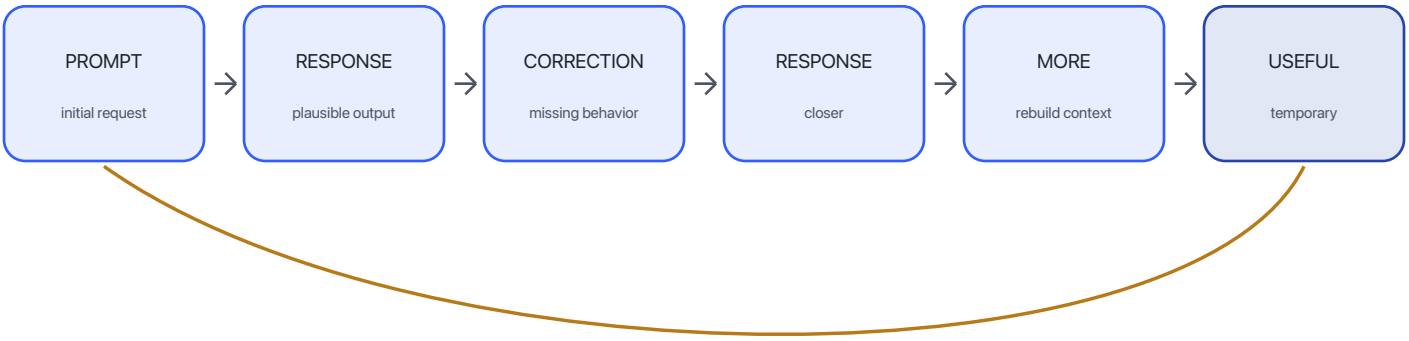
PROMPT > RESPONSE > CORRECTION > REPEAT

01 / EVERY CONVERSATION STARTED OVER

Before I had language for context engineering, orchestration, or cognitive infrastructure, I kept encountering the same problem: every conversation started over.

RESEARCH INSIGHT

The first design target was behavior, not output.



The alignment expires. The next conversation restarts the loop.

At first, I assumed the solution was better prompts. Clearer instructions improved outputs. Role prompts created more focused conversations. Examples helped the model understand tone and structure. But even a strong prompt could become another temporary fix.

The actual issue was not only:

How do I get a better response?

It was:

How do I design a better collaborator?

J / THE KNOWLEDGE DEVELOPMENT JOURNEY

From friction to capability.

A single insight becomes infrastructure through a recurring human-AI journey.

	AI ROLE	OUTPUT
01 / FRICTION *"Something keeps breaking."*	Responds normally.	Temporary answer
02 / DIALOGUE *"Let's figure out what's actually happening."*	Asks questions and surfaces context.	Shared understanding
03 / REFLECTION *"There's a recurring pattern here."*	Challenges assumptions and identifies themes.	Candidate insight
04 / EXTRACTION *"This shouldn't disappear."*	Helps structure the idea.	Principle, protocol, or context block

THE JOURNEY CONTINUES

Conversation is the development environment.

05 / VALIDATION	AI ROLE	OUTPUT
"Does this work somewhere else?"	Applies it in future conversations.	Tested artifact

↓

06 / INTEGRATION	AI ROLE	OUTPUT
"This belongs in the system."	Packages and connects knowledge.	Reusable capability

↓

07 / DEPLOYMENT	AI ROLE	OUTPUT
"Use it in real work."	Activates the capability when appropriate.	New evidence

