The Human Model: A Compositional Architecture for Human-Emulating AI Agents.

Rex Raphael, Member, *IEEE*

***Abstract*— Abstract—AI agent frameworks treat personality and reasoning style as prompt engineering problems. The agent is a system prompt, a set of tools, and a loop. This paper argues that is a dead end for production systems and presents The Human Model, a compositional architecture that decomposes agents into seven structured primitives drawn from cognitive science and personality psychology: Skills, Traits, Behaviors, Cognitive Styles, Communication Styles, Perception, and Personas. We describe Cortex, an open-source Go implementation with multi-tenancy, execution tracking, and human-in-the-loop checkpoints. A survey of cognitive architectures (SOAR, ACT-R, BDI), personality modeling research, and contemporary LLM frameworks shows that no existing system covers all seven dimensions. The Human Model fills that gap by applying validated psychological constructs as engineering primitives rather than prompt decorations.**

***Impact Statement*— The Human Model introduces a practical shift in AI agent design by moving beyond monolithic prompt engineering toward a compositional architecture grounded in cognitive science and personality psychology. By decomposing agents into seven structured primitives—Skills, Traits, Behaviors, Cognitive Styles, Communication Styles, Perception, and Personas—the framework enables more controllable, auditable, and reusable agent behavior in production systems. This approach improves consistency across tasks, supports safer human-in-the-loop oversight, and makes agent capabilities easier to test, compare, and evolve over time. Implemented in Cortex, the architecture also provides operational features such as multi-tenancy and execution tracking, bridging research concepts with real-world deployment needs. The broader impact is a clearer engineering path for building human-aligned AI agents that are reliable, interpretable, and adaptable across domains, from enterprise automation to collaborative decision support.**

***Index Terms*— AI agents, cognitive architecture, personality modeling, agent orchestration, composable primitives, human-computer interaction.**

I. Introduction

THERE is a mismatch in how we build AI agents. The infrastructure has gotten serious. State machines, tool orchestration, memory systems, multi-agent coordination. But the model of the agent itself? Still a system prompt.

Write "You are a helpful assistant who is analytical, detail-oriented, and speaks in a professional tone." Attach some tools. Run it in a loop. That is the state of the art in 2025. LangChain [1] does this. AutoGen [2] does this. CrewAI [3] does this with slightly more structure (a role and a backstory field, both free text). The agent's personality, reasoning approach, communication style, and behavioral dispositions of the agent are all jammed into one blob of natural language and hope.

I ran into the limits of this approach, building production agents for enterprise digital twin systems. We had agents interfacing with engineers at a customer who expected precision, and those same agent configurations getting deployed to contexts where the communication needed to be completely different. Rewiring the personality meant rewriting the prompt, and rewriting the prompt meant retesting everything because you could never change just one thing. Personality dimensions, reasoning depth, communication register, and domain behavior: all tangled together in a paragraph. Pull one thread, and three other things change.

The idea behind what I call The Human Model is not complicated. Psychology has spent decades building validated, quantitative models of how people function. The Big Five personality model [4]. Kahneman's dual-process theory [5]. Gibson's ecological perception [7]. The BDI model of beliefs, desires, and intentions [6]. These frameworks decompose human cognition and behavior into structured, measurable dimensions. They are not speculative philosophy. They have been tested on millions of subjects. And the agent development community has mostly ignored them, preferring to write personality as prose.

Cortex (https://github.com/xraph/cortex) is my implementation. It breaks an agent into seven primitives. Skills for what you can do. Traits for who you are. Behaviors for what you habitually do. Cognitive Styles for how you think. Communication Styles for how you talk. Perception for what you pay attention to. And Personas, which compose the other six into a coherent identity you can swap depending on context.

That last part is important. Goffman wrote in 1959 about how people present different versions of themselves in different social situations [30]. Your agent should be able to do the same thing, and it should not require copy-pasting a prompt and editing sentences.

The rest of this paper covers why these seven dimensions (Section 3), how they relate to prior work (Section 2), and how Cortex implements them (Section 4). Section 5 is where I get into what this buys you over prompts and what is still missing.

This work was submitted for review on February 20, 2026, and not supported by any organizations.

