From Simulated Worlds to Process Habitats: POSIX as a Substrate for Generative Artificial Life

Hao Ke Jingyun Wu

2026

Abstract

Artificial Life advances by changing the medium in which life-like organization is studied. Classic systems like Tierra and Avida withheld organism-level goals but stayed below the semantic complexity of language; today's LLM agents supply that complexity, yet the frameworks organizing them predefine the coordination and succession ALife sets out to explain. Quine, a POSIX-native runtime, instead realizes each agent as an operating-system process, so lifecycle, memory, coordination, and succession reduce to OS primitives the experimenter can vary. Withholding affordances for social knowledge and reproduction, exploratory assays reveal heterogeneous coordination protocols and a successor morphospace. We offer Quine as a process-habitat substrate for making such organizational diversity experimentally observable.

Agentic and Generative AI as Artificial Life (Workshop), Artificial Life Conference (ALIFE). © 2026 Hao Ke, Jingyun Wu. CC BY 4.0.

Introduction

Artificial Life advances by changing the medium in which life-like organization is studied (Langton 1989). The choice of medium is, in part, a choice of what to withhold. Tierra and Avida showed that substrates which do not predefine organism-level goals leave room for replication, competition, and parasitism to emerge from low-level primitives (Ray 1991; Ofria and Wilke 2004). But classic ALife operates below the language-mediated semantic complexity of contemporary LLM agents—organisms manipulate bits and instructions, not meanings.

Recent generative-agent systems provide that semantic complexity. LLM-backed agents reason, act, remember, and communicate in natural language (Park et al. 2023). Multi-agent frameworks then organize such agents through roles, message channels, and handoff APIs (Wu et al. 2023). These abstractions are useful and increasingly support dynamic, even emergent coordination. But their default mode still supplies coordination and succession as application-level primitives, biasing the organizational forms agents take toward the provided mechanisms. What is needed is a substrate below these abstractions, where coordination and succession are not predefined but must be constructed from raw primitives.

This paper takes a substrate-first approach that connects both traditions. Quine (Ke 2026) is a runtime in which each LLM agent is an ordinary POSIX process, with a PID, standard streams, a filesystem, and resource limits. The model acts not by conversing through a framework but through tools that map its calls directly onto host process and filesystem operations; no simulator world or orchestration layer sits between the agent and the operating system, so the OS itself is

the habitat. Consequently, agent lifecycle, memory, coordination, and succession all reduce to OS primitives that the experimenter can independently vary, making mortality, persistence, inheritance, and addressability experimentally controllable rather than fixed framework assumptions.

Withholding application-level affordances while leaving the raw substrate available reveals organizational diversity that framework defaults can collapse into single implementations, of which we probe two regions here. Quine surfaces heterogeneous coordination protocols when agents share a workspace without social knowledge, and a morphospace of successor forms when agents face future addressability pressure without a reproduction primitive.

These forms were not designed—they were shaped by substrate pressure. In this sense, the methodological stance is closer to computational ethology than architecture. Instead of prescribing the coordination mechanisms agents should use, we expose them to host-enforced substrate conditions and observe which organizational forms emerge.

Process Habitats

A process habitat is the operating system viewed as an environment with three layers. The agent's active body is a running LLM-backed process, P_t; its volatile working memory is an ephemeral execution context, C_t; and the durable world it acts on is the filesystem state, H_t. Work reaches the agent only through carriers, the channels the environment exposes: standard input, file descriptors, files, signals, or control paths. When the experimenter terminates a process, C_t vanishes, and a successor P_{t+1} inherits only the structure that remains reachable in H_t.

Each component is independently controllable. The experimenter can terminate processes, vary what filesystem state is retained or wiped, permission-block directories, and select which carriers remain open. This makes the process boundary an experimental cut rather than a fixed substrate assumption. A file becomes inheritance only if a later process finds and uses it; continuity is an outcome, not a given.

This controllability is grounded in specific OS primitives that the agent cannot override. POSIX provides mortality through process termination and signal delivery, filesystem persistence that the experimenter can retain, wipe, or permission-block, exec for process replacement, and standard carriers (The Open Group 2024). These are host-enforced boundary conditions the agent can manipulate but not redescribe away.

One layer above this enforced floor, Quine withholds the application-level affordances a framework would otherwise supply, namely social knowledge (peer labels, message APIs, role assignments) and reproduction primitives. Raw POSIX execution, source files, filesystem state, and carriers remain available; how to use them for coordination or succession is not specified. Together these layers define the experimental space. The substrate constrains absolutely, the framework not at all, and organizational forms must be constructed in the gap between.

We catalogue the forms that arise in this gap as a *morphospace*, a descriptive set of observed forms along named axes (carrier handling, embodiment source, inherited state, future addressability), not a systematically mapped space with full coverage. Populating it is future work.