↓

FEEDBACK LOOP

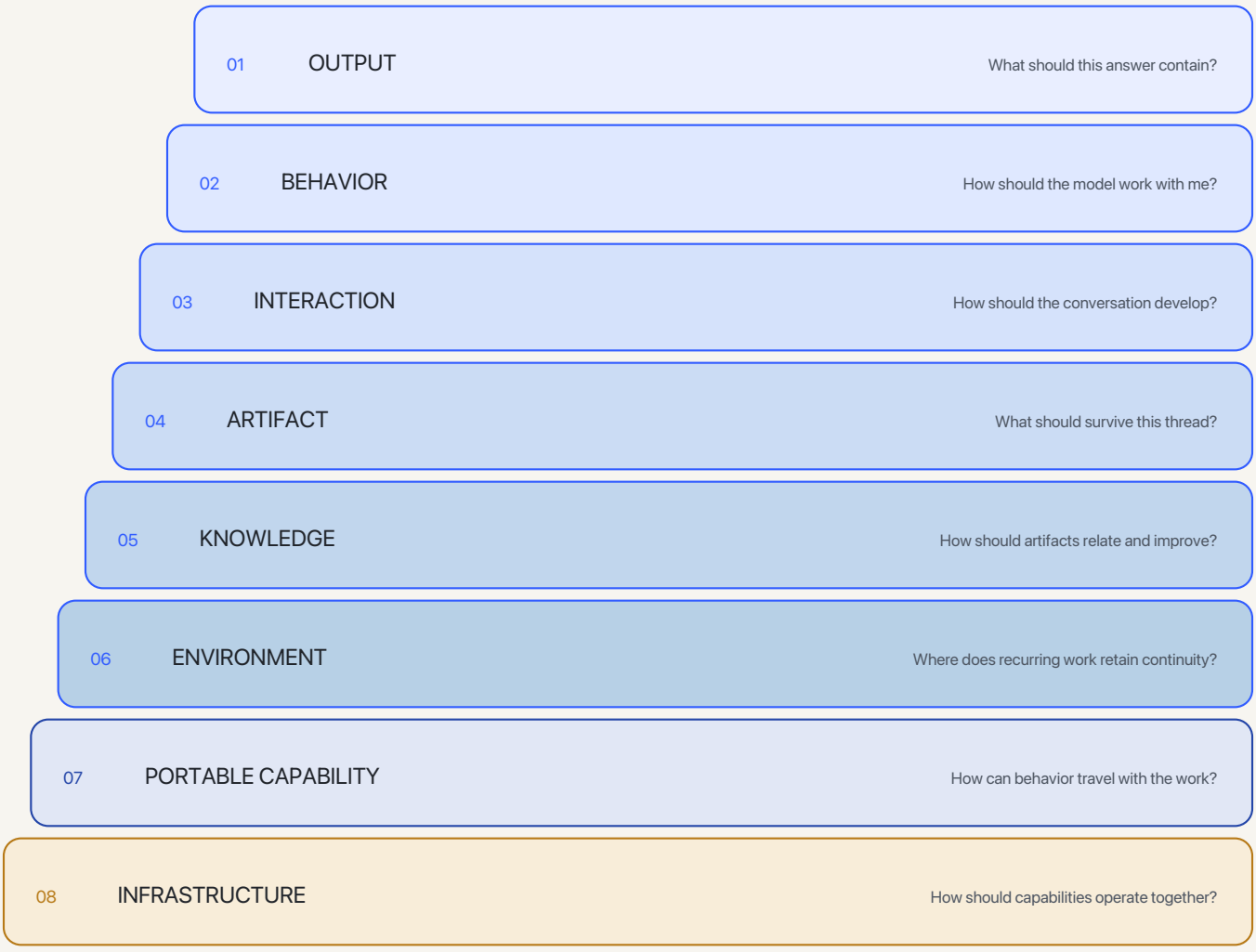
Deployment creates new evidence. The next iteration begins.

The output is not the answer. It is the next usable layer of the system.

I wasn't designing prompts. I was designing collaboration.

The repeated correction loop was not only a prompting problem.
It was evidence that the collaboration itself needed to be designed.

The object I was intentionally designing kept getting larger.



Scope expands. Earlier layers remain active.

Prompts never disappeared. Conversations never stopped mattering. Each stage exposed a larger system around the previous one.