R. I. Raphael works with Kongsberg Digital, Inc, Houston, TX 10777 USA (e-mail: rex@xraph.com).

II. RELATED WORK

A. Classical Cognitive Architectures

SOAR [8] is the granddaddy. Newell's lab at Carnegie Mellon built it in the 1980s as an implementation of his Unified Theory of Cognition [9]. It models thinking as search through a problem space using production rules. When the system reaches an impasse (cannot select an operator), it creates a subgoal and reasons about the impasse. Learning happens through chunking: when a subgoal resolves, SOAR compiles the solution into a new production rule so it doesn't have to reason through it again.

SOAR is impressive engineering. It is also, for our purposes, incomplete. Two SOAR agents loaded with the same production rules are behaviorally identical. There is no way to make one cautious and another bold without writing different rules. SOAR models the mechanics of cognition but not the character of the cognizer.

ACT-R [10] took a different path. Where SOAR is about problem-space search, ACT-R is about memory and learning. It decomposes cognition into modules: declarative memory (facts), procedural memory (skills), and a set of perceptual-motor buffers. The interesting part for personality modeling is the sub symbolic layer. Memory retrieval is governed by activation levels that decay with time and strengthen with use. Production rule selection uses expected utility calculations. So ACT-R agents do develop something like individual character over time as their activation patterns diverge. But it is an emergent property, not a design parameter. You cannot say "make this agent more risk-averse" without understanding and tweaking the sub symbolic math.

LIDA [11] implemented Global Workspace Theory [12], which models consciousness as a broadcast mechanism. Specialized modules compete for access to a shared workspace, and whichever coalition of modules wins gets its content broadcast to the whole system. The relevant contribution here is explicit attentional filtering. Not everything makes it to the workspace. The Human Model's Perception primitive serves a similar function but as a configuration parameter rather than an emergent competition.

Then there is BDI. Rao and Georgeff [6] formalized the intuition that agents have beliefs about the world, desires about outcomes, and intentions that commit them to action. Bratman's philosophical work [13] on intention and practical reason provided the foundation. Platforms like JACK [14] and Jason [15] turned it into runnable code. BDI is elegant and has been enormously influential in multi-agent systems research, but it says nothing about personality or communication. Two BDI agents with identical beliefs and desires will behave identically.

None of these architectures model personality as a first-class concept. None of them model communication style. They are theories of cognition, not theories of persons.

B. Personality Modeling

The Big Five [4, 16] dominates personality psychology. Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism. Costa and McCrae's NEO-PI-R instrument [26] measures them. The dimensions are continuous (not categories), normally distributed in the population, and remarkably stable over the adult lifespan.

Can these dimensions actually change how an agent communicates? Yes. Mairesse and Walker [17] showed that language generation systems parameterized on Big Five dimensions produce text that human raters reliably identify as matching the target personality. This is not a theoretical result. People can tell the difference between text generated with high Extraversion settings and text generated with low Extraversion settings, even when they don't know the system is parameterized that way.

Nass and Lee [18] found a "similarity-attraction" effect for synthetic agents. People prefer interacting with agents whose personality matches their own. The practical implication: if personality is a structured parameter you can configure per-user, you gain something prompt engineering makes very hard to do systematically.

Park et al. [19] published the most relevant contemporary work in 2023. Their "Generative Agents" paper created Smallville, a simulated town where 25 LLM-powered agents maintained memories, formed relationships, planned activities, and exhibited emergent social dynamics. Agents coordinated Valentine's Day parties without being told to. They spread information through gossip chains. The work demonstrated that LLM-based agents can produce remarkably human-like social behavior.

But here is my issue with it. Every agent's identity was a name and a paragraph of natural-language description. There was no structure to personality. You cannot query an agent's extraversion level. You cannot take one agent's communication tendency and transplant it onto another. You cannot A/B test whether making an agent more conscientious improves task completion. The Human Model takes what Park et al. proved was possible (rich agent behavior from personality specifications) and makes it engineering-grade: structured, composable, queryable, testable.

PERSONA-CHAT [20] and Zhang et al. [21] worked on a related problem from the NLP side, conditioning dialogue systems on personality profiles to maintain consistency across turns. Their approach was data-driven (fine-tuning on persona-conditioned dialogues). The Human Model addresses consistency through architecture instead of training.

C. Where Modern Frameworks Stand

LangChain [1] is plumbing. Chains, memory, retrieval, tool calling. It has no opinion on agent identity. That is both its strength (flexibility) and its limitation (every developer reinvents personality from scratch).