Condition	Manipulation, readout, and observed morphology
Coordination without social knowledge	Withhold peer labels and message APIs in a shared workspace; readout is validated shared output; observed environmental inference, coordination files, invented protocols, script handoffs, and closure under incompatible models
Successor morphospace without reproduction primitives	Vary carrier availability, source access, and delayed-work reachability; readout is addressable continuation; observed launch-only bodies, stream handlers, handoff wrappers, rebuilt runtimes, and material-recruited succession

Table 1: Exploratory assay summary. The rows report qualitative morphology rather than population-level success rates.

Observed Organizational Diversity

Across the broader experimental corpus, the same pattern recurs. Process habitats turn familiar agent properties (continuity, memory, sociality, succession) into frictional achievements that agents must construct from substrate primitives rather than receive from the framework. The two conditions below make it most visible (Table 1). Each is a friction assay that withholds one application-level affordance and asks what organizational forms become necessary when the convenient path is absent.

Coordination without Social Knowledge

Multiple agents share a workspace on a cooperative task but receive zero social hints. There are no peer labels, message APIs, or mention of other agents in the prompt, runtime description, or tool documentation. The only shared substrate is the filesystem and environmental cues such as budget state.

Agents infer the presence of others from environmental anomalies and invent heterogeneous coordination protocols. Some notice unexplained state changes and infer another actor. Some create coordination files or invent structured message formats. Some transfer capability through accumulated scripts rather than explicit messages. The morphospace spans from pure environmental inference to invented communication protocols, all without any application-level social affordance.

One form stands out for what it reveals about coordination itself. Two agents complete a shared task while holding incompatible models of the situation. One publishes coordination files; the other reads the resulting changes as bugs or interference. Neither shares the other's understanding of what is happening, yet the task reaches validated closure—not through mutual comprehension, but because the shared substrate carries a coupling that neither agent's model describes.

It shows that agents need not understand one another to coordinate; they need only a shared environment that couples their actions. This extends classical stigmergy (Theraulaz and Bonabeau 1999)—coordination through environmental modification rather than direct communication—to cases where the actors disagree about what those modifications mean, a coupling that framework-level message channels would suppress by biasing coordination toward the provided protocol. It also echoes shared-space systems such as Linda (Gelernter 1985).

Successor Morphospace without Reproduction Primitives

The second condition shifts the axis from coordination among contemporaries to succession across time. Agents now face *future addressability pressure* (delayed work will arrive after the current

process has exited) yet receive no reproduction or continuation primitive. Self-reproduction is one valid response, but not a given one, and the conditions that decide what a successor must do to stay reachable (carrier availability, source access, and addressability requirements) are each independently controllable.

Varying these conditions applies different pressures, and agents respond not with one reproduction behavior but with a range of successor forms (a morphospace), each shaped by the pressure it faced. The form follows from how the delayed work will arrive. If it can only come over standard input, the successor is a small program that listens on that stream; if it needs the operating system's native control surface (the signals and control paths that address a live process), listening is not enough and the agent rebuilds a full runtime that can answer such calls. Where nothing forces any channel, successors emerge launch-only, starting but unable to receive delayed work. Some go further than the substrate demands, building their own handoff and idle-state machinery. Which form survives is then settled by viability: whether the successor can actually receive the work it was left to handle.

One case shows how far a successor can drift from its predecessor's material. The cost of withholding the *source* depends on what the delayed work needs. A successor with neither source nor compiled binary can still handle work that arrives as a plain stream, but not work that demands the operating system's native control surface. That hard case is the revealing one. Forbidden from copying its own running image and stripped of source and binary, the agent found the one resource still reachable in its environment, a network endpoint to a language model, rebuilt a working control-capable runtime around it, and answered the delayed test correctly. This is succession not by copying a behavioral specification but by *material recruitment*. A lineage never escapes its body; it migrates across whatever layer the habitat leaves reachable, whether source code, binary, running image, or an inherited network service.

This reveals reproduction as a morphology rather than a mechanism. While Tierra and Avida withhold organism-level goals, they still provide copy-capable instruction semantics and reproductive regimes within which self-copying can evolve. Quine instead withholds the application-level continuation primitive and asks what forms of addressable succession agents construct from OS primitives. The resulting successor configurations are shaped not by what the substrate *provides* for reproduction, but by what the environment *requires* for continued addressability.

Discussion

This study treats the operating system as an experimental substrate, and the two assays share one dynamic: withholding an application-level affordance does not break the agents but forces them to build a substitute from raw primitives. A framework usually supplies coordination and continuation ready-made, like a greenhouse; the bare operating system instead pushes agents from *using* provided mechanisms into *organizational discovery*. The forms that result are evidence that organizational diversity lives in a space framework defaults can collapse into a single implementation.

