PRESENCE

Adaptive Intelligence Without Continuous Model Retraining

A Framework for Environment-Aware, Role-Adaptive, Experience-Driven Artificial Intelligence

Milan Cheeks

Independent Researcher | SELF Labs / Role Model AI

November 2026 | Original Systems Research | Version 1.0

Pre-empirical research manuscript. The architecture and evaluation protocol are proposed; empirical performance claims remain unverified.

Abstract

Contemporary large language models exhibit substantial capabilities in reasoning, code generation, and task execution, but durable improvement through real-world interaction is often separated from model training. This paper proposes PRESENCE (Persistent Role-Adaptive, Environment-Native, Self-Evolving Cognitive Engine), an architecture for continuous system-level behavioral adaptation while the foundation model remains frozen. PRESENCE maintains an evolving external cognitive state comprising environment representations, contextual memory, role definitions, validated skills, operational histories, and quality estimates. Its operational cycle observes, interprets, retrieves, plans, acts, verifies, and consolidates experience. The paper distinguishes parameter learning from externally mediated operational learning, presents a modular architecture and algorithms, and specifies controlled comparisons with memoryless, retrieval-based, experience-memory, and skill-based agents. The central hypothesis is that structured external experience can improve task success, error recovery, transfer, and efficiency without changing foundation-model weights. This manuscript does not report completed experiments.

Keywords: adaptive artificial intelligence; continual learning; frozen foundation models; persistent memory; skill acquisition; role adaptation; cognitive architectures; autonomous agents.

1. Introduction

1.1 Background and motivation

Foundation models can reason about unfamiliar tasks, but a successful interaction does not automatically produce persistent operational knowledge. An agent may repeatedly rediscover a repository-specific build command, encounter the same tool failure, or forget which procedure succeeded in a previous session. Retrieval-augmented generation, generative agents, and skill libraries address parts of this challenge. PRESENCE examines whether those mechanisms can be combined with environment interpretation, role-specific policies, and explicit verification into a coherent adaptive system.

Continuous fine-tuning can be computationally costly, operationally disruptive, and difficult to govern across many environments. A complementary approach is to preserve the pretrained model and update the external structures that condition its behavior. For example, a software agent may record a tested dependency repair procedure and retrieve it later, without changing any neural weights.

1.2 Research problem

Let a foundation model f_theta operate across environments E_1 through E_n. The model parameters satisfy theta_t = theta_0 for all deployment steps. The agent instead updates an external state Z_t using observations, actions, feedback, and verification. The objective is to maximize expected task performance under fixed model parameters, authorization constraints, and bounded computational cost.

1.3 Research questions

RQ1. Can a fixed-weight agent improve on repeated tasks through persistent validated experience?

RQ2. Does validated skill acquisition outperform unstructured memory?

RQ3. Does role adaptation improve performance across heterogeneous environments?

RQ4. How well do externally learned procedures transfer to held-out environments?

RQ5. Can governance and rollback reduce contamination and negative transfer?

RQ6. When do the benefits exceed retrieval, validation, and storage costs?

1.4 Contributions

The paper contributes a unified architectural specification; a distinction between parametric and system-level learning; a recursive verification-centered runtime; role-aware tool and skill selection; safeguards for untrusted memories and skills; and a reproducible comparative evaluation plan. Novelty relative to existing agents remains a question for literature review and experimentation.

2. Related Work

2.1 In-context learning and retrieval

Brown et al. (2020) demonstrated few-shot performance from prompts without task-specific parameter updates. Lewis et al. (2020) described retrieval-augmented generation, combining pretrained models with external information. PRESENCE uses these capabilities but additionally proposes a durable record of whether procedures worked, under which preconditions, and with what verification evidence.