AutoGen [2] introduced multi-agent conversation as a first-class pattern. Agents take turns in structured conversations. It is a contribution to orchestration. The agents themselves are still prompt-and-tools.

CrewAI [3] tried to add structure. Agents have role, goal, and backstory fields. This is better than nothing, but all three are

free-text strings. You still cannot compose personality from reusable components or inspect which dimension caused a particular behavior.

Semantic Kernel [22] organizes around skills (callable functions with descriptions). Skills are one of seven dimensions in The Human Model. The other six are absent.

ReAct [23] formalized the reasoning loop most agents use: think, act, observe, repeat. Good pattern. But it is a single pattern. Some tasks need brainstorm-then-evaluate. Some need observe-hypothesize-test. Some need no explicit reasoning at all. The Human Model's Cognitive Style primitive makes the reasoning approach itself configurable.

Voyager [24] introduced skill accumulation in Minecraft. Agents discover new capabilities and store them for reuse. The proficiency dimension in The Human Model's Skill primitive reflects a similar insight applied to production rather than game environments.

TABLE I
SHOWS COVERAGE ACROSS ALL SEVEN DIMENSIONS. THE COLUMN ON THE RIGHT IS THE POINT.

	SOAR	**ACT-R**	**BDI**	**Park et al.**	**CrewAI**	**AutoGen**	**LangChain**	**Human Model**
Skill structure	Rules	Procedural mem.	Plans	Implicit	Text	Tool config	Tool config	**Typed + proficiency**
Personality	-	Emergent	-	Free text	Free text	-	-	**Bipolar scales**
Behavioral patterns	Productions	Productions	Intentions	Emergent	-	-	-	**Trigger-action**
Cognitive processing	Fixed cycle	Module buffers	Reasoning cycle	LLM default	LLM default	LLM default	LLM default	**Configurable phases**
Communication	-	Vocal buffer	-	LLM default	-	-	-	**Tone/formality/ adapt.**
Perception/attention	-	Visual buffer	Belief revision	Memory stream	-	-	-	**Attention filters**
Identity composition	-	-	-	Name + paragraph	Role text	Agent config	-	**Persona binding**

III. ARCHITECTURE

A. Skills

Most agent frameworks give you a flat list of tools. Function name, description, parameters, done. Skills in The Human Model carry three additional pieces of information.

First, knowledge references. A skill can point to domain knowledge it requires. A "regulatory compliance" skill might reference specific regulatory documents and standards. The knowledge is not embedded in the skill; it is referenced. This matters when the same knowledge base needs to be shared across skills or when knowledge gets updated independently.

Second, proficiency levels. Dreyfus and Dreyfus [25] described five stages of skill acquisition: novice, advanced beginner, competent, proficient, expert. A novice follows rules explicitly and step by step. An expert recognizes patterns intuitively and acts without deliberation. When an agent has novice-level proficiency in a skill, it should reason about that skill differently. It should be more cautious, more explicit in its reasoning chain, and more likely to double-check. Expert proficiency should mean faster, more pattern-based reasoning. This is not something you get from a tool list.

Third, the tool bindings themselves. Same as everyone else.

B. Traits

A Trait is a bipolar personality dimension with a position and an influence weight. Analytical <-> intuitive. Cautious <-> bold. Empathetic <-> detached. Patient <-> urgent.

Bipolar, not categorical. This matters. Personality psychology figured out decades ago that personality dimensions are continuous and normally distributed [26]. People are not "analytical" or "intuitive." They sit somewhere on a spectrum, and context activates different regions of that spectrum. The influence weight controls how strongly a given Trait shapes the agent's behavior relative to other Traits.

You are not limited to the Big Five dimensions. They are a starting point. Domain-specific dimensions like "risk tolerance" or "detail orientation" or "formality preference" are equally valid. The architecture does not care what the poles are called. It cares that personality is decomposed into independent, configurable dimensions rather than buried in a paragraph.

C. Behaviors

Behaviors are the simplest primitive conceptually. Trigger condition, action, priority. If X happens, do Y. If multiple behaviors trigger at once, the highest priority wins.