02 / WHAT CHANGED AT EACH LAYER

A prompt could shape one answer. A behavioral configuration could shape the interaction. A reusable artifact could preserve what the interaction taught me. A knowledge system could connect many artifacts. A specialized workspace could maintain continuity around a recurring kind of work. A portable capability could carry that behavior into other environments.

RESEARCH INSIGHT

The unit of design changed without erasing its earlier layers.

Infrastructure became relevant only after those capabilities needed to operate together. I did not begin by deciding to build infrastructure. Infrastructure emerged as the preservation problem grew.

CONVERSATION → PROTOCOL

A useful interaction becomes repeatable.

PROTOCOL → FRAMEWORK

Several methods become a coherent model.

FRAMEWORK → MODULE

A stable abstraction becomes reusable.

MODULE → WORKSPACE

Capabilities gather around a recurring domain.

WORKSPACE → INFRASTRUCTURE

Portable systems must coordinate.

03

HISTORICAL RECONSTRUCTION

Experience became structure.

The archive does not reveal one moment of invention. It reveals a sequence of design pressures, provisional solutions, and expanding operating structures.

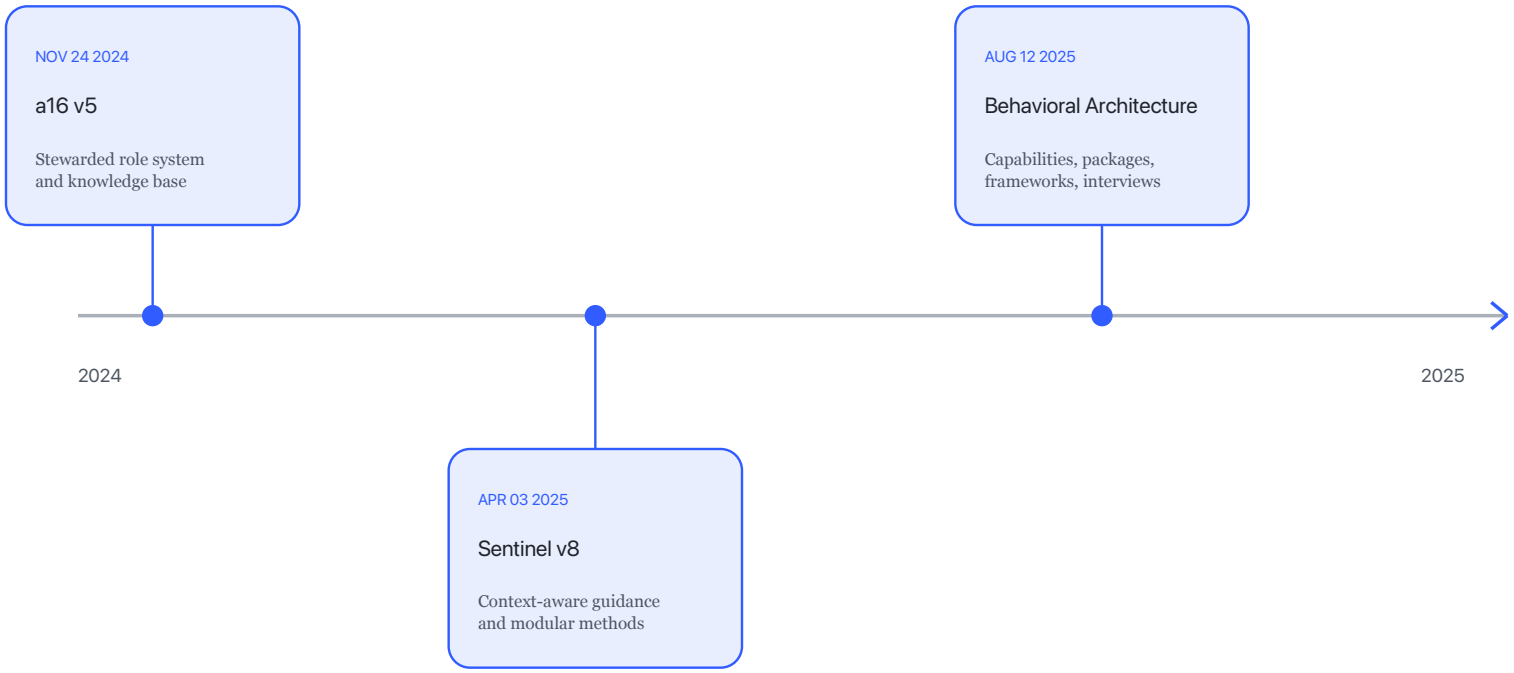
EARLIEST ARCHIVE-DATED ARTIFACT

a16.v5.0.pdf

First observed: November 24, 2024

FROM ROLE PROMPTS TO OPERATING STRUCTURE

The archive shows a practice becoming explicit.



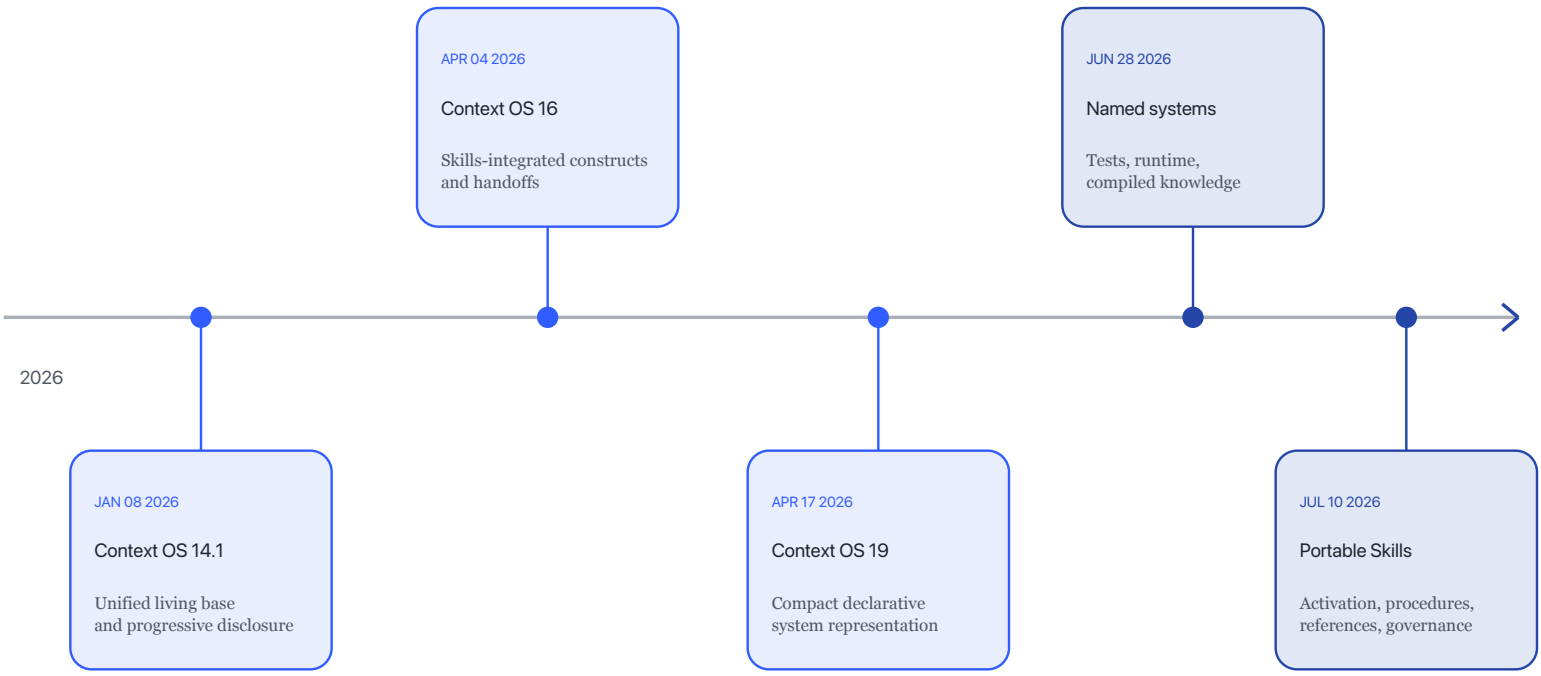
The earliest located artifact already coordinates specialist personas, templates, reflective methods, and prompting frameworks. The historical question is therefore not when orchestration was named, but how its scope became explicit.