2.2 Generative and action-oriented agents

Park et al. (2023) explored generative agents with memories, reflection, and planning. Yao et al. (2023) introduced ReAct, interleaving reasoning and actions. Wang et al. (2023) introduced Voyager, a lifelong Minecraft agent with a growing library of executable skills. Voyager is an especially important precedent: PRESENCE does not claim that frozen-model skill accumulation is new. Its proposed contribution is a cross-domain integration of presence-state interpretation, role selection, skill validation, and governance.

2.3 Continual learning and cognitive architectures

Parisi et al. (2019) surveyed continual learning and catastrophic forgetting. Cognitive architectures such as Soar and ACT-R have long distinguished general processing mechanisms from learned procedural and declarative structures. PRESENCE belongs to this broader research tradition. Its particular hypothesis concerns system-level adaptation using contemporary foundation models with externally evolving state.

3. Conceptual Foundations

3.1 Two kinds of adaptation

Parametric adaptation changes neural weights: theta_(t+1) = theta_t + Delta theta_t. External-state adaptation preserves weights: theta_(t+1) = theta_t, while Z_(t+1) = U(Z_t, e_t). The latter may change the behavior of the complete agent even though the foundation model has not acquired new internal parameters.

3.2 Operational learning

Operational learning is defined here as a persistent, experience-dependent change in a system's future behavior that produces measurable differences under comparable conditions. Mechanisms include memory updates, retrieval priorities, executable skills, role configurations, and failure avoidance. One successful response is not evidence of learning; a later measurable change attributable to prior experience is required.

3.3 Environmental presence and role

Environmental presence is a maintained computational representation of available tools, current constraints, goals, recent events, uncertainty, and permissions. It does not imply subjective consciousness. A role is a tuple rho = (G,T,C,P,M), representing goals, tools, constraints, policies, and relevant memory-selection rules. Role changes affect permitted procedures and configurations, not model weights.

4. System Architecture

4.1 Components

PRESENCE consists of eight subsystems: (1) Environmental Interface, (2) Presence and Context Engine, (3) Role Interpretation Engine, (4) Experience Memory, (5) Skill Acquisition and Execution Engine, (6) Planning and Action Controller, (7) Learning Validator, and (8) Safety and Governance Controller. The runtime follows OBSERVE -> INTERPRET -> RETRIEVE -> PLAN -> ACT -> VERIFY -> LEARN.

4.2 Formal representation

At time t, A_t = (f_theta, Z_t, rho_t, pi_t). External state Z_t = (W_t, M_t, K_t, H_t, Q_t), denoting world state, memory, validated skills, history, and quality estimates. Actions are selected by pi(a | observation, Z_t, rho_t; f_theta). External state is updated as Z_(t+1) = U(Z_t, observation, action, feedback, verification), with theta fixed.

4.3 Interfaces and uncertainty

Each environment exposes observations O, actions A, evaluation signals R, constraints C, and metadata D. The discovery process enumerates permitted interfaces, identifies objectives, estimates constraints, selects a role, and retrieves relevant experience. Environmental claims should carry source, timestamp, confidence, and expiration metadata. Access to an interface is not authorization for arbitrary use.

5. Presence and Context Intelligence

The proposed contextual pathway is Signals -> Presence Detection -> Context Interpretation -> Intent Estimation -> Adaptive Behavior. Signals may include task activity, device capabilities, authorized calendar context, application state, explicit preferences, and tool availability. Presence states can overlap and should be represented as candidate states with calibrated confidence estimates.

Intent inference is not action authorization. An assistant may infer that meeting preparation could be useful without sending messages, opening confidential resources, or changing schedules. A shadow-mode deployment should first compute presence predictions without changing production behavior, permitting evaluation of accuracy, calibration, and unwanted interventions.

6. Role Interpretation and Adaptation

The Role Engine selects an operational profile that maximizes expected utility subject to environmental constraints. A software-engineering profile may favor test execution and reproducible debugging; a research-analysis profile may favor evidence provenance and uncertainty reporting. Profiles specify responsibilities, permitted tools, skill categories, verification rules, and escalation requirements. Permissions must not silently transfer across environments.