This is production-rule territory. SOAR and ACT-R both use production systems internally. The difference is the abstraction level. In SOAR, production rules operate on working memory elements and are written in a specialized syntax. In The Human Model, Behaviors are application-level constructs. "When a user mentions a security concern, always escalate to the security team." "When the conversation has been going for more than 10 exchanges without resolution, offer to transfer to a human." These are things product managers define, not cognitive scientists.

Where Traits influence the tone and style of responses (a high-empathy agent responds warmly, a low-empathy one responds clinically), Behaviors guarantee specific actions in specific situations. The agent escalates security issues regardless of how the language model is feeling that day.

D. Cognitive Style

This one does not have a clean analogue in existing frameworks, and I think it might be the most important primitive in the architecture.

ReAct [23] gave us think-act-observe. Most agents run some variant of that loop. But consider: do you approach every problem the same way? When you are debugging a production outage at 2 AM, you are not brainstorming. You are doing rapid hypothesis testing. When you are designing a new system, you are not doing hypothesis testing. You are diverging, exploring the space, entertaining bad ideas to see where they lead. When a customer asks a routine question, you are not really "thinking" at all; you are pattern-matching and responding.

Cognitive Style makes this configurable. It has three components:

Processing phases define a sequence of reasoning stages. A diagnostic style might be: observe -> hypothesize -> test -> conclude. A creative style: diverge -> evaluate -> converge -> refine. A routine style might skip explicit reasoning entirely.

Strategies are the available methods within each phase. The "test" phase in a diagnostic style might use deductive logic, empirical checking, or cross-referencing with known cases.

Phase transitions govern when and how the agent moves between phases.

The theoretical grounding is Kahneman's System 1 and System 2 [5]. Fast intuitive processing for the familiar, slow deliberative processing for the novel or high-stakes. But instead of it being a fixed property of the architecture (like the single ReAct loop), it is a parameter. You configure where on the spectrum your agent lives, and under what conditions it shifts.

E. Communication Style

Tone. Formality. Verbosity. Adaptation rules.

This is the primitive that most obviously replaces prompt engineering, and the one where the benefit of structure is easiest to see. Everyone has written some version of "Be concise but thorough. Use a professional tone. Explain complex topics simply. Adapt your level of detail to the user's expertise." That is four separate dimensions packed into four sentences, and if you want to change the verbosity without changing the tone, you have to carefully edit the prompt and pray you did not accidentally shift something else.

Communication Accommodation Theory [28] describes how people naturally adjust their communication to match their conversation partner. The adaptation parameter makes this explicit: configure whether and how the agent converges toward the user's register, adjusts formality based on context, or modifies the detail level based on detected expertise.

F. Perception

What the agent pays attention to and what it ignores.

There is a common failure mode with LLM agents where they try to process everything in the context equally. Long conversation, uploaded documents, tool results, system instructions: all weighted the same. Real cognition does not work this way. Gibson's ecological perception [7] described how organisms perceive the environment in terms of affordances, action possibilities relevant to their current goals and capabilities. You do not see a chair as a collection of geometric shapes. You see it as something to sit on. Or, if you are trying to reach a high shelf, something to stand on.

The Perception primitive has two components. Attention filters configure what the agent prioritizes in incoming information. A security-focused agent filters for authentication tokens, access patterns, and anomalous requests. A customer support agent filters for sentiment signals and escalation indicators.

Context windows (semantic, not token-count) define how broadly the agent considers context. Tight focus on the current exchange, or broad consideration of the full conversation history plus external references. This is a separate knob from the LLM's context window size. An agent might have access to 128k tokens of context but be configured to focus primarily on the last three exchanges for routine interactions, broadening only when it detects a complex or unresolved issue.

G. Persona

A Persona binds the other six primitives into a named, switchable identity. A set of Skills, a set of Traits, a set of Behaviors, a Cognitive Style, a Communication Style, and a Perception configuration.

Goffman's dramaturgical model [30] is the reference. People present different selves in different contexts. You act one way in a board meeting and another way at a backyard barbecue. Both are authentically "you" but they emphasize different aspects of your personality and employ different communication registers.

In Cortex, an agent can have multiple Personas and switch between them based on context. A "customer-facing" Persona might have high empathy Traits, a warm Communication Style, careful Cognitive Style, and broad Perception filters. An "internal engineering" Persona for the same agent might have terse Communication Style, rapid diagnostic Cognitive Style, and tight Perception focused on technical signals.