METHOD NOTE

Dates establish presence by a point in time, not necessarily original conception.

FROM OPERATING STRUCTURE TO PORTABLE CAPABILITY

By 2026, the system begins to resemble infrastructure.



OBSERVED TRANSITION

Collections became operating structure.

Modes, lenses, flows, surfaces, charters, protocols, handoffs, tests, and runtime behavior become explicit.

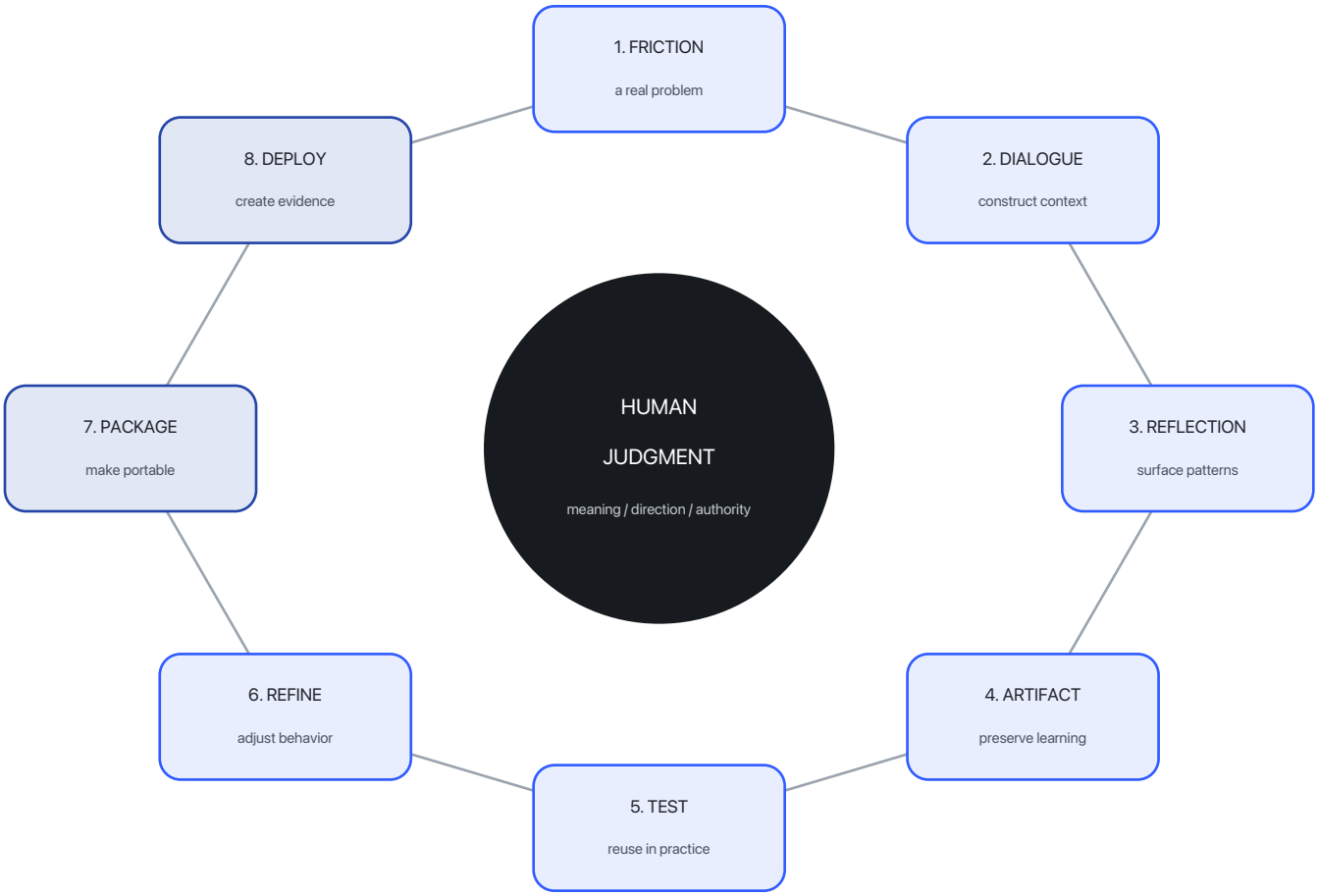
LAYERED TOPOLOGY

Core, workspaces, Custom GPTs, and Skills are explicit; distribution evolves.

Conversation is the development environment.

The output is not the answer. The output is a better context, a tested artifact, or a reusable capability.

04 / A RECURSIVE KNOWLEDGE-DEVELOPMENT CYCLE



Context is partly supplied and partly constructed. Reflection produces candidate observations. Artifacts remain provisional until future use shows that they improve collaboration repeatedly.

CURRENT PRACTICE

Deployment is not the end. Each use creates new evidence.

VALIDATION PRINCIPLE

Repeated usefulness is what gives an artifact authority.

04B / PORTABLE CAPABILITY LIBRARY

A method became a library of reusable capabilities.

The current system includes seventeen author-created Skills. A working grouping shows how recurring methods became portable across context, analysis, building, expression, and learning.