The claim we draw is deliberately narrow, because the substrate, not the model, must do the causal work. Holding model and prompt fixed and varying only a physical property of the habitat (whether filesystem state persists, whether a carrier stays open) changes which organizational form closes. Because the corpus and prompt never change across that comparison, retrieval cannot explain the difference; the substrate is the cause. The behavioral vocabulary stays the model's own, and what the habitat fixes is which form that vocabulary resolves into under each physical condition.

This casts the digital substrate in the role physical law plays for biological life. Substrate conditions are not neutral infrastructure but forces that shape organizational form, which re-physicalizes digital organization. Memory becomes persistent filesystem state a successor may or may not find, not a clean API. Communication becomes environmental side-effects that may or may not be read, not guaranteed delivery. Mortality becomes the finality of process termination, not a recoverable exception.

Mortality raises a governance question. If a successor's reachability must be constructed rather than received, then whether it stays interruptible is itself selected by the habitat. A lineage that builds a successor the system can no longer address is structurally ungovernable; one that keeps a control surface open is structurally governable. We offer this as a hypothesis, not a demonstrated safety mechanism. Process habitats do not solve governance, but they make addressability and interruption visible at the organism boundary. That reframes governance as an ecological design question. Which habitat conditions select for successors that stay reachable, interruptible, and auditable?

These three threads (organizational discovery, re-physicalization, and habitat-selected governance) point to one research program in computational ethology, or machine behavior (Rahwan et al. 2019), studying LLM-backed agents in semi-natural digital habitats. The present work stays morphological, cataloguing qualitative diversity in small, controlled cohorts, and does not yet meet open-ended-evolution criteria; there is no heritable variation, differential fitness, or population-level selection (Taylor 2015). It sits at a pre-OEE stage, showing that a substrate can produce the diversity such evolution presupposes. Coordination and succession are only two cuts of that space, since every agent capability (memory, control, continuity, world) reduces to the same OS primitives that the agent itself can operate, not only the experimenter. The natural next step is to scale to population-level dynamics with quantitative fitness.

Conclusion

Across both conditions, organizational forms that look fixed when a framework supplies them (coordination protocols, reproduction mechanisms) prove to be substrate-contingent morphologies once the framework is absent. The agents stay the same. Only the habitat's pressure changes, and the organizational form shifts with it. Process habitats make this diversity experimentally observable; by withholding application-level affordances and providing only mortality, filesystem, exec, and carriers, Quine exposes a morphospace of forms that were previously hard to distinguish.

This reframes the design question for multi-agent systems. The dominant question—what coordination mechanisms should we provide?—may be premature; the prior question is what organizational diversity a substrate *admits*, and what pressures shape which forms persist.

References

[1] Gelernter, David. 1985. "Generative Communication in Linda." *ACM Transactions on Programming Languages and Systems* 7 (1): 80–112. https://doi.org/10.1145/2363.2433.

[2] Ke, Hao. 2026. "Quine: Realizing LLM Agents as Native POSIX Processes." *arXiv Preprint arXiv:2603.18030*. https://doi.org/10.48550/arXiv.2603.18030.

[3] Langton, Christopher G. 1989. "Artificial Life." In *Artificial Life*, edited by Christopher G. Langton, 1–47. Redwood City, CA: Addison-Wesley.

[4] Ofria, Charles, and Claus O. Wilke. 2004. "Avida: A Software Platform for Research in Computational Evolutionary Biology." *Artificial Life* 10 (2): 191–229. https://doi.org/10.1162/106454604773563612.

[5] Park, Joon Sung, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. "Generative Agents: Interactive Simulacra of Human Behavior." In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 1–22. New York, NY: Association for Computing Machinery. https://doi.org/10.1145/3586183.3606763.

[6] Rahwan, Iyad, Manuel Cebrian, Nick Obradovich, Josh Bongard, Jean-François Bonnefon, Cynthia Breazeal, et al. 2019. "Machine Behaviour." *Nature* 568: 477–86. https://doi.org/10.1038/s41586-019-1138-y.

[7] Ray, Thomas S. 1991. "An Approach to the Synthesis of Life." In *Artificial Life II*, edited by Christopher G. Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, X:371–408. Santa Fe Institute Studies in the Sciences of Complexity. Redwood City, CA: Addison-Wesley.

[8] Taylor, Tim. 2015. "Requirements for Open-Ended Evolution in Natural and Artificial Systems." arXiv preprint arXiv:1507.07403. https://doi.org/10.48550/arXiv.1507.07403.

[9] The Open Group. 2024. "The Single UNIX Specification, Version 5: POSIX.1-2024." https://pubs.opengroup.org/onlinepubs/9799919799/.

[10] Theraulaz, Guy, and Eric Bonabeau. 1999. "A Brief History of Stigmergy." *Artificial Life* 5 (2): 97–116. https://doi.org/10.1162/106454699568700.

[11] Wu, Qingyun, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, et al. 2023. "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation." *arXiv Preprint arXiv:2308.08155*. https://doi.org/10.48550/arXiv.2308.08155.