Cortex also supports a flat mode where primitives attach directly to the agent without a Persona layer. For simple agents that only ever operate in one context, the Persona abstraction is overhead you don't need.

IV. IMPLEMENTATION

A. Why Go Lang

Cortex is written in Go, which is an unusual choice for AI tooling. Most of the ecosystem lives in Python or TypeScript.

The reasoning: agent orchestration is infrastructure. It manages concurrent requests, maintains state, talks to databases, handles multi-tenant isolation, and runs for months without restarts. These are the things Go was designed for. The language model calls are HTTP requests. The intelligence lives in the model. The orchestrator's job is to be boring and reliable.

B. Structure

Twenty-two packages. Each primitive gets its own package with its own types, store interface, and validation logic.

cortex/

Every entity in the system has a type-prefixed identifier based on UUIDv7: agt for agents, skl for skills, trt for traits, bhv for behaviors, prs_ for personas, and so on. Twelve types total. The prefix gives you human-readable type information in logs and debugging sessions. UUIDv7 gives you chronological sorting because the timestamp is in the high bits.

C. Multi-Tenancy and Isolation

Every operation propagates the tenant and application scope through Go's context. Context: The store layer enforces this. You physically cannot query across tenant boundaries without going around the store interface, and the store interface is the only way to touch data. This was a non-negotiable from day one. I have seen what happens when you try to add multi-

```
cortex/
├── engine              - Coordinates
everything
├── agent               - Agent configuration
├── skill, trait, behavior, persona
├── cognitive, communication, perception
├── run                 - Run/Step/ToolCall
state machine
├── memory              - Conversation history
+ working memory
├── checkpoint          - Human-in-the-loop
gates
├── id                  - 12 TypeID types
(UUIDv7)
├── store/postgres      - Production
persistence
├── store/memory        - In-memory store for
tests
├── plugin              - 16 lifecycle hooks
├── observability       - Prometheus metrics
├── audit_hook          - Structured audit
trail
├── api                 - 36 REST endpoints
└── extension           - Forge integration
```

Fig. 1. Code Structure of Cortex

tenancy after the fact to a system that was not designed for it, and I would rather not do that again.

D. Execution Tracking

Runs contain Steps. Steps contain ToolCalls. Each state transition is recorded. When an agent produces a bad response (and they will), you can trace back through the run: what was in the context at each step, which tools were invoked, what they returned, how long each call took, which Persona was active, which Behaviors fired. This is not optional for production deployments, but most agent frameworks treat observability as an afterthought.

E. Checkpoints

Sometimes the right answer is to stop and ask a human. Checkpoints pause agent execution at defined decision points. The run enters a waiting_checkpoint state. It stays there until a human approves or rejects. Then execution resumes or terminates.

This sounds simple, but it is surprisingly rare in agent frameworks. Full autonomy is fine for low-stakes tasks. Anything involving money, customer commitments, legal implications, or irreversible actions needs human oversight at specific points. Checkpoints give you that without killing the automation for the parts that don't need it.

F. Plugins

Sixteen lifecycle hooks covering the full agent execution cycle. The observability system (11 Prometheus counters) and the audit trail (18 tracked actions across 8 resource types) are both implemented as plugins. If you need to add a compliance check before every tool call or log every agent response to a separate system, you write a plugin. Core agent code stays clean.

Type-cached dispatch means the plugin system has zero runtime cost for hooks your plugin does not implement.

V. DISCUSION

A. Case Against "Just Use Better Prompts"

The predictable objection: this is overengineered prompt management. Traits and Cognitive Styles eventually get rendered into text that goes into the model's context window, so why bother with the intermediate representation?

Three reasons, and I think the third one is the most important.

Queryability. When an agent does something unexpected, you need to understand why. With structured primitives, you look at the Trait configuration, check which Behaviors fired, inspect the Cognitive Style that was active. With a free-text prompt, you read a paragraph and try to reverse-engineer which sentence caused the behavior. I have done both. The structured version is not even close to the same debugging experience.

Composability. Take the empathy Traits from your customer support agent. Give them to your sales agent. Take the diagnostic Cognitive Style from your engineering agent. Give it to your DevOps agent. This is trivial with structured primitives. With free-text prompts, it means copy-pasting sentences between paragraphs and hoping the overall coherence survives.