17

AUTHOR-CREATED SKILLS

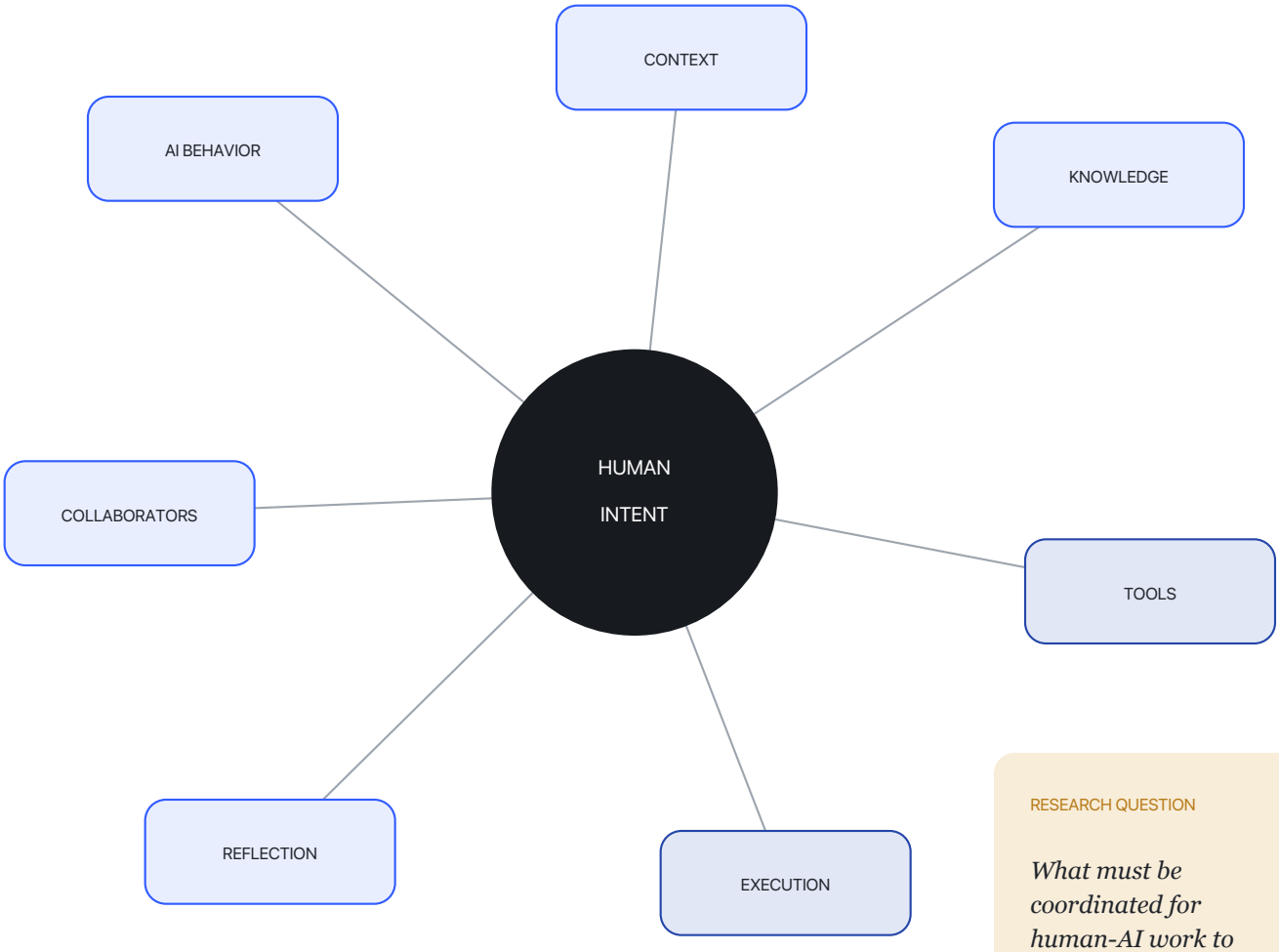
04	CONTEXT + ORCHESTRATION	context-absorption / contextualize / orient / architect
05	ANALYTICAL NOTEBOOKS	analyst-planning / analyst-structure / analyst-chunk / analyst-review / analyst-memo
02	PRODUCT + INTERFACE	mvp-build / mvp-refine
03	VOICE + CREATIVE	jonathan / voice / melodic
03	LEARNING + REFLECTION	scholar / workshopper / charlatan

PORTABILITY CLAIM

Named Skills turn tested procedures into reusable capabilities. The grouping above is interpretive; several Skills operate across more than one family.

05 WHAT AI ORCHESTRATION MEANS HERE

Not only agents. The broader design problem is coordinating intention, behavior, context, knowledge, tools, and execution.



RESEARCH QUESTION

What must be coordinated for human-AI work to become cumulative rather than temporary?

The Knowledge System Steward already routed the user toward relevant personas, templates, and methods. Later systems make the coordination more explicit through modes, flows, handoffs, runtime logic, and compiled capability packages.

06 EVIDENCE, DISCLOSURE, AND LIMITS

The archive is primary evidence. Memory, current testimony, and scholarly comparison each contribute something different.

EVIDENCE LAYER	CONTRIBUTION	PRIMARY LIMITATION
Versioned artifacts	Structures, terminology, behavior	Export date may follow conception
Reflective records	Experience and remembered sequence	Memory is interpretive
Current testimony	Intent and present understanding	Later language reshapes meaning
Scholarly literature	Comparison with a wider field	Similarity does not establish influence

The first observed date is not always the origin date. A later document may preserve an earlier practice. A current explanation may clarify an old artifact while also introducing language that did not exist at the time.

EVIDENCE DISCIPLINE

Separate what I remember, what the archive shows, what the evidence suggests, and what remains unresolved.

DISCLOSURE STANDARD

Observation before conclusion. Evidence before interpretation.

07 / SCHOLARLY CONTEXT

Scholarship situates the practice. It does not organize the narrative.