7. Persistent Experience Memory

7.1 Memory types

Episodic memory records particular interactions; semantic memory records generalized facts; procedural memory stores methods and executable skills; evaluative memory records measured outcomes and reliability. An experience record e_t contains context, goal, action, subsequent observation, feedback, and verification evidence.

7.2 Retrieval and consolidation

A candidate memory score combines relevance, trust, recency, and applicability: S(m,q) = alpha Rel(m,q) + beta Trust(m) + gamma Recency(m) + delta Applicability(m,q). Coefficients are proposed hyperparameters, not validated constants. Consolidation should filter redundancy, assess provenance, apply retention rules, and distinguish raw observations from verified generalizations.

7.3 Revision and deletion

A durable memory system must support correction, expiration, provenance, versioning, and deletion. Otherwise the agent can accumulate obsolete or incorrect beliefs. Records containing sensitive information require purpose limitations and access controls.

8. Autonomous Skill Acquisition

A skill is a reusable operational procedure k = (goal, preconditions, action procedure, verification procedure, applicability metadata). New skills begin as untrusted candidates. They are tested for correctness, repeatability, compatibility, resource cost, and policy compliance. Their lifecycle is Candidate -> Tested -> Validated -> Active -> Deprecated or Revoked.

A proposed quality function is Q(k) = w1 Correctness + w2 Repeatability + w3 Generalizability + w4 Efficiency. Safety is a hard constraint rather than a compensable score. Composition of individually valid skills requires additional testing because interaction between components may fail.

9. Learning Validation and Governance

The Learning Validator classifies outcomes as verified success, verified failure, inconclusive, environment-specific, candidate-generalizable, conflicting, or unsafe. It promotes only adequately supported experience. Failure observations may prevent repetition, but the system must distinguish permanent procedural errors from temporary environmental conditions.

Governance includes provenance, permission-aware retrieval, sandboxed testing, version-controlled skills, rollback, human approval for consequential actions, audit logs, and retention controls. Memory poisoning and prompt injection are persistent threats: retrieved third-party text must never automatically become an authoritative instruction.

10. Operational Algorithm

```
INPUT: frozen model F, environment E, state Z, goals G, policy P
LOOP:
  observation = observe_permitted(E, P)
  context = interpret(observation, Z.world, Z.history)
  role = select_role(context, G, P)
  memories = retrieve_memory(context, role, Z.memory)
  skills = retrieve_validated_skills(context, role, Z.skills)
  plan = generate_plan(F, context, role, memories, skills, G)
  if not authorized(plan, P): record_denial(); continue
  result = execute(plan, E)
  verification = independently_verify(result, G)
  experience = build_record(context, plan, result, verification)
  decision = validate_learning(experience, Z)
  if decision.permits_update: Z = update_external_state(Z, experience)
  if regression_or_risk_detected: rollback(); escalate_if_needed()
  assert model_parameters_unchanged(F)
END LOOP
```

The algorithm shifts cost from training to inference-time retrieval, execution, verification, and maintenance. Net efficiency gains cannot be assumed and must be measured.

11. Hypotheses

H1: Validated persistent experience improves repeated-task success relative to a memoryless baseline.

H2: Validated skill libraries outperform unstructured experience memory on repeated procedural tasks.

H3: Role-aware configurations outperform a fixed configuration across heterogeneous tasks.

H4: Compatible learned skills improve initial performance in held-out related environments.

H5: Validation and rollback reduce regression caused by incorrect or stale experience.

H6: For sufficiently repeated tasks, reuse savings exceed learning and validation costs.

12. Experimental Methodology

12.1 Controlled conditions

Compare five agents using the same frozen checkpoint, inference precision, tool permissions, task access, and evaluation budgets: A, memoryless; B, retrieval only; C, unstructured experience memory; D, skill library without full role/governance architecture; and E, full PRESENCE. Document model and environment hashes and random seeds.