Measurability. And this is the big one. You cannot run a controlled experiment on a paragraph. You can run a controlled experiment on a single Trait value. Change cautious-bold from 0.3 to 0.7, hold everything else constant, measure the effect on task completion rates, user satisfaction, error frequency. This is basic experimental methodology, and it is impossible when every personality dimension is entangled with every other dimension in unstructured text.

B. Auditability and Recognition

The EU AI Act [31] requires that high-risk AI systems provide explanations for their decisions. An agent whose behavior derives from inspectable, structured personality parameters is fundamentally more auditable than one whose behavior emerges from prompt engineering.

You can answer "why did the agent do that?" with "Behavior #7 fired because the trigger condition was met, and the agent's cautious-bold Trait at 0.3 biased the response toward conservative action." Try giving that answer when the personality model is a paragraph of English text that the model interpreted however it felt like interpreting it.

C. What I Haven't Solved

Learning, Traits, and Cognitive Styles are currently static. You configure them, and they stay configured. An agent that becomes more confident after repeated success, or one that adjusts its Communication Style based on what has worked with a particular user before, would be more human-like. The architecture supports this (parameters are just values; updating them is straightforward), but the feedback mechanisms for deciding how and when to update personality parameters based on outcomes are not built yet. Getting that wrong is worse than not having it. An agent that incorrectly "learns" to be overconfident or inappropriately informal is worse than a static one.

Inter-agent relationships. Cortex handles multi-tenant agent deployment, but agents within a tenant do not currently model their relationships with each other. Park et al. [19] showed that relationship dynamics between agents produce genuinely interesting emergent behavior. Adding a relationship layer (trust levels, familiarity, influence weights) between co-tenant agents would enable richer multi-agent interactions. It is on the roadmap.

Controlled experiments. The architecture is grounded in validated psychology, and the framework runs in production. What I have not done yet is run formal comparisons between Human Model agents and prompt-only agents, measuring personality consistency, user satisfaction, and task performance under controlled conditions. The architecture makes these experiments easy to design (that is literally the point of structured parameters), so this is more of a "have not gotten to it yet" gap than a fundamental limitation.

VI. Conclusion

Psychology has spent decades building models of how people think, communicate, and behave. Those models are empirically validated, quantitatively measurable, and well-understood. The Human Model says: use them. Do not reinvent personality in a system prompt.

Seven primitives, each grounded in established theory. Skills with proficiency levels from Dreyfus. Traits as bipolar dimensions from the Big Five tradition. Behaviors as trigger-action rules from the production-system lineage. Cognitive Styles that generalize beyond ReAct's single loop, drawing on Kahneman's dual-process framework. Communication Styles informed by accommodation theory. Perception built on Gibson's affordance-based model. Personas as compositional identities following Goffman's dramaturgical insight.

Cortex is the implementation. Open-source, Go, multi-tenant, production-oriented. The bet is straightforward: as AI agents move from demos into systems that handle real money and real customers and real consequences, we are going to need agent models that are structured, composable, and auditable. Prompt engineering will not cut it.

The code is at https://github.com/xraph/cortex. Documentation at https://cortex.xraph.com.

References

[1] H. Chase, "LangChain: Building applications with LLMs through composability," 2022. https://github.com/langchain-ai/langchain

[2] Q. Wu, G. Bansal, J. Zhang, Y. Wu, et al., "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," arXiv:2308.08155, 2023.

[3] J. Moura, "CrewAI: Framework for orchestrating role-playing autonomous AI agents," 2024. https://github.com/crewAIInc/crewAI

[4] R. R. McCrae and O. P. John, "An introduction to the five-factor model and its applications," J. Personality, vol. 60, no. 2, pp. 175-215, 1992.

[5] D. Kahneman, Thinking, Fast and Slow. Farrar, Straus and Giroux, 2011.

[6] A. S. Rao and M. P. Georgeff, "BDI agents: From theory to practice," in Proc. ICMAS-95, San Francisco, 1995, pp. 312-319.

[7] J. J. Gibson, The Ecological Approach to Visual Perception. Houghton Mifflin, 1979.

[8] J. E. Laird, A. Newell, and P. S. Rosenbloom, "SOAR: An architecture for general intelligence," Artificial Intelligence, vol. 33, no. 1, pp. 1-64, 1987.

[9] A. Newell, Unified Theories of Cognition. Harvard University Press, 1990.