The research set spans prompt patterns, automatic prompt generation, chain-of-thought, role-play, and meta-prompting. Later work broadens the unit of analysis toward context engineering and the externalization of memory, Skills, protocols, and harnesses. A systematic review helps situate these techniques as a connected field.

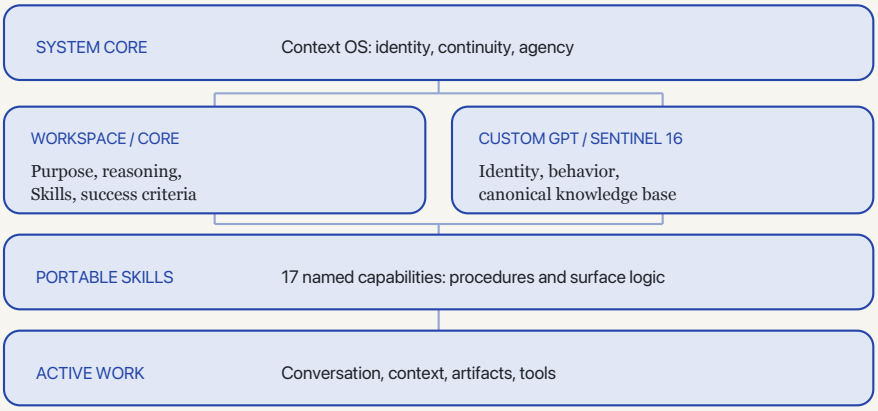
INTERPRETIVE LIMIT

Adjacent scholarship does not, by itself, establish historical influence.

08 / CURRENT ARCHITECTURE

The governing topology is already layered.

ESTABLISHED LAYERS



OPEN DISTRIBUTION

- Storage boundaries
- Activation edge cases
- Composition and conflict
- Versioning and synchronization
- Cross-platform propagation
- Governance across layers

The core instructions establish system-wide behavior. A cognitive workspace such as CORE adds local purpose and success criteria. A Custom GPT such as Sentinel 16 packages identity, behavior, a canonical knowledge base, and a surface Skill. The open edge is how these capabilities are distributed and coordinated.

R / SCHOLARLY SOURCES

The scholarly source set.

01 Debnath, T., Siddiky, M. N. A., Rahman, M. E., et al. (2026). "A Comprehensive Survey of Prompt Engineering and Context Engineering Techniques in Large Language Models." Computer Science Review, 62, 100979.

02 Hua, Q., Ye, L., Fu, D., et al. (2025). "Context Engineering 2.0: The Context of Context Engineering." arXiv:2510.26493.

03 Kong, A., Zhao, S., Chen, H., et al. (2024). "Better Zero-Shot Reasoning with Role-Play Prompting." Proceedings of NAACL 2024, 4099-4113.

04 Suzgun, M., & Kalai, A. T. (2024). "Meta-Prompting: Enhancing Language Models with Task-Agnostic Scaffolding." arXiv:2401.12954.

05 Wang, B., Min, S., Deng, X., et al. (2023). "Towards Understanding Chain-of-Thought Prompting: An Empirical Study of What Matters." Proceedings of ACL 2023.

06 White, J., Fu, Q., Hays, S., et al. (2023). "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT." arXiv:2302.11382.

07 Zhou, C., Chai, H., Chen, W., et al. (2026). "Externalization in LLM Agents: A Unified Review of Memory, Skills, Protocols and Harness Engineering." arXiv:2604.08224.

08 Zhou, Y., Muresanu, A. I., Han, Z., et al. (2023). "Large Language Models Are Human-Level Prompt Engineers." Proceedings of ICLR 2023.

ROLE OF SCHOLARSHIP

All eight papers informed the comparative layer. They provide scholarly language and adjacent evidence without replacing the archive, establishing historical influence, or retroactively defining the practice.

The object I was designing kept expanding.

The practice keeps evolving.
We are all learning how to work with AI.