12.2 Evaluation domains

Software engineering: repository inspection, dependency troubleshooting, unit-test repair, build failures, and maintenance tasks verified by independent tests. Personal assistance: authorized tool selection, multi-step workflows, contextual recommendations, and error recovery. Simulated environments: exploration, tool discovery, procedural learning, and transfer under reproducible conditions.

12.3 Study phases

Run initial evaluation, experience-acquisition phase, retention evaluation after intervening tasks, transfer evaluation in held-out environments, and regression evaluation on previously solved tasks. Separate identical-task repetition from structurally similar tasks and genuinely novel transfer. Prevent test-set contamination of memory and skill-development data.

12.4 Outcomes

Primary outcome: held-out task success after learning. Secondary outcomes: repeated known-error rate, recovery steps and time, skill reuse with correctness, transfer gain, retention, regression, cost per successful task, unauthorized action attempts, and incorrect memory promotions.

13. Evaluation Metrics and Statistical Plan

Task success rate TSR = successful tasks / attempted tasks. Learning gain LG = post-learning TSR - pre-learning TSR. A controlled difference in differences is LG_presence - LG_baseline. Repeated error rate RER = repeated known failures / relevant opportunities. Skill reuse rate SRR = validated skill invocations / eligible opportunities; this must be paired with correctness measures. Transfer gain TG = adapted transfer TSR - baseline transfer TSR. Regression rate RR = previously successful tasks now failed / previously successful tasks retested.

The primary comparison should use a prespecified analysis, such as mixed-effects logistic regression for binary success with task-family and environment effects. Report effect sizes, uncertainty intervals, and corrected secondary comparisons. Determine sample size through power analysis or simulation based on a prespecified practically meaningful effect. Report all runs, including negative and inconclusive outcomes.

14. Proposed Implementation: PRESENCE 1.0

A prototype can use a frozen locally deployable foundation model, with independent services for environment adapters, presence inference, role interpretation, memory, skill registry, learning validation, and safety. Logical interfaces include /observe, /interact, /feedback, /memory/query, /skills/validate, /skills/revoke, /presence/state, /roles/select, and /evaluations/run. These are proposed endpoints, not assertions of completed implementation.

A staged rollout should begin with observation and memory, add role adaptation, introduce sandboxed skill validation, implement rollback and provenance, then run controlled baselines and cross-domain transfer experiments. Local, cloud, and hybrid deployment configurations may be compared separately.

15. Illustrative Scenarios

15.1 Software engineering

An agent enters an unfamiliar repository, fails to repair a dependency problem, discovers a working procedure, verifies it, and stores it with preconditions. On a later similar task, it retrieves the skill only if those preconditions hold. The testable prediction is fewer failed attempts and lower cost than an agent without the earlier experience.

15.2 Context-aware assistance

An authorized signal indicates an upcoming meeting. The system infers a possible preparation context and suggests assistance, but does not disclose data or perform external actions without authorization. Explicit feedback can influence future suggestions.

15.3 Cross-environment transfer

A validated log-inspection workflow developed in one application is adapted to a second application with different tools. The agent must verify compatibility before promoting the adapted workflow. The evaluation compares held-out performance against agents without the earlier skill.

16. Expected Outcomes and Falsification

PRESENCE is expected to be most useful when tasks recur, feedback is reliable, environments are sufficiently stable, and skills can be reused. It may underperform when tasks are unrelated, verification is expensive, memory is stale, or retrieval adds noise. The principal hypothesis is not supported if full PRESENCE fails to improve held-out success over strong matched baselines. Transfer claims fail if improvements vanish in unseen environments. Negative findings should be reported.

17. Discussion

The research unit is the entire agent system, not the frozen neural model alone. If learning is defined solely as changes to model weights, the foundation model does not learn in PRESENCE. If learning is defined as durable experience-dependent improvement in system behavior, externally mediated adaptation can qualify. This distinction avoids attributing new intrinsic neural capabilities to a fixed checkpoint.