[10] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind," Psychological Review, vol. 111, no. 4, pp. 1036-1060, 2004.

[11] S. Franklin, T. Madl, S. D'Mello, and J. Snaider, "LIDA: A systems-level architecture for cognition, emotion, and learning," IEEE Trans. Autonomous Mental Development, vol. 6, no. 1, pp. 19-41, 2014.

[12] B. J. Baars, A Cognitive Theory of Consciousness. Cambridge University Press, 1988.

[13] M. E. Bratman, Intention, Plans, and Practical Reason. Harvard University Press, 1987.

[14] A. L. Howden, R. Ralph, and J. Ronnquist, "JACK intelligent agents - Summary of an agent infrastructure," in Proc. 5th Intl. Conf. Autonomous Agents, Montreal, 2001.

[15] R. H. Bordini, J. F. Hubner, and M. Wooldridge, Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley, 2007.

[16] L. R. Goldberg, "An alternative 'description of personality': The Big-Five factor structure," J. Personality and Social Psychology, vol. 59, no. 6, pp. 1216-1229, 1990.

[17] F. Mairesse and M. A. Walker, "Towards personality-based user adaptation: Psychologically informed stylistic language generation," User Modeling and User-Adapted Interaction, vol. 20, no. 3, pp. 227-278, 2010.

[18] C. Nass and K. M. Lee, "Does computer-synthesized speech manifest personality?," J. Experimental Psychology: Applied, vol. 7, no. 3, pp. 171-181, 2001.

[19] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in Proc. UIST '23, San Francisco, 2023.

[20] J. Li, M. Galley, C. Brockett, G. P. Spithourakis, J. Gao, and B. Dolan, "A persona-based neural conversation model," in Proc. ACL, Berlin, 2016, pp. 994-1003.

[21] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, "Personalizing dialogue agents: I have a dog, do you have pets too?," in Proc. ACL, Melbourne, 2018, pp. 2204-2213.

[22] Microsoft, "Semantic Kernel," 2023. https://github.com/microsoft/semantic-kernel

[23] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in Proc. ICLR, Kigali, 2023.

[24] G. Wang, Y. Xie, Y. Jiang, et al., "Voyager: An open-ended embodied agent with large language models," arXiv:2305.16291, 2023.
[25] H. L. Dreyfus and S. E. Dreyfus, Mind over Machine. The Free Press, 1986.
[26] P. T. Costa and R. R. McCrae, Revised NEO Personality Inventory (NEO-PI-R) and NEO Five-Factor Inventory (NEO-FFI) Professional Manual. Psychological Assessment Resources, 1992.
[27] B. F. Skinner, Science and Human Behavior. Macmillan, 1953.
[28] H. Giles and T. Ogay, "Communication accommodation theory," in Explaining Communication, B. B. Whaley and W. Samter, Eds. Lawrence Erlbaum, 2007, pp. 293-310.
[29] A. M. Treisman and G. Gelade, "A feature-integration theory of attention," Cognitive Psychology, vol. 12, no. 1, pp. 97-136, 1980.
[30] E. Goffman, The Presentation of Self in Everyday Life. Anchor Books, 1959.
[31] European Commission, "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (AI Act)," Official J. European Union, 2024.
[32] Availability. Cortex is open-source under https://github.com/xraph/cortex with documentation at https://cortex.xraph.com.

Rex Raphael, IEEE Young Professionals, IEEE Member, received the B.S. degree in Computing (Hons)*, Arden University,* 2021, and is currently pursuing the M.S. degree in computer science with a concentration in artificial intelligence and machine learning from Western Governors University, Salt Lake City, UT, USA.

He is currently a Staff Engineer with Kongsberg Digital, Inc., Houston, TX, USA, where he maintains and develops Kognitwin, a digital twin platform for industrial applications. He previously held engineering roles with Kongsberg Digital in Oslo, Norway, before transferring to the United States to serve oil and gas customers. His open-source contributions include the Forge distributed backend framework, the Flutter Unity View Widget (2,200+ GitHub stars), and Ultimate Backend (2,800+ GitHub stars), a multi-tenant microservices platform. He is the creator of the FARP protocol and the Cortex agent orchestration framework. His current research interests include API gateway protocol design, schema-driven service discovery, compositional AI agent architectures, and distributed systems for enterprise digital twins.