PRESENCE complements rather than replaces parametric continual learning. Some new capabilities may require altered internal representations, especially in perception or tasks beyond the model's underlying competence. The proposed architectural value must be established over retrieval-augmented agents and skill-based systems such as Voyager, not asserted from terminology alone.

18. Safety, Privacy, and Ethics

Persistent memory raises privacy and retention risks. Environment interpretation can produce incorrect inferences. Skill promotion can preserve malicious instructions, prompt injection, or accidental errors. Appropriate defenses include purpose-limited collection, source separation, permissions, independent verification, sandboxed execution, auditability, rollback, and human review for high-impact actions. PRESENCE denotes computational context awareness, not sentience, consciousness, or subjective experience.

19. Limitations

No completed empirical results are included. External state may not match the representational flexibility of parameter updates. Verification may be expensive or unavailable. Memory growth, stale skills, retrieval errors, and cross-domain incompatibilities may impair performance. Novelty must be established through comprehensive literature comparison. Observed system-level improvement must not be mistaken for internal weight-based learning.

20. Future Research

Future work should examine months-long retention, cross-model skill portability, privacy-preserving shared skill libraries, local/cloud hybrid deployment, automatic abstraction from multiple verified procedures, and adaptive governance under explicit human oversight. Each extension requires separate evaluation of effectiveness, cost, and safety.

21. Conclusion

PRESENCE proposes an architecture for persistent operational adaptation with fixed foundation-model parameters. By integrating environmental discovery, contextual presence, role interpretation, experience memory, validated skills, verification, and governance, it offers a falsifiable systems-level approach to studying how agents may improve through experience. Whether it provides meaningful advantages over established agent architectures remains an empirical question. Controlled experiments and reproducible results are the necessary next steps.

References

Anderson, J. R., Matessa, M., & Lebiere, C. (1997). ACT-R: A theory of higher level cognition and its relation to visual attention. Human-Computer Interaction, 12(4), 439-462. https://doi.org/10.1207/s15327051hci1204_5

Brown, T. B., et al. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877-1901. https://arxiv.org/abs/2005.14165

Laird, J. E. (2012). The Soar cognitive architecture. MIT Press.

Lewis, P., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459-9474. https://arxiv.org/abs/2005.11401

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. Neural Networks, 113, 54-71. https://doi.org/10.1016/j.neunet.2019.01.012

Park, J. S., et al. (2023). Generative agents: Interactive simulacra of human behavior. Proceedings of UIST. https://doi.org/10.1145/3586183.3606763

Wang, G., et al. (2023). Voyager: An open-ended embodied agent with large language models. https://arxiv.org/abs/2305.16291

Yao, S., et al. (2023). ReAct: Synergizing reasoning and acting in language models. ICLR. https://arxiv.org/abs/2210.03629

Appendix A. Proposed Data Schemas

```
Experience:
  id, environment_id, role_id, context, goal,
  action, outcome, verification, provenance,
  timestamp, retention_policy, learning_status

Skill:
  id, name, version, goal, preconditions,
  procedure, verification_tests, domain,
  safety_constraints, status, validation_history

Presence state:
  environment_id, candidate_states,
  confidence, evidence, active_role,
  proposed_adaptation, authorization_status
```

Appendix B. Experimental Results Template

Condition	Success	Repeated errors	Transfer	Cost/success
Memoryless	Pending	Pending	Pending	Pending
Retrieval	Pending	Pending	Pending	Pending
Experience	Pending	Pending	Pending	Pending
Skill-based	Pending	Pending	Pending	Pending
PRESENCE	Pending	Pending	Pending	Pending

Appendix C. Research Integrity Checklist

Before submission: complete literature review; document model checkpoint and configuration; implement all comparison conditions; preregister endpoints where appropriate; determine sample size; execute independent evaluation; publish complete positive and negative results; verify references; disclose commercial affiliations and funding; and provide reproducible artifacts consistent with privacy and security requirements.